RM0486
Reference manual

STM32N6x5/x7xx Arm®-based 32-bit MCUs

Introduction

This document is addressed to application developers. It provides complete information on how to use the STM32N645xx, STM32N647xx, STM32N655xx, and STM32N657xx (referred to as STM32N6x5/x7xx) microcontrollers memory and peripherals.

For ordering information, mechanical and electrical device characteristics, refer to the datasheet.

For information on the Arm® Cortex®-M55 core, refer to the corresponding Arm® Technical Reference Manual available on http://infocenter.arm.com.

STM32N6x5/x7xx microcontrollers include ST state-of-the-art patented technology.

Related documents

- STM32N6x5/x7xx datasheet (DS14791)
- STM32 Cortex®-M55 MCUs programming manual (PM0273)
- UM3234 “*How to proceed with boot ROM on STM32N6 MCUs*”
- STM32N6xx MCU errata sheet (ES0621)

Contents

1 Documentation conventions 156

1.1 General information 156

1.2 List of abbreviations for registers 156

1.3 Register reset value 157

1.4 Glossary 157

1.5 Availability of peripherals 157

2 Memory and bus architecture 158

2.1 System architecture 158

2.1.1 Introduction 158

2.1.2 Bus architecture 158

2.2 Bus network-on-chip (NoC) 162

2.2.1 STNoC AXI 162

2.2.2 NPU_NIC network interconnect 162

2.2.3 Multi-layer AHB interconnect (AHBM) 162

2.3 Memory organization 163

2.3.1 Introduction 163

2.3.2 Memory map and register boundary addresses 164

3 System security 182

3.1 Key security features 182

3.2 Secure boot 183

3.3 Secure provisioning 183

3.4 Temporal isolation 184

3.5 Resource isolation 184

3.5.1 Trustzone filtering using IDAU/SAU 185

3.5.2 Privileged OS-controlled isolation 186

3.5.3 RIF infrastructure 186

3.5.4 Multi-tenancy 187

3.5.5 NPU timeslicing 188

3.5.6 NPU cache (CACHEAXI) management 189

3.5.7 Memory firewalls 189

3.5.8 RIF-aware peripherals 190

3.6 Secure storage . . . 193
3.6.1 Battery backup registers . . . 193
3.6.2 Secure RAM and registers . . . 193
3.6.3 Fuse secrets . . . 193
3.6.4 Hardware key management . . . 193
3.6.5 Unique ID . . . 195
3.7 Tamper detection and response . . . 195
3.8 Crypto engines . . . 197
3.8.1 Accelerators . . . 197
3.8.2 Memory cipher engine (MCE) . . . 199
3.8.3 NPU encryption . . . 199
3.9 Access controlled debug . . . 200

4 Boot and security control (BSEC) . . . 201
4.1 Introduction . . . 201
4.2 BSEC main features . . . 201
4.3 BSEC functional description . . . 202
4.3.1 BSEC block diagram . . . 202
4.3.2 BSEC internal signals . . . 202
4.3.3 BSEC reset and clocks . . . 203
4.3.4 Organization and use of fuses . . . 204
4.3.5 Operations on fuses . . . 205
4.3.6 BSEC read and programming status reporting . . . 207
4.3.7 Lifecycle management . . . 208
4.3.8 Epoch registers . . . 209
4.3.9 BSEC local access control . . . 210
4.3.10 Temporal isolation in BSEC . . . 210
4.3.11 BSEC debug and trace control . . . 211
4.3.12 Device features control . . . 212
4.3.13 Scratch registers . . . 212
4.3.14 JTAG registers . . . 212
4.3.15 STM32 root hardware unique key (RHUK) . . . 212
4.3.16 BSEC error management . . . 213
4.3.17 BSEC tamper response mechanisms . . . 213
4.4 BSEC interrupts . . . 213
4.5 BSEC registers . . . 213

4.5.1 BSEC fuse word w value register (BSEC_FVRw) 214
4.5.2 BSEC sticky programming lock register x (BSEC_SPLOCKx) 215
4.5.3 BSEC sticky write lock register x (BSEC_SWLOCKx) 215
4.5.4 BSEC sticky reload lock register x (BSEC_SRLOCKx) 216
4.5.5 BSEC OTP valid register x (BSEC_OTPVLDRx) 216
4.5.6 BSEC shadowed fuses status register x (BSEC_SFSRx) 217
4.5.7 BSEC OTP control register (BSEC_OTPCR) 217
4.5.8 BSEC write data register (BSEC_WDR) 218
4.5.9 BSEC scratch register x (BSEC_SCRATCHRx) 219
4.5.10 BSEC lock register (BSEC_LOCKR) 219
4.5.11 BSEC JTAG input register (BSEC_JTAGINR) 220
4.5.12 BSEC JTAG output register (BSEC_JTAGOUTR) 220
4.5.13 BSEC unmap register (BSEC_UNMAPR) 220
4.5.14 BSEC status register (BSEC_SR) 221
4.5.15 BSEC OTP status register (BSEC_OTPSR) 222
4.5.16 BSEC epoch register (BSEC_EPOCHRx) 223
4.5.17 BSEC epoch selection control register (BSEC_EPOCHSELCR) 224
4.5.18 BSEC debug control register (BSEC_DBGCR) 224
4.5.19 BSEC AP unlock (BSEC_AP_UNLOCK) 225
4.5.20 BSEC HDPL status register (BSEC_HDPLSR) 226
4.5.21 BSEC HDPL control register (BSEC_HDPLCR) 226
4.5.22 BSEC next HDPL control register (BSEC_NEXTHDPLCR) 227
4.5.23 BSEC write once scratch register x (BSEC_WOSCRx) 227
4.5.24 BSEC hot reset count register (BSEC_HRCR) 228
4.5.25 BSEC warm reset count register (BSEC_WRCR) 228
4.5.26 BSEC register map 228

5 OTP mapping (OTP) 231

6 Resource isolation framework security controller (RIFSC) 248

6.1 Introduction 248
6.2 RIFSC main features 248
6.3 RIFSC functional description 248
6.3.1 RIFSC reset and clocks 248
6.3.2 RISUP 248
6.3.3 RCC security settings for RIF-aware peripherals and RAMs 249
6.3.4 RIMU 250

6.4 RIFSC registers 251
6.4.1 RIFSC RISC slave configuration register x (RIFSC_RISC_CR) 251
6.4.2 RIFSC RISC slave security configuration register x (RIFSC_RISC_SECCFGRx) 252
6.4.3 RIFSC RISFC slave privileged register x (RIFSC_RISC_PRIVCFGRx) 252
6.4.4 RIFSC RISC slave resource configuration lock register x (RIFSC_RISC_RCFGLOCKRx) 253
6.4.5 RIFSC RIMC master configuration register (RIFSC_RIMC_CR) 253
6.4.6 RIFSC RIMC master attribute register x (RIFSC_RIMC_ATTRx) 254
6.4.7 RIFSC peripheral protection status register 0 (RIFSC_PPSR0) 254
6.4.8 RIFSC peripheral protection status register 1 (RIFSC_PPSR1) 255
6.4.9 RIFSC peripheral protection status register 2 (RIFSC_PPSR2) 255
6.4.10 RIFSC peripheral protection status register 3 (RIFSC_PPSR3) 255
6.4.11 RIFSC peripheral protection status register 4 (RIFSC_PPSR4) 256
6.4.12 RIFSC peripheral protection status register 5 (RIFSC_PPSR5) 256
6.4.13 RIFSC register map 257

7 Resource isolation slave unit for address space protection (full version) (RISAF) 258

7.1 RISAF introduction 258
7.2 RISAF main features 258
7.3 RISAF implementation 259
7.4 RISAF functional description 260
7.4.1 RISAF block diagram 260
7.4.2 RISAF internal signals 260
7.4.3 RISAF reset and clocks 261
7.4.4 RISAF address space management 261
7.4.5 RISAF programming 262
7.4.6 RISAF runtime modification of region or subregion configuration 263
7.4.7 Managing illegal accesses 264
7.5 RISAF registers 265
7.5.1 RISAF configuration register (RISAF_CR) 265
7.5.2 RISAF illegal access status register (RISAF_IASR) 265
7.5.3 RISAF illegal access clear register (RISAF_IACR) 266
7.5.4 RISAF illegal access error status register (RISAF_IAESR) 266
7.5.5 RISAF illegal address register (RISAF_IADDR) 267
7.5.6 RISAF region x configuration register (RISAF_REGx_CFGR) 267

7.5.7 RISAF region x start-address register (RISAF_REGx_STARTR) 268
7.5.8 RISAF region x end-address register (RISAF_REGx_ENDR) 269
7.5.9 RISAF region x CID configuration register (RISAF_REGx_CIDCFGR) 269
7.5.10 RISAF region x subregion A configuration register (RISAF_REGx_ACFGR) 270
7.5.11 RISAF region x subregion A start-address register (RISAF_REGx_ASTARTR) 272
7.5.12 RISAF region x subregion A end-address register (RISAF_REGx_AENDR) 272
7.5.13 RISAF region x subregion A nested mode register (RISAF_REGx_ANESTR) 273
7.5.14 RISAF region x subregion B configuration register (RISAF_REGx_BCFGR) 274
7.5.15 RISAF region x subregion B start-address register (RISAF_REGx_BSTARTR) 275
7.5.16 RISAF region x subregion B end-address register (RISAF_REGx_BENDR) 276
7.5.17 RISAF region x subregion B nested mode register (RISAF_REGx_BNESTR) 277
7.5.18 RISAF register map 278

8 Illegal access controller (IAC) 280

8.1 IAC introduction 280
8.2 IAC main features 280
8.3 IAC implementation 280
8.4 IAC functional description 281
8.4.1 IAC block diagram 281
8.4.2 IAC internal signals 282
8.4.3 IAC reset and clocks 282
8.4.4 IAC use in RIF 282
8.4.5 IAC management by trusted application 282
8.5 IAC interrupts 283
8.6 IAC registers 283
8.6.1 IAC interrupt enable register x (IAC_IERx) 283
8.6.2 IAC interrupt status register x (IAC_ISRx) 283
8.6.3 IAC interrupt clear register x (IAC_ICRx) 284
8.6.4 IAC ILAC input status register x (IAC_IISRx) 284
8.6.5 IAC register map 285

9 Boot modes 286

9.1 Boot after system reset 286

9.1.1 Flash boot 286

9.1.2 Serial boot 286

9.1.3 Development boot 286

9.2 Boot from a low-power mode 287

10 SRAM configuration controller (RAMCFG) 288

10.1 RAMCFG introduction 288

10.2 RAMCFG main features 288

10.3 RAMCFG functional description 288

10.3.1 Internal SRAM features 288

10.3.2 FLEXRAM control 289

10.3.3 ECC (BKPSRAM) 292

10.4 RAMCFG low-power modes 293

10.5 RAMCFG interrupts 294

10.6 RAMCFG registers 295

10.6.1 RAMCFG AXISRAM1 control register (RAMCFG_AXISRAM1CR) . . . 295

10.6.2 RAMCFG AXISRAMx interrupt status register (RAMCFG_AXISRAMxISR) 295

10.6.3 RAMCFG AXISRAMx erase key register (RAMCFG_AXISRAMxERKEYR) 296

10.6.4 RAMCFG AXISRAMx control register (RAMCFG_AXISRAMxCR) . . . 296

10.6.5 RAMCFG AHBSRAMx control register (RAMCFG_AHBSRAMxCR) . 297

10.6.6 RAMCFG AHBSRAMx interrupt status register (RAMCFG_AHBSRAMxISR) 297

10.6.7 RAMCFG AHBSRAMx erase key register (RAMCFG_AHBSRAMxERKEYR) 298

10.6.8 RAMCFG VENCRAM control register (RAMCFG_VENCRAMCR) . . . 298

10.6.9 RAMCFG VENCRAM interrupt status register (RAMCFG_VENCRAMISR) 299

10.6.10 RAMCFG VENCRAM erase key register (RAMCFG_VENCRAMERKEYR) 299

10.6.11 RAMCFG BKPSRAM control register (RAMCFG_BKPSRAMCR) . . . 300

10.6.12 RAMCFG BKPSRAM interrupt enable register (RAMCFG_BKPSRAMIER) 301

10.6.13 RAMCFG BKPSRAM interrupt status register (RAMCFG_BKPSRAMISR) 301

10.6.14 RAMCFG BKPSRAM single error address register (RAMCFG_BKPSRAMESEAR) . . . 302
10.6.15 RAMCFG BKPSRAM double error address register (RAMCFG_BKPSRAMEDEAR) . . . 302
10.6.16 RAMCFG BKPSRAM interrupt clear register (RAMCFG_BKPSRAMICR) . . . 303
10.6.17 RAMCFG BKPSRAM ECC key register (RAMCFG_BKPSRAMECCKEYR) . . . 303
10.6.18 RAMCFG BKPSRAM erase key register (RAMCFG_BKPSRAMERKEYR) . . . 304
10.6.19 RAMCFG FLEXRAM control register (RAMCFG_FLEXRAMCR) . . . 304
10.6.20 RAMCFG FLEXRAM interrupt status register (RAMCFG_FLEXRAMISR) . . . 305
10.6.21 RAMCFG FLEXRAM erase key register (RAMCFG_FLEXRAMERKEYR) . . . 305
10.6.22 RAMCFG register map . . . 306

11 Texture cache (ICACHE) . . . 308

11.1 ICACHE introduction . . . 308
11.2 ICACHE main features . . . 308
11.3 ICACHE implementation . . . 309
11.4 ICACHE functional description . . . 309
11.4.1 ICACHE block diagram . . . 310
11.4.2 ICACHE reset and clocks . . . 310
11.4.3 ICACHE TAG memory . . . 311
11.4.4 Direct-mapped ICACHE (1-way cache) . . . 312
11.4.5 ICACHE enable . . . 313
11.4.6 Cacheable and noncacheable traffic . . . 313
11.4.7 Cacheable accesses . . . 314
11.4.8 ICACHE maintenance . . . 314
11.4.9 ICACHE performance monitoring . . . 315
11.4.10 ICACHE boot . . . 315
11.5 ICACHE low-power modes . . . 315
11.6 ICACHE error management and interrupts . . . 316
11.7 ICACHE registers . . . 316
11.7.1 ICACHE control register (ICACHE_CR) . . . 316
11.7.2 ICACHE status register (ICACHE_SR) . . . 317
11.7.3 ICACHE interrupt enable register (ICACHE_IER) . . . 318

11.7.4 ICACHE flag clear register (ICACHE_FCR) 318
11.7.5 ICACHE hit monitor register (ICACHE_HMONR) 319
11.7.6 ICACHE miss monitor register (ICACHE_MMONR) 319
11.7.7 ICACHE register map 319

12 AXI cache (CACHEAXI) 321

12.1 CACHEAXI introduction 321
12.2 CACHEAXI main features 322
12.3 CACHEAXI implementation 323
12.4 CACHEAXI functional description 324
12.4.1 CACHEAXI block diagram 325
12.4.2 CACHEAXI reset and clocks 325
12.4.3 CACHEAXI TAG memory 326
12.4.4 CACHEAXI enable 329
12.4.5 Cacheable and noncacheable AXI traffic to slave cache port 329
12.4.6 Cacheable accesses 330
12.4.7 AXI traffic to master port 331
12.4.8 AXI traffic to slave SRAM port 332
12.4.9 CACHEAXI security 332
12.4.10 CACHEAXI maintenance 332
12.4.11 CACHEAXI performance monitoring 334
12.4.12 CACHEAXI boot 335
12.5 CACHEAXI low-power modes 335
12.6 CACHEAXI error management and interrupts 335
12.7 CACHEAXI registers 337
12.7.1 CACHEAXI control register 1 (CACHEAXI_CR1) 337
12.7.2 CACHEAXI status register (CACHEAXI_SR) 338
12.7.3 CACHEAXI interrupt enable register (CACHEAXI_IER) 339
12.7.4 CACHEAXI flag clear register (CACHEAXI_FCR) 340
12.7.5 CACHEAXI read-hit monitor register (CACHEAXI_RHMONR) 341
12.7.6 CACHEAXI read-miss monitor register (CACHEAXI_RMMONR) 341
12.7.7 CACHEAXI read-allocate miss monitor register (CACHEAXI_RAMMONR) 341
12.7.8 CACHEAXI eviction monitor register (CACHEAXI_EVIMONR) 342
12.7.9 CACHEAXI write-hit monitor register (CACHEAXI_WHMONR) 342
12.7.10 CACHEAXI write-miss monitor register (CACHEAXI_WMMONR) . . . 342

12.7.11 CACHEAXI write-allocate miss monitor register (CACHEAXI_WAMMONR) . . . 343
12.7.12 CACHEAXI write-through monitor register (CACHEAXI_WTMONR) . 343
12.7.13 CACHEAXI control register 2 (CACHEAXI_CR2) . . . 343
12.7.14 CACHEAXI command range start address register (CACHEAXI_CMDRSADDRR) . . . 344
12.7.15 CACHEAXI command range end address register (CACHEAXI_CMDREADDRR) . . . 345
12.7.16 CACHEAXI register map . . . 345

13 Power control (PWR) . . . 347

13.1 PWR introduction . . . 347
13.2 PWR main features . . . 347
13.3 PWR block diagram . . . 348
13.3.1 PWR pins and internal signals . . . 348
13.4 Power supplies . . . 350
13.4.1 System supply startup . . . 353
13.4.2 Core domain . . . 355
13.4.3 PWR external supply . . . 356
13.4.4 Backup domain . . . 357
13.4.5 Retention domain . . . 358
13.4.6 Analog supply . . . 360
13.5 Power supply supervision . . . 360
13.5.1 Power-on reset (POR)/power-down reset (PDR) . . . 361
13.5.2 Brownout reset (BOR) . . . 361
13.5.3 Vdda18pmu_ok reset . . . 362
13.5.4 Vddcore_ok reset . . . 363
13.5.5 V_{DDCORE} monitoring . . . 363
13.5.6 Programmable voltage detector (PVD) . . . 364
13.5.7 Peripheral voltage monitoring (PVM) . . . 365
13.5.8 Battery voltage thresholds . . . 366
13.5.9 Temperature thresholds . . . 367
13.6 Power management . . . 368
13.6.1 Operating modes . . . 368
13.6.2 Voltage scaling . . . 373
13.6.3 Power management examples . . . 374
13.7 Low-power modes . . . 376

13.7.1 Slowing down system clocks . . . 377
13.7.2 Controlling peripheral clocks . . . 377
13.7.3 Entering low-power modes . . . 377
13.7.4 Exiting low-power modes . . . 377
13.7.5 Sleep mode . . . 378
13.7.6 Stop mode . . . 379
13.7.7 Standby mode . . . 381
13.7.8 Power mode output pins . . . 383
13.8 PWR security and privileged protection . . . 383
13.8.1 Secure/nonsecure access filtering . . . 384
13.8.2 Privileged/unprivileged access filtering . . . 385
13.9 PWR registers . . . 385
13.9.1 PWR control register 1 (PWR_CR1) . . . 385
13.9.2 PWR control register 2 (PWR_CR2) . . . 386
13.9.3 PWR control register 3 (PWR_CR3) . . . 387
13.9.4 PWR control register 4 (PWR_CR4) . . . 387
13.9.5 PWR voltage scaling control register (PWR_VOSCR) . . . 388
13.9.6 PWR backup domain control register 1 (PWR_BDCR1) . . . 389
13.9.7 PWR backup domain control register 2 (PWR_BDCR2) . . . 390
13.9.8 PWR disable backup protection control register (PWR_DBPCR) . . . 391
13.9.9 PWR CPU control register (PWR_CPUCR) . . . 391
13.9.10 PWR supply voltage monitoring control register 1 (PWR_SVMCR1) . 392
13.9.11 PWR supply voltage monitoring control register 2 (PWR_SVMCR2) . 394
13.9.12 PWR supply voltage monitoring control register 3 (PWR_SVMCR3) . 395
13.9.13 PWR wake-up clear register (PWR_WKUPCR) . . . 397
13.9.14 PWR wake-up status register (PWR_WKUPSR) . . . 398
13.9.15 PWR wake-up enable and polarity register (PWR_WKUPEPR) . . . 399
13.9.16 PWR security configuration register (PWR_SECCFGR) . . . 401
13.9.17 PWR privilege configuration register (PWR_PRIVCFGR) . . . 402
13.9.18 PWR debug control register 1 (PWR_CRCFG1) . . . 405
13.9.19 PWR debug control register 2 (PWR_CRCFG2) . . . 406
13.9.20 PWR Debug control register 3 (PWR_CRCFG3) . . . 406
13.9.21 PWR register map . . . 408

14 Reset and clock control (RCC) . . . 411

14.1 RCC main features . . . 411
14.2 RCC power domains . . . 411

14.3 RCC block diagram . . . 412
14.4 RCC pins and internal signals . . . 412
14.5 Functional description of RCC reset . . . 414
14.5.1 Reset from the PWR . . . 414
14.5.2 System and application resets (sys_rst, nreset_rstn) . . . 415
14.5.3 NRST reset . . . 416
14.5.4 Low-power mode security reset (lpwr_rst) . . . 416
14.5.5 Backup domain reset . . . 417
14.5.6 CoreSight debug reset . . . 417
14.5.7 Option-byte loading . . . 417
14.5.8 Reset of peripherals . . . 418
14.5.9 Reset pulse control (RPCTL) . . . 418
14.5.10 Reset coverage summary . . . 419
14.5.11 Reset source identification . . . 420
14.5.12 Power-on and wake-up sequences . . . 420
14.6 Functional description of RCC clocks . . . 421
14.6.1 Clock naming convention . . . 424
14.6.2 Oscillator description . . . 424
14.6.3 Clock security system (CSS) . . . 432
14.6.4 Clock output generation (MCO1/MCO2) . . . 435
14.6.5 PLL description . . . 435
14.6.6 System clocks . . . 440
14.6.7 Clock generation in Stop and Standby modes . . . 443
14.6.8 Peripheral clock distribution . . . 444
14.6.9 General clock concept overview . . . 468
14.6.10 Peripheral allocation . . . 469
14.6.11 Peripheral clock-gating control . . . 470
14.6.12 CPU and bus matrix clock-gating control . . . 472
14.6.13 Low-power emulation modes . . . 473
14.7 RCC interrupts . . . 474
14.8 RCC application information . . . 474
14.8.1 HSE crystal auto-detection . . . 474
14.8.2 Calibration and clock frequency measurement using TIMx . . . 475
14.8.3 Clock monitoring . . . 476
14.8.4 Clock frequency limits . . . 476
14.9 RCC security . . . 477

14.9.1 Internal register protection 478
14.9.2 Internal register write-protection 479
14.10 RCC registers 479
14.10.1 RCC control register (RCC_CR) 479
14.10.2 RCC status register (RCC_SR) 481
14.10.3 RCC Stop mode control register (RCC_STOPCR) 482
14.10.4 RCC configuration register 1 (RCC_CFGR1) 483
14.10.5 RCC configuration register 2 (RCC_CFGR2) 484
14.10.6 RCC backup domain protection register (RCC_BDCR) 486
14.10.7 RCC reset status register for hardware (RCC_HWRSR) 487
14.10.8 RCC reset register (RCC_RSR) 488
14.10.9 RCC LSE configuration register (RCC_LSECFGR) 490
14.10.10 RCC MSI configuration register (RCC_MSICFGR) 491
14.10.11 RCC HSI configuration register (RCC_HSICFGR) 492
14.10.12 RCC HSI monitor control register (RCC_HSIMCR) 493
14.10.13 RCC HSI monitor status register (RCC_HSIMSR) 493
14.10.14 RCC HSE configuration register (RCC_HSECFGR) 494
14.10.15 RCC PLL1 configuration register 1 (RCC_PLL1CFGR1) 495
14.10.16 RCC PLL1 configuration register 2 (RCC_PLL1CFGR2) 496
14.10.17 RCC PLL1 configuration register 3 (RCC_PLL1CFGR3) 497
14.10.18 RCC PLL2 configuration register 1 (RCC_PLL2CFGR1) 498
14.10.19 RCC PLL2 configuration register 2 (RCC_PLL2CFGR2) 499
14.10.20 RCC PLL2 configuration register 3 (RCC_PLL2CFGR3) 500
14.10.21 RCC PLL3 configuration register 1 (RCC_PLL3CFGR1) 501
14.10.22 RCC PLL3 configuration register 2 (RCC_PLL3CFGR2) 502
14.10.23 RCC PLL3 configuration register 3 (RCC_PLL3CFGR3) 503
14.10.24 RCC PLL4 configuration register 1 (RCC_PLL4CFGR1) 504
14.10.25 RCC PLL4 configuration register 2 (RCC_PLL4CFGR2) 505
14.10.26 RCC PLL4 configuration register 3 (RCC_PLL4CFGR3) 505
14.10.27 RCC IC1 configuration register (RCC_IC1CFGR) 507
14.10.28 RCC IC2 configuration register (RCC_IC2CFGR) 508
14.10.29 RCC IC3 configuration register (RCC_IC3CFGR) 509
14.10.30 RCC IC4 configuration register (RCC_IC4CFGR) 509
14.10.31 RCC IC5 configuration register (RCC_IC5CFGR) 510
14.10.32 RCC IC6 configuration register (RCC_IC6CFGR) 511
14.10.33 RCC IC7 configuration register (RCC_IC7CFGR) 512
14.10.34 RCC IC8 configuration register (RCC_IC8CFGR) 512

14.10.35 RCC IC9 configuration register (RCC_IC9CFGR) 513
14.10.36 RCC IC10 configuration register (RCC_IC10CFGR) 514
14.10.37 RCC IC11 configuration register (RCC_IC11CFGR) 514
14.10.38 RCC IC12 configuration register (RCC_IC12CFGR) 515
14.10.39 RCC IC13 configuration register (RCC_IC13CFGR) 516
14.10.40 RCC IC14 configuration register (RCC_IC14CFGR) 517
14.10.41 RCC IC15 configuration register (RCC_IC15CFGR) 518
14.10.42 RCC IC16 configuration register (RCC_IC16CFGR) 518
14.10.43 RCC IC17 configuration register (RCC_IC17CFGR) 519
14.10.44 RCC IC18 configuration register (RCC_IC18CFGR) 520
14.10.45 RCC IC19 configuration register (RCC_IC19CFGR) 520
14.10.46 RCC IC20 configuration register (RCC_IC20CFGR) 521
14.10.47 RCC clock-source interrupt enable register (RCC_CIER) 522
14.10.48 RCC clock-source interrupt flag register (RCC_CIFR) 523
14.10.49 RCC clock-source interrupt clear register (RCC_CICR) 525
14.10.50 RCC clock configuration for independent peripheral register 1 (RCC_CCIPR1) 527
14.10.51 RCC clock configuration for independent peripheral register 2 (RCC_CCIPR2) 528
14.10.52 RCC clock configuration for independent peripheral register 3 (RCC_CCIPR3) 530
14.10.53 RCC clock configuration for independent peripheral register 4 (RCC_CCIPR4) 530
14.10.54 RCC clock configuration for independent peripheral register 5 (RCC_CCIPR5) 532
14.10.55 RCC clock configuration for independent peripheral register 6 (RCC_CCIPR6) 534
14.10.56 RCC clock configuration for independent peripheral register 7 (RCC_CCIPR7) 536
14.10.57 RCC clock configuration for independent peripheral register 8 (RCC_CCIPR8) 537
14.10.58 RCC clock configuration for independent peripheral register 9 (RCC_CCIPR9) 538
14.10.59 RCC clock configuration for independent peripheral register 12 (RCC_CCIPR12) 540
14.10.60 RCC clock configuration for independent peripheral register 13 (RCC_CCIPR13) 542
14.10.61 RCC clock configuration for independent peripheral register 14 (RCC_CCIPR14) 544
14.10.62 RCC miscellaneous configurations reset register (RCC_MISCRSTR) 545

14.10.63 RCC embedded memories reset register (RCC_MEMRSTR) 546
14.10.64 RCC AHB1 reset register (RCC_AHB1RSTR) 548
14.10.65 RCC AHB2 reset register (RCC_AHB2RSTR) 549
14.10.66 RCC AHB3 reset register (RCC_AHB3RSTR) 550
14.10.67 RCC AHB4 reset register (RCC_AHB4RSTR) 551
14.10.68 RCC AHB5 reset register (RCC_AHB5RSTR) 553
14.10.69 RCC APB1L reset register (RCC_APB1LRSTR) 556
14.10.70 RCC APB1H reset register (RCC_APB1HRSTR) 560
14.10.71 RCC APB2 reset register (RCC_APB2RSTR) 561
14.10.72 RCC APB4L reset register (RCC_APB4LRSTR) 563
14.10.73 RCC APB4H reset register (RCC_APB4HRSTR) 565
14.10.74 RCC APB5 reset register (RCC_APB5RSTR) 566
14.10.75 RCC IC dividers enable register (RCC_DIVENR) 567
14.10.76 RCC embedded buses enable register (RCC_BUSENR) 570
14.10.77 RCC miscellaneous configurations enable register (RCC_MISCENR) 572
14.10.78 RCC embedded memories enable register (RCC_MEMENR) 573
14.10.79 RCC AHB1 enable register (RCC_AHB1ENR) 575
14.10.80 RCC AHB2 enable register (RCC_AHB2ENR) 575
14.10.81 RCC AHB3 enable register (RCC_AHB3ENR) 576
14.10.82 RCC AHB4 enable register (RCC_AHB4ENR) 577
14.10.83 RCC AHB5 enable register (RCC_AHB5ENR) 580
14.10.84 RCC APB1L enable register (RCC_APB1LENR) 584
14.10.85 RCC APB1H enable register (RCC_APB1HENR) 587
14.10.86 RCC APB2 enable register (RCC_APB2ENR) 588
14.10.87 RCC APB3 enable register (RCC_APB3ENR) 591
14.10.88 RCC APB4L enable register (RCC_APB4LENR) 591
14.10.89 RCC APB4H enable register (RCC_APB4HENR) 593
14.10.90 RCC APB5 enable register (RCC_APB5ENR) 594
14.10.91 RCC embedded buses sleep enable register (RCC_BUSLPENR) . . . 595
14.10.92 RCC miscellaneous configurations sleep enable register (RCC_MISCLPENR) 596
14.10.93 RCC embedded memories sleep enable register (RCC_MEMLPENR) 597
14.10.94 RCC AHB1 sleep enable register (RCC_AHB1LPENR) 599
14.10.95 RCC AHB2 sleep enable register (RCC_AHB2LPENR) 599
14.10.96 RCC AHB3 sleep enable register (RCC_AHB3LPENR) 600
14.10.97 RCC AHB4 sleep enable register (RCC_AHB4LPENR) 601

14.10.98 RCC AHB5 sleep enable register (RCC_AHB5LPENR) 604
14.10.99 RCC APB1L sleep enable register (RCC_APB1LLPENR) 607
14.10.100RCC APB1H sleep enable register (RCC_APB1HLPENR) 611
14.10.101RCC APB2 sleep enable register (RCC_APB2LPENR) 612
14.10.102RCC APB3 sleep enable register (RCC_APB3LPENR) 614
14.10.103RCC APB4L sleep enable register (RCC_APB4LLPENR) 615
14.10.104RCC APB4H sleep enable register (RCC_APB4HLPENR) 617
14.10.105RCC APB5 sleep enable register (RCC_APB5LPENR) 618
14.10.106RCC reset duration control register (RCC_RDCR) 619
14.10.107RCC oscillator secure configuration register 0 (RCC_SECCFGR0) . . 620
14.10.108RCC oscillator privilege configuration register 0 (RCC_PRIVCFGR0) 620
14.10.109RCC oscillator lock configuration register 0 (RCC_LOCKCFGR0) . . . 621
14.10.110RCC oscillator public configuration register 0 (RCC_PUBCFGR0) . . . 622
14.10.111RCC PLL secure configuration register 1 (RCC_SECCFGR1) 623
14.10.112RCC PLL privilege configuration register 1 (RCC_PRIVCFGR1) 624
14.10.113RCC PLL lock configuration register 1 (RCC_LOCKCFGR1) 624
14.10.114RCC PLL public configuration register1 (RCC_PUBCFGR1) 625
14.10.115RCC divider secure configuration register 2 (RCC_SECCFGR2) 626
14.10.116RCC divider privilege configuration register 2 (RCC_PRIVCFGR2) . . 628
14.10.117RCC divider lock configuration register 2 (RCC_LOCKCFGR2) 630
14.10.118RCC divider public configuration register 2 (RCC_PUBCFGR2) 633
14.10.119RCC system secure configuration register 3 (RCC_SECCFGR3) . . . 635
14.10.120RCC system privilege configuration register3 (RCC_PRIVCFGR3) . . 636
14.10.121RCC system lock configuration register 3 (RCC_LOCKCFGR3) 637
14.10.122RCC system public configuration register 3 (RCC_PUBCFGR3) 638
14.10.123RCC bus secure configuration register 4 (RCC_SECCFGR4) 638
14.10.124RCC bus privilege configuration register 4 (RCC_PRIVCFGR4) 640
14.10.125RCC bus lock configuration register 4 (RCC_LOCKCFGR4) 642
14.10.126RCC bus public configuration register 4 (RCC_PUBCFGR4) 643
14.10.127RCC bus public configuration register 4 (RCC_PUBCFGR5) 645
14.10.128RCC control set register (RCC_CSR) 647
14.10.129RCC Stop mode configuration set register (RCC_STOPCSR) 648
14.10.130RCC miscellaneous reset register (RCC_MISCRSTSR) 648
14.10.131RCC memory reset register (RCC_MEMRSTSR) 649
14.10.132RCC AHB1 reset register (RCC_AHB1RSTSR) 650
14.10.133RCC AHB2 reset register (RCC_AHB2RSTSR) 650

14.10.134RCC AHB3 reset register (RCC_AHB3RSTSR) 651
14.10.135RCC AHB4 reset register (RCC_AHB4RSTSR) 652
14.10.136RCC AHB5 reset register (RCC_AHB5RSTSR) 653
14.10.137RCC APB1L reset register (RCC_APB1LRSTSR) 654
14.10.138RCC APB1H reset register (RCC_APB1HRSTSR) 656
14.10.139RCC APB2 reset register (RCC_APB2RSTSR) 657
14.10.140RCC APB4L reset register (RCC_APB4LRSTSR) 658
14.10.141RCC APB4H reset register (RCC_APB4HRSTSR) 659
14.10.142RCC APB5 reset register (RCC_APB5RSTSR) 659
14.10.143RCC divider enable register (RCC_DIVENSR) 660
14.10.144RCC bus enable register (RCC_BUSENSR) 661
14.10.145RCC miscellaneous enable register (RCC_MISCENSR) 662
14.10.146RCC memory enable register (RCC_MEMENSR) 663
14.10.147RCC AHB1 enable register (RCC_AHB1ENSR) 664
14.10.148RCC AHB2 enable register (RCC_AHB2ENSR) 664
14.10.149RCC AHB3 enable register (RCC_AHB3ENSR) 665
14.10.150RCC AHB4 enable register (RCC_AHB4ENSR) 666
14.10.151RCC AHB5 enable register (RCC_AHB5ENSR) 667
14.10.152RCC APB1L enable register (RCC_APB1LENSR) 668
14.10.153RCC APB1H enable register (RCC_APB1HENSR) 670
14.10.154RCC APB2 enable register (RCC_APB2ENSR) 671
14.10.155RCC APB3 enable register (RCC_APB3ENSR) 672
14.10.156RCC APB4L enable register (RCC_APB4LENSR) 672
14.10.157RCC APB4H enable register (RCC_APB4HENSR) 674
14.10.158RCC APB5 enable register (RCC_APB5ENSR) 674
14.10.159RCC bus sleep enable register (RCC_BUSLPENSR) 675
14.10.160RCC miscellaneous sleep enable register (RCC_MISCLPENSR) . . . 675
14.10.161RCC memory sleep enable register (RCC_MEMLPENSR) 676
14.10.162RCC AHB1 sleep enable register (RCC_AHB1LPENSR) 677
14.10.163RCC AHB2 sleep enable register (RCC_AHB2LPENSR) 677
14.10.164RCC AHB3 sleep enable register (RCC_AHB3LPENSR) 678
14.10.165RCC AHB4 sleep enable register (RCC_AHB4LPENSR) 679
14.10.166RCC AHB5 sleep enable register (RCC_AHB5LPENSR) 680
14.10.167RCC APB1L sleep enable register (RCC_APB1LLPENSR) 681
14.10.168RCC APB1H sleep enable register (RCC_APB1HLPENSR) 683
14.10.169RCC APB2 sleep enable register (RCC_APB2LPENSR) 684
14.10.170RCC APB3 sleep enable register (RCC_APB3LPENSR) 685

14.10.171RCC APB4L sleep enable register (RCC_APB4LLPENSR) 685
14.10.172RCC APB4H sleep enable register (RCC_APB4HLPENSR) 687
14.10.173RCC APB5 sleep enable register (RCC_APB5LPENSR) 687
14.10.174RCC oscillator privilege configuration set register 0 (RCC_PRIVCFGSR0) 688
14.10.175RCC oscillator public configuration set register 0 (RCC_PUBCFGSR0) 688
14.10.176RCC PLL privilege configuration set register 1 (RCC_PRIVCFGSR1) 689
14.10.177RCC PLL public configuration set register 1 (RCC_PUBCFGSR1) . . 690
14.10.178RCC divider privilege configuration set register 2 (RCC_PRIVCFGSR2) 691
14.10.179RCC divider public configuration set register 2 (RCC_PUBCFGSR2) 692
14.10.180RCC system privilege configuration set register 3 (RCC_PRIVCFGSR3) 694
14.10.181RCC system public configuration set register 3 (RCC_PUBCFGSR3) 695
14.10.182RCC privilege configuration set register 4 (RCC_PRIVCFGSR4) . . . 696
14.10.183RCC public configuration set register 4 (RCC_PUBCFGSR4) 697
14.10.184RCC public configuration set register 5 (RCC_PUBCFGSR5) 698
14.10.185RCC control clear register (RCC_CCR) 700
14.10.186RCC Stop mode configuration clear register (RCC_STOPCCR) 700
14.10.187RCC miscellaneous reset clear register (RCC_MISCRSTCR) 701
14.10.188RCC memory reset clear register (RCC_MEMRSTCR) 702
14.10.189RCC AHB1 reset clear register (RCC_AHB1RSTCR) 703
14.10.190RCC AHB2 reset clear register (RCC_AHB2RSTCR) 703
14.10.191RCC AHB3 reset clear register (RCC_AHB3RSTCR) 704
14.10.192RCC AHB4 reset clear register (RCC_AHB4RSTCR) 704
14.10.193RCC AHB5 reset clear register (RCC_AHB5RSTCR) 705
14.10.194RCC APB1L reset clear register (RCC_APB1LRSTCR) 707
14.10.195RCC APB1H reset clear register (RCC_APB1HRSTCR) 709
14.10.196RCC APB2 reset clear register (RCC_APB2RSTCR) 709
14.10.197RCC APB4L reset clear register (RCC_APB4LRSTCR) 711
14.10.198RCC APB4H reset clear register (RCC_APB4HRSTCR) 712
14.10.199RCC APB5 reset clear register (RCC_APB5RSTCR) 712
14.10.200RCC divider enable clear register (RCC_DIVENCR) 713
14.10.201RCC bus enable clear register (RCC_BUSENCR) 714
14.10.202RCC miscellaneous enable clear register (RCC_MISCENCR) 715
14.10.203RCC memory enable clear register (RCC_MEMENCR) 715

14.10.204RCC AHB1 enable clear register (RCC_AHB1ENCR) 716
14.10.205RCC AHB2 enable clear register (RCC_AHB2ENCR) 717
14.10.206RCC AHB3 enable clear register (RCC_AHB3ENCR) 717
14.10.207RCC AHB4 enable clear register (RCC_AHB4ENCR) 718
14.10.208RCC AHB5 enable clear register (RCC_AHB5ENCR) 719
14.10.209RCC APB1L enable clear register (RCC_APB1LENCR) 721
14.10.210RCC APB1H enable clear register (RCC_APB1HENCR) 723
14.10.211RCC APB2 enable clear register (RCC_APB2ENCR) 724
14.10.212RCC APB3 enable clear register (RCC_APB3ENCR) 725
14.10.213RCC APB4L enable clear register (RCC_APB4LENCR) 725
14.10.214RCC APB4H enable clear register (RCC_APB4HENCR) 726
14.10.215RCC APB5 enable clear register (RCC_APB5ENCR) 727
14.10.216RCC bus sleep enable clear register (RCC_BUSLPENCR) 728
14.10.217RCC miscellaneous sleep enable clear register (RCC_MISCLPENCR) 728
14.10.218RCC memory sleep enable clear register (RCC_MEMLPENCR) 729
14.10.219RCC AHB1 sleep enable clear register (RCC_AHB1LPENCR) 730
14.10.220RCC AHB2 sleep enable clear register (RCC_AHB2LPENCR) 730
14.10.221RCC AHB3 sleep enable clear register (RCC_AHB3LPENCR) 731
14.10.222RCC AHB4 sleep enable clear register (RCC_AHB4LPENCR) 732
14.10.223RCC AHB5 sleep enable clear register (RCC_AHB5LPENCR) 733
14.10.224RCC APB1L sleep enable clear register (RCC_APB1LLPENCR) . . . 735
14.10.225RCC APB1H sleep enable clear register (RCC_APB1HLPENCR) . . . 736
14.10.226RCC APB2 sleep enable clear register (RCC_APB2LPENCR) 737
14.10.227RCC APB3 sleep enable clear register (RCC_APB3LPENCR) 738
14.10.228RCC APB4L sleep enable clear register (RCC_APB4LLPENCR) . . . 739
14.10.229RCC APB4H sleep enable clear register (RCC_APB4HLPENCR) . . . 740
14.10.230RCC APB5 sleep enable clear register (RCC_APB5LPENCR) 740
14.10.231RCC oscillator privilege configuration clear register 0 (RCC_PRIVCFGCR0) 741
14.10.232RCC oscillator public configuration clear register 0 (RCC_PUBCFGCR0) 742
14.10.233RCC PLL privilege configuration clear register 1 (RCC_PRIVCFGCR1) 742
14.10.234RCC PLL public configuration clear register 1 (RCC_PUBCFGCR1) . 743
14.10.235RCC divider privilege configuration clear register 2 (RCC_PRIVCFGCR2) 744
14.10.236RCC divider public configuration clear register 2 (RCC_PUBCFGCR2) 745

14.10.237RCC system privilege configuration clear register 3 (RCC_PRIVCFGCR3) . . . 747
14.10.238RCC system public configuration clear register 3 (RCC_PUBCFGCR3) . . . 748
14.10.239RCC privilege configuration clear register 4 (RCC_PRIVCFGCR4) . . 749
14.10.240RCC public configuration clear register 4 (RCC_PUBCFGCR4) 751
14.10.241RCC public configuration clear register 4 (RCC_PUBCFGCR5) 752
14.10.242RCC register map . . . 754

15 General-purpose I/Os (GPIO) . . . 783

15.1 Introduction . . . 783
15.2 GPIO main features . . . 783
15.3 GPIO functional description . . . 783
15.3.1 General-purpose I/O (GPIO) . . . 785
15.3.2 I/O pin AF multiplexer and mapping . . . 786
15.3.3 I/O port control registers . . . 786
15.3.4 I/O port data registers . . . 787
15.3.5 I/O data bitwise handling . . . 787
15.3.6 GPIO locking mechanism . . . 787
15.3.7 I/O AF input/output . . . 788
15.3.8 External interrupt/wake-up lines . . . 788
15.3.9 Input configuration . . . 788
15.3.10 Output configuration . . . 788
15.3.11 AF configuration . . . 789
15.3.12 Analog configuration . . . 790
15.3.13 Using HSE or LSE oscillator pins as GPIOs . . . 791
15.3.14 Using GPIO pins in V_{SW} supply domain . . . 791
15.3.15 Advanced I/O configurations . . . 791
15.3.16 I/O pin isolation using TrustZone . . . 791
15.3.17 I/O pin isolation using privilege . . . 793
15.4 GPIO registers . . . 794
15.4.1 GPIO port x mode register (GPIOx_MODER) (x = A to H, N to Q) . . . 794
15.4.2 GPIO port x output type register (GPIOx_OTYPER) (x =A to H, N to Q) . . . 794
15.4.3 GPIO port x output speed register (GPIOx_OSPEEDR) (x =A to H, N to Q) . . . 795
15.4.4 GPIO port x pull-up/pull-down register (GPIOx_PUPDR) (x= A to H, N to Q) . . . 795

15.4.5 GPIO port x input data register (GPIOx_IDR) (x = A to H, N to Q) . . . 796
15.4.6 GPIO port x output data register (GPIOx_ODR) (x = A to H, N to Q) . 796
15.4.7 GPIO port x bit set/reset register (GPIOx_BSRR) (x = A to H, N to Q) 797
15.4.8 GPIO port x configuration lock register (GPIOx_LCKR) (x = A to H, N to Q) . . . 798
15.4.9 GPIO port x AF low register (GPIOx_AFRL) (x = A to H, N to Q) 799
15.4.10 GPIO port x AF high register (GPIOx_AFRH) (x = A to H, N to Q) . . . 799
15.4.11 GPIO port x bit reset register (GPIOx_BRR) (x = A to H, N to Q) 800
15.4.12 GPIO port x secure configuration register (GPIOx_SECCFGR) (x = A to H, N to Q) . . . 800
15.4.13 GPIO port x privileged configuration register (GPIOx_PRIVCFGR) (x = A to H, N to Q) . . . 801
15.4.14 GPIO port x resource configuration lock register (GPIOx_RCFGLOCKR) (x = A to H, N to Q) . . . 802
15.4.15 GPIO port x delay low register (GPIOx_DELAYRL) (x = A to H, N to Q) . . . 802
15.4.16 GPIO port x delay high register (GPIOx_DELAYRH) (x = A to H, N to Q) . . . 803
15.4.17 GPIO port x advanced configuration low register (GPIOx_ADVCFGRL) (x = A to H, N to Q) . . . 803
15.4.18 GPIO port x advanced configuration high register (GPIOx_ADVCFGRH) (x =A to H, N to Q) . . . 804
15.4.19 GPIO register map . . . 806

16 System configuration controller (SYSCFG) . . . 809

16.0.1 I/O compensation cell . . . 809
16.1 SYSCFG registers . . . 810
16.1.1 SYSCFG boot pin control register (SYSCFG_BOOTCR) . . . 810
16.1.2 SYSCFG Cortex-M55 control register (SYSCFG_CM55CR) . . . 811
16.1.3 SYSCFG Cortex-M55 TCM control register (SYSCFG_CM55TCMCR) . . . 811
16.1.4 SYSCFG Cortex-CM55 memory RW margin register (SYSCFG_CM55RWMCR) . . . 812
16.1.5 SYSCFG Cortex-M55 SVTOR control register (SYSCFG_INITSVTORCR) . . . 813
16.1.6 SYSCFG Cortex-M55 NSVTOR control register (SYSCFG_INITNSVTORCR) . . . 813
16.1.7 SYSCFG Cortex-M55 reset type control register (SYSCFG_CM55RSTCR) . . . 814
16.1.8 SYSCFG Cortex-M55 P-AHB write posting control register (SYSCFG_CM55PAHBWPR) . . . 814

16.1.9 SYSCFG VENCRAM control register (SYSCFG_VENCRAMCR) 815
16.1.10 SYSCFG potential tamper reset register (SYSCFG_POTTAMPRSTCR) 815
16.1.11 SYSCFG NPUNIC QoS control register (SYSCFG_NPUNICQOSCR) 816
16.1.12 SYSCFG AHB-AXI bridge early write response control register (SYSCFG_ICNEWRCR) 816
16.1.13 SYSCFG ICN clock gating control register (SYSCFG_ICNCGCR) . . . 817
16.1.14 SYSCFG VDDIO4 compensation cell control register (SYSCFG_VDDIO4CCCR) 818
16.1.15 SYSCFG VDDIO4 compensation cell status register (SYSCFG_VDDIO4CCSR) 818
16.1.16 SYSCFG VDDIO5 compensation cell control register (SYSCFG_VDDIO5CCCR) 819
16.1.17 SYSCFG VDDIO5 compensation cell status register (SYSCFG_VDDIO5CCSR) 820
16.1.18 SYSCFG VDDIO2 compensation cell control register (SYSCFG_VDDIO2CCCR) 820
16.1.19 SYSCFG VDDIO2 compensation cell status register (SYSCFG_VDDIO2CCSR) 821
16.1.20 SYSCFG VDDIO3 compensation cell control register (SYSCFG_VDDIO3CCCR) 822
16.1.21 SYSCFG VDDIO3 compensation cell status register (SYSCFG_VDDIO3CCSR) 822
16.1.22 SYSCFG VDD compensation cell control register (SYSCFG_VDDCCCR) 823
16.1.23 SYSCFG VDD compensation cell status register (SYSCFG_VDDCCSR) 824
16.1.24 SYSCFG control timer break register (SYSCFG_CBR) 824
16.1.25 SYSCFG DMA CID secure control register (SYSCFG_SEC_AIDCR) . 825
16.1.26 SYSCFG FMC retiming logic control register (SYSCFG_FMC_RETIMECR) 826
16.1.27 SYSCFG NPU RAM interleaving control register (SYSCFG_NPU_ICNCR) 826
16.1.28 SYSCFG boot pin status register (SYSCFG_BOOTSR) 827
16.1.29 SYSCFG AHB write posting address error register (SYSCFG_AHBWP_ERROR_SR) 827
16.1.30 SYSCFG SMPS observable signals through HDP selection configuration register (SYSCFG_SMPSHDPCR) 828
16.1.31 SYSCFG DMA CID nonsecure control register (SYSCFG_SECPRIV_AIDCR) 828
16.1.32 SYSCFG device ID register (SYSCFG_DEVICEID) 829
16.1.33 SYSCFG register map 829

17 Peripherals interconnect matrix 833
17.1 Introduction 833
17.2 Connection summary 833

18 High-performance direct memory access controller (HPDMA) 834
18.1 HPDMA introduction 834
18.2 HPDMA main features 834
18.3 HPDMA implementation 835
18.3.1 HPDMA channels 835
18.3.2 HPDMA allowed AXI maximum burst length 836
18.3.3 HPDMA in low-power modes 836
18.3.4 HPDMA requests 836
18.3.5 HPDMA block requests 841
18.3.6 HPDMA channels with peripheral early termination 841
18.3.7 HPDMA triggers 842
18.4 HPDMA functional description 845
18.4.1 HPDMA block diagram 845
18.4.2 HPDMA channel state and direct programming without any linked-list 846
18.4.3 HPDMA channel suspend and resume 847
18.4.4 HPDMA channel abort and restart 848
18.4.5 HPDMA linked-list data structure 849
18.4.6 Linked-list item transfer execution 852
18.4.7 HPDMA channel state and linked-list programming in run-to-completion mode 852
18.4.8 HPDMA channel state and linked-list programming in link step mode 856
18.4.9 HPDMA channel state and linked-list programming 863
18.4.10 HPDMA FIFO-based transfers 865
18.4.11 HPDMA transfer request and arbitration 881
18.4.12 HPDMA triggered transfer 885
18.4.13 HPDMA circular buffering with linked-list programming 886
18.4.14 HPDMA transfer in peripheral flow-control mode 888
18.4.15 HPDMA secure/nonsecure channel 889
18.4.16 HPDMA privileged/unprivileged channel 890
18.4.17 HPDMA error management 890
18.5 HPDMA in debug mode 892

18.6 HPDMA in low-power modes 892
18.7 HPDMA interrupts 893
18.8 HPDMA registers 894
18.8.1 HPDMA secure configuration register (HPDMA_SECCFGR) 894
18.8.2 HPDMA privileged configuration register (HPDMA_PRIVCFGR) 895
18.8.3 HPDMA configuration lock register (HPDMA_RCFGLOCKR) 896
18.8.4 HPDMA nonsecure masked interrupt status register (HPDMA_MISR) 896
18.8.5 HPDMA secure masked interrupt status register (HPDMA_SMISR) . . 897
18.8.6 HPDMA channel x linked-list base address register (HPDMA_CxLBAR) 898
18.8.7 HPDMA channel x CID register (HPDMA_CxCIDCFGR) 899
18.8.8 HPDMA channel x semaphore control register (HPDMA_CxSEMCR) 900
18.8.9 HPDMA channel x flag clear register (HPDMA_CxFCR) 902
18.8.10 HPDMA channel x status register (HPDMA_CxSR) 903
18.8.11 HPDMA channel x control register (HPDMA_CxCR) 904
18.8.12 HPDMA channel x transfer register 1 (HPDMA_CxTR1) 907
18.8.13 HPDMA channel x transfer register 2 (HPDMA_CxTR2) 911
18.8.14 HPDMA channel x block register 1 (HPDMA_CxBR1) 915
18.8.15 HPDMA channel x alternate block register 1 (HPDMA_CxBR1) 916
18.8.16 HPDMA channel x source address register (HPDMA_CxSAR) 919
18.8.17 HPDMA channel x destination address register (HPDMA_CxDAR) . . 921
18.8.18 HPDMA channel x transfer register 3 (HPDMA_CxTR3) 923
18.8.19 HPDMA channel x block register 2 (HPDMA_CxBR2) 924
18.8.20 HPDMA channel x linked-list address register (HPDMA_CxLLR) 925
18.8.21 HPDMA channel x alternate linked-list address register (HPDMA_CxLLR) 927
18.8.22 HPDMA register map 929

19 General purpose direct memory access controller (GPDMA) 931
19.1 GPDMA introduction 931
19.2 GPDMA main features 931
19.3 GPDMA implementation 932
19.3.1 GPDMA channels 932
19.3.2 GPDMA in low-power modes 933
19.3.3 GPDMA requests 933
19.3.4 GPDMA block requests 938

19.3.5 GPDMA channels with peripheral early termination . . . 938
19.3.6 GPDMA triggers . . . 938
19.4 GPDMA functional description . . . 942
19.4.1 GPDMA block diagram . . . 942
19.4.2 GPDMA channel state and direct programming without any linked-list . . . 942
19.4.3 GPDMA channel suspend and resume . . . 943
19.4.4 GPDMA channel abort and restart . . . 944
19.4.5 GPDMA linked-list data structure . . . 945
19.4.6 Linked-list item transfer execution . . . 948
19.4.7 GPDMA channel state and linked-list programming in run-to-completion mode . . . 948
19.4.8 GPDMA channel state and linked-list programming in link step mode . . . 952
19.4.9 GPDMA channel state and linked-list programming . . . 959
19.4.10 GPDMA FIFO-based transfers . . . 961
19.4.11 GPDMA transfer request and arbitration . . . 968
19.4.12 GPDMA triggered transfer . . . 972
19.4.13 GPDMA circular buffering with linked-list programming . . . 973
19.4.14 GPDMA transfer in peripheral flow-control mode . . . 975
19.4.15 GPDMA secure/nonsecure channel . . . 976
19.4.16 GPDMA privileged/unprivileged channel . . . 977
19.4.17 GPDMA error management . . . 977
19.5 GPDMA in debug mode . . . 979
19.6 GPDMA in low-power modes . . . 979
19.7 GPDMA interrupts . . . 980
19.8 GPDMA registers . . . 981
19.8.1 GPDMA secure configuration register (GPDMA_SECCFGR) . . . 981
19.8.2 GPDMA privileged configuration register (GPDMA_PRIVCFGR) . . . 982
19.8.3 GPDMA configuration lock register (GPDMA_RCFGLOCKR) . . . 982
19.8.4 GPDMA nonsecure masked interrupt status register (GPDMA_MISR) . . . 983
19.8.5 GPDMA secure masked interrupt status register (GPDMA_SMISR) . . . 984
19.8.6 GPDMA channel x linked-list base address register (GPDMA_CxLBAR) . . . 984
19.8.7 GPDMA channel x flag clear register (GPDMA_CxFCR) . . . 985
19.8.8 GPDMA channel x status register (GPDMA_CxSR) . . . 986
19.8.9 GPDMA channel x control register (GPDMA_CxCR) . . . 987
19.8.10 GPDMA channel x transfer register 1 (GPDMA_CxTR1) . . . 989

19.8.11 GPDMA channel x transfer register 2 (GPDMA_CxTR2) 993
19.8.12 GPDMA channel x block register 1 (GPDMA_CxBR1) 997
19.8.13 GPDMA channel x alternate block register 1 (GPDMA_CxBR1) 998
19.8.14 GPDMA channel x source address register (GPDMA_CxSAR) 1001
19.8.15 GPDMA channel x destination address register (GPDMA_CxDAR) . 1003
19.8.16 GPDMA channel x transfer register 3 (GPDMA_CxTR3) 1004
19.8.17 GPDMA channel x block register 2 (GPDMA_CxBR2) 1005
19.8.18 GPDMA channel x linked-list address register (GPDMA_CxLLR) . . . 1006
19.8.19 GPDMA channel x alternate linked-list address register (GPDMA_CxLLR) 1008
19.8.20 GPDMA register map 1009

20 Neural-ART accelerator™ (NPU) 1012

20.1 NPU introduction 1012
20.2 NPU implementation 1013
20.3 NPU functional description 1014
20.3.1 NPU block diagram 1014
20.3.2 NPU pins and internal signals 1015
20.3.3 Configuration network 1015
20.3.4 Clock/reset manager 1016
20.3.5 Stream link 1016
20.3.6 Stream switch 1016
20.3.7 Stream engines 1016
20.3.8 Encryption/Decryption unit 1017
20.3.9 Decompression unit 1017
20.3.10 Convolutional accelerator 1019
20.3.11 Pooling unit 1020
20.3.12 Activation unit 1021
20.3.13 Arithmetic unit 1023
20.3.14 Reconfigurable buffer 1024
20.3.15 Epoch controller 1024
20.3.16 Interrupt controller 1025
20.4 Configuration examples 1025
20.4.1 DMA transfer 1025
20.4.2 Simple processing 1025
20.4.3 Multiple processing 1026
20.4.4 Conv-Pool-ReLU 1026

20.4.5 Chained convolutions . . . 1027
20.4.6 Split convolutions . . . 1027
20.5 Address space . . . 1028
20.6 System integration . . . 1029
20.6.1 System considerations . . . 1029
20.6.2 Architecture intent . . . 1029

21 Chrom-ART Accelerator controller (DMA2D) . . . 1032
21.1 DMA2D introduction . . . 1032
21.2 DMA2D main features . . . 1032
21.3 DMA2D functional description . . . 1033
21.3.1 General description . . . 1033
21.3.2 DMA2D internal signals . . . 1034
21.3.3 DMA2D control . . . 1035
21.3.4 DMA2D foreground and background FIFOs . . . 1035
21.3.5 DMA2D foreground and background PFC . . . 1035
21.3.6 DMA2D foreground and background CLUT interface . . . 1037
21.3.7 DMA2D blender . . . 1039
21.3.8 DMA2D output PFC . . . 1039
21.3.9 DMA2D output FIFO . . . 1039
21.3.10 DMA2D output FIFO byte reordering . . . 1040
21.3.11 DMA2D AXI master port timer . . . 1042
21.3.12 DMA2D transactions . . . 1042
21.3.13 DMA2D configuration . . . 1042
21.3.14 YCbCr support . . . 1046
21.3.15 DMA2D transfer control (start, suspend, abort, and completion) . . . 1046
21.3.16 Watermark . . . 1047
21.3.17 Error management . . . 1047
21.3.18 AXI dead time . . . 1047
21.4 DMA2D interrupts . . . 1047
21.5 DMA2D registers . . . 1048
21.5.1 DMA2D control register (DMA2D_CR) . . . 1048
21.5.2 DMA2D interrupt status register (DMA2D_ISR) . . . 1050
21.5.3 DMA2D interrupt flag clear register (DMA2D_IFCR) . . . 1050
21.5.4 DMA2D foreground memory address register (DMA2D_FGMAR) . . . 1051
21.5.5 DMA2D foreground offset register (DMA2D_FGOR) . . . 1051

21.5.6 DMA2D background memory address register (DMA2D_BGMAR) . 1052
21.5.7 DMA2D background offset register (DMA2D_BGOR) 1052
21.5.8 DMA2D foreground PFC control register (DMA2D_FGPFCCR) 1053
21.5.9 DMA2D foreground color register (DMA2D_FGCOLR) 1055
21.5.10 DMA2D background PFC control register (DMA2D_BGPFCCR) . . . 1055
21.5.11 DMA2D background color register (DMA2D_BGCOLR) 1057
21.5.12 DMA2D foreground CLUT memory address register (DMA2D_FGCMAR) 1057
21.5.13 DMA2D background CLUT memory address register (DMA2D_BGCMAR) 1058
21.5.14 DMA2D output PFC control register (DMA2D_OPFCCR) 1058
21.5.15 DMA2D output color register (DMA2D_OCOLR) 1059
21.5.16 DMA2D output color register [alternate] (DMA2D_OCOLR) 1060
21.5.17 DMA2D output color register [alternate] (DMA2D_OCOLR) 1060
21.5.18 DMA2D output color register [alternate] (DMA2D_OCOLR) 1061
21.5.19 DMA2D output memory address register (DMA2D_OMAR) 1061
21.5.20 DMA2D output offset register (DMA2D_OOR) 1062
21.5.21 DMA2D number of line register (DMA2D_NLR) 1062
21.5.22 DMA2D line watermark register (DMA2D_LWR) 1063
21.5.23 DMA2D AXI master timer configuration register (DMA2D_AMTCR) . 1063
21.5.24 DMA2D foreground CLUT (DMA2D_FGCLUTx) 1064
21.5.25 DMA2D background CLUT (DMA2D_BGCLUTx) 1064
21.5.26 DMA2D register map 1065

22 Chrom-GRC (GFXMMU) 1067

22.1 GFXMMU introduction 1067
22.2 GFXMMU main features 1067
22.3 GFXMMU functional and architectural description 1067
22.3.1 GFXMMU block diagram 1067
22.3.2 GFXMMU internal signals 1068
22.3.3 Virtual memory 1068
22.3.4 Packing and unpacking 1070
22.3.5 MMU architecture 1071
22.4 GTZC TZIC GFXMMU interrupts 1074
22.5 GFXMMU registers 1075
22.5.1 GFXMMU configuration register (GFXMMU_CR) 1075
22.5.2 GFXMMU status register (GFXMMU_SR) 1077

22.5.3 GFXMMU flag clear register (GFXMMU_FCR) 1077
22.5.4 GFXMMU default value register (GFXMMU_DVR) 1078
22.5.5 GFXMMU default alpha register (GFXMMU_DAR) 1078
22.5.6 GFXMMU buffer x configuration register (GFXMMU_BxCR) 1079
22.5.7 GFXMMU LUT entry x low (GFXMMU_LUTxL) 1079
22.5.8 GFXMMU LUT entry x high (GFXMMU_LUTxH) 1080
22.5.9 GFXMMU register map 1080

23 Graphic timer (GFXTIM) 1082

23.1 GFXTIM introduction 1082
23.2 GFXTIM main features 1082
23.3 GFXTIM functional description 1082
23.3.1 Block diagram 1082
23.3.2 GFXTIM pins and internal signals 1083
23.3.3 Clock generator 1084
23.3.4 Example of clock generator configuration 1086
23.3.5 Absolute timers 1090
23.3.6 Relative timers 1091
23.3.7 Tearing-effect detection 1092
23.3.8 Event generator 1092
23.3.9 Watchdog timer 1093
23.4 GFXTIM interrupts 1094
23.5 GFXTIM registers 1095
23.5.1 GFXTIM configuration register (GFXTIM_CR) 1095
23.5.2 GFXTIM clock generator configuration register (GFXTIM_CGCR) . . 1096
23.5.3 GFXTIM timers configuration register (GFXTIM_TCR) 1098
23.5.4 GFXTIM timers disable register (GFXTIM_TDR) 1099
23.5.5 GFXTIM events control register (GFXTIM_EVCR) 1100
23.5.6 GFXTIM events selection register (GFXTIM_EVSR) 1100
23.5.7 GFXTIM watchdog timer configuration register (GFXTIM_WDGTCR) 1102
23.5.8 GFXTIM interrupt status register (GFXTIM_ISR) 1104
23.5.9 GFXTIM interrupt clear register (GFXTIM_ICR) 1105
23.5.10 GFXTIM interrupt enable register (GFXTIM_IER) 1107
23.5.11 GFXTIM timers status register (GFXTIM_TSR) 1109
23.5.12 GFXTIM line-clock counter reload register (GFXTIM_LCCRR) 1110
23.5.13 GFXTIM frame-clock counter reload register (GFXTIM_FCCRR) . . . 1110

23.5.14 GFXTIM absolute time register (GFXTIM_ATR) 1110
23.5.15 GFXTIM absolute frame counter register (GFXTIM_AFCR) 1111
23.5.16 GFXTIM absolute line counter register (GFXTIM_ALCR) 1111
23.5.17 GFXTIM absolute frame counter compare 1 register (GFXTIM_AFCC1R) 1112
23.5.18 GFXTIM absolute line counter compare 1 register (GFXTIM_ALCC1R) 1112
23.5.19 GFXTIM absolute line counter compare 2 register (GFXTIM_ALCC2R) 1112
23.5.20 GFXTIM relative frame counter 1 register (GFXTIM_RFC1R) 1113
23.5.21 GFXTIM relative frame counter 1 reload register (GFXTIM_RFC1RR) 1113
23.5.22 GFXTIM relative frame counter 2 register (GFXTIM_RFC2R) 1114
23.5.23 GFXTIM relative frame counter 2 reload register (GFXTIM_RFC2RR) 1114
23.5.24 GFXTIM watchdog counter register (GFXTIM_WDGCR) 1114
23.5.25 GFXTIM watchdog reload register (GFXTIM_WDGRR) 1115
23.5.26 GFXTIM watchdog pre-alarm register (GFXTIM_WDGPAR) 1115
23.5.27 GFXTIM register map 1115

24 Nested vectored interrupt controller (NVIC) 1118

24.1 NVIC features 1118
24.1.1 SysTick calibration value register 1118
24.1.2 Interrupt and exception vectors 1118

25 Extended interrupts and event controller (EXTI) 1126

25.1 EXTI main features 1126
25.2 EXTI implementation 1127
25.3 EXTI block diagram 1127
25.3.1 EXTI connections between peripherals and CPU 1128
25.3.2 EXTI wake-up interrupt list 1129
25.4 EXTI functional description 1131
25.4.1 EXTI configurable event input wake-up 1131
25.4.2 EXTI direct event input wake-up 1132
25.4.3 EXTI multiplexer selection 1132
25.5 EXTI functional behavior 1133
25.6 EXTI event protection 1134
25.6.1 EXTI register security protection 1134

25.6.2 EXTI register privilege protection 1135
25.7 EXTI registers 1135
25.7.1 EXTI rising trigger selection register (EXTI_RTSR1) 1135
25.7.2 EXTI falling trigger selection register (EXTI_FTSR1) 1136
25.7.3 EXTI software interrupt event register (EXTI_SWIER1) 1137
25.7.4 EXTI rising edge pending register (EXTI_RPR1) 1138
25.7.5 EXTI falling edge pending register (EXTI_FPR1) 1138
25.7.6 EXTI security configuration register (EXTI_SECCFGR1) 1139
25.7.7 EXTI privilege configuration register (EXTI_PRIVCFGR1) 1139
25.7.8 EXTI rising trigger selection register (EXTI_RTSR2) 1140
25.7.9 EXTI falling trigger selection register (EXTI_FTSR2) 1141
25.7.10 EXTI software interrupt event register (EXTI_SWIER2) 1142
25.7.11 EXTI rising edge pending register (EXTI_RPR2) 1143
25.7.12 EXTI falling edge pending register (EXTI_FPR2) 1144
25.7.13 EXTI security enable register (EXTI_SECCFGR2) 1145
25.7.14 EXTI privilege enable register (EXTI_PRIVCFGR2) 1146
25.7.15 EXTI rising trigger selection register (EXTI_RTSR3) 1146
25.7.16 EXTI falling trigger selection register (EXTI_FTSR3) 1147
25.7.17 EXTI software interrupt event register (EXTI_SWIER3) 1148
25.7.18 EXTI rising edge pending register (EXTI_RPR3) 1149
25.7.19 EXTI falling edge pending register (EXTI_FPR3) 1149
25.7.20 EXTI security enable register (EXTI_SECCFGR3) 1150
25.7.21 EXTI privilege enable register (EXTI_PRIVCFGR3) 1151
25.7.22 EXTI external interrupt selection register 1 (EXTI_EXTICR1) 1151
25.7.23 EXTI external interrupt selection register 2 (EXTI_EXTICR2) 1154
25.7.24 EXTI external interrupt selection register 3 (EXTI_EXTICR3) 1157
25.7.25 EXTI external interrupt selection register 4 (EXTI_EXTICR4) 1159
25.7.26 EXTI lock register (EXTI_LOCKR) 1162
25.7.27 EXTI CPU wake-up with interrupt mask register 1 (EXTI_IMR1) . . . 1162
25.7.28 EXTI CPU wake-up with event mask register 1 (EXTI_EMR1) 1163
25.7.29 EXTI CPU wake-up with interrupt mask register 2 (EXTI_IMR2) . . . 1163
25.7.30 EXTI CPU wake-up with event mask register 2 (EXTI_EMR2) 1164
25.7.31 EXTI CPU wake-up with interrupt mask register 3 (EXTI_IMR3) . . . 1164
25.7.32 EXTI CPU wake-up with event mask register 3 (EXTI_EMR3) 1165
25.7.33 EXTI register map 1167

26 Cyclic redundancy check calculation unit (CRC) 1170

26.1 CRC introduction 1170
26.2 CRC main features 1170
26.3 CRC functional description 1171
26.3.1 CRC block diagram 1171
26.3.2 CRC internal signals 1171
26.3.3 CRC operation 1171
26.4 CRC registers 1173
26.4.1 CRC data register (CRC_DR) 1173
26.4.2 CRC independent data register (CRC_IDR) 1173
26.4.3 CRC control register (CRC_CR) 1174
26.4.4 CRC initial value (CRC_INIT) 1175
26.4.5 CRC polynomial (CRC_POL) 1175
26.4.6 CRC register map 1176

27 Flexible memory controller (FMC) 1177
27.1 FMC main features 1177
27.2 FMC implementation 1178
27.3 FMC block diagram 1179
27.4 FMC internal signals 1180
27.5 AHB interface 1180
27.6 AXI interface 1180
27.6.1 Supported memories and transactions 1181
27.7 External device address mapping 1182
27.7.1 NOR/PSRAM address mapping 1183
27.7.2 NAND flash memory address mapping 1183
27.7.3 SDRAM address mapping 1184
27.8 NOR flash/PSRAM controller 1187
27.8.1 External memory interface signals 1189
27.8.2 Supported memories and transactions 1191
27.8.3 General timing rules 1193
27.8.4 NOR flash/PSRAM controller asynchronous transactions 1193
27.8.5 Synchronous transactions 1212
27.8.6 NOR/PSRAM controller registers 1217
27.9 NAND flash controller 1226
27.9.1 External memory interface signals 1227
27.9.2 NAND flash supported memories and transactions 1228

27.9.3 Timing diagrams for NAND flash memory 1229
27.9.4 NAND flash operations 1230
27.9.5 NAND flash prewait function 1231
27.9.6 NAND ECC controller 1232
27.9.7 FMC command sequencer 1235
27.9.8 NAND flash controller interrupt 1242
27.9.9 NAND flash controller registers 1243
27.10 SDRAM controller 1267
27.10.1 SDRAM controller main features 1267
27.10.2 SDRAM External memory interface signals 1267
27.10.3 SDRAM controller functional description 1268
27.10.4 Low-power modes 1274
27.10.5 SDRAM controller registers 1276
27.11 FMC register map 1284

28 Extended-SPI interface (XSPI) 1289
28.1 XSPI introduction 1289
28.2 XSPI main features 1289
28.3 XSPI implementation 1290
28.4 XSPI functional description 1291
28.4.1 XSPI block diagram 1291
28.4.2 XSPI pins and internal signals 1295
28.4.3 Clock constraints 1296
28.4.4 XSPI interface to memory modes 1296
28.4.5 XSPI regular-command protocol 1296
28.4.6 XSPI regular-command protocol signal interface 1300
28.4.7 HyperBus protocol 1305
28.4.8 Specific features 1310
28.4.9 XSPI operating modes introduction 1312
28.4.10 XSPI indirect mode 1312
28.4.11 XSPI automatic status-polling mode 1314
28.4.12 XSPI memory-mapped mode 1314
28.4.13 XSPI configuration introduction 1315
28.4.14 XSPI system configuration 1315
28.4.15 XSPI device configuration 1316
28.4.16 XSPI regular-command mode configuration 1318

28.4.17 XSPI HyperBus protocol configuration 1321
28.4.18 XSPI error management 1321
28.4.19 XSPI high-speed interface and calibration 1322
28.4.20 XSPI BUSY and ABORT 1323
28.4.21 XSPI reconfiguration or deactivation 1323
28.4.22 NCS behavior 1324
28.4.23 Software control of two external memories 1324
28.4.24 Hardware-controlled extended memory support 1325
28.5 Address alignment and data number 1326
28.6 XSPI interrupts 1328
28.7 XSPI registers 1328
28.7.1 XSPI control register (XSPI_CR) 1328
28.7.2 XSPI device configuration register 1 (XSPI_DCR1) 1332
28.7.3 XSPI device configuration register 2 (XSPI_DCR2) 1333
28.7.4 XSPI device configuration register 3 (XSPI_DCR3) 1334
28.7.5 XSPI device configuration register 4 (XSPI_DCR4) 1335
28.7.6 XSPI status register (XSPI_SR) 1335
28.7.7 XSPI flag clear register (XSPI_FCR) 1336
28.7.8 XSPI data length register (XSPI_DLR) 1337
28.7.9 XSPI address register (XSPI_AR) 1337
28.7.10 XSPI data register (XSPI_DR) 1338
28.7.11 XSPI polling status mask register (XSPI_PSMKR) 1339
28.7.12 XSPI polling status match register (XSPI_PSMAR) 1339
28.7.13 XSPI polling interval register (XSPI_PIR) 1340
28.7.14 XSPI communication configuration register (XSPI_CCR) 1340
28.7.15 XSPI timing configuration register (XSPI_TCR) 1342
28.7.16 XSPI instruction register (XSPI_IR) 1343
28.7.17 XSPI alternate bytes register (XSPI_ABR) 1343
28.7.18 XSPI low-power timeout register (XSPI_LPTR) 1343
28.7.19 XSPI wrap communication configuration register (XSPI_WPCCR) 1344
28.7.20 XSPI wrap timing configuration register (XSPI_WPTCR) 1346
28.7.21 XSPI wrap instruction register (XSPI_WPIR) 1347
28.7.22 XSPI wrap alternate byte register (XSPI_WPABR) 1347
28.7.23 XSPI write communication configuration register (XSPI_WCCR) 1347
28.7.24 XSPI write timing configuration register (XSPI_WTCR) 1349

28.7.25 XSPI write instruction register (XSPI_WIR) 1350
28.7.26 XSPI write alternate byte register (XSPI_WABR) 1350
28.7.27 XSPI HyperBus latency configuration register (XSPI_HLCR) 1351
28.7.28 XSPI full-cycle calibration configuration (XSPI_CALFCR) 1351
28.7.29 XSPI DLL master calibration configuration (XSPI_CALMR) 1352
28.7.30 XSPI DLL slave output calibration configuration (XSPI_CALSOR) 1353
28.7.31 XSPI DLL slave input calibration configuration (XSPI_CALSIR) 1354
28.7.32 XSPI register map 1354

29 XSPI I/O manager (XSPIM) 1358

29.1 XSPIM introduction 1358
29.2 XSPIM main features 1358
29.3 XSPIM implementation 1358
29.4 XSPIM functional description 1358
29.4.1 XSPIM block diagram 1358
29.4.2 XSPIM input/output pins 1359
29.4.3 XSPIM matrix 1360
29.4.4 XSPIM multiplexed mode 1360
29.5 Use cases description 1361
29.5.1 XSPIs direct octal mode 1361
29.5.2 XSPI direct 16-bit mode 1362
29.5.3 XSPI dual-octal mode 1363
29.5.4 XSPI swapped mode 1364
29.5.5 Two XSPIs multiplexed mode to Port 1 accessing two external memories, and the third XSPI accessing Port 2 1365
29.5.6 Two XSPIs multiplexed mode to Port 2 accessing two external memories, and the third XSPI accessing Port 1 1366
29.5.7 XSPI1 and XSPI2 drive a single external memory 1367
29.5.8 A single XSPI drives two external memories 1368
29.6 XSPIM registers 1370
29.6.1 XSPIM control register (XSPIM_CR) 1370
29.6.2 XSPIM register map 1371

30 Secure digital input/output MultiMediaCard interface (SDMMC) . . 1372

30.1 SDMMC main features 1372
30.2 SDMMC implementation 1372

30.3 SDMMC bus topology 1373
30.4 SDMMC operation modes 1375
30.5 SDMMC functional description 1376
30.5.1 SDMMC block diagram 1376
30.5.2 SDMMC pins and internal signals 1376
30.5.3 General description 1377
30.5.4 SDMMC adapter 1379
30.5.5 SDMMC AHB slave interface 1401
30.5.6 SDMMC AHB master interface 1402
30.5.7 AHB and SDMMC_CK clock relation 1405
30.6 Card functional description 1406
30.6.1 SD I/O mode 1406
30.6.2 CMD12 send timing 1414
30.6.3 Sleep (CMD5) 1418
30.6.4 Interrupt mode (Wait-IRQ) 1419
30.6.5 Boot operation 1420
30.6.6 Response R1b handling 1423
30.6.7 Reset and card cycle power 1424
30.7 Hardware flow control 1425
30.7.1 Hardware flow control during data transfer 1425
30.7.2 Block gap hardware flow control 1426
30.8 Ultra-high-speed phase I (UHS-I) voltage switch 1427
30.9 SDMMC interrupts 1431
30.10 SDMMC registers 1432
30.10.1 SDMMC power control register (SDMMC_POWER) 1432
30.10.2 SDMMC clock control register (SDMMC_CLKCR) 1433
30.10.3 SDMMC argument register (SDMMC_ARGR) 1435
30.10.4 SDMMC command register (SDMMC_CMDR) 1435
30.10.5 SDMMC command response register (SDMMC_RESPCMDR) 1437
30.10.6 SDMMC response x register (SDMMC_RESPxR) 1438
30.10.7 SDMMC data timer register (SDMMC_DTIMER) 1438
30.10.8 SDMMC data length register (SDMMC_DLENR) 1439
30.10.9 SDMMC data control register (SDMMC_DCTRL) 1440
30.10.10 SDMMC data counter register (SDMMC_DCNTR) 1441
30.10.11 SDMMC status register (SDMMC_STAR) 1442
30.10.12 SDMMC interrupt clear register (SDMMC_ICR) 1445

30.10.13 SDMMC mask register (SDMMC_MASKR) 1447
30.10.14 SDMMC acknowledgment timer register (SDMMC_ACKTIMER) . . . 1450
30.10.15 SDMMC data FIFO threshold register (SDMMC_FIFOTHRR) 1450
30.10.16 SDMMC DMA control register (SDMMC_IDMACTRLR) 1451
30.10.17 SDMMC IDMA buffer size register (SDMMC_IDMABSIZER) 1451
30.10.18 SDMMC IDMA buffer base address register (SDMMC_IDMABASER) 1452
30.10.19 SDMMC IDMA linked list address register (SDMMC_IDMALAR) . . . 1452
30.10.20 SDMMC IDMA linked list memory base register (SDMMC_IDMABAR) 1453
30.10.21 SDMMC data FIFO registers x (SDMMC_FIFORx) 1454
30.10.22 SDMMC register map 1454

31 Delay block (DLYB) 1457

31.1 DLYB introduction 1457
31.2 DLYB main features 1457
31.3 DLYB implementation 1457
31.4 DLYB functional description 1457
31.4.1 DLYB diagram 1457
31.4.2 DLYB internal signals 1459
31.4.3 General description 1461
31.4.4 Lock mode procedure 1462
31.4.5 Bypass mode procedure 1462
31.5 DLYB SDMMC registers description 1463
31.5.1 Delay block SDMMC DLL configuration (DLYBSD_CFG) 1463
31.5.2 Delay block SDMMC DLL status (DLYBSD_STATUS) 1464
31.5.3 DLYB register map 1464

32 Analog-to-digital converters (ADC) 1466

32.1 ADC introduction 1466
32.2 ADC main features 1466
32.3 ADC implementation 1468
32.4 ADC functional description 1469
32.4.1 ADC block diagram 1469
32.4.2 ADC pins and internal signals 1470
32.4.3 ADC clocks 1473
32.4.4 ADC connectivity 1475

32.4.5 Slave AHB interface . . . 1477
32.4.6 ADC Deep-power-down mode (DEEPPWD) . . . 1477
32.4.7 Single-ended and differential input channels . . . 1477
32.4.8 Calibration (ADCAL, CALADDOS, ADC_CALFACT) . . . 1478
32.4.9 ADC on-off control (ADEN, ADDIS, ADRDY) . . . 1480
32.4.10 Constraints when writing the ADC control bits . . . 1481
32.4.11 Channel selection (ADC_SQRy, ADC_JSQR) . . . 1482
32.4.12 Channel preselection register (ADC_PCSEL) . . . 1482
32.4.13 Channel-wise programmable sampling time (SMPR1, SMPR2) . . . 1483
32.4.14 Single conversion mode . . . 1484
32.4.15 Continuous conversion mode (CONT = 1) . . . 1485
32.4.16 Discontinuous mode (DISCEN, DISCNUM, JDISCEN) . . . 1486
32.4.17 Starting conversions (ADSTART, JADSTART) . . . 1487
32.4.18 Timing . . . 1488
32.4.19 Stopping an ongoing conversion (ADSTP, JADSTP) . . . 1489
32.4.20 Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN) . . . 1491
32.4.21 Injected channel management . . . 1492
32.4.22 Programmable resolution (RES) - fast conversion mode . . . 1493
32.4.23 End of conversion and end of sampling phase (EOC, JEOC, EOSMP) . . . 1494
32.4.24 End of conversion sequence (EOS, JEOS) . . . 1494
32.4.25 Timing diagram examples (single/continuous modes, hardware/software triggers) . . . 1495
32.4.26 Data management . . . 1497
32.4.27 Managing conversions using the MDF . . . 1505
32.4.28 Dynamic low-power features . . . 1506
32.4.29 Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWDCH of ADC_AWD2CR and ADC_AWD3CR, HTR, LTR, AWDFILT) . . . 1509
32.4.30 Oversampler . . . 1514
32.4.31 Dual ADC modes . . . 1520
32.4.32 VBAT supply monitoring . . . 1536
32.4.33 Monitoring the internal voltage reference . . . 1537
32.4.34 Monitoring the supply voltage . . . 1539
32.5 ADC interrupts . . . 1539
32.6 ADC registers (for each ADC) . . . 1540
32.6.1 ADC interrupt and status register (ADC_ISR) . . . 1540

32.6.2 ADC interrupt enable register (ADC_IER) 1542
32.6.3 ADC control register (ADC_CR) 1544
32.6.4 ADC configuration register (ADC_CFGR1) 1547
32.6.5 ADC configuration register 2 (ADC_CFGR2) 1550
32.6.6 ADC sample time register 1 (ADC_SMPR1) 1553
32.6.7 ADC sample time register 2 (ADC_SMPR2) 1554
32.6.8 ADC channel preselection register (ADC_PCSEL) 1555
32.6.9 ADC regular sequence register 1 (ADC_SQR1) 1555
32.6.10 ADC regular sequence register 2 (ADC_SQR2) 1556
32.6.11 ADC regular sequence register 3 (ADC_SQR3) 1557
32.6.12 ADC regular sequence register 4 (ADC_SQR4) 1558
32.6.13 ADC regular data register (ADC_DR) 1559
32.6.14 ADC injected sequence register (ADC_JSQR) 1559
32.6.15 ADC offset y configuration register (ADC_OFCFGRy) 1561
32.6.16 ADC offset y register (ADC_OFRy) 1562
32.6.17 ADC gain compensation register (ADC_GCOMP) 1563
32.6.18 ADC injected channel y data register (ADC_JDRy) 1564
32.6.19 ADC analog watchdog 2 configuration register (ADC_AWD2CR) . . . 1564
32.6.20 ADC analog watchdog 3 configuration register (ADC_AWD3CR) . . . 1565
32.6.21 ADC analog watchdog 1 lower threshold register (ADC_AWD1LTR) 1565
32.6.22 ADC analog watchdog 1 higher threshold register (ADC_AWD1HTR) 1566
32.6.23 ADC analog watchdog 2 lower threshold register (ADC_AWD2LTR) 1566
32.6.24 ADC analog watchdog 2 higher threshold register (ADC_AWD2HTR) 1567
32.6.25 ADC analog watchdog 3 lower threshold register (ADC_AWD3LTR) 1567
32.6.26 ADC analog watchdog 3 higher threshold register (ADC_AWD3HTR) 1567
32.6.27 ADC differential mode selection register (ADC_DIFSEL) 1568
32.6.28 ADC calibration factors (ADC_CALFACT) 1568
32.6.29 ADC option register (ADC_OR) 1570
32.7 ADC common registers 1571
32.7.1 ADC common status register (ADCC_CSR) 1571
32.7.2 ADC common control register (ADCC_CCR) 1572
32.7.3 ADC common regular data register for dual mode (ADCC_CDR) 1574
32.7.4 ADC common regular data register for dual mode (ADCC_CDR2) 1574

32.8 ADC register map 1575

33 Digital temperature sensor (DTS) 1579

33.1 Introduction 1579
33.2 DTS main features 1579
33.3 DTS functional description 1580
33.3.1 DTS block diagram 1580
33.3.2 DTS pins and internal signals 1580
33.3.3 DTS reset and clocks 1581
33.3.4 DTS serial data adapter (SDA) 1583
33.3.5 Alarms 1589
33.3.6 Main APB programming routines 1590
33.3.7 Temperature sensor configuration 1593
33.4 DTS interrupts 1595
33.5 DTS registers 1596
33.5.1 DTS PVT common registers 1596
33.5.2 PVT IRQ registers 1598
33.5.3 TS common registers 1602
33.5.4 TS individual registers 1609
33.5.5 DTS register map 1616

34 Voltage reference buffer (VREFBUF) 1621

34.1 VREFBUF introduction 1621
34.2 VREFBUF implementation 1621
34.3 VREFBUF functional description 1621
34.4 VREFBUF trimming 1622
34.5 VREFBUF power sequence 1623
34.6 VREFBUF registers 1624
34.6.1 VREFBUF control and status register (VREFBUF_CSR) 1624
34.6.2 VREFBUF calibration control register (VREFBUF_CCR) 1625
34.6.3 VREFBUF register map 1625

35 Multi-function digital filter (MDF) 1626

35.1 MDF introduction 1626
35.2 MDF main features 1627
35.3 MDF implementation 1627

35.4 MDF functional description 1629
35.4.1 MDF block diagram 1629
35.4.2 MDF pins and internal signals 1630
35.4.3 Serial input interfaces (SITF) 1631
35.4.4 ADC slave interface (ADCITF) 1636
35.4.5 Clock generator (CKGEN) 1637
35.4.6 Bitstream matrix (BSMX) 1639
35.4.7 Short-circuit detectors (SCD) 1640
35.4.8 Digital filter processing (DFLT) 1642
35.4.9 Out-of-limit detector (OLD) 1652
35.4.10 Digital filter acquisition modes 1655
35.4.11 Start-up sequence examples 1665
35.4.12 Break interface 1666
35.4.13 Data transfer to memory 1667
35.4.14 Autonomous mode 1672
35.4.15 Register protection 1673
35.5 MDF low-power modes 1674
35.6 MDF interrupts 1674
35.7 MDF application informations 1676
35.7.1 MDF configuration examples for audio capture 1676
35.7.2 Programming examples 1677
35.7.3 Connection examples 1679
35.7.4 Global frequency response 1680
35.7.5 Total MDF gain 1681
35.8 MDF registers 1685
35.8.1 MDF global control register (MDF_GCR) 1685
35.8.2 MDF clock generator control register (MDF_CKGCR) 1686
35.8.3 MDF serial interface control register x (MDF_SITFxCR) 1688
35.8.4 MDF bitstream matrix control register x (MDF_BSMXxCR) 1689
35.8.5 MDF digital filter control register x (MDF_DFLTxCR) 1690
35.8.6 MDF digital filter configuration register x (MDF_DFLTxCICR) 1692
35.8.7 MDF reshape filter configuration register x (MDF_DFLTxRSFR) . . . 1694
35.8.8 MDF integrator configuration register x (MDF_DFLTxINTR) 1695
35.8.9 MDF out-of limit detector control register x (MDF_OLDxCR) 1695
35.8.10 MDF OLDx low threshold register x (MDF_OLDxTHLR) 1697
35.8.11 MDF OLDx high threshold register x (MDF_OLDxTHHR) 1697

35.8.12 MDF delay control register x (MDF_DLYxCR) 1698
35.8.13 MDF short circuit detector control register x (MDF_SCDxCR) 1698
35.8.14 MDF DFLT0 interrupt enable register 0 (MDF_DFLT0IER) 1699
35.8.15 MDF DFLTx interrupt enable register x (MDF_DFLTxIER) 1701
35.8.16 MDF DFLT0 interrupt status register 0 (MDF_DFLT0ISR) 1702
35.8.17 MDF DFLTx interrupt status register x (MDF_DFLTxISR) 1704
35.8.18 MDF offset error compensation control register x (MDF_OECxCR) . 1705
35.8.19 MDF snapshot data register x (MDF_SNPSxDR) 1706
35.8.20 MDF digital filter data register x (MDF_DFLTxDR) 1706
35.8.21 MDF register map 1707

36 Audio digital filter (ADF) 1709

36.1 ADF introduction 1709
36.2 ADF main features 1709
36.3 ADF implementation 1710
36.4 ADF functional description 1711
36.4.1 ADF block diagram 1711
36.4.2 ADF pins and internal signals 1711
36.4.3 Serial input interface (SITF) 1712
36.4.4 ADC slave interface (ADCITF) 1717
36.4.5 Clock generator (CKGEN) 1717
36.4.6 Bitstream matrix (BSMX) 1719
36.4.7 Digital filter processing (DFLT) 1720
36.4.8 Digital filter acquisition modes 1730
36.4.9 Start-up sequence examples 1738
36.4.10 Sound activity detection (SAD) 1739
36.4.11 Data transfer to memory 1747
36.4.12 Autonomous mode 1750
36.4.13 Register protection 1750
36.5 ADF low-power modes 1751
36.6 ADF interrupts 1751
36.7 ADF application information 1753
36.7.1 ADF configuration examples for audio capture 1753
36.7.2 Programming examples 1754
36.7.3 Connection examples 1756
36.7.4 Global frequency response 1757

36.7.5 Total ADF gain . . . 1758
36.7.6 How to compute SAD thresholds . . . 1761
36.8 ADF registers . . . 1765
36.8.1 ADF global control register (ADF_GCR) . . . 1765
36.8.2 ADF clock generator control register (ADF_CKGCR) . . . 1765
36.8.3 ADF serial interface control register 0 (ADF_SITF0CR) . . . 1767
36.8.4 ADF bitstream matrix control register 0 (ADF_BSMX0CR) . . . 1769
36.8.5 ADF digital filter control register 0 (ADF_DFLT0CR) . . . 1769
36.8.6 ADF digital filer configuration register 0 (ADF_DFLT0CICR) . . . 1771
36.8.7 ADF reshape filter configuration register 0 (ADF_DFLT0RSFR) . . . 1772
36.8.8 ADF delay control register 0 (ADF_DLY0CR) . . . 1773
36.8.9 ADF DFLT0 interrupt enable register (ADF_DFLT0IER) . . . 1774
36.8.10 ADF DFLT0 interrupt status register 0 (ADF_DFLT0ISR) . . . 1775
36.8.11 ADF SAD control register (ADF_SADCR) . . . 1776
36.8.12 ADF SAD configuration register (ADF_SADCFGR) . . . 1778
36.8.13 ADF SAD sound level register (ADF_SADSDLVR) . . . 1779
36.8.14 ADF SAD ambient noise level register (ADF_SADANLVR) . . . 1780
36.8.15 ADF digital filter data register 0 (ADF_DFLT0DR) . . . 1780
36.8.16 ADF register map . . . 1780

37 Camera subsystem . . . 1783
37.1 Introduction . . . 1783
37.2 Camera path main features . . . 1783
37.2.1 DCMI path main features . . . 1783
37.2.2 DCMIPP path main features . . . 1784
37.3 Camera subsystem implementation . . . 1785
37.3.1 Hardware settings . . . 1785
37.4 Camera subsystem functional description . . . 1786
37.4.1 Camera subsystem block diagram . . . 1786
37.4.2 Camera subsystem pins and external signal interface . . . 1786
37.4.3 Camera subsystem reset and clocks . . . 1787
37.4.4 Streaming from the camera subsystem to slave peripherals . . . 1787
37.4.5 Camera subsystem security . . . 1788
37.5 Camera subsystem programmable parameters . . . 1789
37.5.1 Low-resolution parallel camera and DCMI . . . 1790
37.5.2 High-resolution parallel camera and DCMIPP . . . 1790

37.5.3 High-resolution CSI2 camera and DCMIPP 1791
37.5.4 Sensors over target resolution 1792
37.5.5 Sensors over target pixel rate 1792
37.5.6 Double sensors 1792
37.6 Camera subsystem interrupts 1793
37.7 Camera subsystem registers 1793

38 Digital camera interface (DCMI) 1794
38.1 DCMI introduction 1794
38.2 DCMI main features 1794
38.3 DCMI functional description 1794
38.3.1 DCMI block diagram 1795
38.3.2 DCMI pins and internal signals 1795
38.3.3 DCMI clocks 1796
38.3.4 DCMI DMA interface 1796
38.3.5 DCMI physical interface 1796
38.3.6 DCMI synchronization 1798
38.3.7 DCMI capture modes 1800
38.3.8 DCMI crop feature 1801
38.3.9 DCMI JPEG format 1802
38.3.10 DCMI FIFO 1802
38.3.11 DCMI data format description 1803
38.4 DCMI interrupts 1805
38.5 DCMI registers 1805
38.5.1 DCMI control register (DCMI_CR) 1805
38.5.2 DCMI status register (DCMI_SR) 1808
38.5.3 DCMI raw interrupt status register (DCMI_RIS) 1808
38.5.4 DCMI interrupt enable register (DCMI_IER) 1809
38.5.5 DCMI masked interrupt status register (DCMI_MIS) 1810
38.5.6 DCMI interrupt clear register (DCMI_ICR) 1811
38.5.7 DCMI embedded synchronization code register (DCMI_ESCR) 1812
38.5.8 DCMI embedded synchronization unmask register (DCMI_ESUR) . 1812
38.5.9 DCMI crop window start (DCMI_CWSTRT) 1813
38.5.10 DCMI crop window size (DCMI_CWSIZE) 1814
38.5.11 DCMI data register (DCMI_DR) 1814
38.5.12 DCMI register map 1815

39 Digital camera interface pixel pipeline (DCMIPP) 1816

39.1 DCMIPP introduction 1816

39.2 DCMIPP main features 1819

39.3 DCMIPP functional description 1821

39.3.1 DCMIPP block diagram 1821

39.3.2 DCMIPP pads and internal signals 1822

39.3.3 DCMIPP reset and clocks 1822

39.3.4 DCMIPP maximum resolution 1825

39.3.5 DCMIPP minimum requirements for frame structure 1826

39.3.6 Description of DCMIPP pixel format support 1826

39.4 DCMIPP input and flow control 1827

39.4.1 DCMIPP common configuration 1828

39.4.2 Parallel input interface 1828

39.4.3 Interface from CSI-2 host 1832

39.4.4 Input selection 1832

39.4.5 Flow selection 1832

39.4.6 Frame counter 1834

39.4.7 Frame control 1835

39.4.8 Pipe deactivation 1840

39.5 Pipe0 (dump pipe) 1841

39.5.1 Overview 1841

39.5.2 Decimation 1842

39.5.3 Crop/statistics selection/suppression 1842

39.5.4 Header insertion 1843

39.5.5 Dump counter 1844

39.5.6 Double buffer mode 1845

39.6 Pipe1 (ISP part) 1846

39.6.1 Overview 1846

39.6.2 Byte-to-pixel conversion 1846

39.6.3 Statistics removal 1847

39.6.4 Bad pixel removal 1848

39.6.5 Input decimation 1849

39.6.6 Black level calibration 1850

39.6.7 Exposure compensation and white-balance calibration 1851

39.6.8 Demosaicing 1852

39.6.9 Color conversion 1854

39.6.10 Contrast enhancement . . . 1857
39.6.11 Statistics extraction . . . 1858
39.7 Pipe1 (post-processing part) . . . 1863
39.7.1 Overview . . . 1863
39.7.2 Pixel 2D cropping . . . 1863
39.7.3 Decimation pre-downsize . . . 1864
39.7.4 Downsize . . . 1865
39.7.5 Regions of interest (ROIs) . . . 1867
39.7.6 Gamma conversion . . . 1868
39.7.7 YUV conversion . . . 1868
39.7.8 Chroma down-sampling . . . 1868
39.7.9 Pixel packing . . . 1869
39.7.10 Overrun detection . . . 1871
39.8 Pipe2 (post-processing) . . . 1872
39.8.1 Overview . . . 1872
39.8.2 Pipe1 and Pipe2 sharing image processing functions . . . 1873
39.9 Application use cases . . . 1873
39.9.1 Parallel interface camera sensor module . . . 1873
39.9.2 CSI2 camera sensor module . . . 1874
39.9.3 Force data type format from CSI-2 data flow (pixel pipes only) . . . 1881
39.10 Pixel format description . . . 1884
39.10.1 Parallel interface formats . . . 1884
39.10.2 Pixel pipe formats . . . 1886
39.10.3 Dump pipe formats . . . 1887
39.10.4 AXI IP-Plug . . . 1889
39.11 Shadow registers . . . 1895
39.12 DCMIPP low power modes . . . 1897
39.13 DCMIPP interrupts . . . 1898
39.13.1 Free-running DCMIPP . . . 1898
39.13.2 Interrupts . . . 1898
39.13.3 Event pins . . . 1901
39.14 DCMIPP registers . . . 1902
39.14.1 DCMIPP IP-Plug global register 1 (DCMIPP_IPGR1) . . . 1902
39.14.2 DCMIPP IP-Plug global register 2 (DCMIPP_IPGR2) . . . 1903
39.14.3 DCMIPP IP-Plug global register 3 (DCMIPP_IPGR3) . . . 1903
39.14.4 DCMIPP IP-Plug identification register (DCMIPP_IPGR8) . . . 1904

39.14.5 DCMIPP IP-Plug Clientx register 1 (DCMIPP_IPCxR1) 1904
39.14.6 DCMIPP IP-Plug Clientx register 2 (DCMIPP_IPCxR2) 1905
39.14.7 DCMIPP IP-Plug Clientx register 3 (DCMIPP_IPCxR3) 1905
39.14.8 DCMIPP parallel interface control register (DCMIPP_PRCR) 1906
39.14.9 DCMIPP parallel interface embedded synchronization code register (DCMIPP_PRESCR) 1907
39.14.10 DCMIPP parallel interface embedded synchronization unmask register (DCMIPP_PRESUR) 1908
39.14.11 DCMIPP parallel interface interrupt enable register (DCMIPP_PRIER) 1909
39.14.12 DCMIPP parallel interface status register (DCMIPP_PRSR) 1909
39.14.13 DCMIPP parallel interface interrupt clear register (DCMIPP_PRFCR) 1910
39.14.14 DCMIPP common configuration register (DCMIPP_CMCR) 1911
39.14.15 DCMIPP common frame counter register (DCMIPP_CMFRCR) 1912
39.14.16 DCMIPP common interrupt enable register (DCMIPP_CMIER) 1912
39.14.17 DCMIPP common status register 1 (DCMIPP_CMSR1) 1913
39.14.18 DCMIPP common status register 2 (DCMIPP_CMSR2) 1915
39.14.19 DCMIPP common interrupt clear register (DCMIPP_CMFCR) 1917
39.14.20 DCMIPP Pipe0 flow selection configuration register (DCMIPP_P0FSCR) 1919
39.14.21 DCMIPP Pipe0 flow control configuration register (DCMIPP_P0FCTCR) 1919
39.14.22 DCMIPP Pipe0 statistic/crop start register (DCMIPP_P0SCSTR) . . . 1920
39.14.23 DCMIPP Pipe0 statistic/crop size register (DCMIPP_P0SCSZR) . . . 1921
39.14.24 DCMIPP Pipe0 dump counter register (DCMIPP_P0DCCNTR) 1921
39.14.25 DCMIPP Pipe0 dump limit register (DCMIPP_P0DCLMTR) 1922
39.14.26 DCMIPP Pipe0 pixel packer configuration register (DCMIPP_P0PPCR) 1922
39.14.27 DCMIPP Pipe0 pixel packer Memory0 address register 1 (DCMIPP_P0PPM0AR1) 1924
39.14.28 DCMIPP Pipe0 pixel packer Memory0 address register 2 (DCMIPP_P0PPM0AR2) 1924
39.14.29 DCMIPP Pipe0 status Memory0 address register (DCMIPP_P0STM0AR) 1924
39.14.30 DCMIPP Pipe0 interrupt enable register (DCMIPP_P0IER) 1925
39.14.31 DCMIPP Pipe0 status register (DCMIPP_P0SR) 1926
39.14.32 DCMIPP Pipe0 interrupt clear register (DCMIPP_P0FCR) 1927
39.14.33 DCMIPP Pipe0 current flow selection configuration register (DCMIPP_P0CFSCR) 1927

39.14.34 DCMIPP Pipe0 current flow control configuration register (DCMIPP_P0CFCTCR) 1928
39.14.35 DCMIPP Pipe0 current statistic/crop start register (DCMIPP_P0CSCSTR) 1929
39.14.36 DCMIPP Pipe0 current statistic/crop size register (DCMIPP_P0CSCSZR) 1929
39.14.37 DCMIPP Pipe0 current pixel packer configuration register (DCMIPP_P0CPPCR) 1930
39.14.38 DCMIPP Pipe0 current pixel packer Memory0 address register 1 (DCMIPP_P0CPPM0AR1) 1932
39.14.39 DCMIPP Pipe0 current pixel packer Memory0 address register 2 (DCMIPP_P0CPPM0AR2) 1932
39.14.40 DCMIPP Pipe1 flow selection configuration register (DCMIPP_P1FSCR) 1933
39.14.41 DCMIPP Pipe1 stat removal configuration register (DCMIPP_P1SRCR) 1934
39.14.42 DCMIPP Pipe1 bad pixel removal control register (DCMIPP_P1BPRCR) 1935
39.14.43 DCMIPP Pipe1 bad pixel removal status register (DCMIPP_P1BPRSR) 1935
39.14.44 DCMIPP Pipe1 decimation register (DCMIPP_P1DECR) 1936
39.14.45 DCMIPP Pipe1 black level calibration control register (DCMIPP_P1BLCCR) 1936
39.14.46 DCMIPP Pipe1 exposure control register 1 (DCMIPP_P1EXCR1) . . 1937
39.14.47 DCMIPP Pipe1 exposure control register 2 (DCMIPP_P1EXCR2) . . 1937
39.14.48 DCMIPP Pipe1 statistics1 control register (DCMIPP_P1ST1CR) . . . 1938
39.14.49 DCMIPP Pipe1 statistics 2 control register (DCMIPP_P1ST2CR) . . . 1939
39.14.50 DCMIPP Pipe1 statistics 3 control register (DCMIPP_P1ST3CR) . . . 1940
39.14.51 DCMIPP Pipe1 statistics window start register (DCMIPP_P1STSTR) 1941
39.14.52 DCMIPP Pipe1 statistics window size register (DCMIPP_P1STSZR) 1941
39.14.53 DCMIPP Pipe1 statistics 1 status register (DCMIPP_P1ST1SR) . . . 1942
39.14.54 DCMIPP Pipe1 statistics 2 status register (DCMIPP_P1ST2SR) . . . 1943
39.14.55 DCMIPP Pipe1 statistics 3 status register (DCMIPP_P1ST3SR) . . . 1943
39.14.56 DCMIPP Pipe1 demosaicing configuration register (DCMIPP_P1DMCR) 1943
39.14.57 DCMIPP Pipe1 ColorConv configuration register (DCMIPP_P1CCCR) 1944
39.14.58 DCMIPP Pipe1 ColorConv red coefficient register 1 (DCMIPP_P1CCRR1) 1945

39.14.59 DCMIPP Pipe1 ColorConv red coefficient register 2 (DCMIPP_P1CCRR2) 1945
39.14.60 DCMIPP Pipe1 ColorConv green coefficient register 1 (DCMIPP_P1CCGR1) 1946
39.14.61 DCMIPP Pipe1 ColorConv green coefficient register 2 (DCMIPP_P1CCGR2) 1946
39.14.62 DCMIPP Pipex ColorConv blue coefficient register 1 (DCMIPP_P1CCBR1) 1947
39.14.63 DCMIPP Pipe1 ColorConv blue coefficient register 2 (DCMIPP_P1CCBR2) 1947
39.14.64 DCMIPP Pipe1 contrast control register 1 (DCMIPP_P1CTCR1) . . . 1948
39.14.65 DCMIPP Pipe1 contrast control register 2 (DCMIPP_P1CTCR2) . . . 1948
39.14.66 DCMIPP Pipe1 contrast control register 3 (DCMIPP_P1CTCR3) . . . 1949
39.14.67 DCMIPP Pipex flow control configuration register (DCMIPP_PxFCTCR) 1949
39.14.68 DCMIPP Pipex crop window start register (DCMIPP_PxCRSTR) . . . 1950
39.14.69 DCMIPP Pipex crop window size register (DCMIPP_PxCRSZR) . . . 1951
39.14.70 DCMIPP Pipex decimation register (DCMIPP_PxDCCR) 1951
39.14.71 DCMIPP Pipex downsize configuration register (DCMIPP_PxDSCR) 1952
39.14.72 DCMIPP Pipex downsize ratio register (DCMIPP_PxDSRTIOR) . . . 1952
39.14.73 DCMIPP Pipex downsize destination size register (DCMIPP_PxDSSZR) 1953
39.14.74 DCMIPP Pipex common ROI configuration register (DCMIPP_PxCMRICR) 1953
39.14.75 DCMIPP Pipe1 ROIx configuration register 1 (DCMIPP_P1RIxCR1) 1955
39.14.76 DCMIPP Pipe1 ROIx configuration register 2 (DCMIPP_P1RIxCR2) 1955
39.14.77 DCMIPP Pipex gamma configuration register (DCMIPP_PxGMCR) . 1956
39.14.78 DCMIPP Pipe1 YUVConv configuration register (DCMIPP_P1YUVCR) 1956
39.14.79 DCMIPP Pipe1 YUVConv red coefficient register 1 (DCMIPP_P1YUVRR1) 1956
39.14.80 DCMIPP Pipe1 YUVConv red coefficient register 2 (DCMIPP_P1YUVRR2) 1957
39.14.81 DCMIPP Pipe1 YUVConv green coefficient register 1 (DCMIPP_P1YUVGR1) 1957
39.14.82 DCMIPP Pipe1 YUVConv green coefficient register 2 (DCMIPP_P1YUVGR2) 1958
39.14.83 DCMIPP Pipe1 YUVConv blue coefficient register 1 (DCMIPP_P1YUVBR1) 1958
39.14.84 DCMIPP Pipe1 YUV blue coefficient register 2 (DCMIPP_P1YUVBR2) 1958

39.14.85 DCMIPP Pipe1 pixel packer configuration register (DCMIPP_P1PPCR) 1959
39.14.86 DCMIPP Pipe1 pixel packer Memory0 address register 1 (DCMIPP_P1PPM0AR1) 1960
39.14.87 DCMIPP Pipe1 pixel packer Memory0 address register 2 (DCMIPP_P1PPM0AR2) 1961
39.14.88 DCMIPP Pipex pixel packer Memory0 pitch register (DCMIPP_PxPPM0PR) 1961
39.14.89 DCMIPP Pipex status Memory0 address register (DCMIPP_PxSTM0AR) 1962
39.14.90 DCMIPP Pipex pixel packer Memory1 address register 1 (DCMIPP_PxPPM1AR1) 1962
39.14.91 DCMIPP Pipex pixel packer Memory1 address register 2 (DCMIPP_PxPPM1AR2) 1962
39.14.92 DCMIPP Pipex pixel packer Memory1 pitch register (DCMIPP_PxPPM1PR) 1963
39.14.93 DCMIPP Pipex status Memory1 address register (DCMIPP_PxSTM1AR) 1963
39.14.94 DCMIPP Pipex pixel packer memory2 address register 1 (DCMIPP_PxPPM2AR1) 1964
39.14.95 DCMIPP Pipex pixel packer memory2 address register 2 (DCMIPP_PxPPM2AR2) 1964
39.14.96 DCMIPP Pipex status Memory2 address register (DCMIPP_PxSTM2AR) 1964
39.14.97 DCMIPP Pipe1 interrupt enable register (DCMIPP_P1IER) 1965
39.14.98 DCMIPP Pipe1 status register (DCMIPP_P1SR) 1966
39.14.99 DCMIPP Pipe1 interrupt clear register (DCMIPP_P1FCR) 1967
39.14.100 DCMIPP Pipe1 current flow selection configuration register (DCMIPP_P1CFSCR) 1967
39.14.101 DCMIPP Pipe1 current bad pixel removal register (DCMIPP_P1CBPRCR) 1968
39.14.102 DCMIPP Pipe1 current black level calibration control register (DCMIPP_P1CBLCCR) 1969
39.14.103 DCMIPP Pipe1 current exposure control register 1 (DCMIPP_P1CEXCR1) 1969
39.14.104 DCMIPP Pipe1 current exposure control register 2 (DCMIPP_P1CEXCR2) 1970
39.14.105 DCMIPP Pipe1 current statistics 1 control register (DCMIPP_P1CST1CR) 1971
39.14.106 DCMIPP Pipe1 current statistics 2 control register (DCMIPP_P1CST2CR) 1972
39.14.107 DCMIPP Pipe1 current statistics 3 control register (DCMIPP_P1CST3CR) 1973

39.14.108DCMIPP Pipe1 current statistics window start register (DCMIPP_P1CSTSTR) 1974
39.14.109DCMIPP Pipe1 current statistics window size register (DCMIPP_P1CSTSZR) 1974
39.14.110DCMIPP Pipe1 current ColorConv configuration register (DCMIPP_P1CCCCR) 1975
39.14.111DCMIPP Pipe1 current ColorConv red coefficient register 1 (DCMIPP_P1CCCRR1) 1975
39.14.112DCMIPP Pipe1 current ColorConv red coefficient register 2 (DCMIPP_P1CCCRR2) 1976
39.14.113DCMIPP Pipe1 current ColorConv green coefficient register 1 (DCMIPP_P1CCCGR1) 1976
39.14.114DCMIPP Pipe1 current ColorConv green coefficient register 2 (DCMIPP_P1CCCGR2) 1977
39.14.115DCMIPP Pipex current ColorConv blue coefficient register 1 (DCMIPP_P1CCCBR1) 1977
39.14.116DCMIPP Pipe1 current ColorConv blue coefficient register 2 (DCMIPP_P1CCCBR2) 1978
39.14.117DCMIPP Pipe1 current contrast control register 1 (DCMIPP_P1CCTCR1) 1978
39.14.118DCMIPP Pipe1 current contrast control register 2 (DCMIPP_P1CCTCR2) 1979
39.14.119DCMIPP Pipe1 current contrast control register 3 (DCMIPP_P1CCTCR3) 1979
39.14.120DCMIPP Pipex current flow control configuration register (DCMIPP_PxCFCTCR) 1980
39.14.121DCMIPP Pipex current crop window start register (DCMIPP_PxCCRSTR) 1981
39.14.122DCMIPP Pipex current crop window size register (DCMIPP_PxCCRSZR) 1981
39.14.123DCMIPP Pipex current decimation register (DCMIPP_PxCDCCR) . 1982
39.14.124DCMIPP Pipex current downsize configuration register (DCMIPP_PxCDSCR) 1982
39.14.125DCMIPP Pipex current downsize ratio register (DCMIPP_PxCDSRTIOR) 1983
39.14.126DCMIPP Pipex current downsize destination size register (DCMIPP_PxCDSSZR) 1983
39.14.127DCMIPP Pipex current common ROI configuration register (DCMIPP_PxCCMRICR) 1984
39.14.128DCMIPP Pipe1 current ROIx configuration register 1 (DCMIPP_P1CRIxCR1) 1985
39.14.129DCMIPP Pipe1 current ROIx configuration register 2 (DCMIPP_P1CRIxCR2) 1985

39.14.130DCMIPP Pipe1 current pixel packer configuration register (DCMIPP_P1CPPCR) 1986
39.14.131DCMIPP Pipe1 current pixel packer Memory0 address register 1 (DCMIPP_P1CPPM0AR1) 1987
39.14.132DCMIPP Pipe1 current pixel packer Memory0 address register 2 (DCMIPP_P1CPPM0AR2) 1988
39.14.133DCMIPP Pipex current pixel packer Memory0 pitch register (DCMIPP_PxCPPM0PR) 1988
39.14.134DCMIPP Pipex current pixel packer Memory1 address register 1 (DCMIPP_PxCPPM1AR1) 1988
39.14.135DCMIPP Pipex current pixel packer Memory1 address register 2 (DCMIPP_PxCPPM1AR2) 1989
39.14.136DCMIPP Pipex current pixel packer Memory1 pitch register (DCMIPP_PxCPPM1PR) 1989
39.14.137DCMIPP Pipex current pixel packer Memory2 address register 1 (DCMIPP_PxCPPM2AR1) 1990
39.14.138DCMIPP Pipex current pixel packer Memory2 address register 1 (DCMIPP_PxCPPM2AR2) 1990
39.14.139DCMIPP Pipe2 flow selection configuration register (DCMIPP_P2FSCR) 1990
39.14.140DCMIPP Pipe2 ROIx configuration register 1 (DCMIPP_P2RIxCR1) 1991
39.14.141DCMIPP Pipe2 ROIx configuration register 2 (DCMIPP_P2RIxCR2) 1992
39.14.142DCMIPP Pipe2 pixel packer configuration register (DCMIPP_P2PPCR) 1992
39.14.143DCMIPP Pipe2 pixel packer Memory0 address register 1 (DCMIPP_P2PPM0AR1) 1994
39.14.144DCMIPP Pipe2 pixel packer Memory0 address register 2 (DCMIPP_P2PPM0AR2) 1994
39.14.145DCMIPP Pipe2 interrupt enable register (DCMIPP_P2IER) 1994
39.14.146DCMIPP Pipe2 status register (DCMIPP_P2SR) 1995
39.14.147DCMIPP Pipe2 interrupt clear register (DCMIPP_P2FCR) 1996
39.14.148DCMIPP Pipe2 current flow selection configuration register (DCMIPP_P2CFSCR) 1997
39.14.149DCMIPP Pipe2 current ROIx configuration register 1 (DCMIPP_P2CRIxCR1) 1998
39.14.150DCMIPP Pipe2 current ROIx configuration register 2 (DCMIPP_P2CRIxCR2) 1998
39.14.151DCMIPP Pipe2 current pixel packer configuration register (DCMIPP_P2CPPCR) 1999
39.14.152DCMIPP Pipe2 current pixel packer Memory0 address register 1 (DCMIPP_P2CPPM0AR1) 2000
39.14.153DCMIPP Pipe2 current pixel packer Memory0 address register 2 (DCMIPP_P2CPPM0AR2) 2000

39.15 DCMIPP register map 2001

40 CSI-2 Host (CSI) 2016

40.1 CSI introduction 2016
40.2 Standard and references 2016
40.3 Glossary 2016
40.4 CSI-2 Host main features 2017
40.5 CSI-2 Host functional description 2017
40.5.1 General description 2017
40.5.2 System level architecture 2019
40.5.3 CSI-2 Host reset and clocks 2020
40.5.4 Lane merger 2021
40.5.5 Clock changer 2022
40.5.6 Low-level protocol (LLP) 2023
40.5.7 CSI-2 Host protocol output 2033
40.6 CSI-2 Host programming guide 2033
40.6.1 CSI-2 PHY setup 2033
40.6.2 CSI-2 Host controller setup and data reception start 2034
40.6.3 Data reception stop 2035
40.7 CSI-2 Host low-power modes 2035
40.8 CSI-2 Host interrupts and errors 2035
40.8.1 Error handling 2037
40.8.2 Interrupt handling 2039
40.9 CSI-2 Host registers 2040
40.9.1 CSI-2 Host control register (CSI_CR) 2040
40.9.2 CSI-2 Host DPHY_RX control register (CSI_PCR) 2042
40.9.3 CSI-2 Host virtual channel x configuration register 1 (CSI_VCxCFGR1) 2043
40.9.4 CSI-2 Host virtual channel x configuration register 2 (CSI_VCxCFGR2) 2045
40.9.5 CSI-2 Host virtual channel x configuration register 3 (CSI_VCxCFGR3) 2046
40.9.6 CSI-2 Host virtual channel x configuration register 4 (CSI_VCxCFGR4) 2047
40.9.7 CSI-2 Host line byte x configuration register (CSI_LBxCFGR) 2048
40.9.8 CSI-2 Host timer x configuration register (CSI_TIMxCFGR) 2049
40.9.9 CSI-2 Host lane merger configuration register (CSI_LMCFGR) 2049

40.9.10 CSI-2 Host program interrupt register (CSI_PRGITR) 2050
40.9.11 CSI-2 Host watchdog register (CSI_WDR) 2054
40.9.12 CSI-2 Host interrupt enable register 0 (CSI_IER0) 2054
40.9.13 CSI-2 Host interrupt enable register 1 (CSI_IER1) 2057
40.9.14 CSI-2 Host status register 0 (CSI_SR0) 2059
40.9.15 CSI-2 Host status register 1 (CSI_SR1) 2061
40.9.16 CSI-2 Host flag clear register 0 (CSI_FCR0) 2063
40.9.17 CSI-2 Host flag clear register 1 (CSI_FCR1) 2064
40.9.18 CSI-2 Host short packet data field register (CSI_SPDFR) 2065
40.9.19 CSI-2 Host error register 1 (CSI_ERR1) 2066
40.9.20 CSI-2 Host error register 2 (CSI_ERR2) 2067
40.10 CSI PHY registers 2067
40.10.1 CSI PHY reset control register (CSI_PRCR) 2067
40.10.2 CSI PHY mode control register (CSI_PMCR) 2068
40.10.3 CSI PHY frequency control register (CSI_PFCR) 2069
40.10.4 CSI PHY test control register 0 (CSI_PTCR0) 2069
40.10.5 CSI PHY test control register 1 (CSI_PTCR1) 2070
40.10.6 CSI PHY test status register (CSI_PTSR) 2070
40.10.7 CSI-2 Host and PHY register map 2070

41 Parallel synchronous slave interface (PSSI) 2074
41.1 PSSI introduction 2074
41.2 PSSI main features 2074
41.3 PSSI functional description 2074
41.3.1 PSSI block diagram 2075
41.3.2 PSSI pins and internal signals 2075
41.3.3 PSSI clock 2076
41.3.4 PSSI data management 2076
41.3.5 PSSI optional control signals 2078
41.4 PSSI interrupts 2081
41.5 PSSI registers 2082
41.5.1 PSSI control register (PSSI_CR) 2082
41.5.2 PSSI status register (PSSI_SR) 2083
41.5.3 PSSI raw interrupt status register (PSSI_RIS) 2084
41.5.4 PSSI interrupt enable register (PSSI_IER) 2085
41.5.5 PSSI masked interrupt status register (PSSI_MIS) 2085

41.5.6 PSSI interrupt clear register (PSSI_ICR) 2086
41.5.7 PSSI data register (PSSI_DR) 2086
41.5.8 PSSI register map 2087

42 Display subsystem 2088
42.1 Introduction 2088
42.2 Display subsystem main features 2088
42.3 Display subsystem implementation 2088
42.4 Display subsystem functional description 2088
42.4.1 Display subsystem block diagram 2088
42.4.2 Display subsystem pins and external signal interface 2089
42.4.3 Display subsystem clocks 2089
42.4.4 Display subsystem security 2089
42.5 Display subsystem programmable parameters 2092
42.6 Display subsystem interrupts 2093
42.7 Display subsystem registers 2093

43 LCD-TFT display controller (LTDC) 2094
43.1 LTDC introduction 2094
43.2 LTDC main features 2094
43.3 LTDC functional description 2095
43.3.1 LTDC block diagram 2095
43.3.2 LTDC pins and internal signals 2095
43.3.3 LTDC reset and clocks 2096
43.4 LTDC configuration parameters 2098
43.4.1 AXI master 2098
43.4.2 Input layer definition and cropping 2099
43.4.3 Input pixel format 2100
43.4.4 YUV planar 2102
43.4.5 YUV-to-RGB color conversion 2103
43.4.6 Horizontal or vertical mirroring 2104
43.4.7 Default layer color 2105
43.4.8 Color look-up table (CLUT) 2105
43.4.9 Transparency color keying 2106
43.4.10 Display composition - windowing 2106
43.4.11 Display composition - blending 2107

43.4.12 Gamma correction 2109
43.4.13 YUV output conversion 2109
43.4.14 Dithering 2110
43.4.15 CRC hashing 2111
43.4.16 Display timing 2111
43.4.17 Output interface polarity 2113
43.4.18 Shadow registers 2114
43.4.19 General control 2115
43.4.20 Hardware trigger generation 2115
43.4.21 Provision for a secure layer 2116
43.5 LTDC interrupts 2117
43.6 LTDC programming procedure 2118
43.7 LTDC registers 2119
43.7.1 LTDC synchronization size configuration register (LTDC_SSCR) . . . 2119
43.7.2 LTDC back porch configuration register (LTDC_BPCR) 2119
43.7.3 LTDC active width configuration register (LTDC_AWCR) 2120
43.7.4 LTDC total width configuration register (LTDC_TWCR) 2121
43.7.5 LTDC global control register (LTDC_GCR) 2121
43.7.6 LTDC shadow reload configuration register (LTDC_SRCR) 2123
43.7.7 LTDC gamma correction configuration register (LTDC_GCCR) 2123
43.7.8 LTDC background color configuration register (LTDC_BCCR) 2124
43.7.9 LTDC interrupt enable register (LTDC_IER) 2125
43.7.10 LTDC interrupt status register (LTDC_ISR) 2126
43.7.11 LTDC interrupt clear register (LTDC_ICR) 2126
43.7.12 LTDC line interrupt position configuration register (LTDC_LIPCR) . . 2127
43.7.13 LTDC current position status register (LTDC_CPSR) 2128
43.7.14 LTDC current display status register (LTDC_CDSR) 2128
43.7.15 LTDC external display control register (LTDC_EDCR) 2129
43.7.16 LTDC interrupt enable register 2 (LTDC_IER2) 2129
43.7.17 LTDC interrupt status register 2 (LTDC_ISR2) 2130
43.7.18 LTDC interrupt clear register 2 (LTDC_ICR2) 2131
43.7.19 LTDC line interrupt position configuration register 2 (LTDC_LIPCR2) 2132
43.7.20 LTDC expected CRC register (LTDC_ECRCR) 2132
43.7.21 LTDC computed CRC register (LTDC_CCRCR) 2133
43.7.22 LTDC FIFO underrun threshold register (LTDC_FUTR) 2133
43.7.23 LTDC layer x configuration 0 register (LTDC_LxC0R) 2134
43.7.24 LTDC layer x configuration 1 register (LTDC_LxC1R) 2135

43.7.25 LTDC layer x reload control register (LTDC_LxRCR) 2135
43.7.26 LTDC layer x control register (LTDC_LxCR) 2136
43.7.27 LTDC layer x window horizontal position configuration register (LTDC_LxWHPCR) 2137
43.7.28 LTDC layer x window vertical position configuration register (LTDC_LxWVPCR) 2138
43.7.29 LTDC layer x color keying configuration register (LTDC_LxCKCR) 2138
43.7.30 LTDC layer x pixel format configuration register (LTDC_LxPFCR) 2139
43.7.31 LTDC layer x constant alpha configuration register (LTDC_LxCACR) 2139
43.7.32 LTDC layer x default color configuration register (LTDC_LxDCCR) 2140
43.7.33 LTDC layer x blending factors configuration register (LTDC_LxBFCR) 2140
43.7.34 LTDC layer x burst length configuration register (LTDC_LxBLCR) 2141
43.7.35 LTDC layer x planar configuration register (LTDC_LxPCR) 2142
43.7.36 LTDC layer x color frame buffer address register (LTDC_LxCFBAR) 2143
43.7.37 LTDC layer x color frame buffer length register (LTDC_LxCFBLR) 2143
43.7.38 LTDC layer x color frame buffer line number register (LTDC_LxCFBLNR) 2144
43.7.39 LTDC layer 1 auxiliary frame buffer address 0 register (LTDC_L1AFBA0R) 2144
43.7.40 LTDC layer 1 auxiliary frame buffer address 1 register (LTDC_L1AFBA1R) 2145
43.7.41 LTDC layer 1 auxiliary frame buffer length register (LTDC_L1AFBLR) 2145
43.7.42 LTDC layer 1 auxiliary frame buffer line number register (LTDC_L1AFBLNR) 2146
43.7.43 LTDC layer x CLUT write register (LTDC_LxCLUTWR) 2146
43.7.44 LTDC layer x conversion YCbCr RGB 0 register (LTDC_LxCYR0R) . 2147
43.7.45 LTDC layer x conversion YCbCr RGB 1 register (LTDC_LxCYR1R) . 2147
43.7.46 LTDC layer x flexible pixel format 0 register (LTDC_LxFPF0R) 2148
43.7.47 LTDC layer x flexible pixel format 1 register (LTDC_LxFPF1R) 2148
43.7.48 LTDC register map 2149

44 Neo-Chrom graphic processor (GPU2D) 2154

44.1 GPU2D introduction . . . 2154
44.2 GPU2D main features . . . 2154
44.3 GPU2D general description . . . 2155
44.3.1 GPU2D block diagram . . . 2155
44.3.2 GPU2D pins and internal signals . . . 2155

45 Video encoder (VENC) . . . 2157
45.1 VENC introduction . . . 2157
45.2 VENC main features . . . 2157
45.3 VENC implementation . . . 2157
45.4 VENC functional description . . . 2158
45.4.1 VENC block diagram . . . 2158
45.4.2 VENC pins and internal signals . . . 2158
45.4.3 VENC reset and clocks . . . 2159
45.4.4 VENC pre-processor . . . 2159
45.4.5 VENC H264 encoder . . . 2159
45.4.6 VENC JPEG encoder . . . 2160
45.4.7 VENC synchronization . . . 2160
45.4.8 VENC security . . . 2161
45.4.9 VENC power-on sequence . . . 2161
45.4.10 VENCRAM power-on sequence for AXI access . . . 2162
45.5 VENC low power modes . . . 2162
45.6 VENC interrupts . . . 2162
45.7 VENC registers . . . 2162
45.7.1 VENC ID register (VENC_SWREG0) . . . 2162
45.7.2 VENC interrupt register (VENC_SWREG1) . . . 2163
45.7.3 VENC bus interface configuration register (VENC_SWREG2) . . . 2163
45.7.4 VENC device configuration register (VENC_SWREG3) . . . 2163
45.7.5 VENC base address for output stream data register (VENC_SWREG5) . . . 2164
45.7.6 VENC base address for output control data register (VENC_SWREG6) . . . 2164
45.7.7 VENC base address for reference luma register (VENC_SWREG7) . . . 2164
45.7.8 VENC base address for reference chroma register (VENC_SWREG8) . . . 2165

45.7.9 VENC base address for reconstructed luma register (VENC_SWREG9) 2165
45.7.10 VENC base address for reconstructed chroma register (VENC_SWREG10) 2165
45.7.11 VENC base address for input picture luma register (VENC_SWREG11) 2166
45.7.12 VENC base address for input picture cb register (VENC_SWREG12) 2166
45.7.13 VENC base address for input picture cr register (VENC_SWREG13) 2166
45.7.14 VENC encoder control register 0 (VENC_SWREG14) 2167
45.7.15 VENC encoder control register 1 (VENC_SWREG15) 2167
45.7.16 VENC encoder control register 2 (VENC_SWREG16) 2167
45.7.17 VENC encoder control register 3 (VENC_SWREG17) 2168
45.7.18 VENC encoder control register 4 (VENC_SWREG18) 2168
45.7.19 VENC encoder control register 5 (VENC_SWREG19) 2168
45.7.20 VENC encoder control register 6 (VENC_SWREG20) 2169
45.7.21 VENC encoder control register 7 (VENC_SWREG21) 2169
45.7.22 VENC stream header remainder MSB bits register (VENC_SWREG22) 2169
45.7.23 VENC stream header remainder LSB bits register (VENC_SWREG23) 2170
45.7.24 VENC stream buffer limit/output stream size register (VENC_SWREG24) 2170
45.7.25 VENC encoder control register 8 (VENC_SWREG25) 2170
45.7.26 VENC intra-slice bitmap register (VENC_SWREG26) 2171
45.7.27 VENC encoder control register 9 (VENC_SWREG27) 2171
45.7.28 VENC encoder control register 10 (VENC_SWREG28) 2171
45.7.29 VENC encoder control register 11 (VENC_SWREG29) 2172
45.7.30 VENC encoder control register 12 (VENC_SWREG30) 2172
45.7.31 VENC encoder control register 13 (VENC_SWREG31) 2172
45.7.32 VENC encoder control register 14 (VENC_SWREG32) 2173
45.7.33 VENC encoder control register 15 (VENC_SWREG33) 2173
45.7.34 VENC encoder control register 16 (VENC_SWREG34) 2173
45.7.35 VENC H.264 checkpoint word error 5-6/encoder control register 17 (VENC_SWREG35) 2174
45.7.36 VENC H.264 checkpoint delta QP 1-8/encoder control register 18 (VENC_SWREG36) 2174
45.7.37 VENC encoder control register 19, stream start offset (VENC_SWREG37) 2174

45.7.38 VENC macroblock count output register (VENC_SWREG38) 2175
45.7.39 VENC base address for next pic luminance register (VENC_SWREG39) 2175
45.7.40 VENC stabilization mode control register (VENC_SWREG40) 2175
45.7.41 VENC stabilization motion sum div8 output register (VENC_SWREG41) 2176
45.7.42 VENC stabilization GMV output, matrix 1, up-left position output register (VENC_SWREG42) 2176
45.7.43 VENC stabilization GMV output, matrix 2, up position output register (VENC_SWREG43) 2176
45.7.44 VENC stabilization matrix 3, up-right position output register (VENC_SWREG44) 2177
45.7.45 VENC stabilization matrix 4, left position output register (VENC_SWREG45) 2177
45.7.46 VENC stabilization matrix 5, GMV position output register (VENC_SWREG46) 2177
45.7.47 VENC stabilization matrix 6, right position output register (VENC_SWREG47) 2178
45.7.48 VENC stabilization matrix 7, down-left position output register (VENC_SWREG48) 2178
45.7.49 VENC stabilization matrix 8, down position output register (VENC_SWREG49) 2178
45.7.50 VENC stabilization matrix 9, down-right position output register (VENC_SWREG50) 2179
45.7.51 VENC base address for cabac context tables H264 register (VENC_SWREG51) 2179
45.7.52 VENC base address for MV output writing register (VENC_SWREG52) 2179
45.7.53 VENC RGB to YUV conversion coefficient A - B register (VENC_SWREG53) 2180
45.7.54 VENC RGB to YUV conversion coefficient C - E register (VENC_SWREG54) 2180
45.7.55 VENC RGB to YUV conversion coefficient F, RGB mask MSB bit position register (VENC_SWREG55) 2180
45.7.56 VENC intra area register (VENC_SWREG56) 2181
45.7.57 VENC CIR intra mb position register (VENC_SWREG57) 2181
45.7.58 VENC intra slice bitmap for slices 0..31/base address for 1st DCT partition register (VENC_SWREG58) 2181
45.7.59 VENC intra slice bitmap for slices 32..63/base address for 2nd DCT partition register (VENC_SWREG59) 2182
45.7.60 VENC 1st ROI area register (VENC_SWREG60) 2182
45.7.61 VENC 2nd ROI area register (VENC_SWREG61) 2182

45.7.62 VENC ROI area delta QP, MV register (VENC_SWREG62) 2183
45.7.63 VENC synthesis configuration register encoder 0 register (VENC_SWREG63) 2183
45.7.64 VENC JPEG luma quantization 1/intra 16x16 mode 0-1 penalty register (VENC_SWREG64) 2183
45.7.65 VENC JPEG luma quantization 2/intra 16x16 mode 2-3 penalty register (VENC_SWREG65) 2184
45.7.66 VENC JPEG luma quantization 3/intra 4x4 mode 0-1 penalty register (VENC_SWREG66) 2184
45.7.67 VENC JPEG luma quantization 4/intra 4x4 mode 2-3 penalty register (VENC_SWREG67) 2184
45.7.68 VENC JPEG luma quantization 5/intra 4x4 mode 4-5 penalty register (VENC_SWREG68) 2185
45.7.69 VENC JPEG luma quantization 6/intra 4x4 mode 6-7 penalty register (VENC_SWREG69) 2185
45.7.70 VENC JPEG luma quantization 7/intra 4x4 mode 8-9 penalty register (VENC_SWREG70) 2185
45.7.71 VENC JPEG luma quantization 8/base address for segmentation map register (VENC_SWREG71) 2186
45.7.72 VENC JPEG luma quantization 9/segment1 parameter register (VENC_SWREG72) 2186
45.7.73 VENC JPEG luma quantization 10/segment1 parameter register (VENC_SWREG73) 2186
45.7.74 VENC JPEG luma quantization 11/segment1 parameter register (VENC_SWREG74) 2187
45.7.75 VENC JPEG luma quantization 12/segment1 parameter register (VENC_SWREG75) 2187
45.7.76 VENC JPEG luma quantization 13/segment1 parameter register (VENC_SWREG76) 2187
45.7.77 VENC JPEG luma quantization 14/segment1 parameter register (VENC_SWREG77) 2188
45.7.78 VENC JPEG luma quantization 15/segment1 parameter register (VENC_SWREG78) 2188
45.7.79 VENC JPEG luma quantization 16/segment2 parameter register (VENC_SWREG79) 2188
45.7.80 VENC JPEG chroma quantization 1/segment2 parameter register (VENC_SWREG80) 2189
45.7.81 VENC JPEG chroma quantization 2/segment2 parameter register (VENC_SWREG81) 2189
45.7.82 VENC JPEG chroma quantization 3/segment2 parameter register (VENC_SWREG82) 2189
45.7.83 VENC JPEG chroma quantization 4/segment2 parameter register (VENC_SWREG83) 2190

45.7.84 VENC JPEG chroma quantization 5/segment2 parameter register (VENC_SWREG84) 2190
45.7.85 VENC JPEG chroma quantization 6/segment2 parameter register (VENC_SWREG85) 2190
45.7.86 VENC JPEG chroma quantization 7/segment2 parameter register (VENC_SWREG86) 2191
45.7.87 VENC JPEG chroma quantization 8/segment2 parameter register (VENC_SWREG87) 2191
45.7.88 VENC JPEG chroma quantization 9/segment3 parameter register (VENC_SWREG88) 2191
45.7.89 VENC JPEG chroma quantization 10/segment3 parameter register (VENC_SWREG89) 2192
45.7.90 VENC JPEG chroma quantization 11/segment3 parameter register (VENC_SWREG90) 2192
45.7.91 VENC JPEG chroma quantization 12/segment3 parameter register (VENC_SWREG91) 2192
45.7.92 VENC JPEG chroma quantization 13/segment3 parameter register (VENC_SWREG92) 2193
45.7.93 VENC JPEG chroma quantization 14/segment3 parameter register (VENC_SWREG93) 2193
45.7.94 VENC JPEG chroma quantization 15/segment3 parameter register (VENC_SWREG94) 2193
45.7.95 VENC JPEG chroma quantization 16/segment3 parameter register (VENC_SWREG95) 2194
45.7.96 VENC DMV 4p/1p penalty values 0-3 register (VENC_SWREG96) . 2194
45.7.97 VENC DMV 4p/1p penalty values 4-7 register (VENC_SWREG97) . 2194
45.7.98 VENC DMV 4p/1p penalty values register (VENC_SWREGx) 2195
45.7.99 VENC DMV 4p/1p penalty values 124-127 register (VENC_SWREG127) 2195
45.7.100 VENC DMV qpel penalty values 0-3 register (VENC_SWREG128) 2195
45.7.101 VENC DMV qpel penalty values 4-7 register (VENC_SWREG129) 2196
45.7.102 VENC DMV qpel penalty values register (VENC_SWREGx) 2196
45.7.103 VENC DMV qpel penalty values 124-127 register (VENC_SWREG159) 2196
45.7.104 VENC base address for output of down-scaled encoder image in YUYV 4:2:2 format register (VENC_SWREG231) 2197
45.7.105 VENC scaling control register (VENC_SWREGx) 2197
45.7.106 VENC squared error output calculated for 13x13 pixels per macroblock register (VENC_SWREG236) 2197
45.7.107 VENC MAD 2 control and output register (VENC_SWREG237) 2198

45.7.108 VENC MAD 3 control and output register (VENC_SWREG238) 2198
45.7.109 VENC segment 1: intra 16x16 mode 0-2 penalty register (VENC_SWREG256) 2198
45.7.110 VENC segment 1: intra 16x16 mode 3, intra 4x4 0-1 penalty register (VENC_SWREG257) 2199
45.7.111 VENC segment 1: intra 4x4 mode 2-4 penalty register (VENC_SWREG258) 2199
45.7.112 VENC segment 1: intra 4x4 mode 5-7 penalty register (VENC_SWREG259) 2199
45.7.113 VENC segment 1: intra 4x4 mode 8-9 penalty, previous mode favor for H.264 register (VENC_SWREG260) 2200
45.7.114 VENC segment 1: bit cost of inter type, intra 16x16 mode favor register (VENC_SWREG261) 2200
45.7.115 VENC segment 1: inter MB mode favor, skip mode penalty, penalty value for 2nd reference frame register (VENC_SWREG262) 2200
45.7.116 VENC segment 1: penalty value register (VENC_SWREGx) 2201
45.7.117 VENC segment 1: deadzone rate multiplier for plane 0-1 register (VENC_SWREG265) 2201
45.7.118 VENC segment 1: deadzone rate multiplier for plane 2-3 register (VENC_SWREG266) 2201
45.7.119 VENC segment 1: deadzone rate for macroblock skip token 0-1, dmv penalty coefficient register (VENC_SWREG267) 2202
45.7.120 VENC segment 2: intra 16x16 mode 0-2 penalty register (VENC_SWREG268) 2202
45.7.121 VENC segment 2: intra 16x16 mode 3 penalty, intra 4x4 mode 0-1 penalty register (VENC_SWREG269) 2202
45.7.122 VENC segment 2: intra 4x4 mode 2-4 penalty register (VENC_SWREG270) 2203
45.7.123 VENC segment 2: intra 4x4 mode 5-7 penalty register (VENC_SWREG271) 2203
45.7.124 VENC segment 2: intra 4x4 mode 8-9 penalty, intra 4x4 previous mode favor for H.264 register (VENC_SWREG272) 2203
45.7.125 VENC segment 2: bit cost of inter type, intra 16x16 mode favor register (VENC_SWREG273) 2204
45.7.126 VENC segment 2: inter MB mode favor, skip mode penalty, penalty value register (VENC_SWREG274) 2204
45.7.127 VENC segment 2: penalty value register (VENC_SWREGx) 2204
45.7.128 VENC segment 2: deadzone rate multiplier for plane 0-1 register (VENC_SWREG277) 2205
45.7.129 VENC segment 2: deadzone rate multiplier for plane 2-3 register (VENC_SWREG278) 2205
45.7.130 VENC segment 2: deadzone rate for macroblock skip token 0-1, dmv penalty coefficient register (VENC_SWREG279) 2205

45.7.131 VENC segment 3: intra 16x16 mode 0-2 penalty register (VENC_SWREG280) 2206
45.7.132 VENC segment 3: intra 16x16 mode 3 penalty, intra 4x4 mode 0-1 penalty register (VENC_SWREG281) 2206
45.7.133 VENC segment 3: intra 4x4 mode 2-4 penalty register (VENC_SWREG282) 2206
45.7.134 VENC segment 3: intra 4x4 mode 5-7 penalty register (VENC_SWREG283) 2207
45.7.135 VENC segment 3: intra 4x4 mode 8-9 penalty, intra 4x4 previous mode favor for H.264 register (VENC_SWREG284) 2207
45.7.136 VENC segment 3: bit cost of inter type, intra 16x16 mode favor register (VENC_SWREG285) 2207
45.7.137 VENC segment 3: inter MB mode favor in intra/inter selection, inter MB mode favor, penalty value for second reference frame register (VENC_SWREG286) 2208
45.7.138 VENC segment 3: penalty value register (VENC_SWREGx) 2208
45.7.139 VENC segment 3: deadzone rate multiplier for plane 0-1 register (VENC_SWREG289) 2208
45.7.140 VENC segment 3: deadzone rate multiplier for plane 2-3 register (VENC_SWREG290) 2209
45.7.141 VENC segment 3: deadzone rate for macroblock skip token 0-1, dmv penalty coefficient register (VENC_SWREG291) 2209
45.7.142 VENC Mb boost register (VENC_SWREG294) 2209
45.7.143 VENC variance control, Pskop conding mode register (VENC_SWREG295) 2210
45.7.144 VENC synthesis configuration register encoder 1 read only register (VENC_SWREG296) 2210
45.7.145 VENC MBRC control register (VENC_SWREG297) 2210
45.7.146 VENC segment 4: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG298) 2211
45.7.147 VENC segment 4: skip mode penalty, inter MB mode favor register (VENC_SWREG299) 2211
45.7.148 VENC segment 4: penalty value register (VENC_SWREGx) 2211
45.7.149 VENC segment 5: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG302) 2212
45.7.150 VENC segment 5: skip mode penalty, inter MB mode favor register (VENC_SWREG303) 2212
45.7.151 VENC segment 5: penalty value register (VENC_SWREGx) 2212
45.7.152 VENC segment 6: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG306) 2213

45.7.153 VENC segment 6: skip mode penalty, inter MB mode favor register (VENC_SWREG307) 2213
45.7.154 VENC segment 6: penalty value register (VENC_SWREGx) 2213
45.7.155 VENC segment 7: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG310) 2214
45.7.156 VENC segment 7: skip mode penalty, inter MB mode favor register (VENC_SWREG311) 2214
45.7.157 VENC segment 7: penalty value register (VENC_SWREGx) 2214
45.7.158 VENC segment 8: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG314) 2215
45.7.159 VENC segment 8: skip mode penalty, inter MB mode favor register (VENC_SWREG315) 2215
45.7.160 VENC segment 8: penalty value register (VENC_SWREGx) 2215
45.7.161 VENC segment 9: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG318) 2216
45.7.162 VENC segment 9: skip mode penalty, inter MB mode favor register (VENC_SWREG319) 2216
45.7.163 VENC segment 9: penalty value register (VENC_SWREGx) 2216
45.7.164 VENC segment 10: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG322) 2217
45.7.165 VENC segment 10: skip mode penalty, inter MB mode favor register (VENC_SWREG323) 2217
45.7.166 VENC segment 10: penalty value register (VENC_SWREGx) 2217
45.7.167 VENC segment 11: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG326) 2218
45.7.168 VENC segment 11: skip mode penalty, inter MB mode favor register (VENC_SWREG327) 2218
45.7.169 VENC segment 11: penalty value register (VENC_SWREGx) 2218
45.7.170 VENC segment 12: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG330) 2219
45.7.171 VENC segment 12: skip mode penalty, inter MB mode favor register (VENC_SWREG331) 2219
45.7.172 VENC segment 12: penalty value register (VENC_SWREGx) 2219
45.7.173 VENC segment 13: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG334) 2220

45.7.174 VENC segment 13: skip mode penalty, inter MB mode favor register (VENC_SWREG335) 2220
45.7.175 VENC segment 13: penalty value register (VENC_SWREGx) 2220
45.7.176 VENC segment 14: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG338) 2221
45.7.177 VENC segment 14: skip mode penalty, inter MB mode favor register (VENC_SWREG339) 2221
45.7.178 VENC segment 14: penalty value register (VENC_SWREGx) 2221
45.7.179 VENC segment 15: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG342) 2222
45.7.180 VENC segment 15: skip mode penalty, inter MB mode favor register (VENC_SWREG343) 2222
45.7.181 VENC segment 15: penalty value register (VENC_SWREGx) 2222
45.7.182 VENC segment 16: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG346) 2223
45.7.183 VENC segment 16: skip mode penalty, inter MB mode favor register (VENC_SWREG347) 2223
45.7.184 VENC segment 16: penalty value register (VENC_SWREGx) 2223
45.7.185 VENC segment 17: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG350) 2224
45.7.186 VENC segment 17: skip mode penalty, inter MB mode favor register (VENC_SWREG351) 2224
45.7.187 VENC segment 17: penalty value register (VENC_SWREGx) 2224
45.7.188 VENC segment 18: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG354) 2225
45.7.189 VENC segment 18: skip mode penalty, inter MB mode favor register (VENC_SWREG355) 2225
45.7.190 VENC segment 18: penalty value register (VENC_SWREGx) 2225
45.7.191 VENC segment 19: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG358) 2226
45.7.192 VENC segment 19: skip mode penalty, inter MB mode favor register (VENC_SWREG359) 2226
45.7.193 VENC segment 19: penalty value register (VENC_SWREGx) 2226
45.7.194 VENC segment 20: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG362) 2227

45.7.195 VENC segment 20: skip mode penalty, inter MB mode favor register (VENC_SWREG363) 2227
45.7.196 VENC segment 20: penalty value register (VENC_SWREGx) 2227
45.7.197 VENC segment 21: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG366) 2228
45.7.198 VENC segment 21: skip mode penalty, inter MB mode favor register (VENC_SWREG367) 2228
45.7.199 VENC segment 21: penalty value register (VENC_SWREGx) 2228
45.7.200 VENC segment 22: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG370) 2229
45.7.201 VENC segment 22: skip mode penalty, inter MB mode favor register (VENC_SWREG371) 2229
45.7.202 VENC segment 22: penalty value register (VENC_SWREGx) 2229
45.7.203 VENC segment 23: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG374) 2230
45.7.204 VENC segment 23: skip mode penalty, inter MB mode favor register (VENC_SWREG375) 2230
45.7.205 VENC segment 23: penalty value register (VENC_SWREGx) 2230
45.7.206 VENC segment 24: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG378) 2231
45.7.207 VENC segment 24: skip mode penalty, inter MB mode favor register (VENC_SWREG379) 2231
45.7.208 VENC segment 24: penalty value register (VENC_SWREGx) 2231
45.7.209 VENC segment 25: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG382) 2232
45.7.210 VENC segment 25: skip mode penalty, inter MB mode favor register (VENC_SWREG383) 2232
45.7.211 VENC segment 25: penalty value register (VENC_SWREGx) 2232
45.7.212 VENC segment 26: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG386) 2233
45.7.213 VENC segment 26: skip mode penalty, inter MB mode favor register (VENC_SWREG387) 2233
45.7.214 VENC segment 26: penalty value register (VENC_SWREGx) 2233
45.7.215 VENC segment 27: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG390) 2234

45.7.216 VENC segment 27: skip mode penalty, inter MB mode favor register (VENC_SWREG391) 2234
45.7.217 VENC segment 27: penalty value register (VENC_SWREGx) 2234
45.7.218 VENC segment 28: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG394) 2235
45.7.219 VENC segment 28: skip mode penalty, inter MB mode favor register (VENC_SWREG395) 2235
45.7.220 VENC segment 28: penalty value register (VENC_SWREGx) 2235
45.7.221 VENC segment 29: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG398) 2236
45.7.222 VENC segment 29: skip mode penalty, inter MB mode favor register (VENC_SWREG399) 2236
45.7.223 VENC segment 29: penalty value register (VENC_SWREGx) 2236
45.7.224 VENC segment 30: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG402) 2237
45.7.225 VENC segment 30: skip mode penalty, inter MB mode favor register (VENC_SWREG403) 2237
45.7.226 VENC segment 30: penalty value register (VENC_SWREGx) 2237
45.7.227 VENC segment 31: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG406) 2238
45.7.228 VENC segment 31: skip mode penalty, inter MB mode favor register (VENC_SWREG407) 2238
45.7.229 VENC segment 31: penalty value register (VENC_SWREGx) 2238
45.7.230 VENC MBRC control, QP, offset, enable register (VENC_SWREG410) 2239
45.7.231 VENC gain of MB QP delta. 8.8 format register (VENC_SWREG411) 2239
45.7.232 VENC average of MB complexity register (VENC_SWREG412) . . . 2239
45.7.233 VENC reference compression control register (VENC_SWREG413) 2240
45.7.234 VENC base address for reference luma register (VENC_SWREG414) 2240
45.7.235 VENC base address for reference chroma register (VENC_SWREG415) 2240
45.7.236 VENC base address for reconstructed luma register (VENC_SWREG416) 2241
45.7.237 VENC base address for reconstructed chroma register (VENC_SWREG417) 2241

45.7.238 VENC base address for second reference luma register (VENC_SWREG418) 2241
45.7.239 VENC base address for second reference chroma register (VENC_SWREG419) 2242
45.7.240 VENC limit of chroma RFC buffer register (VENC_SWREG420) 2242
45.7.241 VENC reorder control register (VENC_SWREG421) 2242
45.7.242 VENC AXI read ID register (VENC_SWREG422) 2243
45.7.243 VENC base address MSB for reference luma compression table register (VENC_SWREG423) 2243
45.7.244 VENC base address MSB for reference chroma compression table register (VENC_SWREG424) 2243
45.7.245 VENC base address MSB for reconstructed luma compression table register (VENC_SWREG425) 2244
45.7.246 VENC base address for reconstructed chroma compression table register (VENC_SWREG426) 2244
45.7.247 VENC base address MSB for second reference luma compression table register (VENC_SWREG427) 2244
45.7.248 VENC base address MSB for second reference chroma compression table register (VENC_SWREG428) 2245
45.7.249 VENC high 32 bits of base address for output stream data register (VENC_SWREG429) 2245
45.7.250 VENC high 32 bits of base address for output control data register (VENC_SWREG430) 2245
45.7.251 VENC high 32 bits of base address for reference luma register (VENC_SWREG431) 2246
45.7.252 VENC high 32 bits of base address for reference chroma register (VENC_SWREG432) 2246
45.7.253 VENC high 32 bits of base address for reconstructed luma register (VENC_SWREG433) 2246
45.7.254 VENC high 32 bits of base address for reconstructed chroma register (VENC_SWREG434) 2247
45.7.255 VENC high 32 bits of base address for input picture luma register (VENC_SWREG435) 2247
45.7.256 VENC high 32 bits of base address for input picture cb register (VENC_SWREG436) 2247
45.7.257 VENC high 32 bits of base address for input picture cr register (VENC_SWREG437) 2248
45.7.258 VENC high 32 bits of base address for second reference luma register (VENC_SWREG438) 2248
45.7.259 VENC high 32 bits of base address for second reference chroma register (VENC_SWREG439) 2248

45.7.260 VENC high 32 bits of H264 secondary ref pic base register (VENC_SWREGx) . . . 2249
45.7.261 VENC high 32 bits of base address for next pic luminance register (VENC_SWREG442) . . . 2249
45.7.262 VENC high 32 bits of base address for cabac context tables H264 register (VENC_SWREG443) . . . 2249
45.7.263 VENC high 32 bits of base address for MV output writing register (VENC_SWREG444) . . . 2250
45.7.264 VENC high 32 bits of base address for output of down-scaled encoder image in YUYV 4:2:2 format register (VENC_SWREG449) 2250
45.7.265 VENC low-latency control register (VENC_SWREG497) . . . 2250
45.7.266 VENC encoder line buffer offset register (VENC_SWREG498) . . . 2251
45.7.267 VENC register map . . . 2251

46 JPEG codec (JPEG) . . . 2265

46.1 JPEG introduction . . . 2265
46.2 JPEG codec main features . . . 2265
46.3 JPEG codec block functional description . . . 2266
46.3.1 General description . . . 2266
46.3.2 JPEG internal signals . . . 2266
46.3.3 JPEG decoding procedure . . . 2267
46.3.4 JPEG encoding procedure . . . 2269
46.4 JPEG codec interrupts . . . 2272
46.5 JPEG codec registers . . . 2272
46.5.1 JPEG codec control register (JPEG_CONFR0) . . . 2272
46.5.2 JPEG codec configuration register 1 (JPEG_CONFR1) . . . 2273
46.5.3 JPEG codec configuration register 2 (JPEG_CONFR2) . . . 2274
46.5.4 JPEG codec configuration register 3 (JPEG_CONFR3) . . . 2274
46.5.5 JPEG codec configuration register x (JPEG_CONFRx) . . . 2275
46.5.6 JPEG control register (JPEG_CR) . . . 2276
46.5.7 JPEG status register (JPEG_SR) . . . 2277
46.5.8 JPEG clear flag register (JPEG_CFR) . . . 2278
46.5.9 JPEG data input register (JPEG_DIR) . . . 2279
46.5.10 JPEG data output register (JPEG_DOR) . . . 2279
46.5.11 JPEG quantization memory x (JPEG_QMEMx_y) . . . 2280
46.5.12 JPEG Huffman min (JPEG_HUFFMINx_y) . . . 2280
46.5.13 JPEG Huffman min x (JPEG_HUFFMINx_y) . . . 2281

46.5.14 JPEG Huffman base (JPEG_HUFFBASEx) 2281
46.5.15 JPEG Huffman symbol (JPEG_HUFFSYMBx) 2282
46.5.16 JPEG DHT memory (JPEG_DHTMEMx) 2283
46.5.17 JPEG Huffman encoder ACx (JPEG_HUFFENC_ACx_y) 2283
46.5.18 JPEG Huffman encoder DCx (JPEG_HUFFENC_DCx_y) 2284
46.5.19 JPEG codec register map 2285

47 True random number generator (RNG) 2287
47.1 RNG introduction 2287
47.2 RNG main features 2287
47.3 RNG functional description 2288
47.3.1 RNG block diagram 2288
47.3.2 RNG internal signals 2288
47.3.3 Random number generation 2288
47.3.4 RNG initialization 2291
47.3.5 RNG operation 2292
47.3.6 RNG clocking 2294
47.3.7 Error management 2294
47.3.8 RNG low-power use 2295
47.4 RNG interrupts 2296
47.5 RNG processing time 2296
47.6 RNG entropy source validation 2297
47.6.1 Introduction 2297
47.6.2 Validation conditions 2297
47.7 RNG registers 2298
47.7.1 RNG control register (RNG_CR) 2298
47.7.2 RNG status register (RNG_SR) 2300
47.7.3 RNG data register (RNG_DR) 2301
47.7.4 RNG noise source control register (RNG_NSCR) 2302
47.7.5 RNG health test control register (RNG_HTCR) 2303
47.7.6 RNG register map 2303

48 Secure AES coprocessor (SAES) 2304
48.1 SAES introduction 2304
48.2 SAES main features 2304
48.3 SAES implementation 2305

48.4 SAES functional description 2305
48.4.1 SAES block diagram 2305
48.4.2 SAES internal signals 2306
48.4.3 SAES reset and clocks 2307
48.4.4 SAES symmetric cipher implementation 2307
48.4.5 SAES encryption or decryption typical usage 2308
48.4.6 SAES authenticated encryption, decryption, and cipher-based message authentication 2310
48.4.7 SAES ciphertext stealing and data padding 2311
48.4.8 SAES suspend and resume operations 2311
48.4.9 SAES basic chaining modes (ECB, CBC) 2312
48.4.10 SAES counter (CTR) mode 2316
48.4.11 SAES Galois/counter mode (GCM) 2318
48.4.12 SAES Galois message authentication code (GMAC) 2322
48.4.13 SAES counter with CBC-MAC (CCM) 2324
48.4.14 SAES operation with wrapped keys 2329
48.4.15 SAES operation with shared keys 2333
48.4.16 SAES data registers and data swapping 2334
48.4.17 SAES key registers 2337
48.4.18 SAES initialization vector registers 2338
48.4.19 SAES error management 2339
48.5 SAES interrupts 2341
48.6 SAES DMA requests 2341
48.7 SAES processing latency 2342
48.8 SAES registers 2344
48.8.1 SAES control register (SAES_CR) 2344
48.8.2 SAES status register (SAES_SR) 2347
48.8.3 SAES data input register (SAES_DINR) 2348
48.8.4 SAES data output register (SAES_DOUTR) 2349
48.8.5 SAES key register 0 (SAES_KEYR0) 2349
48.8.6 SAES key register 1 (SAES_KEYR1) 2350
48.8.7 SAES key register 2 (SAES_KEYR2) 2350
48.8.8 SAES key register 3 (SAES_KEYR3) 2350
48.8.9 SAES initialization vector register 0 (SAES_IVR0) 2351
48.8.10 SAES initialization vector register 1 (SAES_IVR1) 2351
48.8.11 SAES initialization vector register 2 (SAES_IVR2) 2351
48.8.12 SAES initialization vector register 3 (SAES_IVR3) 2352

48.8.13 SAES key register 4 (SAES_KEYR4) 2352
48.8.14 SAES key register 5 (SAES_KEYR5) 2352
48.8.15 SAES key register 6 (SAES_KEYR6) 2353
48.8.16 SAES key register 7 (SAES_KEYR7) 2353
48.8.17 SAES suspend registers (SAES_SUSPRx) 2353
48.8.18 SAES interrupt enable register (SAES_IER) 2354
48.8.19 SAES interrupt status register (SAES_ISR) 2355
48.8.20 SAES interrupt clear register (SAES_ICR) 2356
48.8.21 SAES register map 2357

49 Cryptographic processor (CRYP) 2359

49.1 CRYP introduction 2359
49.2 CRYP main features 2359
49.3 CRYP implementation 2360
49.4 CRYP functional description 2361
49.4.1 CRYP block diagram 2361
49.4.2 CRYP internal signals 2362
49.4.3 CRYP reset and clocks 2362
49.4.4 CRYP symmetric cipher implementation 2362
49.4.5 CRYP encryption/ decryption typical usage 2363
49.4.6 CRYP authenticated encryption, decryption, and cipher-based message authentication 2366
49.4.7 CRYP ciphertext stealing and data padding 2366
49.4.8 CRYP suspend and resume operations 2367
49.4.9 CRYP basic chaining modes (ECB, CBC) 2367
49.4.10 CRYP counter mode (CTR) 2372
49.4.11 CRYP AES Galois/counter mode (GCM) 2374
49.4.12 CRYP AES Galois message authentication code (GMAC) 2380
49.4.13 CRYP AES Counter with CBC-MAC (CCM) 2381
49.4.14 AES key sharing with secure AES co-processor 2386
49.4.15 CRYP data registers and data swapping 2387
49.4.16 CRYP key registers 2390
49.4.17 CRYP initialization vector registers 2390
49.4.18 CRYP error management 2391
49.5 CRYP interrupts 2391
49.6 CRYP DMA requests 2392
49.7 CRYP processing time 2393

49.8 CRYP registers 2394
49.8.1 CRYP control register (CRYP_CR) 2394
49.8.2 CRYP status register (CRYP_SR) 2396
49.8.3 CRYP data input register (CRYP_DINR) 2398
49.8.4 CRYP data output register (CRYP_DOUTR) 2398
49.8.5 CRYP DMA control register (CRYP_DMACR) 2399
49.8.6 CRYP interrupt mask set/clear register (CRYP_IMSCR) 2399
49.8.7 CRYP raw interrupt status register (CRYP_RISR) 2400
49.8.8 CRYP masked interrupt status register (CRYP_MISR) 2400
49.8.9 CRYP key register 0L (CRYP_K0LR) 2401
49.8.10 CRYP key register 0R (CRYP_K0RR) 2402
49.8.11 CRYP key register 1L (CRYP_K1LR) 2402
49.8.12 CRYP key register 1R (CRYP_K1RR) 2402
49.8.13 CRYP key register 2L (CRYP_K2LR) 2403
49.8.14 CRYP key register 2R (CRYP_K2RR) 2403
49.8.15 CRYP key register 3L (CRYP_K3LR) 2404
49.8.16 CRYP key register 3R (CRYP_K3RR) 2404
49.8.17 CRYP initialization vector register 0L (CRYP_IV0LR) 2404
49.8.18 CRYP initialization vector register 0R (CRYP_IV0RR) 2405
49.8.19 CRYP initialization vector register 1L (CRYP_IV1LR) 2405
49.8.20 CRYP initialization vector register 1R (CRYP_IV1RR) 2406
49.8.21 CRYP context swap GCM-CCM registers (CRYP_CSGCMCCMxR) 2406
49.8.22 CRYP context swap GCM registers (CRYP_CSGCMxR) 2407
49.8.23 CRYP register map 2407

50 Hash processor (HASH) 2409

50.1 HASH introduction 2409
50.2 HASH main features 2409
50.3 HASH implementation 2410
50.4 HASH functional description 2410
50.4.1 HASH block diagram 2410
50.4.2 HASH internal signals 2410
50.4.3 About secure hash algorithms 2411
50.4.4 Message data feeding 2411
50.4.5 Message digest computing 2412
50.4.6 Message padding 2414
50.4.7 HMAC operation 2416

50.4.8 HASH suspend/resume operations 2417
50.4.9 HASH DMA interface 2419
50.4.10 HASH error management 2420
50.4.11 HASH processing time 2420
50.5 HASH interrupts 2421
50.6 HASH registers 2421
50.6.1 HASH control register (HASH_CR) 2421
50.6.2 HASH data input register (HASH_DIN) 2423
50.6.3 HASH start register (HASH_STR) 2424
50.6.4 HASH digest registers 2425
50.6.5 HASH interrupt enable register (HASH_IMR) 2427
50.6.6 HASH status register (HASH_SR) 2427
50.6.7 HASH context swap registers 2428
50.6.8 HASH register map 2429

51 Memory cipher engine (MCE) 2431
51.1 MCE introduction 2431
51.2 MCE main features 2431
51.3 MCE implementation 2432
51.4 MCE functional description 2432
51.4.1 MCE block diagram 2432
51.4.2 MCE internal signals 2432
51.4.3 MCE programming 2433
51.4.4 MCE reset and clocks 2434
51.4.5 MCE block cipher encryption mode 2434
51.4.6 MCE stream cipher encryption mode 2436
51.4.7 MCE AXI traffic management 2437
51.4.8 MCE encryption disable options 2437
51.4.9 MCE error management 2438
51.5 MCE interrupts 2438
51.6 MCE registers 2439
51.6.1 MCE configuration register (MCE_CR) 2439
51.6.2 MCE status register (MCE_SR) 2440
51.6.3 MCE illegal access status register (MCE_IASR) 2441
51.6.4 MCE illegal access clear register (MCE_IACR) 2441
51.6.5 MCE illegal access interrupt enable register (MCE_IAIER) 2441

51.6.6 MCE illegal address register (MCE_IADDR) 2442
51.6.7 MCE region x configuration register (MCE_REGCRx) 2442
51.6.8 MCE start address for region x register (MCE_SADDRx) 2444
51.6.9 MCE end address for region x register (MCE_EADDRx) 2444
51.6.10 MCE master key x (MCE_MKEYRx) 2445
51.6.11 MCE fast master key x (MCE_FMKEYRx) 2445
51.6.12 MCE cipher context z configuration register (MCE_CCzCFGR) 2446
51.6.13 MCE cipher context z nonce register 0 (MCE_CCzNR0) 2447
51.6.14 MCE cipher context z nonce register 1 (MCE_CCzNR1) 2448
51.6.15 MCE cipher context z key register 0 (MCE_CCzKEYR0) 2448
51.6.16 MCE cipher context z key register 1 (MCE_CCzKEYR1) 2449
51.6.17 MCE cipher context z key register 2 (MCE_CCzKEYR2) 2449
51.6.18 MCE cipher context z key register 3 (MCE_CCzKEYR3) 2449
51.6.19 MCE register map 2450

52 Public key accelerator (PKA) 2452
52.1 PKA introduction 2452
52.2 PKA main features 2452
52.3 PKA functional description 2453
52.3.1 PKA block diagram 2453
52.3.2 PKA internal signals 2453
52.3.3 PKA reset and clocks 2453
52.3.4 PKA public key acceleration 2454
52.3.5 Typical applications for PKA 2456
52.3.6 PKA procedure to perform an operation 2458
52.3.7 PKA error management 2459
52.4 PKA operating modes 2460
52.4.1 Introduction 2460
52.4.2 Montgomery parameter computation 2461
52.4.3 Modular addition 2461
52.4.4 Modular subtraction 2462
52.4.5 Modular and Montgomery multiplication 2462
52.4.6 Modular exponentiation 2463
52.4.7 Modular inversion 2465
52.4.8 Modular reduction 2465

52.4.9 Arithmetic addition . . . 2466
52.4.10 Arithmetic subtraction . . . 2466
52.4.11 Arithmetic multiplication . . . 2467
52.4.12 Arithmetic comparison . . . 2467
52.4.13 RSA CRT exponentiation . . . 2467
52.4.14 Point on elliptic curve Fp check . . . 2468
52.4.15 ECC Fp scalar multiplication . . . 2469
52.4.16 ECDSA sign . . . 2470
52.4.17 ECDSA verification . . . 2472
52.4.18 ECC complete addition . . . 2473
52.4.19 ECC double base ladder . . . 2473
52.4.20 ECC projective to affine . . . 2474
52.5 Example of configurations and processing times . . . 2475
52.5.1 Supported elliptic curves . . . 2475
52.5.2 Computation times . . . 2477
52.6 PKA interrupts . . . 2479
52.7 PKA registers . . . 2480
52.7.1 PKA control register (PKA_CR) . . . 2480
52.7.2 PKA status register (PKA_SR) . . . 2482
52.7.3 PKA clear flag register (PKA_CLRFR) . . . 2483
52.7.4 PKA RAM . . . 2483
52.7.5 PKA register map . . . 2484

53 Advanced-control timers (TIM1/TIM8) . . . 2485
53.1 TIM1/TIM8 introduction . . . 2485
53.2 TIM1/TIM8 main features . . . 2485
53.3 TIM1/TIM8 functional description . . . 2486
53.3.1 Block diagram . . . 2486
53.3.2 TIM1/TIM8 pins and internal signals . . . 2487
53.3.3 Time-base unit . . . 2491
53.3.4 Counter modes . . . 2493
53.3.5 Repetition counter . . . 2505
53.3.6 External trigger input . . . 2506
53.3.7 Clock selection . . . 2507
53.3.8 Capture/compare channels . . . 2511
53.3.9 Input capture mode . . . 2513

53.3.10 PWM input mode . . . 2514
53.3.11 Forced output mode . . . 2515
53.3.12 Output compare mode . . . 2516
53.3.13 PWM mode . . . 2517
53.3.14 Asymmetric PWM mode . . . 2525
53.3.15 Combined PWM mode . . . 2526
53.3.16 Combined 3-phase PWM mode . . . 2527
53.3.17 Complementary outputs and dead-time insertion . . . 2528
53.3.18 Using the break function . . . 2531
53.3.19 Bidirectional break inputs . . . 2537
53.3.20 Clearing the tim_ocxref signal on an external event . . . 2538
53.3.21 6-step PWM generation . . . 2539
53.3.22 One-pulse mode . . . 2540
53.3.23 Retriggerable One-pulse mode . . . 2542
53.3.24 Pulse on compare mode . . . 2543
53.3.25 Encoder interface mode . . . 2545
53.3.26 Direction bit output . . . 2562
53.3.27 UIF bit remapping . . . 2563
53.3.28 Timer input XOR function . . . 2563
53.3.29 Interfacing with Hall sensors . . . 2563
53.3.30 Timer synchronization . . . 2565
53.3.31 ADC triggers . . . 2570
53.3.32 ADC synchronization . . . 2570
53.3.33 DMA burst mode . . . 2571
53.3.34 TIM1/TIM8 DMA requests . . . 2572
53.3.35 Debug mode . . . 2572
53.4 TIM1/TIM8 low-power modes . . . 2573
53.5 TIM1/TIM8 interrupts . . . 2573
53.6 TIM1/TIM8 registers . . . 2574
53.6.1 TIMx control register 1 (TIMx_CR1)(x = 1, 8) . . . 2574
53.6.2 TIMx control register 2 (TIMx_CR2)(x = 1, 8) . . . 2575
53.6.3 TIMx slave mode control register (TIMx_SMCR)(x = 1, 8) . . . 2579
53.6.4 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 1, 8) . . . 2583
53.6.5 TIMx status register (TIMx_SR)(x = 1, 8) . . . 2584
53.6.6 TIMx event generation register (TIMx_EGR)(x = 1, 8) . . . 2587
53.6.7 TIMx capture/compare mode register 1 (TIMx_CCMR1) (x = 1, 8) . . . 2588

53.6.8 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 1, 8) 2590
53.6.9 TIMx capture/compare mode register 2 (TIMx_CCMR2) (x = 1, 8) 2593
53.6.10 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)(x = 1, 8) 2594
53.6.11 TIMx capture/compare enable register (TIMx_CCER)(x = 1, 8) 2597
53.6.12 TIMx counter (TIMx_CNT)(x = 1, 8) 2601
53.6.13 TIMx prescaler (TIMx_PSC)(x = 1, 8) 2601
53.6.14 TIMx autoreload register (TIMx_ARR)(x = 1, 8) 2602
53.6.15 TIMx repetition counter register (TIMx_RCR)(x = 1, 8) 2602
53.6.16 TIMx capture/compare register 1 (TIMx_CCR1)(x = 1, 8) 2603
53.6.17 TIMx capture/compare register 2 (TIMx_CCR2)(x = 1, 8) 2603
53.6.18 TIMx capture/compare register 3 (TIMx_CCR3)(x = 1, 8) 2604
53.6.19 TIMx capture/compare register 4 (TIMx_CCR4)(x = 1, 8) 2605
53.6.20 TIMx break and dead-time register (TIMx_BDTR)(x = 1, 8) 2606
53.6.21 TIMx capture/compare register 5 (TIMx_CCR5)(x = 1, 8) 2610
53.6.22 TIMx capture/compare register 6 (TIMx_CCR6)(x = 1, 8) 2611
53.6.23 TIMx capture/compare mode register 3 (TIMx_CCMR3) (x = 1, 8) 2612
53.6.24 TIMx timer deadtime register 2 (TIMx_DTR2)(x = 1, 8) 2613
53.6.25 TIMx timer encoder control register (TIMx_ECR)(x = 1, 8) 2614
53.6.26 TIMx timer input selection register (TIMx_TISEL)(x = 1, 8) 2615
53.6.27 TIMx alternate function option register 1 (TIMx_AF1)(x = 1, 8) 2616
53.6.28 TIMx alternate function register 2 (TIMx_AF2)(x = 1, 8) 2619
53.6.29 TIMx DMA control register (TIMx_DCR)(x = 1, 8) 2621
53.6.30 TIMx DMA address for full transfer (TIMx_DMAR)(x = 1, 8) 2623
53.6.31 TIMx register map 2623

54 General-purpose timers (TIM2/TIM3/TIM4/TIM5) 2626

54.1 TIM2/TIM3/TIM4/TIM5 introduction 2626
54.2 TIM2/TIM3/TIM4/TIM5 main features 2626
54.3 TIM2/TIM3/TIM4/TIM5 implementation 2627
54.4 TIM2/TIM3/TIM4/TIM5 functional description 2628
54.4.1 Block diagram 2628
54.4.2 TIM2/TIM3/TIM4/TIM5 pins and internal signals 2629
54.4.3 Time-base unit 2632
54.4.4 Counter modes 2634

54.4.5 Clock selection . . . 2646
54.4.6 Capture/compare channels . . . 2650
54.4.7 Input capture mode . . . 2652
54.4.8 PWM input mode . . . 2653
54.4.9 Forced output mode . . . 2654
54.4.10 Output compare mode . . . 2654
54.4.11 PWM mode . . . 2656
54.4.12 Asymmetric PWM mode . . . 2664
54.4.13 Combined PWM mode . . . 2665
54.4.14 Clearing the tim_ocxref signal on an external event . . . 2666
54.4.15 One-pulse mode . . . 2668
54.4.16 Retriggerable one-pulse mode . . . 2669
54.4.17 Pulse on compare mode . . . 2670
54.4.18 Encoder interface mode . . . 2672
54.4.19 Direction bit output . . . 2690
54.4.20 UIF bit remapping . . . 2691
54.4.21 Timer input XOR function . . . 2691
54.4.22 Timers and external trigger synchronization . . . 2691
54.4.23 Timer synchronization . . . 2695
54.4.24 ADC triggers . . . 2700
54.4.25 ADC synchronization . . . 2701
54.4.26 DMA burst mode . . . 2702
54.4.27 TIM2/TIM3/TIM4/TIM5 DMA requests . . . 2703
54.4.28 Debug mode . . . 2703
54.4.29 TIM2/TIM3/TIM4/TIM5 low-power modes . . . 2703
54.4.30 TIM2/TIM3/TIM4/TIM5 interrupts . . . 2704
54.5 TIM2/TIM3/TIM4/TIM5 registers . . . 2705
54.5.1 TIMx control register 1 (TIMx_CR1)(x = 2 to 5) . . . 2705
54.5.2 TIMx control register 2 (TIMx_CR2)(x = 2 to 5) . . . 2706
54.5.3 TIMx slave mode control register (TIMx_SMCR)(x = 2 to 5) . . . 2708
54.5.4 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 2 to 5) . . . 2712
54.5.5 TIMx status register (TIMx_SR)(x = 2 to 5) . . . 2713
54.5.6 TIMx event generation register (TIMx_EGR)(x = 2 to 5) . . . 2715
54.5.7 TIMx capture/compare mode register 1 (TIMx_CCMR1)(x = 2 to 5) . 2716
54.5.8 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 2 to 5) . . . 2718
54.5.9 TIMx capture/compare mode register 2 (TIMx_CCMR2)(x = 2 to 5) . 2720

54.5.10 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)(x = 2 to 5) . . . 2721
54.5.11 TIMx capture/compare enable register (TIMx_CCER)(x = 2 to 5) . . . 2724
54.5.12 TIMx counter (TIMx_CNT)(x = 2 to 5) . . . 2726
54.5.13 TIMx prescaler (TIMx_PSC)(x = 2 to 5) . . . 2726
54.5.14 TIMx autoreload register (TIMx_ARR)(x = 2 to 5) . . . 2727
54.5.15 TIMx capture/compare register 1 (TIMx_CCR1)(x = 2 to 5) . . . 2727
54.5.16 TIMx capture/compare register 2 (TIMx_CCR2)(x = 2 to 5) . . . 2728
54.5.17 TIMx capture/compare register 3 (TIMx_CCR3)(x = 2 to 5) . . . 2729
54.5.18 TIMx capture/compare register 4 (TIMx_CCR4)(x = 2 to 5) . . . 2730
54.5.19 TIMx timer encoder control register (TIMx_ECR)(x = 2 to 5) . . . 2731
54.5.20 TIMx timer input selection register (TIMx_TISEL)(x = 2 to 5) . . . 2732
54.5.21 TIMx alternate function register 1 (TIMx_AF1)(x = 2 to 5) . . . 2733
54.5.22 TIMx alternate function register 2 (TIMx_AF2)(x = 2 to 5) . . . 2734
54.5.23 TIMx DMA control register (TIMx_DCR)(x = 2 to 5) . . . 2735
54.5.24 TIMx DMA address for full transfer (TIMx_DMAR)(x = 2 to 5) . . . 2736
54.5.25 TIMx register map . . . 2737

55 Basic timers (TIM6/TIM7/TIM18) . . . 2740

55.1 TIM6/TIM7/TIM18 introduction . . . 2740
55.2 TIM6/TIM7/TIM18 main features . . . 2740
55.3 TIM6/TIM7/TIM18 functional description . . . 2740
55.3.1 TIM6/TIM7/TIM18 block diagram . . . 2740
55.3.2 TIM6/TIM7/TIM18 internal signals . . . 2741
55.3.3 TIM6/TIM7/TIM18 clocks . . . 2741
55.3.4 Time-base unit . . . 2742
55.3.5 Counting mode . . . 2744
55.3.6 UIF bit remapping . . . 2750
55.3.7 ADC triggers . . . 2751
55.3.8 ADC synchronization . . . 2751
55.3.9 TIM6/TIM7/TIM18 DMA requests . . . 2752
55.3.10 Debug mode . . . 2752
55.3.11 TIM6/TIM7/TIM18 low-power modes . . . 2752
55.3.12 TIM6/TIM7/TIM18 interrupts . . . 2752
55.4 TIM6/TIM7/TIM18 registers . . . 2753
55.4.1 TIMx control register 1 (TIMx_CR1)(x = 6, 7, 18) . . . 2753
55.4.2 TIMx control register 2 (TIMx_CR2)(x = 6, 7, 18) . . . 2755

55.4.3 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 6, 7, 18) 2755
55.4.4 TIMx status register (TIMx_SR)(x = 6, 7, 18) 2756
55.4.5 TIMx event generation register (TIMx_EGR)(x = 6, 7, 18) 2756
55.4.6 TIMx counter (TIMx_CNT)(x = 6, 7, 18) 2757
55.4.7 TIMx prescaler (TIMx_PSC)(x = 6, 7, 18) 2757
55.4.8 TIMx autoreload register (TIMx_ARR)(x = 6, 7, 18) 2758
55.4.9 TIMx register map 2759

56 General-purpose timers (TIM9/TIM10/TIM11/TIM12/TIM13/TIM14) . 2760

56.1 TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 introduction 2760
56.2 TIM9/TIM12 main features 2760
56.3 TIM10/TIM11/TIM13/TIM14 main features 2761
56.4 TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 functional description 2762
56.4.1 Block diagram 2762
56.4.2 TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 pins and internal signals . . 2763
56.4.3 Time-base unit 2765
56.4.4 Counter modes 2767
56.4.5 Clock selection 2770
56.4.6 Capture/compare channels 2772
56.4.7 Input capture mode 2774
56.4.8 PWM input mode (TIM9/TIM12 only) 2775
56.4.9 Forced output mode 2776
56.4.10 Output compare mode 2777
56.4.11 PWM mode 2778
56.4.12 Combined PWM mode (TIM9/TIM12 only) 2783
56.4.13 One-pulse mode 2784
56.4.14 Retriggerable one pulse mode (TIM9/TIM12 only) 2786
56.4.15 UIF bit remapping 2787
56.4.16 Timer input XOR function 2787
56.4.17 TIM9/TIM12 external trigger synchronization 2787
56.4.18 Slave mode – combined reset + trigger mode 2790
56.4.19 Slave mode – combined reset + gated mode 2790
56.4.20 Timer synchronization (TIM9/TIM12 only) 2790
56.4.21 Using timer output as trigger for other timers (TIM10/TIM11/TIM13/TIM14 only) 2790
56.4.22 ADC triggers (TIM9/TIM12 only) 2790
56.4.23 ADC synchronization (TIM9/TIM12 only) 2791

56.4.24 Debug mode 2792
56.5 TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 low-power modes 2792
56.6 TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 interrupts 2792
56.7 TIM9/TIM12 registers 2792
56.7.1 TIMx control register 1 (TIMx_CR1)(x = 9, 12) 2793
56.7.2 TIM12 control register 2 (TIMx_CR2)(x = 9, 12) 2794
56.7.3 TIMx slave mode control register (TIMx_SMCR)(x = 9, 12) 2795
56.7.4 TIMx interrupt enable register (TIMx_DIER)(x = 9, 12) 2797
56.7.5 TIMx status register (TIMx_SR)(x = 9, 12) 2798
56.7.6 TIMx event generation register (TIMx_EGR)(x = 9, 12) 2799
56.7.7 TIMx capture/compare mode register 1 (TIMx_CCMR1)(x = 9, 12) . 2800
56.7.8 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 9, 12) 2801
56.7.9 TIMx capture/compare enable register (TIMx_CCER)(x = 9, 12) . . . 2804
56.7.10 TIMx counter (TIMx_CNT)(x = 9, 12) 2805
56.7.11 TIMx prescaler (TIMx_PSC)(x = 9, 12) 2806
56.7.12 TIMx autoreload register (TIMx_ARR)(x = 9, 12) 2806
56.7.13 TIMx capture/compare register 1 (TIMx_CCR1)(x = 9, 12) 2807
56.7.14 TIMx capture/compare register 2 (TIMx_CCR2)(x = 9, 12) 2807
56.7.15 TIMx timer input selection register (TIMx_TISEL)(x = 9, 12) 2808
56.7.16 TIM9/TIM12 register map 2809
56.8 TIM10/TIM11/TIM13/TIM14 registers 2811
56.8.1 TIMx control register 1 (TIMx_CR1)(x = 10, 11, 13, 14) 2811
56.8.2 TIMx interrupt enable register (TIMx_DIER)(x = 10, 11, 13, 14) 2812
56.8.3 TIMx status register (TIMx_SR)(x = 10, 11, 13, 14) 2812
56.8.4 TIMx event generation register (TIMx_EGR)(x = 10, 11, 13, 14) 2813
56.8.5 TIMx capture/compare mode register 1 (TIMx_CCMR1)(x = 10, 11, 13, 14) 2814
56.8.6 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 10, 11, 13, 14) 2815
56.8.7 TIMx capture/compare enable register (TIMx_CCER)(x = 10, 11, 13, 14) 2817
56.8.8 TIMx counter (TIMx_CNT)(x = 10, 11, 13, 14) 2818
56.8.9 TIMx prescaler (TIMx_PSC)(x = 10, 11, 13, 14) 2819
56.8.10 TIMx autoreload register (TIMx_ARR)(x = 10, 11, 13, 14) 2819
56.8.11 TIMx capture/compare register 1 (TIMx_CCR1)(x = 10, 11, 13, 14) . 2820
56.8.12 TIMx timer input selection register (TIMx_TISEL)(x = 10, 11, 13, 14) 2820
56.8.13 TIM10/TIM11/TIM13/TIM14 register map 2821

57 General purpose timers (TIM15/TIM16/TIM17) 2823

57.1 TIM15/TIM16/TIM17 introduction 2823

57.2 TIM15 main features 2823

57.3 TIM16/TIM17 main features 2824

57.4 TIM15/TIM16/TIM17 functional description 2825

57.4.1 Block diagram 2825

57.4.2 TIM15/TIM16/TIM17 pins and internal signals 2826

57.4.3 Time-base unit 2829

57.4.4 Counter modes 2831

57.4.5 Repetition counter 2835

57.4.6 Clock selection 2836

57.4.7 Capture/compare channels 2838

57.4.8 Input capture mode 2840

57.4.9 PWM input mode (only for TIM15) 2842

57.4.10 Forced output mode 2843

57.4.11 Output compare mode 2843

57.4.12 PWM mode 2845

57.4.13 Combined PWM mode (TIM15 only) 2850

57.4.14 Complementary outputs and dead-time insertion 2851

57.4.15 Using the break function 2854

57.4.16 Bidirectional break input 2858

57.4.17 Clearing the tim_ocxref signal on an external event 2859

57.4.18 6-step PWM generation 2860

57.4.19 One-pulse mode 2862

57.4.20 Retriggerable one pulse mode (TIM15 only) 2863

57.4.21 UIF bit remapping 2864

57.4.22 Timer input XOR function (TIM15 only) 2864

57.4.23 External trigger synchronization (TIM15 only) 2864

57.4.24 Slave mode – combined reset + trigger mode (TIM15 only) 2867

57.4.25 Slave mode – combined reset + gated mode (TIM15 only) 2867

57.4.26 Timer synchronization (TIM15 only) 2868

57.4.27 Using timer output as trigger for other timers (TIM16/TIM17 only) . . 2868

57.4.28 ADC triggers (TIM15 only) 2868

57.4.29 ADC synchronization (TIM15 only) 2869

57.4.30 DMA burst mode 2869

57.4.31 TIM15/TIM16/TIM17 DMA requests 2870

57.4.32 Debug mode 2871
57.5 TIM15/TIM16/TIM17 low-power modes 2871
57.6 TIM15/TIM16/TIM17 interrupts 2871
57.7 TIM15 registers 2872
57.7.1 TIM15 control register 1 (TIM15_CR1) 2872
57.7.2 TIM15 control register 2 (TIM15_CR2) 2873
57.7.3 TIM15 slave mode control register (TIM15_SMCR) 2875
57.7.4 TIM15 DMA/interrupt enable register (TIM15_DIER) 2877
57.7.5 TIM15 status register (TIM15_SR) 2878
57.7.6 TIM15 event generation register (TIM15_EGR) 2880
57.7.7 TIM15 capture/compare mode register 1 (TIM15_CCMR1) 2881
57.7.8 TIM15 capture/compare mode register 1 [alternate] (TIM15_CCMR1) 2883
57.7.9 TIM15 capture/compare enable register (TIM15_CCER) 2885
57.7.10 TIM15 counter (TIM15_CNT) 2888
57.7.11 TIM15 prescaler (TIM15_PSC) 2888
57.7.12 TIM15 autoreload register (TIM15_ARR) 2889
57.7.13 TIM15 repetition counter register (TIM15_RCR) 2889
57.7.14 TIM15 capture/compare register 1 (TIM15_CCR1) 2890
57.7.15 TIM15 capture/compare register 2 (TIM15_CCR2) 2891
57.7.16 TIM15 break and dead-time register (TIM15_BDTR) 2891
57.7.17 TIM15 timer deadtime register 2 (TIM15_DTR2) 2894
57.7.18 TIM15 input selection register (TIM15_TISEL) 2895
57.7.19 TIM15 alternate function register 1 (TIM15_AF1) 2896
57.7.20 TIM15 alternate function register 2 (TIM15_AF2) 2898
57.7.21 TIM15 DMA control register (TIM15_DCR) 2899
57.7.22 TIM15 DMA address for full transfer (TIM15_DMAR) 2900
57.7.23 TIM15 register map 2900
57.8 TIM16/TIM17 registers 2903
57.8.1 TIMx control register 1 (TIMx_CR1)(x = 16 to 17) 2903
57.8.2 TIMx control register 2 (TIMx_CR2)(x = 16 to 17) 2904
57.8.3 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 16 to 17) 2905
57.8.4 TIMx status register (TIMx_SR)(x = 16 to 17) 2906
57.8.5 TIMx event generation register (TIMx_EGR)(x = 16 to 17) 2907
57.8.6 TIMx capture/compare mode register 1 (TIMx_CCMR1) (x = 16 to 17) 2908

57.8.7 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 16 to 17) . . . 2909
57.8.8 TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17) . 2911
57.8.9 TIMx counter (TIMx_CNT)(x = 16 to 17) . . . 2914
57.8.10 TIMx prescaler (TIMx_PSC)(x = 16 to 17) . . . 2914
57.8.11 TIMx auto-reautoreload register (TIMx_ARR)(x = 16 to 17) . . . 2915
57.8.12 TIMx repetition counter register (TIMx_RCR)(x = 16 to 17) . . . 2915
57.8.13 TIMx capture/compare register 1 (TIMx_CCR1)(x = 16 to 17) . . . 2916
57.8.14 TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17) . . . 2917
57.8.15 TIMx timer deadtime register 2 (TIMx_DTR2)(x = 16 to 17) . . . 2920
57.8.16 TIMx input selection register (TIMx_TISEL)(x = 16 to 17) . . . 2921
57.8.17 TIMx alternate function register 1 (TIMx_AF1)(x = 16 to 17) . . . 2921
57.8.18 TIMx alternate function register 2 (TIMx_AF2)(x = 16 to 17) . . . 2924
57.8.19 TIMx DMA control register (TIMx_DCR)(x = 16 to 17) . . . 2924
57.8.20 TIM16/TIM17 DMA address for full transfer (TIMx_DMAR)(x = 16 to 17) . . . 2925
57.8.21 TIM16/TIM17 register map . . . 2927

58 Low-power timer (LPTIM) . . . 2929

58.1 LPTIM introduction . . . 2929
58.2 LPTIM main features . . . 2929
58.3 LPTIM implementation . . . 2930
58.4 LPTIM functional description . . . 2931
58.4.1 LPTIM block diagram . . . 2931
58.4.2 LPTIM pins and internal signals . . . 2932
58.4.3 LPTIM input and trigger mapping . . . 2934
58.4.4 LPTIM reset and clocks . . . 2935
58.4.5 Glitch filter . . . 2936
58.4.6 Prescaler . . . 2937
58.4.7 Trigger multiplexer . . . 2937
58.4.8 Operating mode . . . 2938
58.4.9 Timeout function . . . 2940
58.4.10 Waveform generation . . . 2940
58.4.11 Register update . . . 2941
58.4.12 Counter mode . . . 2942
58.4.13 Timer enable . . . 2942
58.4.14 Timer counter reset . . . 2943

58.4.15 Encoder mode . . . 2943
58.4.16 Repetition counter . . . 2945
58.4.17 Capture/compare channels . . . 2946
58.4.18 Input capture mode . . . 2947
58.4.19 PWM mode . . . 2949
58.4.20 DMA requests . . . 2951
58.4.21 Debug mode . . . 2952
58.5 LPTIM low-power modes . . . 2952
58.6 LPTIM interrupts . . . 2952
58.7 LPTIM registers . . . 2953
58.7.1 LPTIMx interrupt and status register (LPTIMx_ISR)(x = 4 to 5) . . . 2954
58.7.2 LPTIMx interrupt and status register [alternate] (LPTIMx_ISR) (x = 1 to 3) . . . 2955
58.7.3 LPTIMx interrupt and status register [alternate] (LPTIMx_ISR) (x = 1 to 3) . . . 2957
58.7.4 LPTIMx interrupt clear register (LPTIMx_ICR)(x = 4 to 5) . . . 2959
58.7.5 LPTIMx interrupt clear register [alternate] (LPTIMx_ICR) (x = 1 to 3) . . . 2960
58.7.6 LPTIMx interrupt clear register [alternate] (LPTIMx_ICR) (x = 1 to 3) . . . 2961
58.7.7 LPTIMx interrupt enable register (LPTIMx_DIER)(x = 4 to 5) . . . 2962
58.7.8 LPTIMx interrupt enable register [alternate] (LPTIMx_DIER) (x = 1 to 3) . . . 2964
58.7.9 LPTIMx interrupt enable register [alternate] (LPTIMx_DIER) (x = 1 to 3) . . . 2965
58.7.10 LPTIM configuration register (LPTIM_CFGR) . . . 2967
58.7.11 LPTIM control register (LPTIM_CR) . . . 2970
58.7.12 LPTIM compare register 1 (LPTIM_CCR1) . . . 2971
58.7.13 LPTIM autoreload register (LPTIM_ARR) . . . 2972
58.7.14 LPTIM counter register (LPTIM_CNT) . . . 2972
58.7.15 LPTIM configuration register 2 (LPTIM_CFGR2) . . . 2973
58.7.16 LPTIM repetition register (LPTIM_RCR) . . . 2974
58.7.17 LPTIM capture/compare mode register 1 (LPTIM_CCMR1) . . . 2974
58.7.18 LPTIM compare register 2 (LPTIM_CCR2) . . . 2977
58.7.19 LPTIM register map . . . 2977

59 Independent watchdog (IWDG) . . . 2980

59.1 IWDG introduction . . . 2980

59.2 IWDG main features 2980
59.3 IWDG implementation 2980
59.4 IWDG functional description 2981
59.4.1 IWDG block diagram 2981
59.4.2 IWDG internal signals 2982
59.4.3 Software and hardware watchdog modes 2982
59.4.4 Window option 2983
59.4.5 Debug 2986
59.4.6 Register access protection 2986
59.5 IWDG low power modes 2987
59.6 IWDG interrupts 2987
59.7 IWDG registers 2989
59.7.1 IWDG key register (IWDG_KR) 2990
59.7.2 IWDG prescaler register (IWDG_PR) 2990
59.7.3 IWDG reload register (IWDG_RLR) 2991
59.7.4 IWDG status register (IWDG_SR) 2991
59.7.5 IWDG window register (IWDG_WINR) 2993
59.7.6 IWDG early wake-up interrupt register (IWDG_EWCR) 2993
59.7.7 IWDG interrupt clear register (IWDG_ICR) 2994
59.7.8 IWDG register map 2995

60 System window watchdog (WWDG) 2996
60.1 WWDG introduction 2996
60.2 WWDG main features 2996
60.3 WWDG implementation 2996
60.4 WWDG functional description 2997
60.4.1 WWDG block diagram 2997
60.4.2 WWDG internal signals 2997
60.4.3 Enabling the watchdog 2998
60.4.4 Controlling the down-counter 2998
60.4.5 How to program the watchdog timeout 2998
60.4.6 Debug mode 2999
60.5 WWDG interrupts 3000
60.6 WWDG registers 3000
60.6.1 WWDG control register (WWDG_CR) 3000
60.6.2 WWDG configuration register (WWDG_CFR) 3001

60.6.3 WWDG status register (WWDG_SR) 3002
60.6.4 WWDG register map 3002

61 Real-time clock (RTC) 3003
61.1 RTC introduction 3003
61.2 RTC main features 3003
61.3 RTC functional description 3004
61.3.1 RTC block diagram 3004
61.3.2 RTC pins and internal signals 3006
61.3.3 GPIOs controlled by the RTC and TAMP 3007
61.3.4 RTC secure protection modes 3010
61.3.5 RTC privilege protection modes 3011
61.3.6 Clock and prescalers 3012
61.3.7 Real-time clock and calendar 3013
61.3.8 Calendar ultra-low power mode 3014
61.3.9 Programmable alarms 3014
61.3.10 Periodic auto-wake-up 3014
61.3.11 RTC initialization and configuration 3015
61.3.12 Reading the calendar 3018
61.3.13 Resetting the RTC 3019
61.3.14 RTC synchronization 3020
61.3.15 RTC reference clock detection 3020
61.3.16 RTC smooth digital calibration 3021
61.3.17 Timestamp function 3023
61.3.18 Calibration clock output 3024
61.3.19 Tamper and alarm output 3024
61.4 RTC low-power modes 3025
61.5 RTC interrupts 3025
61.6 RTC registers 3027
61.6.1 RTC time register (RTC_TR) 3027
61.6.2 RTC date register (RTC_DR) 3028
61.6.3 RTC subsecond register (RTC_SSR) 3029
61.6.4 RTC initialization control and status register (RTC_ICSR) 3030
61.6.5 RTC prescaler register (RTC_PRER) 3032
61.6.6 RTC wake-up timer register (RTC_WUTR) 3033
61.6.7 RTC control register (RTC_CR) 3033

61.6.8 RTC privilege mode control register (RTC_PRIVCFGR) 3037
61.6.9 RTC secure configuration register (RTC_SECCFGR) 3039
61.6.10 RTC write protection register (RTC_WPR) 3040
61.6.11 RTC calibration register (RTC_CALR) 3041
61.6.12 RTC shift control register (RTC_SHIFTR) 3042
61.6.13 RTC timestamp time register (RTC_TSTR) 3043
61.6.14 RTC timestamp date register (RTC_TSDR) 3044
61.6.15 RTC timestamp subsecond register (RTC_TSSSR) 3045
61.6.16 RTC alarm A register (RTC_ALRMAR) 3045
61.6.17 RTC alarm A subsecond register (RTC_ALRMASSR) 3047
61.6.18 RTC alarm B register (RTC_ALRMBR) 3048
61.6.19 RTC alarm B subsecond register (RTC_ALRMBSSR) 3049
61.6.20 RTC status register (RTC_SR) 3050
61.6.21 RTC nonsecure masked interrupt status register (RTC_MISR) 3051
61.6.22 RTC secure masked interrupt status register (RTC_SMISR) 3052
61.6.23 RTC status clear register (RTC_SCR) 3053
61.6.24 RTC alarm A binary mode register (RTC_ALRABINR) 3054
61.6.25 RTC alarm B binary mode register (RTC_ALRBBINR) 3055
61.6.26 RTC register map 3056

62 Tamper and backup registers (TAMP) 3058

62.1 TAMP introduction 3058
62.2 TAMP main features 3059
62.3 TAMP functional description 3060
62.3.1 TAMP block diagram 3060
62.3.2 TAMP pins and internal signals 3061
62.3.3 GPIOs controlled by the RTC and TAMP 3063
62.3.4 TAMP register write protection 3063
62.3.5 TAMP secure protection modes 3063
62.3.6 Backup registers protection zones 3064
62.3.7 TAMP privilege protection modes 3064
62.3.8 Boot hardware key (BHK) 3065
62.3.9 Tamper detection 3065
62.3.10 TAMP backup registers and other device secrets erase 3066
62.3.11 Tamper detection configuration and initialization 3068
62.4 TAMP low-power modes 3075
62.5 TAMP interrupts 3075

62.6 TAMP registers . . . 3076
62.6.1 TAMP control register 1 (TAMP_CR1) . . . 3076
62.6.2 TAMP control register 2 (TAMP_CR2) . . . 3077
62.6.3 TAMP control register 3 (TAMP_CR3) . . . 3080
62.6.4 TAMP filter control register (TAMP_FLTCR) . . . 3081
62.6.5 TAMP active tamper control register 1 (TAMP_ATCR1) . . . 3083
62.6.6 TAMP active tamper seed register (TAMP_ATSEEDR) . . . 3085
62.6.7 TAMP active tamper output register (TAMP_ATOR) . . . 3085
62.6.8 TAMP active tamper control register 2 (TAMP_ATCR2) . . . 3086
62.6.9 TAMP secure configuration register (TAMP_SECCFGR) . . . 3089
62.6.10 TAMP privilege configuration register (TAMP_PRIVCFGR) . . . 3090
62.6.11 TAMP interrupt enable register (TAMP_IER) . . . 3091
62.6.12 TAMP status register (TAMP_SR) . . . 3093
62.6.13 TAMP nonsecure masked interrupt status register (TAMP_MISR) . . 3095
62.6.14 TAMP secure masked interrupt status register (TAMP_SMISR) 3097
62.6.15 TAMP status clear register (TAMP_SCR) . . . 3098
62.6.16 TAMP monotonic counter 1 register (TAMP_COUNT1R) . . . 3100
62.6.17 TAMP option register (TAMP_OR) . . . 3100
62.6.18 TAMP resources protection configuration register (TAMP_RPCFGR) 3101
62.6.19 TAMP backup x register (TAMP_BKPxR) . . . 3102
62.6.20 TAMP register map . . . 3103

63 Inter-integrated circuit interface (I2C) . . . 3105
63.1 I2C introduction . . . 3105
63.2 I2C main features . . . 3105
63.3 I2C implementation . . . 3106
63.4 I2C functional description . . . 3106
63.4.1 I2C block diagram . . . 3107
63.4.2 I2C pins and internal signals . . . 3107
63.4.3 I2C clock requirements . . . 3108
63.4.4 I2C mode selection . . . 3108
63.4.5 I2C initialization . . . 3109
63.4.6 I2C reset . . . 3113
63.4.7 I2C data transfer . . . 3114
63.4.8 I2C target mode . . . 3116
63.4.9 I2C controller mode . . . 3125
63.4.10 I2C_TIMINGR register configuration examples . . . 3136

63.4.11 SMBus specific features . . . 3138
63.4.12 SMBus initialization . . . 3140
63.4.13 SMBus I2C_TIMEOUTR register configuration examples . . . 3142
63.4.14 SMBus target mode . . . 3143
63.4.15 SMBus controller mode . . . 3146
63.4.16 Wake-up from Stop mode on address match . . . 3149
63.4.17 Error conditions . . . 3150
63.5 I2C in low-power modes . . . 3152
63.6 I2C interrupts . . . 3152
63.7 I2C DMA requests . . . 3153
63.7.1 Transmission using DMA . . . 3153
63.7.2 Reception using DMA . . . 3153
63.8 I2C debug modes . . . 3154
63.9 I2C registers . . . 3154
63.9.1 I2C control register 1 (I2C_CR1) . . . 3154
63.9.2 I2C control register 2 (I2C_CR2) . . . 3157
63.9.3 I2C own address 1 register (I2C_OAR1) . . . 3159
63.9.4 I2C own address 2 register (I2C_OAR2) . . . 3159
63.9.5 I2C timing register (I2C_TIMINGR) . . . 3160
63.9.6 I2C timeout register (I2C_TIMEOUTR) . . . 3161
63.9.7 I2C interrupt and status register (I2C_ISR) . . . 3162
63.9.8 I2C interrupt clear register (I2C_ICR) . . . 3165
63.9.9 I2C PEC register (I2C_PECR) . . . 3166
63.9.10 I2C receive data register (I2C_RXDR) . . . 3166
63.9.11 I2C transmit data register (I2C_TXDR) . . . 3167
63.9.12 I2C register map . . . 3168

64 Improved inter-integrated circuit (I3C) . . . 3169
64.1 I3C introduction . . . 3169
64.2 I3C main features . . . 3169
64.3 I3C implementation . . . 3171
64.3.1 I3C instantiation . . . 3171
64.3.2 I3C wake-up from low-power mode(s) . . . 3171
64.3.3 I3C FIFOs . . . 3171
64.3.4 I3C triggers . . . 3171
64.3.5 I3C interrupt(s) . . . 3171

64.3.6 I3C MIPI® support . . . 3173
64.4 I3C block diagram . . . 3174
64.5 I3C pins and internal signals . . . 3174
64.6 I3C reset and clocks . . . 3175
64.6.1 I3C reset . . . 3175
64.6.2 I3C clocks and requirements . . . 3175
64.7 I3C peripheral state and programming . . . 3177
64.7.1 I3C peripheral state . . . 3177
64.7.2 I3C controller state and programming sequence . . . 3178
64.7.3 I3C target state and programming sequence . . . 3182
64.8 I3C registers and programming . . . 3186
64.8.1 I3C register set, as controller/target . . . 3186
64.8.2 I3C registers and fields use versus peripheral state, as controller . . . 3187
64.8.3 I3C registers and fields usage versus peripheral state, as target . . . 3190
64.9 I3C bus transfers and programming . . . 3192
64.9.1 I3C command set (CCCs), as controller/target . . . 3192
64.9.2 I3C broadcast/direct CCC transfer (except ENTDAA, RSTACT), as controller . . . 3196
64.9.3 I3C broadcast ENTDAA CCC transfer, as controller . . . 3198
64.9.4 I3C broadcast/direct RSTACT CCC transfer, as controller . . . 3198
64.9.5 I3C broadcast/direct CCC transfer (except ENTDAA, DEFTGTS, DEFGRPA), as target . . . 3200
64.9.6 I3C broadcast ENTDAA CCC transfer, as target . . . 3202
64.9.7 I3C broadcast DEFTGTS CCC transfer, as target . . . 3203
64.9.8 I3C broadcast DEFGRPA CCC transfer, as target . . . 3204
64.9.9 I3C direct GETSTATUS CCC response, as target . . . 3205
64.9.10 I3C private read/write transfer, as controller . . . 3206
64.9.11 I3C private read/write transfer, as target . . . 3207
64.9.12 Legacy I2C read/write transfer, as controller . . . 3208
64.9.13 I3C IBI transfer, as controller/target . . . 3209
64.9.14 I3C hot-join request transfer, as controller/target . . . 3211
64.9.15 I3C controller-role request transfer, as controller/target . . . 3212
64.10 I3C FIFOs management, as controller . . . 3213
64.10.1 C-FIFO management, as controller . . . 3213
64.10.2 TX-FIFO management, as controller . . . 3214
64.10.3 RX-FIFO management, as controller . . . 3217
64.10.4 S-FIFO management, as controller . . . 3219

64.11 I3C FIFOs management, as target . . . 3221
64.11.1 RX-FIFO management, as target . . . 3221
64.11.2 TX-FIFO management, as target . . . 3222
64.12 I3C error management . . . 3225
64.12.1 Controller error management . . . 3225
64.12.2 Target error management . . . 3227
64.13 I3C wake-up from low-power mode(s) . . . 3228
64.13.1 Wake-up from Stop . . . 3228
64.14 I3C in low-power modes . . . 3231
64.15 I3C interrupts . . . 3232
64.16 I3C registers . . . 3233
64.16.1 I3C message control register (I3C_CR) . . . 3233
64.16.2 I3C message control register [alternate] (I3C_CR) . . . 3235
64.16.3 I3C configuration register (I3C_CFGR) . . . 3237
64.16.4 I3C receive data byte register (I3C_RDR) . . . 3242
64.16.5 I3C receive data word register (I3C_RDWR) . . . 3242
64.16.6 I3C transmit data byte register (I3C_TDR) . . . 3243
64.16.7 I3C transmit data word register (I3C_TDWR) . . . 3244
64.16.8 I3C IBI payload data register (I3C_IBIDR) . . . 3246
64.16.9 I3C target transmit configuration register (I3C_TGTTDR) . . . 3247
64.16.10 I3C status register (I3C_SR) . . . 3248
64.16.11 I3C status error register (I3C_SER) . . . 3249
64.16.12 I3C received message register (I3C_RMR) . . . 3251
64.16.13 I3C event register (I3C_EVR) . . . 3252
64.16.14 I3C interrupt enable register (I3C_IER) . . . 3256
64.16.15 I3C clear event register (I3C_CEVR) . . . 3258
64.16.16 I3C own device characteristics register (I3C_DEVR0) . . . 3260
64.16.17 I3C device x characteristics register (I3C_DEVRx) . . . 3262
64.16.18 I3C maximum read length register (I3C_MAXRLR) . . . 3264
64.16.19 I3C maximum write length register (I3C_MAXWLR) . . . 3265
64.16.20 I3C timing register 0 (I3C_TIMINGR0) . . . 3266
64.16.21 I3C timing register 1 (I3C_TIMINGR1) . . . 3267
64.16.22 I3C timing register 2 (I3C_TIMINGR2) . . . 3269
64.16.23 I3C bus characteristics register (I3C_BCR) . . . 3270
64.16.24 I3C device characteristics register (I3C_DCR) . . . 3271
64.16.25 I3C get capability register (I3C_GETCAPR) . . . 3272

64.16.26 I3C controller-role capability register (I3C_CRCAPR) 3273
64.16.27 I3C get max data speed register (I3C_GETMXDSR) 3274
64.16.28 I3C extended provisioned ID register (I3C_EPIDR) 3276
64.16.29 I3C register map 3277

65 Universal synchronous/asynchronous receiver transmitter (USART/UART) 3280

65.1 USART introduction 3280
65.2 USART main features 3280
65.3 USART extended features 3281
65.4 USART implementation 3281
65.5 USART functional description 3283
65.5.1 USART block diagram 3283
65.5.2 USART pins and internal signals 3283
65.5.3 USART clocks 3285
65.5.4 USART character description 3285
65.5.5 USART FIFOs and thresholds 3288
65.5.6 USART transmitter 3288
65.5.7 USART receiver 3291
65.5.8 USART baud rate generation 3298
65.5.9 Tolerance of the USART receiver to clock deviation 3300
65.5.10 USART auto baud rate detection 3301
65.5.11 USART multiprocessor communication 3303
65.5.12 USART Modbus communication 3305
65.5.13 USART parity control 3306
65.5.14 USART LIN (local interconnection network) mode 3307
65.5.15 USART synchronous mode 3309
65.5.16 USART single-wire half-duplex communication 3313
65.5.17 USART receiver timeout 3313
65.5.18 USART smartcard mode 3314
65.5.19 USART IrDA SIR ENDEC block 3318
65.5.20 Continuous communication using USART and DMA 3321
65.5.21 RS232 hardware flow control and RS485 driver enable 3323
65.5.22 USART low-power management 3326
65.6 USART in low-power modes 3329
65.7 USART interrupts 3329

65.8 USART registers 3332
65.8.1 USART control register 1 (USART_CR1) 3332
65.8.2 USART control register 1 [alternate] (USART_CR1) 3336
65.8.3 USART control register 2 (USART_CR2) 3339
65.8.4 USART control register 3 (USART_CR3) 3343
65.8.5 USART control register 3 [alternate] (USART_CR3) 3347
65.8.6 USART baud rate register (USART_BRR) 3350
65.8.7 USART guard time and prescaler register (USART_GTPR) 3351
65.8.8 USART receiver timeout register (USART_RTOR) 3352
65.8.9 USART request register (USART_RQR) 3353
65.8.10 USART interrupt and status register (USART_ISR) 3354
65.8.11 USART interrupt and status register [alternate] (USART_ISR) 3360
65.8.12 USART interrupt flag clear register (USART_ICR) 3365
65.8.13 USART receive data register (USART_RDR) 3366
65.8.14 USART transmit data register (USART_TDR) 3367
65.8.15 USART prescaler register (USART_PRESC) 3367
65.8.16 USART register map 3368

66 Low-power universal asynchronous receiver transmitter (LPUART) 3370

66.1 LPUART introduction 3370
66.2 LPUART main features 3370
66.3 LPUART implementation 3371
66.4 LPUART functional description 3373
66.4.1 LPUART block diagram 3373
66.4.2 LPUART pins and internal signals 3374
66.4.3 LPUART clocks 3375
66.4.4 LPUART character description 3375
66.4.5 LPUART FIFOs and thresholds 3377
66.4.6 LPUART transmitter 3377
66.4.7 LPUART receiver 3381
66.4.8 LPUART baud rate generation 3385
66.4.9 Tolerance of the LPUART receiver to clock deviation 3386
66.4.10 LPUART multiprocessor communication 3387
66.4.11 LPUART parity control 3389
66.4.12 LPUART single-wire half-duplex communication 3390
66.4.13 Continuous communication using DMA and LPUART 3390

66.4.14 RS232 hardware flow control and RS485 driver enable . . . 3393
66.4.15 LPUART low-power management . . . 3395
66.5 LPUART in low-power modes . . . 3398
66.6 LPUART interrupts . . . 3399
66.7 LPUART registers . . . 3400
66.7.1 LPUART control register 1 (LPUART_CR1) . . . 3400
66.7.2 LPUART control register 1 [alternate] (LPUART_CR1) . . . 3403
66.7.3 LPUART control register 2 (LPUART_CR2) . . . 3406
66.7.4 LPUART control register 3 (LPUART_CR3) . . . 3408
66.7.5 LPUART control register 3 [alternate] (LPUART_CR3) . . . 3411
66.7.6 LPUART baud rate register (LPUART_BRR) . . . 3413
66.7.7 LPUART request register (LPUART_RQR) . . . 3413
66.7.8 LPUART interrupt and status register (LPUART_ISR) . . . 3414
66.7.9 LPUART interrupt and status register [alternate] (LPUART_ISR) . . . 3419
66.7.10 LPUART interrupt flag clear register (LPUART_ICR) . . . 3422
66.7.11 LPUART receive data register (LPUART_RDR) . . . 3423
66.7.12 LPUART transmit data register (LPUART_TDR) . . . 3423
66.7.13 LPUART prescaler register (LPUART_PRESC) . . . 3424
66.7.14 LPUART register map . . . 3425

67 Serial peripheral interface (SPI) . . . 3427
67.1 SPI introduction . . . 3427
67.2 SPI main features . . . 3427
67.3 SPI implementation . . . 3428
67.4 SPI functional description . . . 3429
67.4.1 SPI block diagram . . . 3429
67.4.2 SPI pins and internal signals . . . 3430
67.4.3 SPI communication general aspects . . . 3431
67.4.4 Communications between one master and one slave . . . 3431
67.4.5 Standard multislave communication . . . 3434
67.4.6 Multimaster communication . . . 3437
67.4.7 Slave select (NSS pin) management . . . 3438
67.4.8 Ready pin (RDY) management . . . 3442
67.4.9 Communication formats . . . 3442
67.4.10 Configuring the SPI . . . 3444
67.4.11 Enabling the SPI . . . 3445

67.4.12 SPI data transmission and reception procedures . . . 3446
67.4.13 Disabling the SPI . . . 3450
67.4.14 Communication using DMA (direct memory addressing) . . . 3451
67.5 SPI specific modes and control . . . 3453
67.5.1 TI mode . . . 3453
67.5.2 SPI error flags . . . 3453
67.5.3 CRC computation . . . 3457
67.6 SPI in low-power modes . . . 3458
67.7 SPI interrupts . . . 3458
67.8 I2S main features . . . 3460
67.9 I2S functional description . . . 3460
67.9.1 I2S general description . . . 3460
67.9.2 Pin sharing with SPI function . . . 3461
67.9.3 Bitfields usable in I2S/PCM mode . . . 3461
67.9.4 Slave and master modes . . . 3462
67.9.5 Supported audio protocols . . . 3462
67.9.6 Additional serial interface flexibility . . . 3468
67.9.7 Startup sequence . . . 3470
67.9.8 Stop sequence . . . 3472
67.9.9 Clock generator . . . 3473
67.9.10 Internal FIFOs . . . 3475
67.9.11 FIFO status flags . . . 3476
67.9.12 Handling of underrun situation . . . 3476
67.9.13 Handling of overrun situation . . . 3477
67.9.14 Frame error detection . . . 3478
67.9.15 DMA interface . . . 3480
67.9.16 Programing examples . . . 3480
67.10 I2S interrupts . . . 3482
67.11 SPI/I2S registers . . . 3483
67.11.1 SPI/I2S control register 1 (SPI_CR1) . . . 3483
67.11.2 SPI/I2S control register 2 (SPI_CR2) . . . 3485
67.11.3 SPI/I2S configuration register 1 (SPI_CFG1) . . . 3485
67.11.4 SPI/I2S configuration register 2 (SPI_CFG2) . . . 3488
67.11.5 SPI/I2S interrupt enable register (SPI_IER) . . . 3490
67.11.6 SPI/I2S status register (SPI_SR) . . . 3491
67.11.7 SPI/I2S interrupt/status flags clear register (SPI_IFCR) . . . 3494

67.11.8 SPI/I2S transmit data register (SPI_TXDR) . . . 3495
67.11.9 SPI/I2S receive data register (SPI_RXDR) . . . 3495
67.11.10 SPI/I2S polynomial register (SPI_CRCPOLY) . . . 3496
67.11.11 SPI/I2S transmitter CRC register (SPI_TXCRC) . . . 3496
67.11.12 SPI/I2S receiver CRC register (SPI_RXCRC) . . . 3497
67.11.13 SPI/I2S underrun data register (SPI_UDRDR) . . . 3498
67.11.14 SPI/I2S configuration register (SPI_I2SCFGR) . . . 3498
67.11.15 SPI/I2S register map . . . 3500

68 Serial audio interface (SAI) . . . 3502
68.1 SAI introduction . . . 3502
68.2 SAI main features . . . 3502
68.3 SAI implementation . . . 3503
68.4 SAI functional description . . . 3504
68.4.1 SAI block diagram . . . 3504
68.4.2 SAI pins and internal signals . . . 3505
68.4.3 Main SAI modes . . . 3506
68.4.4 SAI synchronization mode . . . 3507
68.4.5 Audio data size . . . 3508
68.4.6 Frame synchronization . . . 3508
68.4.7 Slot configuration . . . 3511
68.4.8 SAI clock generator . . . 3513
68.4.9 Internal FIFOs . . . 3516
68.4.10 PDM interface . . . 3518
68.4.11 AC’97 link controller . . . 3526
68.4.12 SPDIF output . . . 3528
68.4.13 Specific features . . . 3531
68.4.14 Error flags . . . 3535
68.4.15 Disabling the SAI . . . 3538
68.4.16 SAI DMA interface . . . 3538
68.5 SAI interrupts . . . 3539
68.6 SAI registers . . . 3541
68.6.1 SAI global configuration register (SAI_GCR) . . . 3541
68.6.2 SAI configuration register 1 (SAI_ACR1) . . . 3541
68.6.3 SAI configuration register 2 (SAI_ACR2) . . . 3544
68.6.4 SAI frame configuration register (SAI_AFRCR) . . . 3546

68.6.5 SAI slot register (SAI_ASLOTR) . . . 3547
68.6.6 SAI interrupt mask register (SAI_AIM) . . . 3548
68.6.7 SAI status register (SAI_ASR) . . . 3550
68.6.8 SAI clear flag register (SAI_ACLRFR) . . . 3552
68.6.9 SAI data register (SAI_ADR) . . . 3553
68.6.10 SAI configuration register 1 (SAI_BCR1) . . . 3553
68.6.11 SAI configuration register 2 (SAI_BCR2) . . . 3556
68.6.12 SAI frame configuration register (SAI_BFRCR) . . . 3558
68.6.13 SAI slot register (SAI_BSLOTR) . . . 3559
68.6.14 SAI interrupt mask register (SAI_BIM) . . . 3560
68.6.15 SAI status register (SAI_BSR) . . . 3561
68.6.16 SAI clear flag register (SAI_BCLRFR) . . . 3563
68.6.17 SAI data register (SAI_BDR) . . . 3564
68.6.18 SAI PDM control register (SAI_PDMCR) . . . 3565
68.6.19 SAI PDM delay register (SAI_PDMDLY) . . . 3566
68.6.20 SAI register map . . . 3568

69 SPDIF receiver interface (SPDIFRX) . . . 3570

69.1 SPDIFRX interface introduction . . . 3570
69.2 SPDIFRX main features . . . 3570
69.3 SPDIFRX functional description . . . 3570
69.3.1 SPDIFRX pins and internal signals . . . 3571
69.3.2 S/PDIF protocol (IEC-60958) . . . 3572
69.3.3 SPDIFRX decoder (SPDIFRX_DC) . . . 3574
69.3.4 SPDIFRX tolerance to clock deviation . . . 3578
69.3.5 SPDIFRX synchronization . . . 3578
69.3.6 SPDIFRX handling . . . 3580
69.3.7 Data reception management . . . 3582
69.3.8 Dedicated control flow . . . 3584
69.3.9 Reception errors . . . 3585
69.3.10 Clocking strategy . . . 3587
69.3.11 Symbol clock generation . . . 3587
69.3.12 DMA interface . . . 3589
69.3.13 Interrupt generation . . . 3590
69.3.14 Register protection . . . 3591
69.4 Programming procedures . . . 3591
69.4.1 Initialization phase . . . 3592

69.4.2 Handling of interrupts coming from SPDIFRX . . . 3593
69.4.3 Handling of interrupts coming from DMA . . . 3593
69.5 SPDIFRX interface registers . . . 3594
69.5.1 SPDIFRX control register (SPDIFRX_CR) . . . 3594
69.5.2 SPDIFRX interrupt mask register (SPDIFRX_IMR) . . . 3596
69.5.3 SPDIFRX status register (SPDIFRX_SR) . . . 3597
69.5.4 SPDIFRX interrupt flag clear register (SPDIFRX_IFCR) . . . 3599
69.5.5 SPDIFRX data input register (SPDIFRX_FMT0_DR) . . . 3600
69.5.6 SPDIFRX data input register (SPDIFRX_FMT1_DR) . . . 3600
69.5.7 SPDIFRX data input register (SPDIFRX_FMT2_DR) . . . 3601
69.5.8 SPDIFRX channel status register (SPDIFRX_CSR) . . . 3602
69.5.9 SPDIFRX debug information register (SPDIFRX_DIR) . . . 3602
69.5.10 SPDIFRX interface register map . . . 3603

70 Management data input/output (MDIOS) . . . 3604
70.1 MDIOS introduction . . . 3604
70.2 MDIOS main features . . . 3604
70.3 MDIOS functional description . . . 3605
70.3.1 MDIOS block diagram . . . 3605
70.3.2 MDIOS pins and internal signals . . . 3605
70.3.3 MDIOS protocol . . . 3605
70.3.4 MDIOS enabling and disabling . . . 3606
70.3.5 MDIOS data . . . 3607
70.3.6 MDIOS APB frequency . . . 3608
70.3.7 Write/read flags and interrupts . . . 3608
70.3.8 MDIOS error management . . . 3609
70.3.9 MDIOS in Stop mode . . . 3610
70.3.10 MDIOS interrupts . . . 3610
70.4 MDIOS registers . . . 3610
70.4.1 MDIOS configuration register (MDIOS_CR) . . . 3610
70.4.2 MDIOS write flag register (MDIOS_WRFR) . . . 3611
70.4.3 MDIOS clear write flag register (MDIOS_CWRFR) . . . 3612
70.4.4 MDIOS read flag register (MDIOS_RDFR) . . . 3612
70.4.5 MDIOS clear read flag register (MDIOS_CRDFR) . . . 3613
70.4.6 MDIOS status register (MDIOS_SR) . . . 3613
70.4.7 MDIOS clear flag register (MDIOS_CLRFR) . . . 3614
70.4.8 MDIOS input data register x (MDIOS_DINRx) . . . 3614

70.4.9 MDIOS output data register x (MDIOS_DOUTRx) 3615
70.4.10 MDIOS register map 3615

71 Controller area network with flexible data rate (FDCAN) 3617

71.1 FDCAN introduction 3617
71.2 FDCAN main features 3620
71.3 FDCAN implementation 3620
71.4 FDCAN functional description 3621
71.4.1 Operating modes 3622
71.4.2 Error management 3631
71.4.3 Message RAM 3631
71.4.4 FIFO acknowledge handling 3643
71.4.5 Bit timing 3644
71.4.6 Clock calibration on CAN 3645
71.4.7 Application 3649
71.4.8 TTCAN operations (FDCAN1 only) 3650
71.4.9 TTCAN configuration 3651
71.4.10 Message scheduling 3653
71.4.11 TTCAN gap control 3660
71.4.12 Stop watch 3661
71.4.13 Local time, cycle time, global time, and external clock synchronization 3661
71.4.14 TTCAN error level 3664
71.4.15 TTCAN message handling 3665
71.4.16 TTCAN interrupt and error handling 3668
71.4.17 Level 0 3669
71.4.18 Synchronization to external time schedule 3671
71.4.19 FDCAN Rx buffer and FIFO element 3672
71.4.20 FDCAN Tx buffer element 3674
71.4.21 FDCAN Tx event FIFO element 3676
71.4.22 FDCAN standard message ID filter element 3677
71.4.23 FDCAN extended message ID filter element 3679
71.4.24 FDCAN trigger memory element 3680
71.5 FDCAN registers 3682
71.5.1 FDCAN core release register (FDCAN_CREL) 3682
71.5.2 FDCAN Endian register (FDCAN_ENDN) 3682
71.5.3 FDCAN data bit timing and prescaler register (FDCAN_DBTP) 3682

71.5.4 FDCAN test register (FDCAN_TEST) 3683
71.5.5 FDCAN RAM watchdog register (FDCAN_RWD) 3684
71.5.6 FDCAN CC control register (FDCAN_CCCR) 3685
71.5.7 FDCAN nominal bit timing and prescaler register (FDCAN_NBTP) . 3687
71.5.8 FDCAN timestamp counter configuration register (FDCAN_TSCC) . 3688
71.5.9 FDCAN timestamp counter value register (FDCAN_TSCV) 3688
71.5.10 FDCAN timeout counter configuration register (FDCAN_TOCC) . . . 3689
71.5.11 FDCAN timeout counter value register (FDCAN_TOCV) 3690
71.5.12 FDCAN error counter register (FDCAN_ECR) 3690
71.5.13 FDCAN protocol status register (FDCAN_PSR) 3691
71.5.14 FDCAN transmitter delay compensation register (FDCAN_TDCR) . . 3693
71.5.15 FDCAN interrupt register (FDCAN_IR) 3693
71.5.16 FDCAN interrupt enable register (FDCAN_IE) 3696
71.5.17 FDCAN interrupt line select register (FDCAN_ILS) 3698
71.5.18 FDCAN interrupt line enable register (FDCAN_ILE) 3699
71.5.19 FDCAN global filter configuration register (FDCAN_GFC) 3700
71.5.20 FDCAN standard ID filter configuration register (FDCAN_SIDFC) . . 3701
71.5.21 FDCAN extended ID filter configuration register (FDCAN_XIDFC) . . 3701
71.5.22 FDCAN extended ID and mask register (FDCAN_XIDAM) 3702
71.5.23 FDCAN high priority message status register (FDCAN_HPMS) 3703
71.5.24 FDCAN new data 1 register (FDCAN_NDAT1) 3703
71.5.25 FDCAN new data 2 register (FDCAN_NDAT2) 3704
71.5.26 FDCAN Rx FIFO 0 configuration register (FDCAN_RXF0C) 3704
71.5.27 FDCAN Rx FIFO 0 status register (FDCAN_RXF0S) 3705
71.5.28 FDCAN Rx FIFO 0 acknowledge register (FDCAN_RXF0A) 3706
71.5.29 FDCAN Rx buffer configuration register (FDCAN_RXBC) 3706
71.5.30 FDCAN Rx FIFO 1 configuration register (FDCAN_RXF1C) 3707
71.5.31 FDCAN Rx FIFO 1 status register (FDCAN_RXF1S) 3708
71.5.32 FDCAN Rx FIFO 1 acknowledge register (FDCAN_RXF1A) 3709
71.5.33 FDCAN Rx buffer element size configuration register (FDCAN_RXESC) 3709
71.5.34 FDCAN Tx buffer configuration register (FDCAN_TXBC) 3710
71.5.35 FDCAN Tx FIFO/queue status register (FDCAN_TXFQS) 3711
71.5.36 FDCAN Tx buffer element size configuration register (FDCAN_TXESC) 3712
71.5.37 FDCAN Tx buffer request pending register (FDCAN_TXBRP) 3712
71.5.38 FDCAN Tx buffer add request register (FDCAN_TXBAR) 3713
71.5.39 FDCAN Tx buffer cancellation request register (FDCAN_TXBCR) . . 3714

71.5.40 FDCAN Tx buffer transmission occurred register (FDCAN_TXBTO) 3714
71.5.41 FDCAN Tx buffer cancellation finished register (FDCAN_TXBCF) . . 3715
71.5.42 FDCAN Tx buffer transmission interrupt enable register (FDCAN_TXBTIE) 3715
71.5.43 FDCAN Tx buffer cancellation finished interrupt enable register (FDCAN_TXBCIE) 3716
71.5.44 FDCAN Tx event FIFO configuration register (FDCAN_TXEFC) . . . 3716
71.5.45 FDCAN Tx event FIFO status register (FDCAN_TXEFS) 3717
71.5.46 FDCAN Tx event FIFO acknowledge register (FDCAN_TXEFA) . . . 3718
71.5.47 FDCAN register map 3718
71.6 TTCAN registers 3722
71.6.1 FDCAN TT trigger memory configuration register (FDCAN_TTTMC) 3722
71.6.2 FDCAN TT reference message configuration register (FDCAN_TTRMC) 3722
71.6.3 FDCAN TT operation configuration register (FDCAN_TTOCF) 3723
71.6.4 FDCAN TT matrix limits register (FDCAN_TTMLM) 3725
71.6.5 FDCAN TUR configuration register (FDCAN_TURCF) 3726
71.6.6 FDCAN TT operation control register (FDCAN_TTOCN) 3727
71.6.7 FDCAN TT global time preset register (FDCAN_TTGTP) 3729
71.6.8 FDCAN TT time mark register (FDCAN_TTTMK) 3729
71.6.9 FDCAN TT interrupt register (FDCAN_TTIR) 3730
71.6.10 FDCAN TT interrupt enable register (FDCAN_TTIE) 3732
71.6.11 FDCAN TT interrupt line select register (FDCAN_TTILS) 3734
71.6.12 FDCAN TT operation status register (FDCAN_TTOST) 3735
71.6.13 FDCAN TUR numerator actual register (FDCAN_TURNA) 3737
71.6.14 FDCAN TT local and global time register (FDCAN_TTLGT) 3738
71.6.15 FDCAN TT cycle time and count register (FDCAN_TTCTC) 3738
71.6.16 FDCAN TT capture time register (FDCAN_TTCPT) 3739
71.6.17 FDCAN TT cycle sync mark register (FDCAN_TTCSM) 3739
71.6.18 FDCAN TT trigger select register (FDCAN_TTTS) 3740
71.6.19 FDCAN TT register map 3740
71.7 CCU registers 3742
71.7.1 Clock calibration unit core release register (FDCAN_CCU_CREL) . . 3742
71.7.2 Calibration configuration register (FDCAN_CCU_CCFG) 3742
71.7.3 Calibration status register (FDCAN_CCU_CSTAT) 3744
71.7.4 Calibration watchdog register (FDCAN_CCU_CWD) 3744
71.7.5 Clock calibration unit interrupt register (FDCAN_CCU_IR) 3745

71.7.6 Clock calibration unit interrupt enable register (FDCAN_CCU_IE) . . 3746
71.7.7 CCU register map 3746

72 USB subsystem (USBSS) 3748
72.1 Overview 3748
72.2 USB2 OTG high-speed Port 1 3748
72.2.1 USB2 OTG high-speed Port 1 main features 3748
72.2.2 USB2 OTG high-speed Port 1 functional description 3749
72.2.3 USB2 OTG high-speed Port 1 interrupts 3751
72.2.4 Battery charging 3751
72.2.5 ID management 3752
72.2.6 VBUS detection 3752
72.3 USB2 OTG high-speed Port 2 3752
72.3.1 USB2 OTG high-speed Port 2 main features 3752
72.3.2 USB2 OTG high-speed Port 2 functional description 3753
72.3.3 USB2 OTG high-speed Port 2 interrupts 3755
72.3.4 Battery charging 3755
72.3.5 ID management 3755
72.3.6 VBUS management 3755
72.4 USB Type-C and power delivery interface 3756
72.4.1 USB Type-C and power delivery interface main features 3756
72.4.2 USB Type-C and power delivery interface implementation 3756
72.4.3 USB Type-C and power delivery interface functional description . . . 3758
72.4.4 USB Type-C and power delivery interrupts 3759
72.5 Debug signals 3760

73 USB on-the-go high-speed (OTG) 3761
73.1 OTG introduction 3761
73.2 OTG main features 3762
73.2.1 General features 3762
73.2.2 Host-mode features 3763
73.2.3 Peripheral-mode features 3763
73.3 OTG implementation 3763
73.4 OTG functional description 3764
73.4.1 OTG block diagram 3764
73.4.2 OTG pin and internal signals 3764

73.4.3 OTG core . . . 3765
73.4.4 OTG detections . . . 3765
73.4.5 High-speed OTG PHY connected to OTG . . . 3765
73.4.6 Battery charging detection . . . 3765
73.5 OTG dual role device (DRD) . . . 3766
73.5.1 ID line detection . . . 3766
73.6 OTG as a USB peripheral . . . 3766
73.6.1 Peripheral states . . . 3767
73.6.2 Peripheral endpoints . . . 3768
73.7 OTG as a USB host . . . 3770
73.7.1 USB host states . . . 3771
73.7.2 Host channels . . . 3772
73.7.3 Host scheduler . . . 3773
73.8 OTG SOF trigger . . . 3774
73.8.1 Host SOFs . . . 3775
73.8.2 Peripheral SOFs . . . 3775
73.9 OTG low-power modes . . . 3775
73.10 OTG Dynamic update of the OTG_HFIR register . . . 3776
73.11 OTG data FIFOs . . . 3777
73.11.1 Peripheral FIFO architecture . . . 3777
73.11.2 Host FIFO architecture . . . 3778
73.11.3 FIFO RAM allocation . . . 3780
73.12 **OTG** interrupts . . . 3781
73.13 **OTG** control and status registers . . . 3783
73.13.1 CSR memory map . . . 3783
73.14 **OTG** registers . . . 3788
73.14.1 OTG control and status register (OTG_GOTGCTL) . . . 3788
73.14.2 OTG interrupt register (OTG_GOTGINT) . . . 3790
73.14.3 OTG AHB configuration register (OTG_GAHBCFG) . . . 3791
73.14.4 OTG USB configuration register (OTG_GUSBCFG) . . . 3792
73.14.5 OTG reset register (OTG_GRSTCTL) . . . 3794
73.14.6 OTG core interrupt register [alternate] (OTG_GINTSTS) . . . 3797
73.14.7 OTG core interrupt register [alternate] (OTG_GINTSTS) . . . 3801
73.14.8 OTG interrupt mask register [alternate] (OTG_GINTMSK) . . . 3806
73.14.9 OTG interrupt mask register [alternate] (OTG_GINTMSK) . . . 3807

73.14.10 OTG receive status debug read register [alternate] (OTG_GRXSTSR) . . . 3809
73.14.11 OTG receive status debug read register [alternate] (OTG_GRXSTSR) . . . 3810
73.14.12 OTG status read and pop registers (OTG_GRXSTSP) . . . 3811
73.14.13 OTG status read and pop registers [alternate] (OTG_GRXSTSP) . . 3812
73.14.14 OTG receive FIFO size register (OTG_GRXFSIZ) . . . 3813
73.14.15 OTG host non-periodic transmit FIFO size register [alternate] (OTG_HNPTXFSIZ) . . . 3814
73.14.16 Endpoint 0 transmit FIFO size [alternate] (OTG_DIEPTXF0) . . . 3814
73.14.17 OTG non-periodic transmit FIFO/queue status register (OTG_HNPTXSTS) . . . 3815
73.14.18 OTG general core configuration register (OTG_GCCFG) . . . 3816
73.14.19 OTG core ID register (OTG_CID) . . . 3817
73.14.20 OTG core LPM configuration register (OTG_GLPMCFG) . . . 3818
73.14.21 OTG host periodic transmit FIFO size register (OTG_HPTXFSIZ) . . . 3822
73.14.22 OTG device IN endpoint transmit FIFO x size register (OTG_DIEPTXFx) . . . 3822
73.14.23 Host-mode registers . . . 3822
73.14.24 OTG host configuration register (OTG_HCFG) . . . 3823
73.14.25 OTG host frame interval register (OTG_HFIR) . . . 3823
73.14.26 OTG host frame number/frame time remaining register (OTG_HFNUM) . . . 3824
73.14.27 OTG_Host periodic transmit FIFO/queue status register (OTG_HPTXSTS) . . . 3825
73.14.28 OTG host all channels interrupt register (OTG_HAINT) . . . 3826
73.14.29 OTG host all channels interrupt mask register (OTG_HAINTMSK) . . . 3826
73.14.30 OTG host port control and status register (OTG_HPRT) . . . 3827
73.14.31 OTG host channel x characteristics register (OTG_HCCHARx) . . . 3829
73.14.32 OTG host channel x split control register (OTG_HCSPLTx) . . . 3830
73.14.33 OTG host channel x interrupt register (OTG_HCINTx) . . . 3831
73.14.34 OTG host channel x interrupt mask register (OTG_HCINTMSKx) . . 3832
73.14.35 OTG host channel x transfer size register (OTG_HCTSIZx) . . . 3833
73.14.36 OTG host channel x DMA address register(OTG_HCDMAx) . . . 3834
73.14.37 Device-mode registers . . . 3834
73.14.38 OTG device configuration register (OTG_DCFG) . . . 3834
73.14.39 OTG device control register (OTG_DCTL) . . . 3836
73.14.40 OTG device status register (OTG_DSTS) . . . 3838

73.14.41 OTG device IN endpoint common interrupt mask register (OTG_DIEPMSK) . . . 3839
73.14.42 OTG device OUT endpoint common interrupt mask register (OTG_DOEPMSK) . . . 3840
73.14.43 OTG device all endpoints interrupt register (OTG_DAINT) . . . 3841
73.14.44 OTG all endpoints interrupt mask register (OTG_DAINTMSK) . . . 3842
73.14.45 OTG device threshold control register (OTG_DTHRCTL) . . . 3842
73.14.46 OTG device IN endpoint FIFO empty interrupt mask register (OTG_DIEPEMPMSK) . . . 3843
73.14.47 OTG device IN endpoint x control register [alternate] (OTG_DIEPCTLx) . . . 3844
73.14.48 OTG device IN endpoint x control register [alternate] (OTG_DIEPCTLx) . . . 3846
73.14.49 OTG device IN endpoint x interrupt register (OTG_DIEPINTx) . . . 3848
73.14.50 OTG device IN endpoint 0 transfer size register (OTG_DIEPTSIZ0) . . . 3849
73.14.51 OTG device IN endpoint x DMA address register (OTG_DIEPDMAx) . . . 3850
73.14.52 OTG device IN endpoint transmit FIFO status register (OTG_DTXFSTSx) . . . 3850
73.14.53 OTG device IN endpoint x transfer size register (OTG_DIEPTSIZx) . 3851
73.14.54 OTG device control OUT endpoint 0 control register (OTG_DOEPCTL0) . . . 3851
73.14.55 OTG device OUT endpoint x interrupt register (OTG_DOEPINTx) . . 3853
73.14.56 OTG device OUT endpoint 0 transfer size register (OTG_DOEPTSIZ0) . . . 3855
73.14.57 OTG device OUT endpoint x DMA address register (OTG_DOEPDMAx) . . . 3856
73.14.58 OTG device OUT endpoint x control register [alternate] (OTG_DOEPCTLx) . . . 3856
73.14.59 OTG device OUT endpoint x control register [alternate] (OTG_DOEPCTLx) . . . 3858
73.14.60 OTG device OUT endpoint x transfer size register (OTG_DOEPTSIZx) . . . 3860
73.14.61 OTG power and clock gating control register (OTG_PCGCCTL) . . . 3861
73.14.62 OTG power and clock gating control register 1 (OTG_PCGCCTL1) . 3862
73.14.63 OTG register map . . . 3863
73.15 **OTG** programming model . . . 3870
73.15.1 Core initialization . . . 3870
73.15.2 Host initialization . . . 3871

73.15.3 Device initialization . . . 3872
73.15.4 DMA mode . . . 3872
73.15.5 Host programming model . . . 3872
73.15.6 Device programming model . . . 3905
73.15.7 Worst case response time . . . 3925
73.15.8 OTG programming model . . . 3927

74 USB HS PHY controller (USBPHYC) . . . 3928
74.1 Introduction . . . 3928
74.2 USBPHYC main features . . . 3928
74.3 USBPHYC implementation . . . 3928
74.4 USBPHYC functional description . . . 3928
74.4.1 USBPHYC block diagram . . . 3928
74.4.2 USBPHYC reset and clocks . . . 3928
74.4.3 USBPHYC programmable parameters . . . 3929
74.4.4 USBPHYC trimming of electrical parameters . . . 3929
74.5 USBPHYC registers . . . 3929
74.5.1 USBPHYC control register (USBPHYC_CR) . . . 3929
74.5.2 USBPHYC trimming 1 register (USBPHYC_TRIM1CR) . . . 3931
74.5.3 USBPHYC trimming 2 register (USBPHYC_TRIM2CR) . . . 3935
74.5.4 USBPHYC register map . . . 3936

75 USB Type-C®/USB Power Delivery interface (UCPD) . . . 3937
75.1 UCPD introduction . . . 3937
75.2 UCPD main features . . . 3937
75.3 UCPD implementation . . . 3938
75.4 UCPD functional description . . . 3938
75.4.1 UCPD block diagram . . . 3939
75.4.2 UCPD reset and clocks . . . 3940
75.4.3 Physical layer protocol . . . 3940
75.4.4 UCPD BMC transmitter . . . 3947
75.4.5 UCPD BMC receiver . . . 3948
75.4.6 UCPD Type-C pull-ups (Rp) and pull-downs (Rd) . . . 3950
75.4.7 UCPD Type-C voltage monitoring and de-bouncing . . . 3950
75.4.8 UCPD fast role swap (FRS) . . . 3951
75.4.9 UCPD DMA Interface . . . 3951

75.4.10 Wake-up from Stop mode . . . 3951
75.5 UCPD programming sequences . . . 3951
75.5.1 Initialization phase . . . 3951
75.5.2 Type-C state machine handling . . . 3952
75.5.3 USB PD transmit . . . 3953
75.5.4 USB PD receive . . . 3954
75.6 UCPD low-power modes . . . 3955
75.7 UCPD interrupts . . . 3956
75.8 UCPD registers . . . 3957
75.8.1 UCPD configuration register 1 (UCPD_CFGR1) . . . 3957
75.8.2 UCPD configuration register 2 (UCPD_CFGR2) . . . 3959
75.8.3 UCPD control register (UCPD_CR) . . . 3959
75.8.4 UCPD interrupt mask register (UCPD_IMR) . . . 3962
75.8.5 UCPD status register (UCPD_SR) . . . 3963
75.8.6 UCPD interrupt clear register (UCPD_ICR) . . . 3966
75.8.7 UCPD Tx ordered set type register (UCPD_TX_ORDSETR) . . . 3967
75.8.8 UCPD Tx payload size register (UCPD_TX_PAYSZR) . . . 3967
75.8.9 UCPD Tx data register (UCPD_TXDR) . . . 3968
75.8.10 UCPD Rx ordered set register (UCPD_RX_ORDSETR) . . . 3968
75.8.11 UCPD Rx payload size register (UCPD_RX_PAYSZR) . . . 3969
75.8.12 UCPD receive data register (UCPD_RXDR) . . . 3969
75.8.13 UCPD Rx ordered set extension register 1 (UCPD_RX_ORDEXTR1) . . . 3970
75.8.14 UCPD Rx ordered set extension register 2 (UCPD_RX_ORDEXTR2) . . . 3970
75.8.15 UCPD register map . . . 3971

76 Ethernet (ETH): gigabit media access control (GMAC) with DMA controller . . . 3973

76.1 Ethernet introduction . . . 3973
76.2 Ethernet main features . . . 3973
76.2.1 Standard compliance . . . 3973
76.2.2 MAC features . . . 3973
76.2.3 Transaction layer (MTL) features . . . 3975
76.2.4 DMA block features . . . 3976
76.2.5 Bus interface features . . . 3977
76.2.6 Audio and video features . . . 3978

76.2.7 Time sensitive networking features . . . 3978
76.2.8 Generic queuing features . . . 3978
76.3 Ethernet pins and internal signals . . . 3978
76.4 Ethernet architecture . . . 3980
76.4.1 DMA controller . . . 3981
76.4.2 MTL . . . 3988
76.4.3 MAC . . . 3988
76.4.4 Multiple channels and queues support . . . 3993
76.5 Ethernet functional description: MAC . . . 4005
76.5.1 Double VLAN processing . . . 4005
76.5.2 Source address and VLAN insertion, replacement, or deletion . . . 4006
76.5.3 Queue/channel-based VLAN tag insertion on Tx . . . 4007
76.5.4 Packet filtering . . . 4008
76.5.5 IEEE 1588 timestamp support . . . 4020
76.5.6 Time sensitive networking (TSN) . . . 4046
76.5.7 Checksum offload engine . . . 4075
76.5.8 TCP segmentation offload . . . 4081
76.5.9 IPv4 ARP offload . . . 4087
76.5.10 Loopback . . . 4088
76.5.11 Flow control . . . 4089
76.5.12 MAC management counters . . . 4092
76.5.13 Interrupts generated by the MAC . . . 4094
76.5.14 MAC and MMC register descriptions . . . 4094
76.6 Ethernet functional description: PHY interfaces . . . 4095
76.6.1 Station management agent (SMA) . . . 4095
76.6.2 Media independent interface (MII) . . . 4102
76.6.3 Reduced media independent interface (RMII) . . . 4103
76.6.4 Reduced gigabit media independent interface (RGMII) . . . 4106
76.7 Ethernet low-power modes . . . 4108
76.7.1 Low-power management . . . 4108
76.7.2 Energy-efficient Ethernet (EEE) . . . 4114
76.8 Ethernet interrupts . . . 4121
76.8.1 DMA interrupts . . . 4121
76.8.2 MTL interrupts . . . 4123
76.8.3 MAC interrupts . . . 4123
76.9 Ethernet programming model . . . 4124

76.9.1 DMA initialization 4124
76.9.2 MTL initialization 4125
76.9.3 MAC initialization 4126
76.9.4 Performing normal receive and transmit operation 4127
76.9.5 Stopping and starting transmission 4127
76.9.6 Programming guidelines for switching to new descriptor list in RxDMA 4128
76.9.7 Programming guidelines for switching the AXI clock frequency 4128
76.9.8 Programming guidelines for multichannel multiqueue operation 4128
76.9.9 Programming guidelines for GMII link state transitions 4131
76.9.10 Programming guidelines for IEEE 1588 timestamping 4132
76.9.11 Programming guidelines for PTP offload feature 4133
76.9.12 Programming guidelines for AV feature 4137
76.9.13 Programming guidelines for Energy Efficient Ethernet (EEE) 4139
76.9.14 Programming guidelines for flexible pulse-per-second (PPS) output 4140
76.9.15 Programming guidelines for IEEE 1588 auxiliary snapshot 4142
76.9.16 Programming guidelines for TSO 4143
76.9.17 Programming guidelines to perform VLAN filtering on the receiver . . 4144
76.9.18 Programming guidelines for extended VLAN filtering and routing on reception 4144
76.9.19 Programming sequence for queue/channel-based VLAN inclusion register 4145
76.9.20 Programming guidelines for EST 4146
76.9.21 Programming the launch time in time-based scheduling 4147
76.9.22 Enabling the frame preemption function 4147
76.10 Descriptors 4148
76.10.1 Descriptor overview 4148
76.10.2 Descriptor structure 4149
76.10.3 Transmit descriptor 4152
76.10.4 Receive descriptor 4165
76.10.5 Enhanced descriptor for time-based scheduling 4178
76.11 Ethernet registers 4182
76.11.1 Ethernet register maps 4182
76.11.2 Ethernet DMA registers 4182
76.11.3 Ethernet MTL registers 4212
76.11.4 Ethernet MAC and MMC registers 4243

77 Hardware debug port (HDP) 4368

77.1 HDP introduction 4368
77.2 HDP functional description 4368
77.3 HDP block diagram 4370
77.4 HDP register description 4370
77.4.1 HDP control register (HDP_CTRL) 4370
77.4.2 HDP multiplexer control register (HDP_MUX) 4371
77.4.3 HDP read back value register (HDP_VAL) 4371
77.4.4 HDP general-purpose output set register (HDP_GPOSET) 4371
77.4.5 HDP general purpose output clear register (HDP_GPOCLR) 4372
77.4.6 HDP general purpose output value register (HDP_GPOVAL) 4372
77.4.7 HDP register map 4372

78 Debug support (DBG) 4374
78.1 DBG introduction 4374
78.2 DBG functional description 4375
78.3 DBG block diagram 4377
78.4 DBG pins 4377
78.5 DBG power, clock, and reset 4378
78.5.1 DBG power domains 4378
78.5.2 DBG clocks 4378
78.5.3 Debug reset 4379
78.6 Security 4379
78.6.1 BSEC control over debug 4380
78.6.2 Authentication signals 4380
78.7 Debug authentication 4381
78.7.1 Main characteristics 4381
78.7.2 Debug authentication protocol 4381
78.8 Chip-level TAP controller (CLTAPC) 4382
78.9 Serial-wire and JTAG debug port (SWJ-DP) 4382
78.9.1 JTAG debug port 4383
78.9.2 SWD debug port 4386
78.9.3 Debug port registers 4388
78.10 Access ports 4396
78.10.1 AP0 registers 4398
78.10.2 AP1 registers 4401
78.10.3 AP2 registers 4405

78.10.4 ROM tables 4410
78.10.5 System/MCU ROM registers 4412
78.11 Cortex-M55 debug features 4417
78.11.1 Processor ROM table 4417
78.11.2 Processor ROM registers 4419
78.11.3 Data watchpoint and trace unit (DWT) 4423
78.11.4 Cortex-M55 DWT registers 4424
78.11.5 Cortex-M55 performance monitoring unit (PMU) 4438
78.11.6 Cortex-M55 PMU registers 4439
78.11.7 Instrumentation trace macrocell (ITM) 4452
78.11.8 Cortex-M55 ITM registers 4453
78.11.9 Breakpoint Unit (BPU) 4461
78.11.10 Cortex-M55 BPU registers 4461
78.11.11 Embedded Trace Macrocell (ETM) 4468
78.11.12 Cortex-M55 ETM registers 4468
78.11.13 Cross trigger interface (CTI) 4495
78.12 Trace and debug subsystem 4495
78.12.1 Trace subsystem ROM table registers 4498
78.12.2 Global timestamp generator (TSGEN) 4502
78.12.3 TSGEN registers 4503
78.12.4 Cross trigger interface (CTI) and matrix (CTM) 4510
78.12.5 CTI registers 4514
78.12.6 Trace funnel (CSTF) 4529
78.12.7 Trace funnel registers 4529
78.12.8 Embedded trace FIFO (ETF) 4539
78.12.9 Embedded trace FIFO registers 4540
78.12.10 Embedded trace router (ETR) 4558
78.12.11 ETR registers 4562
78.12.12 Trace port interface unit (TPIU) 4582
78.12.13 TPIU registers 4583
78.12.14 Trace replicator (REP) 4599
78.12.15 Trace replicator registers 4599
78.12.16 Serial-wire output (SWO) 4609
78.12.17 SWO registers 4610
78.12.18 System trace macrocell (STM) 4620
78.12.19 STM registers 4623
78.12.20 Microcontroller debug unit (DBGMCU) 4649

78.12.21 DBGMCU registers 4650
78.13 References 4661

79 Device electronic signature 4662
79.1 Unique device ID register (96 bits) (UID) 4662
79.2 Device part number (RPN) 4663
79.3 Package data register (PKG) 4664
79.4 Device version 4664
79.5 Boot ROM version 4665

80 Important security notice 4666

81 Revision history 4667

List of tables

Table 1. Memory map based on IDAU mapping . . . 164
Table 2. Memory map and peripheral register boundary addresses . . . 169
Table 3. Peripheral register boundary addresses . . . 175
Table 4. Aliasing . . . 185
Table 5. Nonsecure peripheral functions that can be connected to secure I/Os . . . 192
Table 6. Internal tampers in TAMP . . . 195
Table 7. Effect of low-power modes on TAMP . . . 197
Table 8. Accelerated cryptographic operations . . . 198
Table 9. BSEC internal input/output signals . . . 202
Table 10. BSEC initial status reporting structure . . . 207
Table 11. BSEC ad-hoc error/status reporting structure . . . 207
Table 12. BSEC states definition . . . 208
Table 13. Current HDPL coding . . . 210
Table 14. BSEC Next HDPL register usage . . . 211
Table 15. BSEC debug register legal values . . . 211
Table 16. BSEC register map and reset values . . . 228
Table 17. OTP mapping . . . 231
Table 18. OTP fuse description (lower OTP region) . . . 235
Table 19. OTP fuse description (mid OTP region) . . . 246
Table 20. RISUP indexes . . . 249
Table 21. RISC indexes purely for RCC security control . . . 250
Table 22. RIMU resource assignment . . . 250
Table 23. RIFSC register map and reset values . . . 257
Table 24. RISAF resource assignment . . . 259
Table 25. RISAF internal input/output signals . . . 260
Table 26. RISAF subregion security setup matrix . . . 262
Table 27. RISAF register map and reset values . . . 278
Table 28. Peripheral indexes in IAC . . . 280
Table 29. IAC internal signals . . . 282
Table 30. IAC interrupt request . . . 283
Table 31. IAC register map and reset values . . . 285
Table 32. Boot modes . . . 286
Table 33. Internal SRAM features . . . 289
Table 34. Hardware-erase conditions for internal SRAMs . . . 289
Table 35. FLEXRAM supported configurations . . . 290
Table 36. FLEXMEM versus retention . . . 290
Table 37. RAMCFG interrupt requests . . . 294
Table 38. RAMCFG register map and reset values . . . 306
Table 39. ICACHE features . . . 309
Table 40. TAG memory dimensioning parameters for n-way set associative operating mode (default) . . . 311
Table 41. TAG memory dimensioning parameters for direct-mapped cache mode . . . 312
Table 42. ICACHE cacheability for AHB transaction . . . 313
Table 43. ICACHE interrupts . . . 316
Table 44. ICACHE register map and reset values . . . 319
Table 45. CACHEAXI features . . . 323
Table 46. TAG memory dimensioning parameters . . . 328
Table 47. CACHEAXI supported AXI 4-bit memory/cache attribute . . . 329

Table 48. CACHEAXI interrupts . . . 336
Table 49. CACHEAXI register map and reset values . . . 345
Table 50. PWR input/output signals connected to package pins or balls . . . 349
Table 51. PWR internal input/output signals . . . 349
Table 52. Wake-up source selection . . . 350
Table 53. Supply configuration control . . . 353
Table 54. Operating mode summary . . . 369
Table 55. Functionalities depending on system operating mode . . . 370
Table 56. Sleep mode . . . 378
Table 57. Stop mode SVOS high . . . 380
Table 58. Stop mode SVOS low . . . 380
Table 59. Standby and Stop flags . . . 382
Table 60. Standby mode . . . 382
Table 61. Power mode output states versus MCU power modes . . . 383
Table 62. PWR register security overview . . . 384
Table 63. PWR register map and reset values . . . 408
Table 64. RCC input/output signals connected to package pins or balls . . . 412
Table 65. RCC internal input/output signals . . . 413
Table 66. Reset coverage summary . . . 419
Table 67. Reset source identification . . . 420
Table 68. Oscillator states versus system modes . . . 424
Table 69. Clock output selection . . . 435
Table 70. STOPWUCK description . . . 443
Table 71. HSISTOPEN and MSISTOPEN behavior . . . 444
Table 72. Peripheral clock distribution summary . . . 445
Table 73. SDMMC interface clock constraints . . . 464
Table 74. Peripheral clock enabling . . . 471
Table 75. Interrupt sources and control . . . 474
Table 76. Maximum peripheral clock frequencies . . . 476
Table 77. RCC register map and reset values . . . 754
Table 78. Port x bit configurations . . . 784
Table 79. Secure AF between peripherals and allocated I/Os . . . 792
Table 80. GPIO secured bits . . . 792
Table 81. GPIO register map and reset values . . . 806
Table 82. SYSCFG register map and reset values . . . 829
Table 83. Connectivity matrix . . . 833
Table 84. Implementation of HPDMA1 channels . . . 836
Table 85. HPDMA1 in low-power modes . . . 836
Table 86. Programmed HPDMA1 request . . . 837
Table 87. Programmed HPDMA1 request as a block request . . . 841
Table 88. HPDMA1 channel with peripheral early termination . . . 841
Table 89. Programmed HPDMA1 request with peripheral early termination . . . 841
Table 90. Programmed HPDMA1 trigger . . . 842
Table 91. Programmed HPDMA source/destination burst . . . 865
Table 92. Programmed data handling . . . 870
Table 93. Effect of low-power modes on HPDMA . . . 892
Table 94. HPDMA interrupt requests . . . 893
Table 95. HPDMA register map and reset values . . . 929
Table 96. GPDMA1 channel implementation . . . 933
Table 97. GPDMA1 wake-up in low-power modes . . . 933
Table 98. Programmed GPDMA1 request . . . 933
Table 99. Programmed GPDMA1 request as a block request . . . 938

Table 100. GPDMA1 channel with peripheral early termination . . . 938
Table 101. Programmed GPDMA1 request with peripheral early termination . . . 938
Table 102. Programmed GPDMA1 trigger . . . 938
Table 103. Programmed GPDMA source/destination burst . . . 961
Table 104. Programmed data handling . . . 966
Table 105. Effect of low-power modes on GPDMA . . . 979
Table 106. GPDMA interrupt requests . . . 980
Table 107. GPDMA register map and reset values . . . 1009
Table 108. Neural-ART 14 NPU configuration . . . 1013
Table 109. Neural-ART 14 NPU signals . . . 1015
Table 110. Activation functions example . . . 1021
Table 111. Neural-ART 14 functional units memory base addresses . . . 1028
Table 112. DMA2D internal signals . . . 1034
Table 113. DMA2D trigger interconnections . . . 1034
Table 114. Supported color mode in input . . . 1035
Table 115. Data order in memory . . . 1036
Table 116. Alpha mode configuration . . . 1037
Table 117. Supported CLUT color mode . . . 1038
Table 118. CLUT data order in memory . . . 1038
Table 119. Supported color mode in output . . . 1039
Table 120. Data order in memory . . . 1040
Table 121. Standard data order in memory . . . 1040
Table 122. Output FIFO byte reordering steps . . . 1041
Table 123. MCU order in memory . . . 1046
Table 124. DMA2D interrupt requests . . . 1047
Table 125. DMA2D register map and reset values . . . 1065
Table 126. GFXMMU internal input/output signals . . . 1068
Table 127. GFXMMU interrupt requests . . . 1074
Table 128. GFXMMU register map and reset values . . . 1080
Table 129. GFXTIM input/output pins . . . 1083
Table 130. GFXTIM internal signals . . . 1083
Table 131. GFXTIM trigger interconnections . . . 1084
Table 132. Graphic timer interrupt requests . . . 1094
Table 133. GFXTIM register map and reset values . . . 1115
Table 134. STM32N6x5/x7xx vector table . . . 1118
Table 135. EXTI implementation . . . 1127
Table 136. EXTI signals . . . 1127
Table 137. EVG signals . . . 1128
Table 138. EXTI line connection . . . 1129
Table 139. EXTI event input configurations and register control . . . 1131
Table 140. Masking functionality . . . 1133
Table 141. Register protection overview . . . 1134
Table 142. EXTI register map sections . . . 1135
Table 143. EXTI register map and reset values . . . 1167
Table 144. CRC internal input/output signals . . . 1171
Table 145. CRC register map and reset values . . . 1176
Table 146. FMC features . . . 1178
Table 147. FMC internal signals . . . 1180
Table 148. FMC memory regions (default mapping) . . . 1182
Table 149. FMC memory region remap using BMAP[1:0] . . . 1182
Table 150. NOR/PSRAM subregion selection . . . 1183
Table 151. NOR/PSRAM External memory address . . . 1183

Table 152. NAND access memory map . . . 1183
Table 153. SDRAM device selection . . . 1184
Table 154. SDRAM address mapping . . . 1184
Table 155. SDRAM address mapping with 8-bit data bus width . . . 1185
Table 156. SDRAM address mapping with 16-bit data bus width . . . 1186
Table 157. SDRAM address mapping with 32-bit data bus width . . . 1186
Table 158. Programmable NOR/PSRAM access parameters . . . 1188
Table 159. Non-multiplexed I/O NOR flash memory . . . 1189
Table 160. 16-bit multiplexed I/O NOR flash memory . . . 1189
Table 161. Non-multiplexed I/Os PSRAM/SRAM . . . 1190
Table 162. 16-bit multiplexed I/O PSRAM . . . 1190
Table 163. NOR flash/PSRAM: Example of supported memories and transactions . . . 1191
Table 164. FMC_BCRx bitfields (mode 1) . . . 1195
Table 165. FMC_BTRx bitfields (mode 1) . . . 1195
Table 166. FMC_BCRx bitfields (mode A) . . . 1197
Table 167. FMC_BTRx bitfields (mode A) . . . 1198
Table 168. FMC_BWTRx bitfields (mode A) . . . 1198
Table 169. FMC_BCRx bitfields (mode 2/B) . . . 1200
Table 170. FMC_BTRx bitfields (mode 2/B) . . . 1201
Table 171. FMC_BWTRx bitfields (mode 2/B) . . . 1201
Table 172. FMC_BCRx bitfields (mode C) . . . 1203
Table 173. FMC_BTRx bitfields (mode C) . . . 1203
Table 174. FMC_BWTRx bitfields (mode C) . . . 1204
Table 175. FMC_BCRx bitfields (mode D) . . . 1205
Table 176. FMC_BTRx bitfields (mode D) . . . 1206
Table 177. FMC_BWTRx bitfields (mode D) . . . 1206
Table 178. FMC_BCRx bitfields (muxed mode) . . . 1208
Table 179. FMC_BTRx bitfields (muxed mode) . . . 1209
Table 180. FMC_BCRx bitfields (Synchronous multiplexed read mode) . . . 1214
Table 181. FMC_BTRx bitfields (Synchronous multiplexed read mode) . . . 1215
Table 182. FMC_BCRx bitfields (Synchronous multiplexed write mode) . . . 1216
Table 183. FMC_BTRx bitfields (Synchronous multiplexed write mode) . . . 1217
Table 184. Programmable NAND flash access parameters . . . 1227
Table 185. 8-bit NAND flash memory . . . 1227
Table 186. 16-bit NAND flash memory . . . 1228
Table 187. Supported memories and transactions . . . 1228
Table 188. Number of ECC parity bytes per sector . . . 1233
Table 189. FMC NAND controller Interrupt request . . . 1242
Table 190. HPR relevant bits . . . 1248
Table 191. ECC result relevant bits . . . 1249
Table 192. SDRAM signals . . . 1267
Table 193. FMC register map . . . 1284
Table 194. XSPI implementation . . . 1290
Table 195. XSPI input/output pins . . . 1295
Table 196. XSPI internal signals . . . 1295
Table 197. Command/address phase description . . . 1306
Table 198. OctaRAM command address bit assignment (based on 64 Mb OctaRAM) . . . 1318
Table 199. Address alignment cases . . . 1326
Table 200. XSPI interrupt requests . . . 1328
Table 201. XSPI register map and reset values . . . 1354
Table 202. XSPIM implementation . . . 1358

Table 203. XSPIM input/output pins . . . 1359
Table 204. Use cases . . . 1361
Table 205. XSPIM register map and reset values . . . 1371
Table 206. SDMMC features . . . 1372
Table 207. SDMMC operation modes SD and SDIO . . . 1375
Table 208. SDMMC operation modes e•MMC . . . 1375
Table 209. SDMMC internal input/output signals . . . 1376
Table 210. SDMMC pins . . . 1377
Table 211. SDMMC Command and data phase selection . . . 1378
Table 212. Command token format . . . 1384
Table 213. Short response with CRC token format . . . 1385
Table 214. Short response without CRC token format . . . 1385
Table 215. Long response with CRC token format . . . 1385
Table 216. Specific Commands overview . . . 1386
Table 217. Command path status flags . . . 1387
Table 218. Command path error handling . . . 1387
Table 219. Data token format . . . 1395
Table 220. Data path status flags and clear bits . . . 1395
Table 221. Data path error handling . . . 1397
Table 222. Data FIFO access . . . 1398
Table 223. Transmit FIFO status flags . . . 1399
Table 224. Receive FIFO status flags . . . 1401
Table 225. AHB and SDMMC_CK clock frequency relation . . . 1405
Table 226. SDIO special operation control . . . 1406
Table 227. 4-bit mode Start, interrupt, and CRC-status Signaling detection . . . 1410
Table 228. CMD12 use cases . . . 1415
Table 229. Data block gap hardware flow control behavior . . . 1427
Table 230. Hardware flow control selection . . . 1427
Table 231. SDMMC interrupts . . . 1431
Table 232. Response type and SDMMC_RESPxR registers . . . 1438
Table 233. SDMMC register map . . . 1454
Table 234. STM32N6x5/x7xx features . . . 1457
Table 235. DLYB functional signals . . . 1459
Table 236. DLYB interconnection . . . 1460
Table 237. DLYB register map and reset values . . . 1464
Table 238. ADC features . . . 1468
Table 239. ADC input/output pins . . . 1470
Table 240. ADC internal input/output signals . . . 1470
Table 241. ADC1/2 interconnection . . . 1470
Table 242. Configuring the trigger polarity for regular external triggers . . . 1491
Table 243. Configuring the trigger polarity for injected external triggers . . . 1491
Table 244. TSAR timings depending on resolution . . . 1494
Table 245. Offset computation versus data resolution . . . 1498
Table 246. 12-bit data formats . . . 1500
Table 247. Numerical examples for 32-bit or 16-bit format (POSSPFF = 0) . . . 1501
Table 248. Analog window watchdog features . . . 1509
Table 249. Analog watchdog 1 channel selection . . . 1510
Table 250. Analog watchdog 1, 2, 3 comparison . . . 1511
Table 251. Maximum output results versus N and M . . . 1516
Table 252. Oversampler operating mode summary . . . 1520
Table 253. DELAY bits versus ADC resolution . . . 1526
Table 254. ADC interrupts . . . 1539

Table 255. ADC register map and reset values for each ADC . . . 1575
Table 256. ADC register map and reset values (master and slave ADC common registers) . . . 1578
Table 257. DTS internal input/output signals . . . 1580
Table 258. DTS interconnection . . . 1581
Table 259. SDA slave registers . . . 1586
Table 260. SDA TS control register (SDATS_CR) . . . 1586
Table 261. SDA TS configuration register (SDATS_CFGR) . . . 1587
Table 262. SDA TS data register (SDATS_DR) . . . 1588
Table 263. SDA TS timer register (SDATS_TIMERR) . . . 1589
Table 264. Output resolution configuration . . . 1594
Table 265. DTS interrupts . . . 1595
Table 266. DTS register memory map . . . 1596
Table 267. PVT common register map and reset values . . . 1616
Table 268. PVT IRQ register map and reset values . . . 1616
Table 269. DTS TSC clock synthesizer register map and reset values . . . 1617
Table 270. DTS TS individual register map and reset values . . . 1619
Table 271. VREFBUF typical values . . . 1621
Table 272. VREF buffer modes . . . 1622
Table 273. VREFBUF register map and reset values . . . 1625
Table 274. MDF features . . . 1627
Table 275. MDF external pins . . . 1630
Table 276. MDF internal signals . . . 1630
Table 277. MDF trigger connections . . . 1630
Table 278. MDF break connections . . . 1631
Table 279. Control of the common clock generation . . . 1638
Table 280. Clock constraints with respect to the incoming stream . . . 1639
Table 281. Data size according to CIC order and CIC decimation values . . . 1646
Table 282. Maximum decimation ratio versus order and input data size . . . 1647
Table 283. Possible gain values . . . 1648
Table 284. Recommended maximum gain values versus CIC decimation ratios . . . 1650
Table 285. Most common microphone settings . . . 1650
Table 286. HPF 3 dB cut-off frequencies examples . . . 1652
Table 287. Register protection summary . . . 1673
Table 288. Effect of low-power modes on MDF . . . 1674
Table 289. MDF interrupt requests . . . 1676
Table 290. Examples of MDF settings for microphone capture . . . 1677
Table 291. Programming sequence . . . 1677
Table 292. Output signal levels . . . 1684
Table 293. MDF register map and reset values . . . 1707
Table 294. ADF features . . . 1710
Table 295. ADF external pins . . . 1711
Table 296. ADF internal signals . . . 1712
Table 297. ADF trigger connections . . . 1712
Table 298. Control of the common clock generation . . . 1718
Table 299. Clock constraints with respect to the incoming stream . . . 1719
Table 300. Data size according to CIC order and CIC decimation values . . . 1724
Table 301. Possible gain values . . . 1725
Table 302. Recommended maximum gain values versus CIC decimation ratios . . . 1727
Table 303. Most common microphone settings . . . 1728
Table 304. HPF 3 dB cut-off frequency examples . . . 1730

Table 305. ANSLP values versus FRSIZE and sampling rates . . . 1743
Table 306. Threshold values according SNTHR . . . 1744
Table 307. Register protection summary . . . 1750
Table 308. Effect of low-power modes on ADF . . . 1751
Table 309. ADF interrupt requests . . . 1752
Table 310. Examples of ADF settings for microphone capture . . . 1753
Table 311. Programming sequence (CIC4) . . . 1754
Table 312. Programming sequence (CIC5) . . . 1755
Table 313. Output signal levels . . . 1760
Table 314. ADF register map and reset values . . . 1780
Table 315. Camera subsystem RIF peripheral ID . . . 1789
Table 316. DCMI input/output pins . . . 1795
Table 317. DCMI internal input/output signals . . . 1795
Table 318. Positioning of captured data bytes in 32-bit words (8-bit width) . . . 1797
Table 319. Positioning of captured data bytes in 32-bit words (10-bit width) . . . 1797
Table 320. Positioning of captured data bytes in 32-bit words (12-bit width) . . . 1797
Table 321. Positioning of captured data bytes in 32-bit words (14-bit width) . . . 1798
Table 322. Data storage in monochrome progressive video format . . . 1803
Table 323. Data storage in RGB progressive video format . . . 1804
Table 324. Data storage in YCbCr progressive video format . . . 1804
Table 325. Data storage in YCbCr progressive video format - Y extraction mode . . . 1804
Table 326. DCMI interrupts . . . 1805
Table 327. DCMI register map and reset values . . . 1815
Table 328. Available pipelines . . . 1816
Table 329. Glossary . . . 1817
Table 330. DCMIPP input/output pads . . . 1822
Table 331. DCMIPP input/output pins . . . 1822
Table 332. DCMIPP clocks . . . 1822
Table 333. DCMIPP resets . . . 1823
Table 334. Parallel interface maximum resolution (80 MHz) . . . 1824
Table 335. Supported pixel formats . . . 1826
Table 336. DCMIPP_CMCR bit function . . . 1828
Table 337. DCMIPP_PRCR bit function . . . 1831
Table 338. DCMIPP_PRESCR and DCMIPP_PRESUR bit function . . . 1831
Table 339. DCMIPP_PxFCTCR bit function . . . 1839
Table 340. DCMIPP_P0PPCR bit function . . . 1842
Table 341. DCMIPP_P0DCCNTR and DCMIPP_P0DCLMTR bit function . . . 1845
Table 342. DCMIPP_P1SRCR bit function . . . 1848
Table 343. DCMIPP_P1BPRCR and DCMIPP_P1BPRSR bit function . . . 1848
Table 344. DCMIPP_P1DECR bit function . . . 1850
Table 345. DCMIPP_P1BLCCR bit function . . . 1850
Table 346. DCMIPP_P1EXCR1 and DCMIPP_P1EXCR2 bit function . . . 1851
Table 347. DCMIPP_P1DMCR bit function . . . 1853
Table 348. DCMIPP_PxCCyy (yy = R0, R1, G0, G1, B0, B1) bit function . . . 1854
Table 349. Color conversion: examples of coefficients . . . 1855
Table 350. DCMIPP_P1CTCR1,2,3 bit function . . . 1857
Table 351. DCMIPP_P1STyCR (y = 1, 2, 3), DCMIPP_P1STySR (y = 1, 2, 3), DCMIPP_P1STSTR and DCMIPP_P1STSZR bit function . . . 1859
Table 352. Statistics extraction: collected data vs. modes . . . 1860
Table 353. DCMIPP_PxCRSTR and DCMIPP_PxCRSZR bit function . . . 1864
Table 354. DCMIPP_PxDCCR bit function . . . 1865
Table 355. DCMIPP_PxDSCR, DCMIPP_PxDSRTIOR, DCMIPP_PxDSSZR . . . 1866

Table 356. DCMIPP_PxPPCR bit function . . . 1869
Table 357. Parallel interface input pixel formats . . . 1884
Table 358. Correspondence between index and DCMIPP_PRCR register values . . . 1885
Table 359. Parallel interface input pixel formats . . . 1886
Table 360. Pixel pipe output pixel format . . . 1886
Table 361. Dump pipe OUTPUT pixel formats . . . 1888
Table 362. Shadow and physical registers . . . 1895
Table 363. DCMIPP low power modes . . . 1897
Table 364. DCMIPP interrupts . . . 1900
Table 365. Event connection . . . 1901
Table 366. DCMIPP registers organization . . . 1902
Table 367. DCMIPP register map and reset values . . . 2001
Table 368. Glossary of terms . . . 2016
Table 369. Effect of low-power modes on CSI-2 Host controller . . . 2035
Table 370. List of errors detected by the CSI-2 Host controller . . . 2035
Table 371. List of events that trigger an interrupt . . . 2036
Table 372. CSI-2 Host and PHY register map and reset values . . . 2070
Table 373. PSSI input/output pins . . . 2076
Table 374. PSSI internal input/output signals . . . 2076
Table 375. Positioning of captured data bytes in 32-bit words (8-bit width) . . . 2077
Table 376. Positioning of captured data bytes in 32-bit words (16-bit width) . . . 2078
Table 377. PSSI interrupt requests . . . 2081
Table 378. PSSI register map and reset values . . . 2087
Table 379. Display subsystem RIF peripheral ID . . . 2092
Table 380. LTDC pins . . . 2095
Table 381. LTDC trigger interconnections . . . 2096
Table 382. Clock domain for each register . . . 2096
Table 383. LTDC register access and updated durations . . . 2098
Table 384. Pixel data mapping versus color format . . . 2101
Table 385. LTDC hardware triggers . . . 2116
Table 386. LTDC interrupt requests . . . 2118
Table 387. LTDC register map and reset values . . . 2149
Table 388. GPU2D internal input/output signals . . . 2155
Table 389. GPU2D trigger connections . . . 2156
Table 390. VENC register map and reset values . . . 2251
Table 391. JPEG internal signals . . . 2266
Table 392. JPEG trigger connections . . . 2267
Table 393. JPEG codec interrupt requests . . . 2272
Table 394. JPEG codec register map and reset values . . . 2285
Table 395. RNG internal input/output signals . . . 2288
Table 396. RNG interrupt requests . . . 2296
Table 397. RNG initialization times . . . 2297
Table 398. RNG configurations . . . 2297
Table 399. Configuration selection . . . 2298
Table 400. RNG register map and reset map . . . 2303
Table 401. CRYP versus SAES features . . . 2305
Table 402. SAES internal input/output signals . . . 2306
Table 403. SAES approved symmetric key functions . . . 2307
Table 404. Counter mode initialization vector definition . . . 2317
Table 405. Initialization of IV registers in GCM mode . . . 2320
Table 406. GCM last block definition . . . 2320
Table 407. Initialization of IV registers in CCM mode . . . 2326

Table 408. AES data swapping example . . . 2335
Table 409. Key endianness in SAES_KEYRx registers (128/256-bit keys) . . . 2337
Table 410. IVI bitfield spread over SAES_IVRx registers . . . 2339
Table 411. SAES interrupt requests . . . 2341
Table 412. Processing latency for ECB, CBC and CTR . . . 2342
Table 413. Processing latency for GCM and CCM (in SAES kernel clock cycles) . . . 2342
Table 414. SAES register map and reset values . . . 2357
Table 415. CRYP versus SAES features . . . 2360
Table 416. CRYP internal input/output signals . . . 2362
Table 417. CRYP approved AES symmetric key functions . . . 2362
Table 418. Counter mode initialization vector definition . . . 2373
Table 419. GCM mode IVI registers initialization . . . 2376
Table 420. GCM last block definition . . . 2376
Table 421. CCM mode IVI registers initialization . . . 2383
Table 422. AES data swapping example . . . 2388
Table 423. Key endianness in CRYP_KxR/LR registers (128/192/256-bit keys) . . . 2390
Table 424. Initialization vector endianness in CRYP_IVxR registers (AES) . . . 2390
Table 425. CRYP interrupt requests . . . 2392
Table 426. Processing latency for ECB, CBC and CTR . . . 2393
Table 427. Processing latency for GCM and CCM (in clock cycles) . . . 2394
Table 428. CRYP register map and reset values . . . 2407
Table 429. HASH internal input/output signals . . . 2410
Table 430. Information on supported hash algorithms . . . 2411
Table 431. Hash processor outputs . . . 2414
Table 432. Processing time (in clock cycle) . . . 2420
Table 433. HASH interrupt requests . . . 2421
Table 434. HASH1 register map and reset values . . . 2429
Table 435. MCE internal input/output signals . . . 2432
Table 436. MCE block cipher latencies . . . 2436
Table 437. MCE stream cipher latencies . . . 2437
Table 438. MCE interrupt requests . . . 2438
Table 439. MCE register map and reset values . . . 2450
Table 440. Internal input/output signals . . . 2453
Table 441. PKA integer arithmetic functions list . . . 2454
Table 442. PKA prime field (Fp) elliptic curve functions list . . . 2455
Table 443. Example of ‘a’ curve coefficient for ECC Fp scalar . . . 2461
Table 444. Montgomery parameter computation . . . 2461
Table 445. Modular addition . . . 2462
Table 446. Modular subtraction . . . 2462
Table 447. Montgomery multiplication . . . 2463
Table 448. Modular exponentiation (normal mode) . . . 2464
Table 449. Modular exponentiation (fast mode) . . . 2464
Table 450. Modular exponentiation (protected mode) . . . 2465
Table 451. Modular inversion . . . 2465
Table 452. Modular reduction . . . 2466
Table 453. Arithmetic addition . . . 2466
Table 454. Arithmetic subtraction . . . 2466
Table 455. Arithmetic multiplication . . . 2467
Table 456. Arithmetic comparison . . . 2467
Table 457. CRT exponentiation . . . 2468
Table 458. Point on elliptic curve Fp check . . . 2469
Table 459. ECC Fp scalar multiplication . . . 2469

Table 460. ECDSA sign - Inputs . . . 2471
Table 461. ECDSA sign - Outputs . . . 2471
Table 462. Extended ECDSA sign - additional outputs . . . 2472
Table 463. ECDSA verification - inputs . . . 2472
Table 464. ECDSA verification - outputs . . . 2473
Table 465. ECC complete addition . . . 2473
Table 466. ECC double base ladder . . . 2474
Table 467. ECC projective to affine . . . 2475
Table 468. Family of supported curves for ECC operations . . . 2476
Table 469. Modular exponentiation . . . 2477
Table 470. ECC scalar multiplication . . . 2477
Table 471. ECDSA signature average computation time . . . 2478
Table 472. ECDSA verification average computation times . . . 2478
Table 473. ECC double base ladder average computation times . . . 2478
Table 474. ECC projective to affine average computation times . . . 2478
Table 475. ECC complete addition average computation times . . . 2478
Table 476. Point on elliptic curve Fp check average computation times . . . 2478
Table 477. Montgomery parameters average computation times . . . 2479
Table 478. PKA interrupt requests . . . 2479
Table 479. PKA register map and reset values . . . 2484
Table 480. TIM input/output pins . . . 2487
Table 481. TIM internal input/output signals . . . 2487
Table 482. Interconnect to the tim_ti1 input multiplexer . . . 2488
Table 483. Interconnect to the tim_ti2 input multiplexer . . . 2488
Table 484. Interconnect to the tim_ti3 input multiplexer . . . 2489
Table 485. Interconnect to the tim_ti4 input multiplexer . . . 2489
Table 486. Internal trigger connection . . . 2489
Table 487. Interconnect to the tim_etr input multiplexer . . . 2490
Table 488. Timer break interconnect . . . 2490
Table 489. Timer break2 interconnect . . . 2490
Table 490. System break interconnect . . . 2490
Table 491. CCR and ARR register change dithering pattern . . . 2523
Table 492. CCR register change dithering pattern in center-aligned PWM mode . . . 2524
Table 493. Behavior of timer outputs versus tim_brk/tim_brk2 inputs . . . 2536
Table 494. Break protection disarming conditions . . . 2538
Table 495. Counting direction versus encoder signals (CC1P = CC2P = 0) . . . 2546
Table 496. Counting direction versus encoder signals and polarity settings . . . 2550
Table 497. DMA request . . . 2572
Table 498. Effect of low-power modes on TIM1/TIM8 . . . 2573
Table 499. Interrupt requests . . . 2573
Table 500. Output control bits for complementary tim_ocx and tim_ocxn channels with break feature . . . 2600
Table 501. TIMx register map and reset values . . . 2623
Table 502. STM32N6x5/x7xx general purpose timers . . . 2627
Table 503. TIM input/output pins . . . 2629
Table 504. TIM internal input/output signals . . . 2629
Table 505. Interconnect to the tim_ti1 input multiplexer . . . 2630
Table 506. Interconnect to the tim_ti2 input multiplexer . . . 2630
Table 507. Interconnect to the tim_ti3 input multiplexer . . . 2630
Table 508. Interconnect to the tim_ti4 input multiplexer . . . 2631
Table 509. TIMx internal trigger connection . . . 2631
Table 510. Interconnect to the tim_etr input multiplexer . . . 2631

Table 511. CCR and ARR register change dithering pattern . . . 2663
Table 512. CCR register change dithering pattern in center-aligned PWM mode . . . 2664
Table 513. Counting direction versus encoder signals(CC1P = CC2P = 0) . . . 2673
Table 514. Counting direction versus encoder signals and polarity settings . . . 2678
Table 515. DMA request . . . 2703
Table 516. Effect of low-power modes on TIM2/TIM3/TIM4/TIM5 . . . 2703
Table 517. Interrupt requests . . . 2704
Table 518. Output control bit for standard tim_ocx channels . . . 2725
Table 519. TIM2/TIM3/TIM4/TIM5 register map and reset values . . . 2737
Table 520. TIM internal input/output signals . . . 2741
Table 521. TIMx_ARR register change dithering pattern . . . 2750
Table 522. DMA request . . . 2752
Table 523. Effect of low-power modes on TIM6/TIM7/TIM18 . . . 2752
Table 524. Interrupt request . . . 2752
Table 525. TIMx register map and reset values . . . 2759
Table 526. TIM input/output pins . . . 2763
Table 527. TIM internal input/output signals . . . 2763
Table 528. Interconnect to the tim_ti1 input multiplexer . . . 2764
Table 529. Interconnect to the tim_ti2 input multiplexer . . . 2764
Table 530. TIMx internal trigger connection . . . 2764
Table 531. CCR and ARR register change dithering pattern . . . 2782
Table 532. Effect of low-power modes on TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 . . . 2792
Table 533. Interrupt requests . . . 2792
Table 534. Output control bit for standard tim_ocx channels . . . 2805
Table 535. TIMx register map and reset values (x = 9, 12) . . . 2809
Table 536. Output control bit for standard tim_ocx channels . . . 2818
Table 537. TIM10/TIM11/TIM13/TIM14 register map and reset values . . . 2821
Table 538. TIM input/output pins . . . 2826
Table 539. TIM internal input/output signals . . . 2827
Table 540. Interconnect to the tim_ti1 input multiplexer . . . 2828
Table 541. Interconnect to the tim_ti2 input multiplexer . . . 2828
Table 542. TIMx internal trigger connection . . . 2828
Table 543. Timer break interconnect . . . 2829
Table 544. System break interconnect . . . 2829
Table 545. CCR and ARR register change dithering pattern . . . 2849
Table 546. Break protection disarming conditions . . . 2858
Table 547. DMA request . . . 2871
Table 548. Effect of low-power modes on TIM15/TIM16/TIM17 . . . 2871
Table 549. Interrupt requests . . . 2872
Table 550. Output control bits for complementary tim_ocx and tim_ocxn channels with break feature (TIM15) . . . 2887
Table 551. TIM15 register map and reset values . . . 2900
Table 552. Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17) . . . 2913
Table 553. TIM16/TIM17 register map and reset values . . . 2927
Table 554. STM32N6x5/x7xx LPTIM features . . . 2930
Table 555. LPTIM1/2/3 input/output pins . . . 2932
Table 556. LPTIM4/5 input/output pins . . . 2932
Table 557. LPTIM1/2/3 internal signals . . . 2933
Table 558. LPTIM4/5 internal signals . . . 2933
Table 559. LPTIM1/2/3/4/5 external trigger connections . . . 2934
Table 560. LPTIM1/2/3 input 1 connections . . . 2934

Table 561. LPTIM1/2/3 input 2 connections . . . 2934
Table 562. LPTIM1/2/3 input capture 1 connections . . . 2934
Table 563. LPTIM1 input capture 2 connections . . . 2935
Table 564. LPTIM2 input capture 2 connections . . . 2935
Table 565. LPTIM3 input capture 2 connections . . . 2935
Table 566. Prescaler division ratios . . . 2937
Table 567. Encoder counting scenarios . . . 2944
Table 568. Input capture Glitch filter latency (in counter step unit) . . . 2948
Table 569. Effect of low-power modes on the LPTIM . . . 2952
Table 570. Interrupt events . . . 2953
Table 571. LPTIM register map and reset values . . . 2977
Table 572. IWDG features . . . 2980
Table 573. IWDG delays versus actions . . . 2981
Table 574. IWDG internal input/output signals . . . 2982
Table 575. Effect of low power modes on IWDG . . . 2987
Table 576. IWDG interrupt request . . . 2989
Table 577. IWDG register map and reset values . . . 2995
Table 578. WWDG features . . . 2996
Table 579. WWDG internal input/output signals . . . 2997
Table 580. WWDG interrupt requests . . . 3000
Table 581. WWDG register map and reset values . . . 3002
Table 582. RTC input/output pins . . . 3006
Table 583. RTC internal input/output signals . . . 3006
Table 584. RTC interconnection . . . 3007
Table 585. RTC pin PC13 configuration . . . 3007
Table 586. RTC_OUT mapping . . . 3009
Table 587. Effect of low-power modes on RTC . . . 3025
Table 588. RTC pins functionality over modes . . . 3025
Table 589. Nonsecure interrupt requests . . . 3026
Table 590. Secure interrupt requests . . . 3026
Table 591. RTC register map and reset values . . . 3056
Table 592. TAMP input/output pins . . . 3061
Table 593. TAMP internal input/output signals . . . 3061
Table 594. TAMP interconnection . . . 3062
Table 595. Device resource x tamper protection . . . 3067
Table 596. Active tamper output change period . . . 3071
Table 597. Minimum ATPER value . . . 3072
Table 598. Active tamper filtered pulse duration . . . 3073
Table 599. Effect of low-power modes on TAMP . . . 3075
Table 600. TAMP pins functionality over modes . . . 3075
Table 601. Interrupt requests . . . 3075
Table 602. TAMP register map and reset values . . . 3103
Table 603. I2C implementation . . . 3106
Table 604. I2C input/output pins . . . 3107
Table 605. I2C internal input/output signals . . . 3108
Table 606. Comparison of analog and digital filters . . . 3110
Table 607. I²C-bus and SMBus specification data setup and hold times . . . 3112
Table 608. I2C configuration . . . 3116
Table 609. I²C-bus and SMBus specification clock timings . . . 3127
Table 610. Timing settings for f_{I2CCLK} of 8 MHz . . . 3137
Table 611. Timing settings for f_{I2CCLK} of 16 MHz . . . 3137
Table 612. SMBus timeout specifications . . . 3139

Table 613. SMBus with PEC configuration . . . 3141
Table 614. TIMEOUTA[11:0] for maximum $t_{TIMEOUT}$ of 25 ms . . . 3142
Table 615. TIMEOUTB[11:0] for maximum $t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ of 8 ms . . . 3142
Table 616. TIMEOUTA[11:0] for maximum t_{IDLE} of 50 µs . . . 3142
Table 617. Effect of low-power modes to I2C . . . 3152
Table 618. I2C interrupt requests . . . 3152
Table 619. I2C register map and reset values . . . 3168
Table 620. I3C instantiation . . . 3171
Table 621. I3C wake-up . . . 3171
Table 622. I3C FIFOs implementation . . . 3171
Table 623. I3C interrupt(s) . . . 3172
Table 624. I3C peripheral controller/target features versus MIPI v1.1 . . . 3173
Table 625. I3C input/output pins . . . 3174
Table 626. I3C internal input/output signals . . . 3174
Table 627. I3C register usage . . . 3186
Table 628. I3C registers/fields usage versus controller state . . . 3187
Table 629. I3C registers/fields usage versus target state . . . 3190
Table 630. List of supported I3C CCCs, as controller/target . . . 3193
Table 631. I3C controller error management . . . 3225
Table 632. I3C target error management . . . 3227
Table 633. Effect of low-power modes . . . 3231
Table 634. I3C interrupt requests . . . 3232
Table 635. I3C register map and reset values . . . 3277
Table 636. Instance implementation on STM32N6x5/x7xx . . . 3281
Table 637. USART/LPUART features . . . 3282
Table 638. USART/UART input/output pins . . . 3284
Table 639. USART internal input/output signals . . . 3285
Table 640. Noise detection from sampled data . . . 3297
Table 641. Tolerance of the USART receiver when BRR [3:0] = 0000 . . . 3301
Table 642. Tolerance of the USART receiver when BRR[3:0] is different from 0000 . . . 3301
Table 643. USART frame formats . . . 3306
Table 644. Effect of low-power modes on the USART . . . 3329
Table 645. USART interrupt requests . . . 3330
Table 646. USART register map and reset values . . . 3368
Table 647. Instance implementation on STM32N6x5/x7xx . . . 3371
Table 648. USART/LPUART features . . . 3371
Table 649. LPUART input/output pins . . . 3374
Table 650. LPUART internal input/output signals . . . 3374
Table 651. Error calculation for programmed baud rates at lpuart_ker_ck_pres= 32.768 kHz . . . 3385
Table 652. Tolerance of the LPUART receiver . . . 3386
Table 654. Effect of low-power modes on the LPUART . . . 3398
Table 655. LPUART interrupt requests . . . 3399
Table 656. LPUART register map and reset values . . . 3425
Table 657. SPI features . . . 3428
Table 658. SPI/I2S input/output pins . . . 3431
Table 659. SPI internal input/output signals . . . 3431
Table 660. Effect of low-power modes on the SPI . . . 3458
Table 661. SPI wake-up and interrupt requests . . . 3459
Table 662. Bitfields usable in PCM/I2S mode . . . 3461
Table 663. WS and CK level before SPI/I2S is enabled when AFCNTR = 1 . . . 3470
Table 664. Serial data line swapping . . . 3470
Table 665. CLKGEN programming examples for usual I2S frequencies . . . 3474

Table 666. I2S interrupt requests . . . 3483
Table 667. SPI register map and reset values . . . 3500
Table 668. SAI features . . . 3503
Table 669. SAI internal input/output signals . . . 3505
Table 670. SAI input/output pins . . . 3505
Table 671. External synchronization selection . . . 3508
Table 672. MCLK_x activation conditions . . . 3513
Table 673. Clock generator programming examples . . . 3516
Table 674. SAI_A configuration for TDM mode . . . 3523
Table 675. TDM frame configuration examples . . . 3525
Table 676. SOPD pattern . . . 3529
Table 677. Parity bit calculation . . . 3529
Table 678. Audio sampling frequency versus symbol rates . . . 3530
Table 679. SAI interrupt sources . . . 3539
Table 680. SAI register map and reset values . . . 3568
Table 681. SPDIFRX internal input/output signals . . . 3571
Table 682. SPDIFRX pins . . . 3571
Table 683. Transition sequence for preamble . . . 3577
Table 684. Minimum spdifrx_ker_ck frequency versus audio sampling rate . . . 3587
Table 685. Conditions of spdifrx_symb_ck generation . . . 3588
Table 686. Bit field property versus SPDIFRX state . . . 3591
Table 687. SPDIFRX interface register map and reset values . . . 3603
Table 688. MDIOS input/output signals connected to package pins or balls . . . 3605
Table 689. MDIOS internal input/output signals . . . 3605
Table 690. Interrupt control bits . . . 3610
Table 691. MDIOS register map and reset values . . . 3615
Table 692. CAN subsystem I/O signals . . . 3617
Table 693. CAN subsystem I/O pins . . . 3618
Table 694. CAN triggers . . . 3618
Table 695. Main features . . . 3620
Table 696. DLC coding in FDCAN . . . 3625
Table 697. Example of filter configuration for Rx buffers . . . 3638
Table 698. Example of filter configuration for Debug messages . . . 3639
Table 699. Possible configurations for frame transmission . . . 3639
Table 700. Tx buffer/FIFO - Queue element size . . . 3640
Table 701. First byte of level 1 reference message . . . 3650
Table 702. First four bytes of level 2 reference message . . . 3651
Table 703. First four bytes of level 0 reference message . . . 3651
Table 704. TUR configuration example . . . 3652
Table 705. System matrix, Node A . . . 3657
Table 706. Trigger list, Node A . . . 3658
Table 707. Number of data bytes transmitted with a reference message . . . 3665
Table 708. Rx buffer and FIFO element . . . 3672
Table 709. Rx buffer and FIFO element description . . . 3672
Table 710. Tx buffer and FIFO element . . . 3674
Table 711. Tx buffer element description . . . 3674
Table 712. Tx Event FIFO element . . . 3676
Table 713. Tx Event FIFO element description . . . 3676
Table 714. Standard message ID filter element . . . 3677
Table 715. Standard message ID filter element field description . . . 3678
Table 716. Extended message ID filter element . . . 3679
Table 717. Extended message ID filter element field description . . . 3679

Table 718. Trigger memory element . . . 3680
Table 719. Trigger memory element description . . . 3680
Table 720. FDCAN register map and reset values . . . 3718
Table 721. FDCAN TT register map and reset values . . . 3740
Table 722. CCU register map and reset values . . . 3746
Table 723. USB2 OTG high-speed Port 1 pins . . . 3749
Table 724. USB2 OTG high-speed Port 1 internal signals . . . 3750
Table 725. USB2 OTG high-speed Port 1 interrupt . . . 3751
Table 726. USB2 OTG high-speed Port 1 wake-up events . . . 3751
Table 727. USB2 OTG high-speed Port 2 pins . . . 3753
Table 728. USB2 OTG high-speed Port 2 internal signals . . . 3753
Table 729. USB2 OTG high-speed Port 2 interrupts . . . 3755
Table 730. USB2 OTG high-speed Port 2 wake-up events . . . 3755
Table 731. USB Type-C and power delivery pins . . . 3758
Table 732. USB Type-C and power delivery internal signals . . . 3758
Table 733. USB Type-C and power delivery interrupt . . . 3759
Table 734. USB Type-C and power delivery events . . . 3759
Table 735. OTG speeds supported . . . 3761
Table 736. OTG implementation . . . 3763
Table 737. OTG input/output pins . . . 3764
Table 738. OTG input/output signals . . . 3765
Table 739. Compatibility of STM32 low power modes with the OTG . . . 3775
Table 740. Core global control and status registers (CSRs) . . . 3783
Table 741. Host-mode control and status registers (CSRs) . . . 3784
Table 742. Device-mode control and status registers . . . 3785
Table 743. Data FIFO (DFIFO) access register map . . . 3787
Table 744. Power and clock gating control and status registers . . . 3788
Table 745. TRDT values . . . 3794
Table 746. Minimum duration for soft disconnect . . . 3837
Table 747. OTG register map and reset values . . . 3863
Table 748. USBPHYC implementation . . . 3928
Table 749. USBPHYC register map and reset values . . . 3936
Table 750. UCPD implementation . . . 3938
Table 751. UCPD signals on pins . . . 3939
Table 752. UCPD internal signals . . . 3939
Table 753. 4b5b symbol encoding table . . . 3941
Table 754. Ordered sets . . . 3943
Table 755. Validation of ordered sets . . . 3943
Table 756. Data size . . . 3943
Table 757. Coding for ANAMODE, ANASUBMODE and link with TYPEC_VSTATE_CCx . . . 3952
Table 758. Type-C sequence (source: 3A); cable/sink connected (Rd on CC1; Ra on CC2) . . . 3953
Table 759. Effect of low power modes on the UCPD . . . 3955
Table 760. UCPD interrupt requests . . . 3956
Table 761. UCPD register map and reset values . . . 3971
Table 762. Ethernet peripheral pins . . . 3978
Table 763. Ethernet internal input/output signals . . . 3979
Table 764. Fixed priority scheme for DMA channels . . . 3993
Table 765. Routing DA/SA failed packets to Rx queue . . . 3997
Table 766. Routing DA/SA pass but VLAN filter fail packets to Rx queue . . . 3998
Table 767. Priority routing of filter pass packets when OMCBCQ = 0 . . . 3998
Table 768. Priority routing of filter pass packets when OMCBCQ = 1 . . . 3998
Table 769. Weight for DMA channels . . . 4001

Table 770. Example of credit value per transmit cycle 4002
Table 771. Double VLAN processing features in Tx path 4005
Table 772. Double VLAN processing in Rx path 4006
Table 773. VLAN insertion or replacement based on VLTI bit 4007
Table 774. Destination address filtering 4011
Table 775. Source address filtering 4012
Table 776. VLAN match status 4013
Table 777. OTS and ITS bit values with at least one perfect filter enabled 4016
Table 778. OTS and ITS bit values with only VLAN Hash filter enabled 4017
Table 779. Rx Queue routing table for Unicast-tagged packets 4018
Table 780. Ordinary clock: PTP messages for snapshot 4021
Table 781. End-to-end transparent clock: PTP messages for snapshot 4022
Table 782. Peer-to-peer transparent clock: PTP messages for snapshot 4023
Table 783. Egress and ingress latency for PHY interfaces 4027
Table 784. Minimum PTP clock frequency example 4027
Table 785. Message format defined in IEEE 1588-2008 4028
Table 786. Message format defined in IEEE 1588-2008 4029
Table 787. IPv6-UDP PTP packet fields required for control and status 4030
Table 788. Ethernet PTP packet fields required for control and status 4031
Table 789. Timestamp Snapshot Dependency on ETH_MACTSCR bits 4032
Table 790. PTP message generation criteria 4038
Table 791. Common PTP message header fields 4040
Table 792. MAC Transmit PTP mode and one-step timestamping operation 4043
Table 793. External memory used for holding the two gate-control lists 4048
Table 794. GCL and associated registers - BTR and CTR 4052
Table 795. GCL and associated registers - BTR and CTR, execution time > CycleTime 4053
Table 796. GCL and Associated Registers - BTR and CTR 4055
Table 797. Extending to New List By Truncating the Current List 4056
Table 798. Switching to New List By Extending the Current List 4057
Table 799. Gate Control List When FPE is disabled 4059
Table 800. Gate Control List When FPE is enabled 4059
Table 801. Valid SMD Values of mPacket 4062
Table 802. Valid frag_count values 4062
Table 803. Current and Previous SMD Values 4063
Table 804. Current and previous SMD values 4063
Table 805. Preemption Control Values on MRI Interface for Various Frame Types 4067
Table 806. MMC Counters and Associated Interrupt Registers 4068
Table 807. Transmit checksum offload engine functions for different packet types 4077
Table 808. Receive checksum offload engine functions for different packet types 4080
Table 809. TSO: TCP and IP header fields 4084
Table 810. Pause packet fields 4089
Table 811. Tx MAC flow control 4090
Table 812. Rx MAC flow control 4091
Table 813. Size of the maximum receive packet 4093
Table 814. MCD clock selection 4096
Table 815. MDIO Clause 45 frame structure 4097
Table 816. MDIO Clause 22 frame structure 4098
Table 817. Remote wake-up packet filter register 4111
Table 818. Description of the remote wake-up filter fields 4111
Table 819. Remote wake-up packet and PMT interrupt generation 4113
Table 820. Transfer complete interrupt behavior 4122
Table 821. TDES0 normal descriptor (read format) 4152

Table 822. TDES1 normal descriptor (read format) . . . 4153
Table 823. TDES2 normal descriptor (read format) . . . 4153
Table 824. TDES3 normal descriptor (read format) . . . 4154
Table 825. TDES0 normal descriptor (write-back format). . . . 4157
Table 826. TDES1 normal descriptor (write-back format). . . . 4157
Table 827. TDES2 normal descriptor (write-back format). . . . 4158
Table 828. TDES3 normal descriptor (write-back format). . . . 4158
Table 829. TDES0 context descriptor . . . 4162
Table 830. TDES1 context descriptor . . . 4162
Table 831. TDES2 context descriptor . . . 4163
Table 832. TDES3 context descriptor . . . 4163
Table 833. RDES0 normal descriptor (read format) . . . 4166
Table 834. RDES1 normal descriptor (read format) . . . 4166
Table 835. RDES2 normal descriptor (read format) . . . 4166
Table 836. RDES3 normal descriptor (read format) . . . 4167
Table 837. RDES0 normal descriptor (write-back format) . . . 4168
Table 838. RDES1 normal descriptor (write-back format) . . . 4169
Table 839. RDES2 normal descriptor (write-back format) . . . 4171
Table 840. RDES3 normal descriptor (write-back format) . . . 4173
Table 841. RDES0 context descriptor . . . 4176
Table 842. RDES1 context descriptor . . . 4177
Table 843. RDES2 context descriptor . . . 4177
Table 844. RDES3 context descriptor . . . 4177
Table 845. Enhanced normal descriptor (read format) . . . 4178
Table 846. ETH_DMA common register map and reset values . . . 4207
Table 847. ETH_DMA_CH0 register map and reset values . . . 4208
Table 848. ETH_DMA_CH1 register map and reset values . . . 4210
Table 849. ETH_MTL register map and reset values . . . 4239
Table 850. Giant Packet Status based on S2KP and JE Bits . . . 4248
Table 851. Packet Length based on the CST and ACS bits . . . 4248
Table 852. Ethernet MAC register map and reset values . . . 4356
Table 853. Muxing of HDP signals . . . 4368
Table 854. HDP register map and reset values . . . 4372
Table 855. SWJ-DP pins . . . 4377
Table 856. Trace port pins . . . 4377
Table 857. Serial-wire trace port pins . . . 4377
Table 858. Trigger pins . . . 4378
Table 859. Authentication signal initial states . . . 4381
Table 860. JTAG-DP data registers . . . 4385
Table 861. Packet request . . . 4386
Table 862. ACK response . . . 4387
Table 863. Data transfer . . . 4387
Table 864. Debug port registers . . . 4388
Table 865. Debug port register map and reset values . . . 4395
Table 866. AP1-2 authentication behavior . . . 4397
Table 867. AP0 register map and reset values . . . 4401
Table 868. AP1 register map and reset values . . . 4405
Table 869. AP2 register map and reset values . . . 4410
Table 870. System ROM table . . . 4411
Table 871. MCU ROM table . . . 4411
Table 872. System ROM table register map and reset values . . . 4416
Table 873. Processor ROM table . . . 4417

Table 874. Processor ROM register map and reset values 4422
Table 875. DWT register map and reset values 4435
Table 876. PMU register map and reset values 4450
Table 877. ITM register map and reset values 4460
Table 878. BPU register map and reset values 4467
Table 879. ETM register map and reset values 4492
Table 880. Trace subsystem ROM table 4496
Table 881. Trace subsystem ROM table register map and reset values 4502
Table 882. TSGEN register map and reset values 4509
Table 883. Trace and debug subsystem CTI_0 inputs 4512
Table 884. Trace and debug subsystem CTI_0 outputs 4512
Table 885. Trace and debug subsystem CTI_1 inputs 4512
Table 886. Trace and debug subsystem CTI_1 outputs 4513
Table 887. Cortex-M55 CTI inputs 4513
Table 888. Cortex-M55 CTI outputs 4513
Table 889. CTI register map and reset values 4527
Table 890. CSTF register map and reset values 4537
Table 891. ETF register map and reset values 4556
Table 892. ETR scatter-gather page format 4561
Table 893. ETR register map and reset values 4580
Table 894. TPIU register map and reset values 4597
Table 895. ETR replicator allocation 4599
Table 896. Replicator register map and reset values 4608
Table 897. SWO register map and reset values 4618
Table 898. STM extended stimulus port memory map 4620
Table 899. STM trace packet ID mapping to AXI masters 4621
Table 900. Hardware event mapping 4622
Table 901. STM register map and reset values 4645
Table 902. DBGMCU register map and reset values 4658
Table 903. Document revision history 4667

List of figures

Figure 1. Interconnect top view - STM32N6x7 devices . . . 160
Figure 2. Interconnect top view - STM32N6x5 devices . . . 161
Figure 3. Key management principle . . . 194
Figure 4. BSEC block diagram . . . 202
Figure 5. BSEC fuse mapping overview . . . 204
Figure 6. RISAF block diagram . . . 260
Figure 7. RISAF region programming . . . 261
Figure 8. RISAF region on-the-fly resizing . . . 264
Figure 9. IAC block diagram . . . 281
Figure 10. FLEXMEM versus retention . . . 291
Figure 11. ICACHE block diagram . . . 310
Figure 12. ICACHE TAG and data memories functional view . . . 312
Figure 13. CACHEAXI block diagram . . . 325
Figure 14. CACHEAXI TAG and data memories functional view . . . 328
Figure 15. Power control block diagram . . . 348
Figure 16. Power supply overview . . . 352
Figure 17. System supply configurations . . . 353
Figure 18. Device startup (VCORE supplied directly from SMPS step-down converter) . . . 354
Figure 19. Device startup (VCORE supplied from an external regulator) . . . 355
Figure 20. Backup domain . . . 358
Figure 21. Retention domain . . . 359
Figure 22. POR/PDR waveform . . . 361
Figure 23. BOR thresholds . . . 362
Figure 24. Vdda18pmu_ok reset threshold . . . 362
Figure 25. Vddcore_ok reset thresholds . . . 363
Figure 26. V_{DDCORE} monitoring . . . 364
Figure 27. PVD thresholds . . . 364
Figure 28. V_{08CAP} thresholds . . . 367
Figure 29. Temperature thresholds . . . 368
Figure 30. V_{CORE} voltage scaling versus system power modes . . . 373
Figure 31. Dynamic voltage scaling in Run mode . . . 374
Figure 32. Dynamic voltage scaling behavior in Stop mode . . . 375
Figure 33. Dynamic Voltage Scaling from Standby mode . . . 376
Figure 34. RCC block diagram . . . 412
Figure 35. Simplified reset circuit . . . 415
Figure 36. NRST reset pulse control . . . 418
Figure 37. Boot sequences versus system states . . . 421
Figure 38. Top-level clock tree . . . 423
Figure 39. HSE clock source . . . 425
Figure 40. HSE clock generation . . . 426
Figure 41. LSE clock generation . . . 427
Figure 42. HSI calibration flow . . . 430
Figure 43. MSI calibration flow . . . 431
Figure 44. PLL block diagram . . . 436
Figure 45. Spread spectrum modulation . . . 438
Figure 46. Core and bus clock generation . . . 442
Figure 47. Key signals controlling low-power modes . . . 443
Figure 48. Clock distribution for the NPU . . . 454

Figure 49. Clock distribution for PSSI, CSI, and DCMIPP . . . 455
Figure 50. Clock distribution for GPU, ICACHE, and GFXMMU . . . 456
Figure 51. Clock distribution for LTDC . . . 456
Figure 52. Clock distribution for VENC . . . 457
Figure 53. Clock distribution for OTG1, OTG2, and UCPD1 . . . 458
Figure 54. Clock distribution for ETH1 . . . 459
Figure 55. Clock management for ETH1 . . . 460
Figure 56. Clock distribution for MDIOS . . . 461
Figure 57. Clock distribution for FMC and MCE4 . . . 462
Figure 58. Clock distribution for XSPIs and MCE1/2/3 . . . 463
Figure 59. Clock distribution for SDMMCx and companions . . . 465
Figure 60. Clock distribution for ADCs . . . 466
Figure 61. Clock distribution for RTC . . . 466
Figure 62. Clock distribution for IWDG and WWDG . . . 467
Figure 63. Clock distribution for trace and debug . . . 468
Figure 64. Kernel clock switching . . . 469
Figure 65. Enable logic details for peripheral kernel clock . . . 470
Figure 66. Basic structure of an I/O port bit . . . 784
Figure 67. Input floating/pull-up/pull-down configurations . . . 788
Figure 68. Output configuration . . . 789
Figure 69. AF configuration . . . 790
Figure 70. Programmable analog configuration . . . 790
Figure 71. I/O compensation cell control overview . . . 810
Figure 72. HPDMA block diagram . . . 845
Figure 73. HPDMA channel direct programming without linked-list (HPDMA_CxLLR = 0) . . . 847
Figure 74. HPDMA channel suspend and resume sequence . . . 848
Figure 75. HPDMA channel abort and restart sequence . . . 849
Figure 76. Static linked-list data structure (all Uxx = 1) of a linear addressing channel x . . . 850
Figure 77. Static linked-list data structure (all Uxx = 1) of a 2D addressing channel x . . . 851
Figure 78. HPDMA dynamic linked-list data structure of linear addressing channel x . . . 852
Figure 79. HPDMA dynamic linked-list data structure of a 2D addressing channel x . . . 852
Figure 80. HPDMA channel execution and linked-list programming in run-to-completion mode (HPDMA_CxCR.LSM = 0) . . . 854
Figure 81. Inserting a LLIn with an auxiliary HPDMA channel y . . . 856
Figure 82. HPDMA channel execution and linked-list programming in link step mode (HPDMA_CxCR.LSM = 1) . . . 858
Figure 83. Building LLIn+1: HPDMA dynamic linked-lists in link step mode . . . 859
Figure 84. Replace with a new LLIn' in register file in link step mode . . . 860
Figure 85. Replace with a new LLIn' and LLIn+1' in memory in link step mode (option 1) . . . 861
Figure 86. Replace with a new LLIn' and LLIn+1' in memory in link step mode (option 2) . . . 862
Figure 87. HPDMA channel execution and linked-list programming . . . 864
Figure 88. Programmed 2D addressing . . . 867
Figure 89. HPDMA arbitration policy . . . 882
Figure 90. Trigger hit, memorization and overrun waveform . . . 886
Figure 91. HPDMA circular buffer programming: update of the memory start address with a linear addressing channel . . . 887
Figure 92. Shared HPDMA channel with circular buffering: update of the memory start address with a linear addressing channel . . . 888
Figure 93. GPDMA block diagram . . . 942
Figure 94. GPDMA channel direct programming without linked-list (GPDMA_CxLLR = 0) . . . 943
Figure 95. GPDMA channel suspend and resume sequence . . . 944
Figure 96. GPDMA channel abort and restart sequence . . . 945

Figure 97. Static linked-list data structure (all Uxx = 1) of a linear addressing channel x . . . 946
Figure 98. Static linked-list data structure (all Uxx = 1) of a 2D addressing channel x . . . 947
Figure 99. GPDMA dynamic linked-list data structure of a linear addressing channel x . . . 948
Figure 100. GPDMA dynamic linked-list data structure of a 2D addressing channel x . . . 948
Figure 101. GPDMA channel execution and linked-list programming in run-to-completion mode (GPDMA_CxCR.LSM = 0) . . . 950
Figure 102. Inserting a LLIn with an auxiliary GPDMA channel y . . . 952
Figure 103. GPDMA channel execution and linked-list programming in link step mode (GPDMA_CxCR.LSM = 1) . . . 954
Figure 104. Building LLIn+1: GPDMA dynamic linked-lists in link step mode . . . 955
Figure 105. Replace with a new LLIn' in register file in link step mode . . . 956
Figure 106. Replace with a new LLIn' and LLIn+1' in memory in link step mode (option 1) . . . 957
Figure 107. Replace with a new LLIn' and LLIn+1' in memory in link step mode (option 2) . . . 958
Figure 108. GPDMA channel execution and linked-list programming . . . 960
Figure 109. Programmed 2D addressing . . . 963
Figure 110. GPDMA arbitration policy . . . 970
Figure 111. Trigger hit, memorization, and overrun waveform . . . 973
Figure 112. GPDMA circular buffer programming: update of the memory start address with a linear addressing channel . . . 974
Figure 113. Shared GPDMA channel with circular buffering: update of the memory start address with a linear addressing channel . . . 975
Figure 114. Stream processing engine . . . 1012
Figure 115. Chaining of processing units . . . 1013
Figure 116. Neural-ART 14 NPU accelerator . . . 1014
Figure 117. Stream engines . . . 1017
Figure 118. Two dimensional lossy compression . . . 1018
Figure 119. Decompression unit (DECUN) . . . 1018
Figure 120. Convolutional accelerator (CONVACC) . . . 1019
Figure 121. Example of pooling operations . . . 1020
Figure 122. Pooling unit (POOL) . . . 1020
Figure 123. Activation unit (ACTIV) . . . 1021
Figure 124. Arithmetic unit (ARITH) . . . 1024
Figure 125. Memory to memory transfer . . . 1025
Figure 126. Simple processing . . . 1026
Figure 127. Multiple processing . . . 1026
Figure 128. Classical Conv-Pool-ReLU . . . 1026
Figure 129. Chained convolutions . . . 1027
Figure 130. Split convolution . . . 1027
Figure 131. Neural-ART 14 integration . . . 1030
Figure 132. DMA2D block diagram . . . 1034
Figure 133. Intel 8080 16-bit mode (RGB565) . . . 1041
Figure 134. Intel 8080 18/24-bit mode (RGB888) . . . 1041
Figure 135. GFXMMU block diagram . . . 1067
Figure 136. Virtual buffer . . . 1069
Figure 137. Virtual buffer and physical buffer memory map . . . 1070
Figure 138. MMU block diagram . . . 1071
Figure 139. Block validation/comparator implementation . . . 1073
Figure 140. GFXTIM block diagram . . . 1083

Figure 141. Clock generator . . . 1085
Figure 142. Waveforms in standalone . . . 1087
Figure 143. Active counters and signals in standalone . . . 1087
Figure 144. Waveforms with external HSYNC and VSYNC . . . 1087
Figure 145. Waveforms with external HSYNC only . . . 1088
Figure 146. Active counters and signals with external HSYNC only . . . 1088
Figure 147. Waveforms with external VSYNC only . . . 1088
Figure 148. Active counters with external VSYNC only . . . 1089
Figure 149. Prescaling when external VSYNC only . . . 1089
Figure 150. Waveforms with external CSYNC only . . . 1089
Figure 151. Active counters and signals with external CSYNC only . . . 1090
Figure 152. Prescaling when external CSYNC only . . . 1090
Figure 153. Tearing-effect configurations . . . 1092
Figure 154. Watchdog timer . . . 1093
Figure 155. EXTI block diagram . . . 1127
Figure 156. Configurable event trigger logic CPU wake-up . . . 1131
Figure 157. EXTI direct events . . . 1132
Figure 158. EXTI mux GPIO selection . . . 1133
Figure 159. CRC calculation unit block diagram . . . 1171
Figure 160. FMC block diagram . . . 1179
Figure 161. Mode 1 read access waveforms . . . 1194
Figure 162. Mode 1 write access waveforms . . . 1194
Figure 163. Mode A read access waveforms . . . 1196
Figure 164. Mode A write access waveforms . . . 1197
Figure 165. Mode 2 and mode B read access waveforms . . . 1199
Figure 166. Mode 2 write access waveforms . . . 1199
Figure 167. Mode B write access waveforms . . . 1200
Figure 168. Mode C read access waveforms . . . 1202
Figure 169. Mode C write access waveforms . . . 1202
Figure 170. Mode D read access waveforms . . . 1204
Figure 171. Mode D write access waveforms . . . 1205
Figure 172. Muxed read access waveforms . . . 1207
Figure 173. Muxed write access waveforms . . . 1208
Figure 174. Asynchronous wait during a read access waveforms . . . 1211
Figure 175. Asynchronous wait during a write access waveforms . . . 1211
Figure 176. Wait configuration waveforms . . . 1213
Figure 177. Synchronous multiplexed read mode waveforms - NOR, PSRAM (CRAM) . . . 1214
Figure 178. Synchronous multiplexed write mode waveforms - PSRAM (CRAM) . . . 1216
Figure 179. NAND flash controller waveforms for common memory access . . . 1229
Figure 180. NAND flash controller waveforms for TCLR and TAR timings . . . 1230
Figure 181. Access to NAND flash memory . . . 1231
Figure 182. NAND flash page (2 Kbytes) layout with BCH . . . 1235
Figure 183. Example of 2 Kbyte page program sequence with 4-bit BCH . . . 1237
Figure 184. Chained-page read command timings . . . 1239
Figure 185. Example of 2-Kbyte page read sequence with 4-bit BCH . . . 1240
Figure 186. SDRAM burst write waveforms (AXI 4-word transfer, no page boundary crossing) . . . 1269
Figure 187. SDRAM burst read waveforms (AXI 4-word transfer, no page boundary crossing) . . . 1270
Figure 188. SDRAM burst read crossing a row boundary . . . 1271
Figure 189. SDRAM burst write crossing a row boundary . . . 1272
Figure 190. Read followed by write operation . . . 1272

Figure 191. Write followed by read operation . . . 1273
Figure 192. SDRAM Self-refresh waveforms . . . 1274
Figure 193. Power-down mode . . . 1275
Figure 194. Power-down with Auto-refresh . . . 1276
Figure 195. XSPI block diagram for 16-bit configuration(1) . . . 1291
Figure 196. XSPI block diagram for dual-octal configuration(1) . . . 1292
Figure 197. XSPI block diagram for octal configuration . . . 1293
Figure 198. XSPI block diagram in quad configuration . . . 1294
Figure 199. XSPI block diagram for dual-quad configuration . . . 1295
Figure 200. SDR read command in 16-bit configuration . . . 1297
Figure 201. DTR read in octal-SPI mode with DQS (Macronix mode) example . . . 1300
Figure 202. SDR write command in octal-SPI mode example . . . 1302
Figure 203. DTR write in octal-SPI mode (Macronix mode) example . . . 1303
Figure 204. Example of HyperBus read operation (8-bit data mode) . . . 1306
Figure 205. HyperBus write operation with initial latency (8-bit data mode) . . . 1307
Figure 206. HyperBus read operation with additional latency (8-bit data mode) . . . 1308
Figure 207. HyperBus write operation with additional latency (8-bit data mode) . . . 1308
Figure 208. HyperBus write operation with no latency (register write) . . . 1309
Figure 209. HyperBus read operation page crossing with latency (8-bit data mode) . . . 1309
Figure 210. HyperBus write operation with initial latency (16-bit mode) . . . 1310
Figure 211. D0/D1 data ordering in octal-SPI DTR mode (Micron) - Read access . . . 1317
Figure 212. OctaRAM read operation with reverse data ordering D1/D0 . . . 1317
Figure 213. NCS when CKMODE = 0 (T = CLK period) . . . 1324
Figure 214. Example of software control of two external memories . . . 1325
Figure 215. Example of hardware-controlled extended memory support . . . 1326
Figure 216. XSPIM block diagram . . . 1359
Figure 217. XSPI direct octal mode . . . 1362
Figure 218. XSPI direct 16-bit mode . . . 1363
Figure 219. XSPI dual-octal mode . . . 1364
Figure 220. XSPI swapped (octal) mode . . . 1365
Figure 221. XSPI multiplexed mode to Port1 . . . 1366
Figure 222. XSPI multiplexed (octal and dual-octal) mode to Port 2 . . . 1367
Figure 223. XSPI1 and XSPI2 drive a single external memory (octal mode) . . . 1368
Figure 224. Single XSPI driving two external memories . . . 1369
Figure 225. SDMMC “no response” and “no data” operations . . . 1373
Figure 226. SDMMC (multiple) block read operation . . . 1373
Figure 227. SDMMC (multiple) block write operation . . . 1374
Figure 228. SDMMC (sequential) stream read operation . . . 1374
Figure 229. SDMMC (sequential) stream write operation . . . 1374
Figure 230. SDMMC block diagram . . . 1376
Figure 231. SDMMC Command and data phase relation . . . 1378
Figure 232. Control unit . . . 1380
Figure 233. Command/response path . . . 1381
Figure 234. Command path state machine (CPSM) . . . 1382
Figure 235. Data path . . . 1388
Figure 236. DDR mode data packet clocking . . . 1389
Figure 237. DDR mode CRC status / boot acknowledgment clocking . . . 1389
Figure 238. Data path state machine (DPSM) . . . 1390
Figure 239. CLKMUX unit . . . 1401
Figure 240. Linked list structures . . . 1404
Figure 241. Asynchronous interrupt generation . . . 1407
Figure 242. Synchronous interrupt period data read . . . 1408

Figure 243. Synchronous interrupt period data write . . . 1408
Figure 244. Asynchronous interrupt period data read . . . 1409
Figure 245. Asynchronous interrupt period data write . . . 1410
Figure 246. Clock stop with SDMMC_CK for DS, HS, SDR12, SDR25 . . . 1413
Figure 247. Clock stop with SDMMC_CK for DDR50, SDR50, SDR104 . . . 1413
Figure 248. Read Wait with SDMMC_CK < 50 MHz . . . 1414
Figure 249. Read Wait with SDMMC_CK > 50 MHz . . . 1414
Figure 250. CMD12 stream timing . . . 1417
Figure 251. CMD5 Sleep Awake procedure . . . 1419
Figure 252. Normal boot mode operation . . . 1421
Figure 253. Alternative boot mode operation . . . 1422
Figure 254. Command response R1b busy signaling . . . 1423
Figure 255. SDMMC state control . . . 1424
Figure 256. Card cycle power / power up diagram . . . 1425
Figure 257. Read block gap hardware flow timing . . . 1426
Figure 258. CMD11 signal voltage switch sequence . . . 1428
Figure 259. Voltage switch transceiver typical application . . . 1430
Figure 260. DLYB block diagram . . . 1458
Figure 261. ADC block diagram . . . 1469
Figure 262. ADC clock scheme . . . 1473
Figure 263. ADC1 connectivity . . . 1475
Figure 264. ADC2 connectivity . . . 1476
Figure 265. Enabling/disabling the ADC . . . 1481
Figure 266. Bulb mode timing diagram . . . 1484
Figure 267. Analog-to-digital conversion time . . . 1489
Figure 268. Stopping ongoing regular conversions . . . 1490
Figure 269. Stopping ongoing regular and injected conversions . . . 1490
Figure 270. Triggers shared between ADC master and slave . . . 1492
Figure 271. Injected conversion latency during ongoing regular conversion . . . 1493
Figure 272. Single conversions of a sequence, software trigger . . . 1495
Figure 273. Continuous conversion of a sequence, software trigger . . . 1495
Figure 274. Single conversions of a sequence, hardware trigger . . . 1496
Figure 275. Continuous conversions of a sequence, hardware trigger . . . 1496
Figure 276. Right alignment (offset disabled, unsigned value) . . . 1498
Figure 277. Right alignment (offset enabled, signed value) . . . 1499
Figure 278. Left alignment (offset disabled, unsigned value) . . . 1499
Figure 279. Left alignment (offset enabled, signed value) . . . 1500
Figure 280. Example of overrun (OVRMOD = 0) . . . 1503
Figure 281. Example of overrun (OVRMOD = 1) . . . 1503
Figure 282. AUTDLY = 1, regular conversion in continuous mode, software trigger . . . 1507
Figure 283. AUTDLY = 1, regular hardware conversions interrupted by injected conversions (DISCEN = 0; JDISCEN = 0) . . . 1507
Figure 284. AUTDLY = 1, regular hardware conversions interrupted by injected conversions (DISCEN = 1, JDISCEN = 1) . . . 1508
Figure 285. AUTDLY = 1, regular continuous conversions interrupted by injected conversions . . . 1508
Figure 286. AUTDLY = 1 in autoinjection mode (JAUTO = 1) . . . 1509
Figure 287. Analog watchdog guarded area . . . 1510
Figure 288. ADC_AWDy_OUT signal generation (on all regular channels) . . . 1512
Figure 289. ADC_AWDy_OUT signal generation (AWDy flag not cleared by software) . . . 1513
Figure 290. ADC_AWDy_OUT signal generation (on a single regular channel) . . . 1513
Figure 291. ADC_AWDy_OUT signal generation (on all injected channels) . . . 1514
Figure 292. 12-bit result oversampling with 10-bit right shift and rounding . . . 1515

Figure 293. Triggered regular oversampling mode (TROVS bit = 1) . . . 1517
Figure 294. Regular oversampling modes (4x ratio) . . . 1518
Figure 295. Regular and injected oversampling modes used simultaneously . . . 1519
Figure 296. Triggered regular oversampling with injection . . . 1519
Figure 297. Oversampling in autoinjection mode . . . 1520
Figure 298. Dual ADC block diagram . . . 1522
Figure 299. Injected simultaneous mode on four channels: Dual ADC mode . . . 1523
Figure 300. Regular simultaneous mode on 16 channels: dual ADC mode . . . 1525
Figure 301. Interleaved mode on 1 channel in continuous conversion mode: Dual ADC mode . . . 1527
Figure 302. Interleaved mode on 1 channel in single conversion mode: Dual ADC mode . . . 1527
Figure 303. Interleaved conversion with injection . . . 1528
Figure 304. Alternate trigger: injected group of each ADC . . . 1529
Figure 305. Alternate trigger: 4 injected channels (each ADC) in discontinuous mode . . . 1530
Figure 306. Alternate + regular simultaneous . . . 1530
Figure 307. Case of trigger occurring during injected conversion . . . 1531
Figure 308. Interleaved single channel CH0 with injected sequence CH11, CH12 . . . 1531
Figure 309. Two interleaved channels (CH1, CH2) with injected sequence CH11, CH12 - case 1: Master interrupted first . . . 1532
Figure 310. Two interleaved channels (CH1, CH2) with injected sequence CH11, CH12 - case 2: Slave interrupted first . . . 1532
Figure 311. DMA requests in regular simultaneous mode when DAMDF[1:0] = 0b00 . . . 1533
Figure 312. DMA requests in interleaved mode when DAMDF = 0b10 . . . 1534
Figure 313. VBAT channel block diagram . . . 1537
Figure 314. VREFINT channel block diagram . . . 1537
Figure 315. DTS block diagram . . . 1580
Figure 316. Serial data adapter . . . 1583
Figure 317. SDA data format . . . 1584
Figure 318. Falling alarms . . . 1589
Figure 319. Rising alarms . . . 1590
Figure 320. VREFBUF block diagram . . . 1621
Figure 321. MDF block diagram . . . 1629
Figure 322. SITFx overview . . . 1632
Figure 323. SPI timing example . . . 1633
Figure 324. Manchester timing example (SITFMOD = 11) . . . 1635
Figure 325. CKGEN overview . . . 1638
Figure 326. BSMX overview . . . 1640
Figure 327. SCD functional view . . . 1641
Figure 328. SCD timing example . . . 1641
Figure 329. DFLT overview . . . 1643
Figure 330. Programmable delay . . . 1644
Figure 331. CIC3 and CIC5 frequency response with decimation ratio = 32 . . . 1646
Figure 332. Reshape filter frequency response normalized (FRS / 2 = 1) . . . 1651
Figure 333. Out-of-limit detector thresholds . . . 1654
Figure 334. Trigger logic for DFLT and CKGEN . . . 1656
Figure 335. Asynchronous continuous mode (ACQMOD[2:0] = 0) . . . 1657
Figure 336. Asynchronous single-shot mode (ACQMOD[2:0] = 001) . . . 1658
Figure 337. Synchronous continuous mode (ACQMOD[2:0] = 010) . . . 1659
Figure 338. Synchronous single-shot mode (ACQMOD[2:0] = 011) . . . 1660
Figure 339. Window continuous mode (ACQMOD[2:0] = 100) . . . 1661
Figure 340. Snapshot mode example . . . 1663

Figure 341. Discard function example . . . 1664
Figure 342. Start sequence with DFLTEN, in continuous mode, audio configuration . . . 1665
Figure 343. Start sequence with trigger input, in continuous mode, motor configuration . . . 1666
Figure 344. Break interface simplified view . . . 1667
Figure 345. MDF_DFLTxDR data format . . . 1667
Figure 346. Data resynchronization . . . 1668
Figure 347. Data transfer in interleaved-transfer mode . . . 1669
Figure 348. Data path for interleaved- and independent-transfer modes . . . 1671
Figure 349. Example of overflow and transfer to memory . . . 1672
Figure 350. MDF interrupt interface . . . 1675
Figure 351. Sensor connection examples . . . 1680
Figure 352. Global frequency response . . . 1681
Figure 353. Detailed frequency response . . . 1681
Figure 354. Simplified DFLT view with gain information . . . 1683
Figure 355. ADF block diagram . . . 1711
Figure 356. SITF overview . . . 1713
Figure 357. SPI timing example . . . 1714
Figure 358. Manchester timing example (SITFMOD = 11) . . . 1715
Figure 359. CKGEN overview . . . 1718
Figure 360. BSMX overview . . . 1720
Figure 361. DFLT overview . . . 1721
Figure 362. Programmable delay . . . 1722
Figure 363. CIC4 and CIC5 frequency response with decimation ratio = 32 or 16 . . . 1723
Figure 364. Reshape filter frequency response normalized (FRS / 2 = 1) . . . 1729
Figure 365. Trigger logic for DFLT and CKGEN . . . 1731
Figure 366. Asynchronous continuous mode (ACQMOD[2:0] = 0) . . . 1732
Figure 367. Asynchronous single-shot mode (ACQMOD[2:0] = 001) . . . 1733
Figure 368. Synchronous continuous mode (ACQMOD[2:0] = 010) . . . 1734
Figure 369. Synchronous single-shot mode (ACQMOD[2:0] = 011) . . . 1735
Figure 370. Window continuous mode (ACQMOD[2:0] = 100) . . . 1736
Figure 371. Discard function example . . . 1738
Figure 372. Start sequence with DFLTEN, in continuous mode, audio configuration . . . 1738
Figure 373. SAD block diagram . . . 1739
Figure 374. SAD flow diagram . . . 1741
Figure 375. SAD timing diagram example . . . 1747
Figure 376. ADF_DFLTxDR data format . . . 1747
Figure 377. Data resynchronization . . . 1748
Figure 378. Example of overflow and transfer to memory . . . 1749
Figure 379. ADF interrupt interface . . . 1752
Figure 380. Sensor connection examples . . . 1757
Figure 381. Global frequency response . . . 1757
Figure 382. Detailed frequency response . . . 1758
Figure 383. Simplified DFLT view with gain information . . . 1760
Figure 384. SAD example working with SADMOD = 01 . . . 1763
Figure 385. SAD example working with SADMOD = 1x . . . 1764
Figure 386. Camera subsystem block diagram . . . 1786
Figure 387. Camera subsystem clock diagram . . . 1787
Figure 388. DCMI block diagram . . . 1795
Figure 389. DCMI signal waveforms . . . 1796
Figure 390. Timing diagram . . . 1798
Figure 391. Frame capture waveforms in snapshot mode . . . 1800
Figure 392. Frame capture waveforms in continuous grab mode . . . 1801

Figure 393. Coordinates and size of the window after cropping . . . 1801
Figure 394. Data capture waveforms . . . 1802
Figure 395. Pixel raster scan order . . . 1803
Figure 396. DCMIPP overview . . . 1817
Figure 397. DCMIPP block diagram . . . 1821
Figure 398. VC flow processed by one pipe . . . 1836
Figure 399. Two exclusive VC flows processed by one reconfigured pipe . . . 1837
Figure 400. Two overlapping VC flows processed by two pipes . . . 1837
Figure 401. Snapshot (CPTMODE = 1) and Continuous (CPTMODE = 0) capture modes . . . 1838
Figure 402. Pipe0 (dump) architecture overview . . . 1841
Figure 403. Pipe1 ISP architecture view . . . 1846
Figure 404. Block diagram . . . 1863
Figure 405. Downsize with coverage filtering . . . 1866
Figure 406. Example with two ROIs . . . 1867
Figure 407. Pipe2 architecture overview . . . 1872
Figure 408. Use case: Pipe0 (statistics), Pipe1 (pixels) . . . 1874
Figure 409. Statistics and RGB data for single virtual channel data flow . . . 1875
Figure 410. Three pipes within a single virtual channel in CSI2 mode . . . 1875
Figure 411. Handling interleaved packet . . . 1876
Figure 412. Handling interleaved packets across two pixel pipes . . . 1876
Figure 413. Interlaced video (based on single pixel pipe) . . . 1877
Figure 414. Interlaced video (based on two pixel pipes) . . . 1877
Figure 415. 3D sensors (top-bottom) processed by two pixel pipes . . . 1882
Figure 416. CSI-2 Host block diagram . . . 2018
Figure 417. CSI-2 Host architecture . . . 2019
Figure 418. CSI-2 Host clock domain . . . 2021
Figure 419. Lane merger input and output (one data lane) . . . 2022
Figure 420. Lane merger input and output (two data lanes) . . . 2022
Figure 421. Data lane mapping . . . 2022
Figure 422. Low-level protocol overview . . . 2023
Figure 423. CSI2- long-packet structure for a D-PHY physical layer . . . 2024
Figure 424. Short-packet structure based on D-PHY physical layer . . . 2024
Figure 425. DATA ID structure in short or long packet format . . . 2025
Figure 426. Data interleaving using data type . . . 2026
Figure 427. Data interleaving using virtual channels . . . 2027
Figure 428. RAW6 (BPP6) input and output format . . . 2028
Figure 429. Raw7 (BPP7) input and output format . . . 2029
Figure 430. Raw8 (BPP8) input and output format . . . 2030
Figure 431. Raw 10 (BPP10) input and output format . . . 2030
Figure 432. Raw12 (BPP12) input/output format . . . 2031
Figure 433. Raw14 (BPP14) input format . . . 2032
Figure 434. Raw16 (BPP16) input/output format . . . 2033
Figure 435. PSSI block diagram . . . 2075
Figure 436. Top-level block diagram . . . 2075
Figure 437. Data enable in receive mode waveform diagram (CKPOL=0) . . . 2079
Figure 438. Data enable waveform diagram in transmit mode (CKPOL=1) . . . 2079
Figure 439. Ready in receive mode waveform diagram (CKPOL=0) . . . 2080
Figure 440. Bidirectional PSSI_DE/PSSI_RDY waveform . . . 2081
Figure 441. Bidirectional PSSI_DE/PSSI_RDY connection diagram . . . 2081
Figure 442. Display subsystem block diagram . . . 2089
Figure 443. Display subsystem clock diagram . . . 2089
Figure 444. LTDC block diagram . . . 2095

Figure 445. Layer window programmable parameters . . . 2107
Figure 446. Blending two layers . . . 2108
Figure 447. LTDC synchronous timings . . . 2112
Figure 448. GPU2D block diagram . . . 2155
Figure 449. VENC block diagram . . . 2158
Figure 450. JPEG codec block diagram . . . 2266
Figure 451. RNG block diagram . . . 2288
Figure 452. NIST SP800-90B entropy source model . . . 2289
Figure 453. RNG initialization overview . . . 2292
Figure 454. SAES block diagram . . . 2306
Figure 455. Encryption/ decryption typical usage . . . 2308
Figure 456. Typical operation with authentication . . . 2310
Figure 457. Example of suspend mode management . . . 2311
Figure 458. ECB encryption . . . 2312
Figure 459. ECB decryption . . . 2312
Figure 460. CBC encryption . . . 2313
Figure 461. CBC decryption . . . 2313
Figure 462. Message construction in CTR mode . . . 2316
Figure 463. CTR encryption . . . 2317
Figure 464. Message construction in GCM . . . 2318
Figure 465. GCM authenticated encryption . . . 2320
Figure 466. Message construction in GMAC mode . . . 2323
Figure 467. GMAC authentication mode . . . 2323
Figure 468. Message construction in CCM mode . . . 2324
Figure 469. CCM mode authenticated encryption . . . 2326
Figure 470. Operation with wrapped keys for SAES in ECB and CBC modes . . . 2329
Figure 471. Operation with wrapped keys for SAES in CTR mode . . . 2332
Figure 472. Usage of Shared-key mode . . . 2333
Figure 473. 128-bit block construction according to the data type . . . 2336
Figure 474. Key protection mechanisms . . . 2338
Figure 475. CRYP block diagram . . . 2361
Figure 476. Encryption/ decryption typical usage . . . 2363
Figure 477. Typical operation with authentication . . . 2366
Figure 478. Example of suspend mode management . . . 2367
Figure 479. ECB encryption . . . 2368
Figure 480. ECB decryption . . . 2368
Figure 481. CBC encryption . . . 2369
Figure 482. CBC decryption . . . 2369
Figure 483. Message construction in CTR mode . . . 2372
Figure 484. CTR encryption . . . 2373
Figure 485. Message construction in GCM . . . 2375
Figure 486. GCM authenticated encryption . . . 2376
Figure 487. Message construction in GMAC mode . . . 2380
Figure 488. GMAC authentication mode . . . 2380
Figure 489. Message construction in CCM mode . . . 2381
Figure 490. CCM mode authenticated encryption . . . 2383
Figure 491. 128-bit block construction according to the data type . . . 2389
Figure 492. HASH block diagram . . . 2410
Figure 493. Message data swapping feature . . . 2412
Figure 494. HASH suspend/resume mechanism . . . 2418
Figure 495. MCE block diagram . . . 2432
Figure 496. MCE region programming . . . 2433

Figure 497. MCE implementation of block ciphers . . . 2435
Figure 498. MCE implementation of stream cipher . . . 2436
Figure 499. PKA block diagram . . . 2453
Figure 500. Advanced-control timer block diagram . . . 2486
Figure 501. Counter timing diagram with prescaler division change from 1 to 2 . . . 2492
Figure 502. Counter timing diagram with prescaler division change from 1 to 4 . . . 2492
Figure 503. Counter timing diagram, internal clock divided by 1 . . . 2494
Figure 504. Counter timing diagram, internal clock divided by 2 . . . 2494
Figure 505. Counter timing diagram, internal clock divided by 4 . . . 2495
Figure 506. Counter timing diagram, internal clock divided by N . . . 2495
Figure 507. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded) . . . 2496
Figure 508. Counter timing diagram, update event when ARPE = 1 (TIMx_ARR preloaded) . . . 2497
Figure 509. Counter timing diagram, internal clock divided by 1 . . . 2498
Figure 510. Counter timing diagram, internal clock divided by 2 . . . 2499
Figure 511. Counter timing diagram, internal clock divided by 4 . . . 2499
Figure 512. Counter timing diagram, internal clock divided by N . . . 2500
Figure 513. Counter timing diagram, update event when repetition counter is not used . . . 2500
Figure 514. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6 . . . 2502
Figure 515. Counter timing diagram, internal clock divided by 2 . . . 2502
Figure 516. Counter timing diagram, internal clock divided by 4, TIMx_ARR = 0x36 . . . 2503
Figure 517. Counter timing diagram, internal clock divided by N . . . 2503
Figure 518. Counter timing diagram, update event with ARPE = 1 (counter underflow) . . . 2504
Figure 519. Counter timing diagram, Update event with ARPE = 1 (counter overflow) . . . 2505
Figure 520. Update rate examples depending on mode and TIMx_RCR register settings . . . 2506
Figure 522. Control circuit in normal mode, internal clock divided by 1 . . . 2508
Figure 523. tim_ti2 external clock connection example . . . 2508
Figure 524. Control circuit in external clock mode 1 . . . 2509
Figure 525. External trigger input block . . . 2510
Figure 526. Control circuit in external clock mode 2 . . . 2511
Figure 527. Capture/compare channel (example: channel 1 input stage) . . . 2511
Figure 528. Capture/compare channel 1 main circuit . . . 2512
Figure 529. Output stage of capture/compare channel (channel 1, idem ch. 2, 3 and 4) . . . 2512
Figure 530. Output stage of capture/compare channel (channel 5, idem ch. 6) . . . 2513
Figure 531. PWM input mode timing . . . 2515
Figure 532. Output compare mode, toggle on tim_oc1 . . . 2517
Figure 533. Edge-aligned PWM waveforms (ARR = 8) . . . 2518
Figure 534. Center-aligned PWM waveforms (ARR = 8) . . . 2519
Figure 535. Dithering principle . . . 2520
Figure 536. Data format and register coding in dithering mode . . . 2521
Figure 537. PWM resolution vs frequency . . . 2522
Figure 538. PWM dithering pattern . . . 2523
Figure 539. Dithering effect on duty cycle in center-aligned PWM mode . . . 2524
Figure 540. Generation of 2 phase-shifted PWM signals with 50% duty cycle . . . 2526
Figure 541. Combined PWM mode on channel 1 and 3 . . . 2527
Figure 542. 3-phase combined PWM signals with multiple trigger pulses per period . . . 2528
Figure 543. Complementary output with symmetrical dead-time insertion . . . 2529
Figure 544. Asymmetrical deadtime . . . 2530
Figure 545. Dead-time waveforms with delay greater than the negative pulse . . . 2530
Figure 546. Dead-time waveforms with delay greater than the positive pulse . . . 2530
Figure 547. Break and Break2 circuitry overview . . . 2533

Figure 548. Various output behavior in response to a break event on tim_brk (OSSI = 1) 2535
Figure 549. PWM output state following tim_brk and tim_brk2 assertion (OSSI = 1) 2536
Figure 550. PWM output state following tim_brk assertion (OSSI = 0) 2537
Figure 551. Output redirection (tim_brk2 request not represented) 2538
Figure 552. Clearing TIMx tim_ocxref 2539
Figure 553. 6-step generation, COM example (OSSR = 1) 2540
Figure 554. Example of one pulse mode. 2541
Figure 555. Retriggerable one-pulse mode 2543
Figure 556. Pulse generator circuitry 2543
Figure 557. Pulse generation on compare event, for edge-aligned and encoder modes 2544
Figure 558. Extended pulsewidth in case of concurrent triggers 2545
Figure 559. Example of counter operation in encoder interface mode. 2547
Figure 560. Example of encoder interface mode with tim_ti1fp1 polarity inverted. 2547
Figure 561. Quadrature encoder counting modes 2548
Figure 562. Direction plus clock encoder mode 2549
Figure 563. Directional clock encoder mode (CC1P = CC2P = 0) 2549
Figure 564. Directional clock encoder mode (CC1P = CC2P = 1) 2550
Figure 565. Index gating options 2551
Figure 566. Jittered Index signals 2551
Figure 567. Index generation for IPOS[1:0] = 11 2552
Figure 568. Counter reading with index gated on channel A (IPOS[1:0] = 11) 2552
Figure 569. Counter reading with index ungated (IPOS[1:0] = 00) 2553
Figure 570. Counter reading with index gated on channel A and B. 2553
Figure 571. Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11) 2554
Figure 572. Counter reset Narrow index pulse (closer view, ARR = 0x07) 2555
Figure 573. Index behavior in x1 and x2 mode (IPOS[1:0] = 01) 2556
Figure 574. Directional index sensitivity. 2556
Figure 575. Counter reset as function of FIDX bit setting 2557
Figure 576. Index blanking. 2557
Figure 577. Index behavior in clock + direction mode, IPOS[0] = 1 2558
Figure 578. Index behavior in directional clock mode, IPOS[0] = 1 2558
Figure 579. State diagram for quadrature encoded signals 2559
Figure 580. Up-counting encoder error detection 2560
Figure 581. Down-counting encode error detection 2561
Figure 582. Encoder mode change with preload transferred on update (SMSPS = 0) 2562
Figure 583. Measuring time interval between edges on three signals 2563
Figure 584. Example of Hall sensor interface 2565
Figure 585. Control circuit in reset mode 2566
Figure 586. Control circuit in Gated mode 2567
Figure 587. Control circuit in trigger mode 2568
Figure 588. Control circuit in external clock mode 2 + trigger mode 2569
Figure 589. General-purpose timer block diagram 2628
Figure 590. Counter timing diagram with prescaler division change from 1 to 2 2633
Figure 591. Counter timing diagram with prescaler division change from 1 to 4 2634
Figure 592. Counter timing diagram, internal clock divided by 1 2635
Figure 593. Counter timing diagram, internal clock divided by 2 2635
Figure 594. Counter timing diagram, internal clock divided by 4 2636
Figure 595. Counter timing diagram, internal clock divided by N. 2636
Figure 596. Counter timing diagram, Update event when ARPE = 0 (TIMx_ARR not preloaded). . . 2637
Figure 597. Counter timing diagram, Update event when ARPE = 1 (TIMx_ARR preloaded). 2638
Figure 598. Counter timing diagram, internal clock divided by 1 2639
Figure 599. Counter timing diagram, internal clock divided by 2 2640

Figure 600. Counter timing diagram, internal clock divided by 4 2640
Figure 601. Counter timing diagram, internal clock divided by N. 2641
Figure 602. Counter timing diagram, Update event 2641
Figure 603. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6 2643
Figure 604. Counter timing diagram, internal clock divided by 2 2643
Figure 605. Counter timing diagram, internal clock divided by 4, TIMx_ARR = 0x36 2644
Figure 606. Counter timing diagram, internal clock divided by N. 2644
Figure 607. Counter timing diagram, Update event with ARPE = 1 (counter underflow). 2645
Figure 608. Counter timing diagram, Update event with ARPE = 1 (counter overflow). 2646
Figure 609. Control circuit in normal mode, internal clock divided by 1 2647
Figure 610. tim_ti2 external clock connection example 2647
Figure 611. Control circuit in external clock mode 1 2648
Figure 612. External trigger input block 2649
Figure 613. Control circuit in external clock mode 2 2650
Figure 614. Capture/compare channel (example: channel 1 input stage) 2650
Figure 615. Capture/compare channel 1 main circuit 2651
Figure 616. Output stage of capture/compare channel (channel 1, idem ch.2, 3 and 4). 2651
Figure 617. PWM input mode timing 2654
Figure 618. Output compare mode, toggle on tim_oc1 2656
Figure 619. Edge-aligned PWM waveforms (ARR = 8) 2657
Figure 620. Center-aligned PWM waveforms (ARR = 8). 2658
Figure 621. Dithering principle 2659
Figure 622. Data format and register coding in dithering mode. 2660
Figure 623. PWM resolution vs frequency (16-bit mode). 2661
Figure 624. PWM resolution vs frequency (32-bit mode). 2661
Figure 625. PWM dithering pattern 2662
Figure 626. Dithering effect on duty cycle in center-aligned PWM mode 2663
Figure 627. Generation of two phase-shifted PWM signals with 50% duty cycle 2665
Figure 628. Combined PWM mode on channels 1 and 3 2666
Figure 629. OCREF_CLR input selection multiplexer 2667
Figure 630. Clearing TIMx tim_ocxref 2667
Figure 631. Example of One-pulse mode 2668
Figure 632. Retriggerable one-pulse mode 2670
Figure 633. Pulse generator circuitry 2671
Figure 634. Pulse generation on compare event, for edge-aligned and encoder modes 2671
Figure 635. Extended pulse width in case of concurrent triggers 2672
Figure 636. Example of counter operation in encoder interface mode 2674
Figure 637. Example of encoder interface mode with tim_ti1fp1 polarity inverted 2674
Figure 638. Quadrature encoder counting modes 2675
Figure 639. Direction plus clock encoder mode. 2676
Figure 640. Directional clock encoder mode (CC1P = CC2P = 0). 2677
Figure 641. Directional clock encoder mode (CC1P = CC2P = 1). 2677
Figure 642. Index gating options 2679
Figure 643. Jittered Index signals 2679
Figure 644. Index generation for IPOS[1:0] = 11 2680
Figure 645. Counter reading with index gated on channel A (IPOS[1:0] = 11) 2680
Figure 646. Counter reading with index ungated (IPOS[1:0] = 00) 2681
Figure 647. Counter reading with index gated on channel A and B. 2681
Figure 648. Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11). 2682
Figure 649. Counter reset Narrow index pulse (closer view, ARR = 0x07) 2683
Figure 650. Index behavior in x1 and x2 mode (IPOS[1:0] = 01). 2684
Figure 651. Directional index sensitivity. 2684

Figure 652. Counter reset as function of FIDX bit setting . . . 2685
Figure 653. Index blanking . . . 2685
Figure 654. Index behavior in clock + direction mode, IPOS[0] = 1 . . . 2686
Figure 655. Index behavior in directional clock mode, IPOS[0] = 1 . . . 2686
Figure 656. State diagram for quadrature encoded signals . . . 2687
Figure 657. Up-counting encoder error detection . . . 2688
Figure 658. Down-counting encode error detection . . . 2689
Figure 659. Encoder mode change with preload transferred on update (SMSPS = 0) . . . 2690
Figure 660. Control circuit in reset mode . . . 2692
Figure 661. Control circuit in gated mode . . . 2693
Figure 662. Control circuit in trigger mode . . . 2693
Figure 663. Control circuit in external clock mode 2 + trigger mode . . . 2695
Figure 664. Master/Slave timer example . . . 2695
Figure 665. Master/slave connection example with 1 channel only timers . . . 2696
Figure 666. Gating TIM_slv with tim_oc1ref of TIM_mstr . . . 2697
Figure 667. Gating TIM_slv with Enable of TIM_mstr . . . 2698
Figure 668. Triggering TIM_slv with update of TIM_mstr . . . 2699
Figure 669. Triggering TIM_slv with Enable of TIM_mstr . . . 2699
Figure 670. Triggering TIM_mstr and TIM_slv with TIM_mstr tim_ti1 input . . . 2700
Figure 671. Basic timer block diagram . . . 2740
Figure 672. Control circuit in normal mode, internal clock divided by 1 . . . 2741
Figure 673. Counter timing diagram with prescaler division change from 1 to 2 . . . 2743
Figure 674. Counter timing diagram with prescaler division change from 1 to 4 . . . 2743
Figure 675. Counter timing diagram, internal clock divided by 1 . . . 2744
Figure 676. Counter timing diagram, internal clock divided by 2 . . . 2745
Figure 677. Counter timing diagram, internal clock divided by 4 . . . 2745
Figure 678. Counter timing diagram, internal clock divided by N . . . 2746
Figure 679. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded) . . . 2746
Figure 680. Counter timing diagram, update event when ARPE = 1 (TIMx_ARR preloaded) . . . 2747
Figure 681. Dithering principle . . . 2748
Figure 682. Data format and register coding in dithering mode . . . 2748
Figure 683. FCnt resolution vs frequency . . . 2749
Figure 684. PWM dithering pattern . . . 2749
Figure 685. General-purpose timer block diagram (TIM9/TIM12) . . . 2762
Figure 686. General-purpose timer block diagram (TIM10/TIM11/TIM13/TIM14) . . . 2763
Figure 687. Counter timing diagram with prescaler division change from 1 to 2 . . . 2766
Figure 688. Counter timing diagram with prescaler division change from 1 to 4 . . . 2766
Figure 689. Counter timing diagram, internal clock divided by 1 . . . 2767
Figure 690. Counter timing diagram, internal clock divided by 2 . . . 2768
Figure 691. Counter timing diagram, internal clock divided by 4 . . . 2768
Figure 692. Counter timing diagram, internal clock divided by N . . . 2769
Figure 693. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded) . . . 2769
Figure 694. Counter timing diagram, update event when ARPE = 1 (TIMx_ARR preloaded) . . . 2770
Figure 695. Control circuit in normal mode, internal clock divided by 1 . . . 2771
Figure 696. tim_ti2 external clock connection example . . . 2771
Figure 697. Control circuit in external clock mode 1 . . . 2772
Figure 698. Capture/compare channel 1 input stage (TIM10/TIM11/TIM13/TIM14) . . . 2773
Figure 699. Capture/compare channel 1 input stage (TIM9/TIM12) . . . 2773

Figure 700. Capture/compare channel 1 main circuit . . . 2774
Figure 701. Output stage of capture/compare channel 1 . . . 2774
Figure 702. PWM input mode timing . . . 2776
Figure 703. Output compare mode, toggle on tim_oc1 . . . 2778
Figure 704. Edge-aligned PWM waveforms (ARR = 8) . . . 2779
Figure 705. Dithering principle . . . 2780
Figure 706. Data format and register coding in dithering mode . . . 2780
Figure 707. PWM resolution vs frequency . . . 2781
Figure 708. PWM dithering pattern . . . 2782
Figure 709. Combined PWM mode on channel 1 and 2 . . . 2784
Figure 710. Example of one pulse mode . . . 2785
Figure 711. Retriggerable one pulse mode . . . 2786
Figure 712. Measuring time interval between edges on 2 signals . . . 2787
Figure 713. Control circuit in reset mode . . . 2788
Figure 714. Control circuit in gated mode . . . 2789
Figure 715. Control circuit in trigger mode . . . 2789
Figure 716. TIM15 block diagram . . . 2825
Figure 717. TIM16/TIM17 block diagram . . . 2826
Figure 718. Counter timing diagram with prescaler division change from 1 to 2 . . . 2830
Figure 719. Counter timing diagram with prescaler division change from 1 to 4 . . . 2831
Figure 720. Counter timing diagram, internal clock divided by 1 . . . 2832
Figure 721. Counter timing diagram, internal clock divided by 2 . . . 2833
Figure 722. Counter timing diagram, internal clock divided by 4 . . . 2833
Figure 723. Counter timing diagram, internal clock divided by N . . . 2834
Figure 724. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded) . . . 2834
Figure 725. Counter timing diagram, update event when ARPE = 1 (TIMx_ARR preloaded) . . . 2835
Figure 726. Update rate examples depending on mode and TIMx_RCR register settings . . . 2836
Figure 727. Control circuit in normal mode, internal clock divided by 1 . . . 2837
Figure 728. tim_ti2 external clock connection example . . . 2837
Figure 729. Control circuit in external clock mode 1 . . . 2838
Figure 730. Capture/compare channel (example: channel 1 input stage) . . . 2839
Figure 731. Capture/compare channel 1 main circuit . . . 2839
Figure 732. Output stage of capture/compare channel (channel 1) . . . 2840
Figure 733. Output stage of capture/compare channel (channel 2 for TIM15) . . . 2840
Figure 734. PWM input mode timing . . . 2843
Figure 735. Output compare mode, toggle on tim_oc1 . . . 2845
Figure 736. Edge-aligned PWM waveforms (ARR = 8) . . . 2846
Figure 737. Dithering principle . . . 2847
Figure 738. Data format and register coding in dithering mode . . . 2847
Figure 739. PWM resolution vs frequency . . . 2848
Figure 740. PWM dithering pattern . . . 2849
Figure 741. Combined PWM mode on channel 1 and 2 . . . 2851
Figure 742. Complementary output with symmetrical dead-time insertion . . . 2852
Figure 743. Asymmetrical deadtime . . . 2853
Figure 744. Dead-time waveforms with delay greater than the negative pulse . . . 2853
Figure 745. Dead-time waveforms with delay greater than the positive pulse . . . 2853
Figure 746. Break circuitry overview . . . 2855
Figure 747. Output behavior in response to a break event on tim_brk . . . 2857
Figure 748. Output redirection . . . 2859
Figure 749. tim_ocref_clr input selection multiplexer . . . 2860

Figure 750. 6-step generation, COM example (OSSR = 1) . . . 2861
Figure 751. Example of one pulse mode. . . . 2862
Figure 752. Retriggerable one pulse mode . . . 2864
Figure 753. Measuring time interval between edges on 2 signals . . . 2864
Figure 754. Control circuit in reset mode . . . 2865
Figure 755. Control circuit in gated mode . . . 2866
Figure 756. Control circuit in trigger mode . . . 2867
Figure 757. LPTIM1/2/3 block diagram(1) . . . 2931
Figure 758. LPTIM4/5 block diagram(1) . . . 2932
Figure 759. Glitch filter timing diagram . . . 2936
Figure 760. LPTIM output waveform, single-counting mode configuration when repetition register content is different than zero (with PRELOAD = 1) . . . 2938
Figure 761. LPTIM output waveform, single-counting mode configuration and Set-once mode activated (WAVE bit is set) . . . 2939
Figure 762. LPTIM output waveform, Continuous counting mode configuration . . . 2939
Figure 763. Waveform generation . . . 2941
Figure 764. Encoder mode counting sequence . . . 2945
Figure 765. Continuous counting mode when repetition register LPTIM_RCR different from zero (with PRELOAD = 1). . . . 2946
Figure 766. Capture/compare input stage (channel 1) . . . 2947
Figure 767. Capture/compare output stage (channel 1) . . . 2947
Figure 768. Edge-aligned PWM mode (PRELOAD = 1) . . . 2949
Figure 769. Edge-aligned PWM waveforms (ARR=8 and CCxP = 0) . . . 2950
Figure 770. PWM mode with immediate update versus preloaded update . . . 2951
Figure 771. Independent watchdog block diagram . . . 2981
Figure 772. Reset timing due to timeout . . . 2983
Figure 773. Reset timing due to refresh in the not allowed area . . . 2984
Figure 774. Changing PR, RL, and performing a refresh(1) . . . 2985
Figure 775. Window comparator update(1) . . . 2986
Figure 776. Independent watchdog interrupt timing diagram . . . 2988
Figure 777. Early wake-up comparator update(1) . . . 2989
Figure 778. Watchdog block diagram . . . 2997
Figure 779. Window watchdog timing diagram . . . 2999
Figure 780. RTC block diagram . . . 3005
Figure 781. TAMP block diagram . . . 3060
Figure 782. Backup registers protection zones . . . 3064
Figure 783. Tamper sampling with precharge pulse . . . 3069
Figure 784. Low level detection with precharge and filtering . . . 3070
Figure 785. Active tamper filtering . . . 3073
Figure 786. Block diagram . . . 3107
Figure 787. I²C-bus protocol . . . 3109
Figure 788. Setup and hold timings . . . 3111
Figure 789. I2C initialization flow . . . 3113
Figure 790. Data reception . . . 3114
Figure 791. Data transmission . . . 3115
Figure 792. Target initialization flow . . . 3118
Figure 793. Transfer sequence flow for I2C target transmitter, NOSTRETCH = 0 . . . 3120
Figure 794. Transfer sequence flow for I2C target transmitter, NOSTRETCH = 1 . . . 3121
Figure 795. Transfer bus diagrams for I2C target transmitter (mandatory events only) . . . 3122
Figure 796. Transfer sequence flow for I2C target receiver, NOSTRETCH = 0 . . . 3123
Figure 797. Transfer sequence flow for I2C target receiver, NOSTRETCH = 1 . . . 3124
Figure 798. Transfer bus diagrams for I2C target receiver

(mandatory events only) . . . 3124
Figure 799. Controller clock generation . . . 3126
Figure 800. Controller initialization flow . . . 3128
Figure 801. 10-bit address read access with HEAD10R = 0 . . . 3128
Figure 802. 10-bit address read access with HEAD10R = 1 . . . 3129
Figure 803. Transfer sequence flow for I2C controller transmitter, N ≤ 255 bytes . . . 3130
Figure 804. Transfer sequence flow for I2C controller transmitter, N > 255 bytes . . . 3131
Figure 805. Transfer bus diagrams for I2C controller transmitter (mandatory events only) . . . 3132
Figure 806. Transfer sequence flow for I2C controller receiver, N ≤ 255 bytes . . . 3134
Figure 807. Transfer sequence flow for I2C controller receiver, N > 255 bytes . . . 3135
Figure 808. Transfer bus diagrams for I2C controller receiver (mandatory events only) . . . 3136
Figure 809. Timeout intervals for $t_{LOW:SEXT}$, $t_{LOW:MEXT}$ . . . 3140
Figure 810. Transfer sequence flow for SMBus target transmitter N bytes + PEC . . . 3143
Figure 811. Transfer bus diagram for SMBus target transmitter (SBC = 1) . . . 3144
Figure 812. Transfer sequence flow for SMBus target receiver N bytes + PEC . . . 3145
Figure 813. Bus transfer diagrams for SMBus target receiver (SBC = 1) . . . 3146
Figure 814. Bus transfer diagrams for SMBus controller transmitter . . . 3147
Figure 815. Bus transfer diagrams for SMBus controller receiver . . . 3149
Figure 816. I3C block diagram . . . 3174
Figure 817. I3C (primary) controller state and programming sequence diagram . . . 3178
Figure 818. I3C target state and programing sequence diagram . . . 3183
Figure 819. I3C CCC messages, as controller . . . 3197
Figure 820. I3C broadcast ENTDAA CCC, as controller . . . 3198
Figure 821. I3C broadcast, direct read and direct write RSTACT CCC, as controller . . . 3199
Figure 822. I3C CCC messages, as target . . . 3201
Figure 823. I3C broadcast ENTDAA CCC, as target . . . 3202
Figure 824. I3C broadcast DEFTGTS CCC, as target . . . 3203
Figure 825. I3C broadcast DEFGRPA CCC, as target . . . 3204
Figure 826. I3C private read/write messages, as controller . . . 3206
Figure 827. I3C private read/write messages, as target . . . 3207
Figure 828. Legacy I2C read/write messages, as controller . . . 3208
Figure 829. IBI transfer, as controller/target . . . 3209
Figure 830. Hot-join request transfer, as controller/target . . . 3211
Figure 831. Controller-role request transfer, as controller/target . . . 3212
Figure 832. C-FIFO management, as controller . . . 3213
Figure 833. TX-FIFO management, as controller . . . 3215
Figure 834. RX-FIFO management, as controller . . . 3217
Figure 835. S-FIFO management, as controller . . . 3220
Figure 836. RX-FIFO management, as target on the I3C bus . . . 3221
Figure 837. TX-FIFO management with I3C_TGTTDR, as target on the I3C bus . . . 3223
Figure 838. TX-FIFO management by software without I3C_TGTTDR if reading less bytes than TX-FIFO size, as target . . . 3225
Figure 839. USART block diagram . . . 3283
Figure 840. Word length programming . . . 3287
Figure 841. Configurable stop bits . . . 3289
Figure 842. TC/TXE behavior when transmitting . . . 3291
Figure 843. Start bit detection when oversampling by 16 or 8 . . . 3292
Figure 844. usart_ker_ck clock divider block diagram . . . 3295
Figure 845. Data sampling when oversampling by 16 . . . 3296
Figure 846. Data sampling when oversampling by 8 . . . 3297

Figure 847. Mute mode using Idle line detection . . . 3304
Figure 848. Mute mode using address mark detection . . . 3305
Figure 849. Break detection in LIN mode (11-bit break length - LBDL bit is set) . . . 3308
Figure 850. Break detection in LIN mode vs. Framing error detection . . . 3309
Figure 851. USART example of synchronous master transmission . . . 3310
Figure 852. USART data clock timing diagram in synchronous master mode (M bits = 00) . . . 3310
Figure 853. USART data clock timing diagram in synchronous master mode (M bits = 01) . . . 3311
Figure 854. USART data clock timing diagram in synchronous slave mode (M bits = 00) . . . 3312
Figure 855. ISO 7816-3 asynchronous protocol . . . 3314
Figure 856. Parity error detection using the 1.5 stop bits . . . 3316
Figure 857. IrDA SIR ENDEC block diagram . . . 3320
Figure 858. IrDA data modulation (3/16) - normal mode . . . 3320
Figure 859. Transmission using DMA . . . 3322
Figure 860. Reception using DMA . . . 3323
Figure 861. Hardware flow control between two USARTs . . . 3323
Figure 862. RS232 RTS flow control . . . 3324
Figure 863. RS232 CTS flow control . . . 3325
Figure 864. Wake-up event verified (wake-up event = address match, FIFO disabled) . . . 3328
Figure 865. Wake-up event not verified (wake-up event = address match, FIFO disabled) . . . 3328
Figure 866. LPUART block diagram . . . 3373
Figure 867. LPUART word length programming . . . 3376
Figure 868. Configurable stop bits . . . 3378
Figure 869. TC/TXE behavior when transmitting . . . 3380
Figure 870. lpuart_ker_ck clock divider block diagram . . . 3384
Figure 871. Mute mode using Idle line detection . . . 3388
Figure 872. Mute mode using address mark detection . . . 3389
Figure 873. Transmission using DMA . . . 3391
Figure 874. Reception using DMA . . . 3392
Figure 875. Hardware flow control between two LPUARTs . . . 3393
Figure 876. RS232 RTS flow control . . . 3393
Figure 877. RS232 CTS flow control . . . 3394
Figure 878. Wake-up event verified (wake-up event = address match, FIFO disabled) . . . 3397
Figure 879. Wake-up event not verified (wake-up event = address match, FIFO disabled) . . . 3397
Figure 880. SPI/I2S block diagram . . . 3429
Figure 881. Full-duplex single master/ single slave application . . . 3432
Figure 882. Half-duplex single master/ single slave application . . . 3433
Figure 883. Simplex single master / single slave application (master in transmit-only / slave in receive-only mode) . . . 3434
Figure 884. Master and three independent slaves connected in star topology . . . 3435
Figure 885. Master and three slaves connected in circular (daisy chain) topology . . . 3437
Figure 886. Multimaster application . . . 3438
Figure 887. Scheme of NSS control logic . . . 3440
Figure 888. Data flow timing control (SSOE = 1, SSOM = 0, SSM = 0) . . . 3440
Figure 889. NSS interleaving pulses between data (SSOE = 1, SSOM = 1, SSM = 0) . . . 3441
Figure 890. Data clock timing diagram . . . 3443
Figure 891. Data alignment when data size is not equal to 8, 16 or 32 bits . . . 3444

Figure 892. TI mode transfer . . . 3453
Figure 893. Optional configurations of the slave behavior when an underrun condition is detected . 3455
Figure 894. Waveform examples . . . 3463
Figure 895. Master I2S Philips protocol waveforms (16/32-bit full accuracy) . . . 3464
Figure 896. I2S Philips standard waveforms . . . 3464
Figure 897. Master MSB-justified 16- or 32-bit full-accuracy length . . . 3465
Figure 898. Master MSB-justified 16- or 24-bit data length . . . 3465
Figure 899. Slave MSB-justified 16-, 24- or 32-bit data length . . . 3466
Figure 900. LSB-justified 16 or 24-bit data length . . . 3466
Figure 901. Master PCM when the frame length is equal the data length . . . 3467
Figure 902. Master PCM standard waveforms (16 or 24-bit data length) . . . 3467
Figure 903. Slave PCM waveforms . . . 3468
Figure 904. Startup sequence, I2S Philips standard, master . . . 3471
Figure 905. Startup sequence, I2S Philips standard, slave . . . 3472
Figure 906. Stop sequence, I2S Philips standard, master . . . 3472
Figure 907. I^2S clock generator architecture . . . 3473
Figure 908. Data Format . . . 3475
Figure 909. Handling of underrun situation . . . 3477
Figure 910. Handling of overrun situation . . . 3478
Figure 911. Frame error detection, with FIXCH = 0 . . . 3479
Figure 912. Frame error detection, with FIXCH = 1 . . . 3479
Figure 913. SAI functional block diagram . . . 3504
Figure 914. Audio frame . . . 3508
Figure 915. FS role is start of frame + channel side identification (FSDEF = TRIS = 1) . . . 3510
Figure 916. FS role is start of frame (FSDEF = 0) . . . 3511
Figure 917. Slot size configuration with FBOFF = 0 in SAI_xSLOTR . . . 3512
Figure 918. First bit offset . . . 3512
Figure 919. Audio block clock generator overview . . . 3514
Figure 920. PDM typical connection and timing . . . 3518
Figure 921. Detailed PDM interface block diagram . . . 3519
Figure 922. Start-up sequence . . . 3520
Figure 923. SAI_ADR format in TDM mode, 32-bit slot width . . . 3521
Figure 924. SAI_ADR format in TDM mode, 16-bit slot width . . . 3522
Figure 925. SAI_ADR format in TDM mode, 8-bit slot width . . . 3523
Figure 926. AC'97 audio frame . . . 3526
Figure 927. Example of typical AC'97 configuration on devices featuring at least two embedded SAIs (three external AC'97 decoders) . . . 3527
Figure 928. SPDIF format . . . 3528
Figure 929. SAI_xDR register ordering . . . 3529
Figure 930. Data companding hardware in an audio block in the SAI . . . 3532
Figure 931. Tristate strategy on SD output line on an inactive slot . . . 3534
Figure 932. Tristate on output data line in a protocol like I2S . . . 3535
Figure 933. Overrun detection error . . . 3536
Figure 934. FIFO underrun event . . . 3536
Figure 935. SPDIFRX block diagram . . . 3571
Figure 936. S/PDIF sub-frame format . . . 3572
Figure 937. S/PDIF block format . . . 3573
Figure 938. S/PDIF preambles . . . 3573
Figure 939. Channel coding example . . . 3574
Figure 940. SPDIFRX decoder . . . 3575
Figure 941. Noise filtering and edge detection . . . 3575
Figure 942. Thresholds . . . 3577

Figure 943. Synchronization flowchart . . . 3579
Figure 944. Synchronization process scheduling . . . 3580
Figure 945. SPDIFRX states . . . 3581
Figure 946. SPDIFRX_FMTx_DR register format . . . 3583
Figure 947. Channel/user data format . . . 3584
Figure 948. S/PDIF overrun error when RXSTEO = 0 . . . 3586
Figure 949. S/PDIF overrun error when RXSTEO = 1 . . . 3587
Figure 950. SPDIFRX interface interrupt mapping diagram . . . 3590
Figure 951. MDIOS block diagram . . . 3605
Figure 952. MDIO protocol write frame waveform . . . 3606
Figure 953. MDIO protocol read frame waveform . . . 3606
Figure 954. CAN subsystem . . . 3619
Figure 955. FDCAN block diagram . . . 3621
Figure 956. Transceiver delay measurement . . . 3626
Figure 957. Pin control in bus monitoring mode . . . 3628
Figure 958. Pin control in loop back mode . . . 3630
Figure 959. CAN error state diagram . . . 3631
Figure 960. Message RAM configuration . . . 3632
Figure 961. Standard message ID filter path . . . 3635
Figure 962. Extended message ID filter path . . . 3636
Figure 963. Example of mixed configuration dedicated Tx buffers / Tx FIFO . . . 3642
Figure 964. Example of mixed configuration dedicated Tx buffers / Tx queue . . . 3642
Figure 965. Bit timing . . . 3644
Figure 966. Bypass operation . . . 3646
Figure 967. FSM calibration . . . 3647
Figure 968. Cycle time and global time synchronization . . . 3662
Figure 969. TTCAN level 0 and level 2 drift compensation . . . 3663
Figure 970. Level 0 schedule synchronization state machine . . . 3670
Figure 971. Level 0 master to slave relation . . . 3671
Figure 972. USB subsystem overview . . . 3748
Figure 973. USB2 OTG high-speed Port 1 block diagram . . . 3749
Figure 974. USB2 OTG high-speed Port 1 resets . . . 3750
Figure 975. USB2 OTG high-speed Port 2 block diagram . . . 3753
Figure 976. USB2 OTG high-speed Port 2 resets . . . 3754
Figure 977. USB Type-C implementation: OTG HS Port 1 example . . . 3757
Figure 978. USB Type-C and power delivery block diagram . . . 3758
Figure 979. USB Type-C and power delivery reset . . . 3759
Figure 980. Multiplexing of debug signals in the USB subsystem . . . 3760
Figure 981. OTG1/OTG2 high-speed block diagram . . . 3764
Figure 982. OTG A-B device connection . . . 3766
Figure 983. OTG peripheral-only connection . . . 3767
Figure 984. OTG host-only connection . . . 3771
Figure 985. SOF connectivity (SOF trigger output to TIM and ITR1 connection) . . . 3774
Figure 986. Updating OTG_HFIR dynamically (RLDCTRL = 1) . . . 3776
Figure 987. Device-mode FIFO address mapping and AHB FIFO access mapping . . . 3777
Figure 988. Host-mode FIFO address mapping and AHB FIFO access mapping . . . 3778
Figure 989. Interrupt hierarchy . . . 3782
Figure 990. Transmit FIFO write task . . . 3875
Figure 991. Receive FIFO read task . . . 3876
Figure 992. Normal bulk/control OUT/SETUP . . . 3877
Figure 993. Bulk/control IN transactions . . . 3881
Figure 994. Normal interrupt OUT . . . 3884

Figure 995. Normal interrupt IN . . . 3889
Figure 996. Isochronous OUT transactions . . . 3891
Figure 997. Isochronous IN transactions . . . 3894
Figure 998. Normal bulk/control OUT/SETUP transactions - DMA . . . 3896
Figure 999. Normal bulk/control IN transaction - DMA. . . . 3898
Figure 1000. Normal interrupt OUT transactions - DMA mode . . . 3899
Figure 1001. Normal interrupt IN transactions - DMA mode . . . 3900
Figure 1002. Normal isochronous OUT transaction - DMA mode . . . 3901
Figure 1003. Normal isochronous IN transactions - DMA mode . . . 3902
Figure 1004. Receive FIFO packet read . . . 3908
Figure 1005. Processing a SETUP packet . . . 3910
Figure 1006. Bulk OUT transaction . . . 3917
Figure 1007. TRDT max timing case . . . 3926
Figure 1008. UCPD block diagram . . . 3939
Figure 1009. Clock division and timing elements. . . . 3940
Figure 1010. K-code transmission . . . 3942
Figure 1011. Transmit order for various sizes of data . . . 3944
Figure 1012. Packet format . . . 3944
Figure 1013. Line format of Hard Reset. . . . 3945
Figure 1014. Line format of Cable Reset . . . 3945
Figure 1015. BIST test data frame. . . . 3946
Figure 1016. BIST Carrier Mode 2 frame. . . . 3947
Figure 1017. UCPD BMC transmitter architecture. . . . 3947
Figure 1018. UCPD BMC receiver architecture . . . 3949
Figure 1019. Ethernet high-level block diagram . . . 3981
Figure 1020. DMA transmission flow (standard mode) . . . 3983
Figure 1021. DMA transmission flow (OSP mode) . . . 3985
Figure 1022. Receive DMA flow . . . 3987
Figure 1023. Overview of MAC transmission flow . . . 3990
Figure 1024. MAC reception flow . . . 3992
Figure 1025. Packet filtering sequence . . . 4009
Figure 1026. Networked time synchronization . . . 4023
Figure 1027. Propagation delay calculation in clocks supporting peer-to-peer path correction . . . 4024
Figure 1028. System time update using fine correction method . . . 4034
Figure 1029. Time aware shaper implementation for traffic scheduling. . . . 4046
Figure 1030. Implementing a guard band to avoid delays due to interfering frames . . . 4047
Figure 1031. GCL governing gate close and open events block diagram (from IEEE 802.1 Qbv specifications) . . . 4048
Figure 1032. GCL and associated registers - BaseTime and CycleTime . . . 4052
Figure 1033. GCL and associated registers - BaseTime and CycleTime list execution time > CycleTime . . . 4053
Figure 1034. Switching to a new Configuration that is Aligned with the Existing Configuration. . . . 4054
Figure 1035. Switching to the New List by Truncating the Current List . . . 4056
Figure 1036. Switching to New List By Extending the Current List . . . 4057
Figure 1037. mPacket formats. . . . 4061
Figure 1038. MAC data Alignment Feature . . . 4066
Figure 1039. Time-based scheduler implementation. . . . 4074
Figure 1040. Time-based scheduling flow . . . 4075
Figure 1041. TCP segmentation offload overview . . . 4081
Figure 1042. TCP segmentation offload flow. . . . 4082
Figure 1043. Header and payload fields of segmented packets . . . 4085

Figure 1044. Supported PHY interfaces 4095
Figure 1045. SMA Interface block 4096
Figure 1046. MDIO packet structure (Clause 45) 4097
Figure 1047. MDIO packet structure (Clause 22) 4098
Figure 1048. SMA write operation flow 4099
Figure 1049. Write data packet 4100
Figure 1050. Read data packet 4100
Figure 1051. Media independent interface (MII) signals 4102
Figure 1052. RMII block diagram 4104
Figure 1053. Transmission bit order 4105
Figure 1054. Receive bit order 4106
Figure 1055. RGMII block diagram and interface signals 4107
Figure 1056. LPI transitions (Transmit, 100 Mbds) 4116
Figure 1057. LPI Tx clock gating (when LPITCSE = 1) 4117
Figure 1058. LPI transitions (receive, 100 Mbit/s) 4118
Figure 1059. Descriptor ring structure 4149
Figure 1060. DMA descriptor ring 4150
Figure 1061. Descriptor tail pointer example 1 4151
Figure 1062. Descriptor tail pointer example 2 4151
Figure 1063. Transmit descriptor (read format) 4152
Figure 1064. Transmit descriptor write-back format 4157
Figure 1065. Transmit context descriptor format 4162
Figure 1066. Receive normal descriptor (read format) 4165
Figure 1067. Receive normal descriptor (write-back format) 4168
Figure 1068. Receive context descriptor 4176
Figure 1069. Enhanced normal descriptor (read format) 4178
Figure 1070. Enhanced normal descriptor (write format) 4179
Figure 1071. Enhanced context descriptor (read format) 4180
Figure 1072. Enhanced context descriptor (write format) 4181
Figure 1073. Generation of ETH_DMAISR flags 4200
Figure 1074. HDP block diagram 4370
Figure 1075. Block diagram of debug infrastructure 4377
Figure 1076. Debug reset control logic 4379
Figure 1077. JTAG TAP state machine 4384
Figure 1078. SWD successful data transfer 4387
Figure 1079. Debug and access port connections 4396
Figure 1080. Cortex-M55 debug topology 4418
Figure 1081. Trace subsystem CoreSight topology 4497
Figure 1082. CoreSight timestamp distribution 4503
Figure 1083. Embedded cross trigger 4511
Figure 1084. Mapping trigger inputs to outputs 4514
Figure 1085. ETR state transition diagram 4560
Figure 1086. Scatter-gather operation 4561
Figure 1087. System trace macrocell 4620

1 Documentation conventions

1.1 General information

The STM32N6x5/x7xx devices have an Arm®(a) Cortex®-M55 core with Arm®(a) TrustZone®.

arm

1.2 List of abbreviations for registers

The following abbreviations(b) are used in register descriptions:

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

a. Arm and TrustZone are registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

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

1.3 Register reset value

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

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

1.4 Glossary

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

- Debug port definition/acronyms:
  - **JTAG**: Joint Test Action Group protocol, provides 4-pin standard + 1 optional
  - **SWD**: Serial Wire Debug protocol, provides 2-pin interface
- **Double word**: data of 64-bit length.
- **Word**: data of 32-bit length.
- **Half-word**: data of 16-bit length.
- **Byte**: data of 8-bit length.
- **Option bytes**: product configuration bits stored in internal fuses.
- **OTP**: one-time programming (fuses).
- **AHB**: advanced high-performance bus.
- **APB**: advanced peripheral bus.
- **AXI**: Advanced extensible interface protocol.
- **ECC**: error code correction.
- **DMA**: direct memory access.

1.5 Availability of peripherals

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

2 Memory and bus architecture

2.1 System architecture

2.1.1 Introduction

The STM32N6x5 and STM32N6x7 device architecture relies on an Arm Cortex-M55 core optimized for execution:

- Main masters:
  - Cortex-M55 with Arm TrustZone mainline with two master ports:
    - > M-AXI provides access to the memory and to peripherals
    - > P-AHB provides access to peripherals
  - NPU (neural processor unit), which includes two master AXI ports (STM32N6x7 devices only)
- Memories:
  - AHB and APB peripherals
  - 4.2 Mbytes of SRAM
  - 64 Kbytes I-TCM RAM with ECC for critical real-time routines, and 128 Kbytes of D-TCM RAM with ECC for critical real-time data
  - 8 Kbytes of backup SRAM (BKPSRAM) active in V_{BAT} mode
  - 2 x 16-Kbyte AHB RAMs
  - Flexible external memory controller (FLEXMEM) with cypher engine supporting up to 32-bit data bus: SRAM, PSRAM, SDRAM, LPSDR SDRAM, NOR/NAND flash memories
  - XSPIM port2 8-bit configuration with cypher engine
  - XSPIM port1 16-bit configuration with cypher engine

2.1.2 Bus architecture

The bus architecture is divided in two domains, high- and low-performance (see *Figure 1* and *Figure 2*).

High-speed, multi-frequency domain, AXI compliant interconnect

This high-performance interconnect is mainly used by the CPU, the NPU (STM32N6x7 devices only), and by the high-bandwidth masters (GPU, DMA2D, GFXMMU, VENC, DCMIPP, LTDC, ETH1, OTG1/2, SDMMC1/2, HPDMA1 AXI port).

This domain manages accesses to:

- external flash memories and SRAMs (through the FMC and the XSPI manager)
- main system internal memories AXISRAM1/2/3/4/5/6, boot ROM, VENCRAM (when the VENC is off), and CACHEAXI RAM (STM32N6x7 devices when NPU is off) or SRAM7 (STM32N6x5 devices)
- the CPU internal memory (TCM) when accessed by the HPDMA1 AXI and the NPU

  Note: The CPU has internally access to the TCM.
- the GFXMMU
- the STM (system trace macrocell)

The CPU uses its M-AXI master port to access the above-described targets.

The NPU uses a local interconnect (NPU_NIC) to have a direct access to AXISRAM3/4/5/6 and to the CPU TCM. This results in a high-performance and low-latency bus when accessing these targets.

In addition, data interleaving can be enabled on AXISRAM3/4/5/6 to balance the traffic and to improve access performance.

The CPU and high-bandwidth masters use another local interconnect called CPU_NOC to have high-performance access on AXISRAM1/2 and external flash memories.

Note: The boot ROM is accessible only by the CPU through an intermediate interconnect called CPU_NIC.

Multi-layer AHB interconnect

This interconnect is used to access AHBSRAM1/2, BKPSRAM, and peripherals.

The CPU can reach peripherals mapped in the peripheral region (address range from 0x4000 0000 to 0x5FFF FFFF), using the following masters ports:

- the CPU P-AHB master port for data accesses
- the CPU M-AXI master port for instruction accesses

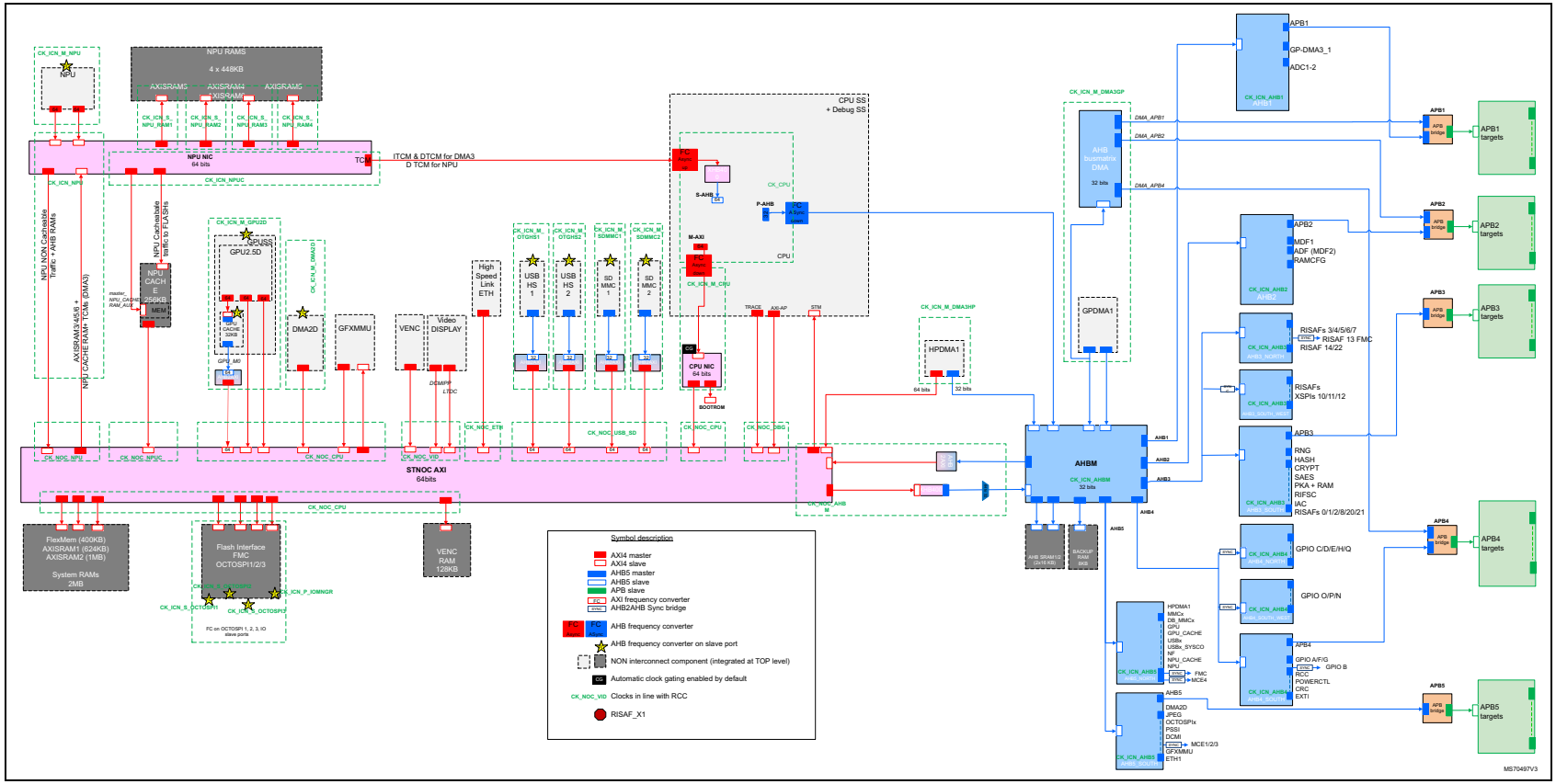
  When P-AHB is disabled, data accesses are performed on M-AXI.

The CPU accesses AHBSRAM1/2 and BKPSRAM through its M-AXI master port. The CPU memory accesses start on the high-performance interconnect before reaching the AHB interconnect.

GPDMA1 and HPDMA1 AHB ports are the other masters on this interconnect.

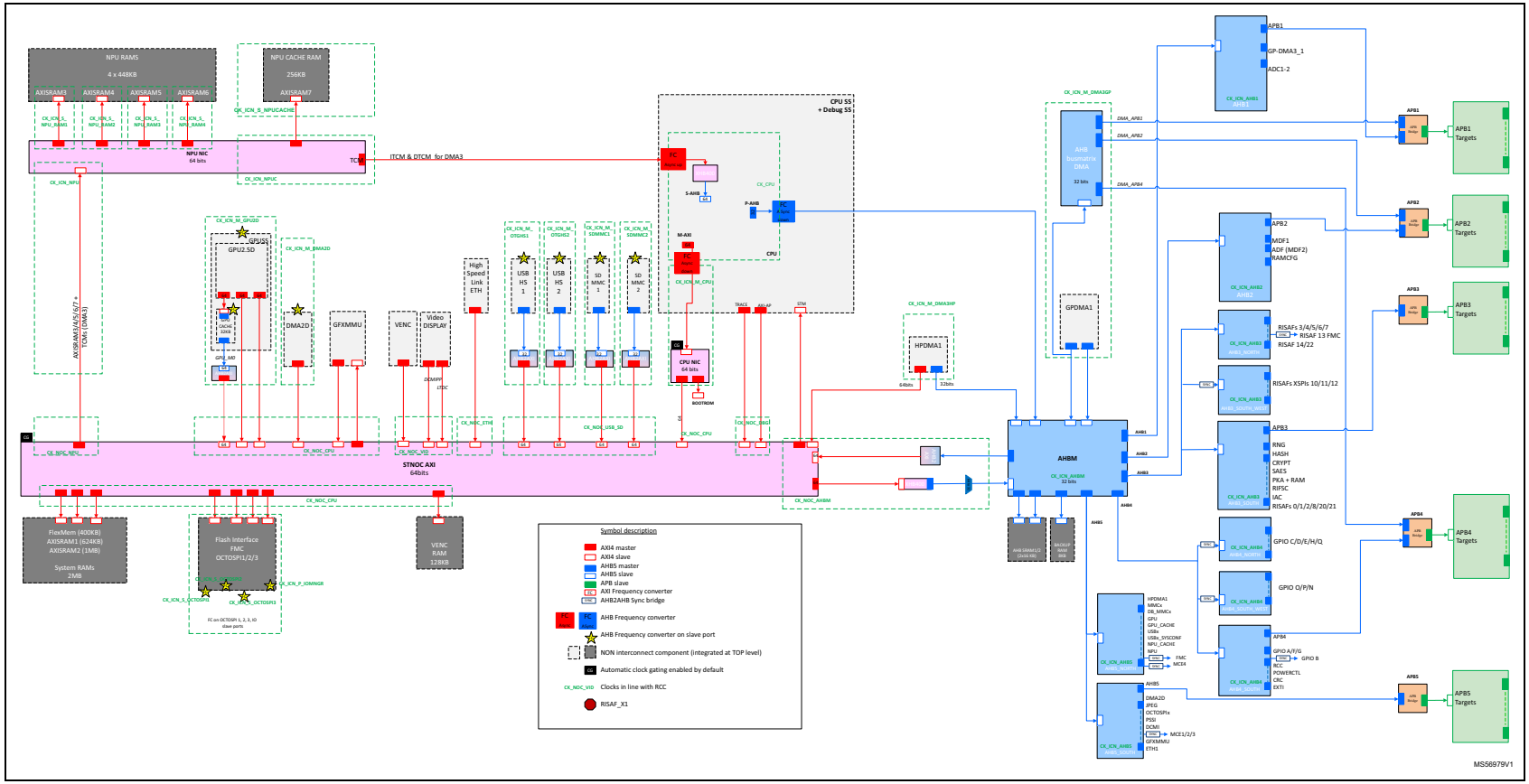
The GPDMA1_P has a dedicated access to APB1/2/4 peripherals.

Figure 1. Interconnect top view - STM32N6x7 devices



1. The high-performance domain is shown in pink. The low-performance domain is shown in blue.

Figure 2. Interconnect top view - STM32N6x5 devices



1. The high-performance domain is shown in pink. The low-performance domain is shown in blue.

2.2 Bus network-on-chip (NoC)

2.2.1 STNoC AXI

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

- Source routed, packet based, wormhole switching forwarding scheme for simultaneous and parallel/pipelined accesses from master to slave
- Independent read and write input/output points to cope with AXI independent channels
- Size and frequency conversion, asynchronism management
- Fine grain automatic clock gating shutting down the clocks to unused network parts at any point in time, with zero delay clock resume
- Internal packets buffering for improved traffic efficiency
- Independent request and response network
- Software accessible registers to observe and report traffic statistics, performance, blocking sources (if any) over the network.

2.2.2 NPU_NIC network interconnect

The NPU_NIC interconnect is based on Arm NIC-400 Network Interconnect. Its main features are:

- High-performance and low-latency interconnect
- Independent read/write request/response channel as defined in AXI protocol
- Asynchronism management
- Address interleaving to AXISRAM3/4/5/6/7 (can be disabled by software)
- CACHEAXI / SRAM7:
  - for STM32N6x7, when CACHEAXI is disabled, 256 KB SRAM7 can be accessible
  - for STM32N6x5, no CACHEAXI feature, only 256 KB accessible
- Static QoS assigned by software used at any arbitration node. The highest value has the highest priority. Arbitration with same QoS value uses an LRU (least recently used) algorithm.

2.2.3 Multi-layer AHB interconnect (AHBM)

The multi-layer AHB main features are:

- 32-bit bus matrix running at 200 MHz using a master clock from the RCC
- Simultaneous accesses from different masters to the slaves connected to separated layers
- Five master ports: CPU P-AHB, HPDMA1, GPDMA1_P /M, and CPU_NOC
- Nine slave ports: memories (AHB SRAM1/2 and BKPSRAM), peripherals for AHBx/APBx targets, and CPU_NOC target to reach AXI.

Note: The GPDMA1_P has a direct path to APB1, APB2, and APB4 targets. It bypasses this multi-layer AHB.

2.3 Memory organization

2.3.1 Introduction

Program memory, data memory, registers and I/O ports are organized within the same linear 4-Gbyte address space.

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

2.3.2 Memory map and register boundary addresses

Program memory, data memory, registers, and I/O ports are organized within the same linear 4-Gbyte address space.

The bytes are coded in memory in little-endian format. The lowest numbered byte in a word is considered the word least significant byte, and the highest numbered byte the most significant.

All memory map areas that are not allocated to on-chip memories and peripherals are considered reserved.

The IDAU mapping indicates if a particular memory address is secure or nonsecure.

Table 1. Memory map based on IDAU mapping

IDAU security type	Mapping	Description
-	0xFFFFFFFF	Reserved
-	0xE0100000	Reserved
Cortex-M55 nonsecure	0xE0000000	Cortex-M55 internal peripherals, debug and trace
SDRAM nonsecure	0xD0000000	SDRAM 2
	0xC0000000	SDRAM 1
-	0xA0000000	Reserved
XSPI bank nonsecure	0x90000000	XSPI1
	0x80000000	XSPI3
	0x70000000	XSPI2
SDRAM nonsecure	0x60000000	FMC NOR/SRAM

Table 1. Memory map based on IDAU mapping (continued)

IDAU security type	Mapping	Description
Reserved peripheral secure	0x5E000000	Trace and debug
	0x58120000	Reserved
	0x58020000	AHB5 peripherals
	0x58010000	Reserved for APB
	0x58000000	APB5 peripherals
	0x56030000	Reserved
	0x56020000	AHB4 peripherals
	0x56010000	Reserved for APB
	0x56000000	APB4 peripherals
	0x54040000	Reserved
	0x54020000	AHB3 peripherals
	0x54010000	Reserved for APB
	0x54000000	APB3 peripherals
	0x52040000	Reserved
	0x52020000	AHB2 peripherals
	0x52010000	Reserved for APB
	0x52000000	APB2 peripherals
	0x50188000	Reserved
	0x50020000	AHB1 peripherals
	0x50010000	Reserved
	0x50000000	APB1 peripherals

Table 1. Memory map based on IDAU mapping (continued)

IDAU security type	Mapping	Description
Peripheral nonsecure	0x4E000000	Trace and debug
	0x48120000	Reserved
	0x48020000	AHB5 peripherals
	0x48010000	Reserved for APB
	0x48000000	APB5 peripherals
	0x46030000	Reserved
	0x46020000	AHB4 peripherals
	0x46010000	Reserved for APB
	0x46000000	APB4 peripherals
	0x44040000	Reserved
	0x44020000	AHB3 peripherals
	0x44010000	Reserved for APB
	0x44000000	APB3 peripherals
	0x42040000	Reserved
	0x42020000	AHB2 peripherals
	0x42010000	Reserved for APB
	0x42000000	APB2 peripherals
	0x40188000	Reserved
	0x40020000	AHB1 peripherals
	0x40010000	Reserved
	0x40000000	APB1 peripherals
Reserved SRAM/AXI bank secure	0x3C002000	Reserved
	0x3C000000	Backup SRAM (8 Kbytes)
	0x38008000	Reserved
	0x38004000	AHBSRAM2
	0x38000000	AHBSRAM1
	0x37F00000	STM500 channels (system trace)
	0x37EFF000	FMC-NAND
	0x36000000	Reserved
	0x35000000	GFXMMU SLV
	0x34420000	Reserved
	0x34400000	VENCRAM/AXISRAM8
	0x343C0000	CACHEAXI/AXISRAM7
	0x34350000	AXISRAM6
	0x342E0000	AXISRAM5

Table 1. Memory map based on IDAU mapping (continued)

IDAU security type	Mapping	Description
Reserved SRAM/AXI bank secure	0x34270000	AXISRAM4
	0x34200000	AXISRAM3
	0x34100000	AXISRAM2
	0x34064000	AXISRAM1
Data secure	0x34000000	FLEXRAM
	0x30040000	Reserved
	0x30000000	DTCM - Baseline
SRAM/AXI bank nonsecure	0x2C002000	Reserved
	0x2C000000	Backup SRAM (8 Kbytes)
	0x28008000	Reserved
	0x28004000	AHBSRAM2
	0x28000000	AHBSRAM1
	0x27F00000	STM500 channels (system trace)
	0x27EFF000	FMC-NAND
	0x26000000	Reserved
	0x25000000	GFXMMU SLV
	0x24420000	Reserved
	0x24400000	VENCRAM/AXISRAM8
	0x243C0000	CACHEAXI/AXISRAM7
	0x24350000	AXISRAM6
	0x242E0000	AXISRAM5
	0x24270000	AXISRAM4
	0x24200000	AXISRAM3
	0x24100000	AXISRAM2
	0x24064000	AXISRAM1
Data nonsecure	0x24000000	FLEXRAM
	0x20040000	Reserved
	0x20000000	DTCM - Baseline
Code secure	0x18020000	Reserved
	0x18000000	BootROM 128 Kbytes
	0x10040000	Reserved
	0x10000000	ITCM

Table 1. Memory map based on IDAU mapping (continued)

IDAU security type	Mapping	Description
Code nonsecure	0x08020000	Reserved
	0x08000000	BootROM 128 Kbytes
	0x00040000	Reserved
	0x00000000	ITCM

Table 2. Memory map and peripheral register boundary addresses

Zone	Name	IDAU security type	Mapping	Description	Size (Mbytes)
-	Reserved	-	0xE0100000 - 0xFFFFFFFF	Reserved	512
	-	-	0xE0000000 - 0xE00FFFFF	Cortex-M55 internal peripherals, trace and debug	
AXI(D1)	External device	Nonsecure	0xD0000000 - 0xDFFFFFFF	SDRAM 2 - Reserved (through FMC)	256
			0xC0000000 - 0xCFFFFFFF	SDRAM 1 - Remap NOR/SRAM bank (through FMC)	256
			0xA0000000 - 0xBFFFFFFF	Reserved	512
	External RAM		0x90000000 - 0x9FFFFFFF	XSPI1	256
			0x80000000 - 0x8FFFFFFF	XSPI3	256
			0x70000000 - 0x7FFFFFFF	XSPI2	256
			0x60000000 - 0x6FFFFFFF	NOR/SRAM - Remap SDRAM 1 (through FMC)	256
Peripherals	Trace and debug	Secure	0x5E000000 - 0x5FFFFFFF	Trace and debug	32
	Reserved		0x5C000000 - 0x5DFFFFFF	Reserved	32
	Reserved		0x5A000000 - 0x5BFFFFFF	Reserved	32
	Reserved		0x58120000 - 0x59FFFFFF	Reserved	32
	AHB5		0x58020000 - 0x5811FFFF	AHB5 peripherals	
	Reserved		0x58010000 - 0x5801FFFF	Reserved for APB	
	APB5		0x58000000 - 0x5800FFFF	APB5 peripherals	
	Reserved	Secure	0x56030000 - 0x57FFFFFF	Reserved	32
	AHB4		0x56020000 - 0x5602FFFF	AHB4 peripherals	
	Reserved		0x56010000 - 0x5601FFFF	Reserved for APB	
	APB4		0x56000000 - 0x5600FFFF	APB4 peripherals	

Table 2. Memory map and peripheral register boundary addresses (continued)

Zone	Name	IDAU security type	Mapping	Description	Size (Mbytes)
Peripherals	Reserved	Secure	0x54040000 - 0x55FFFFFF	Reserved	32
	AHB3		0x54020000 - 0x5403FFFF	AHB3 peripherals	
	Reserved		0x54010000 - 0x5401FFFF	Reserved for APB	
	Reserved		0x54000000 - 0x5400FFFF	APB3 peripherals	
	Reserved		0x52040000 - 0x53FFFFFF	Reserved	32
	AHB2		0x52020000 - 0x5203FFFF	AHB2 peripherals	
	Reserved		0x52010000 - 0x5201FFFF	Reserved for APB	
	Reserved		0x52000000 - 0x5200FFFF	APB2 peripherals	
	Reserved		0x50188000 - 0x51FFFFFF	Reserved	32
	AHB1		0x50020000 - 0x50187FFF	AHB1 peripherals	
	APB2		0x50010000 - 0x5001FFFF	Reserved	
	APB1		0x50000000 - 0x5000FFFF	APB1 peripherals	
	Trace and debug	Nonsecure	0x4E000000 - 0x4FFFFFFF	Trace and debug	32
	Reserved		0x4C000000 - 0x4DFFFFFF	Reserved	32
	Reserved		0x4A000000 - 0x4BFFFFFF	Reserved	32
	Reserved		0x48120000 - 0x49FFFFFF	Reserved	32
	AHB5		0x48020000 - 0x4811FFFF	AHB5 peripherals	
	Reserved		0x48010000 - 0x4801FFFF	Reserved for APB	
	APB5		0x48000000 - 0x4800FFFF	APB5 peripherals	
	Reserved		0x46030000 - 0x47FFFFFF	Reserved	32
	AHB4		0x46020000 - 0x4602FFFF	AHB4 peripherals	
	Reserved		0x46010000 - 0x4601FFFF	Reserved for APB	
	APB4		0x46000000 - 0x4600FFFF	APB4 peripherals	

Table 2. Memory map and peripheral register boundary addresses (continued)

Zone	Name	IDAU security type	Mapping	Description	Size (Mbytes)
Peripherals	Reserved	Nonsecure	0x44040000 - 0x45FFFFFF	Reserved	32
	AHB3		0x44020000 - 0x4403FFFF	AHB3 peripherals	
	Reserved		0x44010000 - 0x4401FFFF	Reserved for APB	
	Reserved		0x44000000 - 0x4400FFFF	APB3 peripherals	
	Reserved		0x42040000 - 0x43FFFFFF	Reserved	32
	AHB2		0x42020000 - 0x4203FFFF	AHB2 peripherals	
	Reserved		0x42010000 - 0x4201FFFF	Reserved for APB	
	Reserved		0x42000000 - 0x4200FFFF	APB2 peripherals	
	Reserved		0x40188000 - 0x41FFFFFF	Reserved	32
	AHB1		0x40020000 - 0x40187FFF	AHB1 peripherals	
	APB2		0x40010000 - 0x4001FFFF	Reserved	
	APB1		0x40000000 - 0x4000FFFF	APB1 peripherals	
SRAM	-	Secure	0x3C002000 - 0x3FFFFFFF	Reserved	64
	AHB_IC2_L5		0x3C000000 - 0x3C001FFF	Backup SRAM	
	-		0x38008000 - 0x3BFFFFFF	Reserved	64
	AHB_IC2_L1		0x38004000 - 0x38007FFF	AHBSRAM2	
	AHB_IC2_L0		0x38000000 - 0x38003FFF	AHBSRAM1	

Table 2. Memory map and peripheral register boundary addresses (continued)

Zone	Name	IDAU security type	Mapping	Description	Size (Mbytes)
SRAM	-	Secure	0x37F00000 - 0x37FFFFFF	STM channels (system trace)	64
	-		0x37EFF000 - 0x37EFFFFF	FMC-NAND	
	AXI_ICI_GFX		0x36000000 - 0x37EFEFFF	Reserved	
	-		0x35000000 - 0x35FFFFFF	GFXMMU SLV	
	-		0x34420000 - 0x34FFFFFF	Reserved	
	-		0x34400000 - 0x3441FFFF	VENCRAM/AXISRAM8	
	AXI_IC1_L4-C		0x343C0000 - 0x343FFFFF	CACHEAXI/AXISRAM7	
	-		0x34350000 - 0x343BFFFF	AXISRAM6	
	AXI_IC1_L4		0x342E0000 - 0x3434FFFF	AXISRAM5	
	-		0x34270000 - 0x342DFFFF	AXISRAM4	
	AXI_IC1_L3		0x34200000 - 0x3426FFFF	AXISRAM3	
	AXI_IC1_L2		0x34100000 - 0x341FFFFF	AXISRAM2	
	AXI_IC1_L1		0x34000000 - 0x340FFFFF	AXISRAM1 (FLEXMEM extension bites on the lower end)	
	DTCM		0x30040000 - 0x33FFFFFF	Reserved	64
			0x30020000 - 0x3003FFFF	DTCM - FLEXMEM extension	
			0x30000000 - 0x3001FFFF	DTCM - Base line	
	-	Nonsecure	0x2C002000 - 0x2FFFFFFF	Reserved	64
	AHB_IC2_L5		0x2C000000 - 0x2C001FFF	BKPSRAM	
	-		0x28008000 - 0x2BFFFFFF	Reserved	64
	AHB_IC2_L1		0x28004000 - 0x28007FFF	AHBSRAM2	
	AHB_IC2_L0		0x28000000 - 0x28003FFF	AHBSRAM1	

Table 2. Memory map and peripheral register boundary addresses (continued)

Zone	Name	IDAU security type	Mapping	Description	Size (Mbytes)
SRAM	-	Nonsecure	0x27F00000 - 0x27FFFFFF	STM channels (system trace)	64
	-		0x27EFF000 - 0x27EFFFFF	FMC-NAND	
	AXI_ICI_GFX		0x26000000 - 0x27EFEFFF	Reserved	
	-		0x25000000 - 0x25FFFFFF	GFXMMU SLV	
	-		0x24420000 - 0x24FFFFFF	Reserved	
	-		0x24400000 - 0x2441FFFF	VENCRAM/AXISRAM8	
	AXI_IC1_L4-C		0x243C0000 - 0x243FFFFF	CACHEAXI/AXISRAM7	
	-		0x24350000 - 0x243BFFFF	AXISRAM6	
	AXI_IC1_L4		0x242E0000 - 0x2434FFFF	AXISRAM5	
	-		0x24270000 - 0x242DFFFF	AXISRAM4	
	AXI_IC1_L3		0x24200000 - 0x2426FFFF	AXISRAM3	
	AXI_IC1_L2		0x24100000 - 0x241FFFFF	AXISRAM2	
	AXI_IC1_L1		0x24000000 - 0x240FFFFF	AXISRAM1 (FLEXMEM extension bites on the lower end)	
	DTCM		0x20040000 - 0x23FFFFFF	Reserved	64
			0x20020000 - 0x2003FFFF	DTCM - FLEXMEM extension	
			0x20000000 - 0x2001FFFF	DTCM - Base line	
CODE	AXI_IC1_L0	Secure	0x1C080000 - 0x1FFFFFFF	Reserved	128
			0x1C000000 - 0x1C07FFFF	Reserved	
			0x18020000 - 0x1BFFFFFF	Reserved	
			0x18000000 - 0x1801FFFF	BootROM	
	ITCM		0x10040000 - 0x17FFFFFF	Reserved	128
			0x10010000 - 0x1003FFFF	ITCM - FLEXMEM extension	
			0x10000000 - 0x1000FFFF	ITCM - Base line	

Table 2. Memory map and peripheral register boundary addresses (continued)

Zone	Name	IDAU security type	Mapping	Description	Size (Mbytes)
CODE	AXI_IC1_L0	Nonsecure	0x0C080000 - 0x0FFFFFFF	Reserved	128
			0x0C000000 - 0x0C07FFFF	Reserved	
			0x08020000 - 0x0BFFFFFF	Reserved	
			0x08000000 - 0x0801FFFF	BootROM	
	ITCM		0x00040000 - 0x07FFFFFF	Reserved	128
			0x00010000 - 0x0003FFFF	ITCM - FLEXMEM extension	
			0x00000000 - 0x0000FFFF	ITCM - Baseline	

Table 3. Peripheral register boundary addresses

Bus	Secure boundary address	Nonsecure boundary address	Description	Size (Kbytes)
AHB5	0x58100000 - 0x5811FFFF	0x48100000 - 0x4811FFFF	Reserved (for AHB peripherals)	128
	0x580E0000 - 0x580FFFFF	0x480E0000 - 0x480FFFFF	NPU	128
	0x580DFC00 - 0x580DFFFF	0x480DFC00 - 0x480DFFFF	CACHEAXI (STM32N6x7xx only)	1
	0x580C0400 - 0x580DFBFF	0x480C0400 - 0x480DFBFF	Reserved (NPU)	126
	0x580C0000 - 0x580C03FF	0x480C0000 - 0x480C03FF	OTG2PHYCTL	1
	0x58080000 - 0x580BFFFF	0x48080000 - 0x480BFFFF	OTG2_HS	256
	0x58040000 - 0x5807FFFF	0x48040000 - 0x4807FFFF	OTG1_HS	256
	0x5803FC00 - 0x5803FFFF	0x4803FC00 - 0x4803FFFF	OTG1PHYCTL	1
	0x58037400 - 0x580377FF	0x48037400 - 0x480377FF	Reserved	34
	0x58036000 - 0x580373FF	0x48036000 - 0x480373FF	ETH1	5
	0x58035400 - 0x58035FFF	0x48035400 - 0x48035FFF	Reserved (ETH1)	3
	0x58035000 - 0x580353FF	0x48035000 - 0x480353FF	GPU cache (ICACHE)	1
	0x58034000 - 0x58034FFF	0x48034000 - 0x48034FFF	GPU2D	4
	0x58030000 - 0x58033FFF	0x48030000 - 0x48033FFF	GFXMMU	16
	0x5802E400 - 0x5802FFFF	0x4802E400 - 0x4802FFFF	Reserved	7
	0x5802E000 - 0x5802E3FF	0x4802E000 - 0x4802E3FF	MCE4	1
	0x5802D000 - 0x5802DFFF	0x4802D000 - 0x4802DFFF	XSPI3	4
	0x5802C400 - 0x5802CFFF	0x4802C400 - 0x4802CFFF	Reserved (XSPI3)	3
	0x5802C000 - 0x5802C3FF	0x4802C000 - 0x4802C3FF	MCE3	1
	0x5802BC00 - 0x5802BFFF	0x4802BC00 - 0x4802BFFF	MCE2 (former OTFDEC2)	1
	0x5802B800 - 0x5802BBFF	0x4802B800 - 0x4802BBFF	MCE1 (former OTFDEC1)	1
	0x5802B400 - 0x5802B7FF	0x4802B400 - 0x4802B7FF	XSPIM (XSPI I/O manager)	1
	0x5802B000 - 0x5802B3FF	0x4802B000 - 0x4802B3FF	Reserved (delay block XSPI2)	1
	0x5802A000 - 0x5802AFFF	0x4802A000 - 0x4802AFFF	XSPI2	4
	0x58028800 - 0x58029FFF	0x48028800 - 0x48029FFF	Reserved	6
	0x58028400 - 0x580287FF	0x48028400 - 0x480287FF	DCMI	1
	0x58028000 - 0x580283FF	0x48028000 - 0x480283FF	DLYB1	1
	0x58027000 - 0x58027FFF	0x48027000 - 0x48027FFF	SDMMC1	4
	0x58026C00 - 0x58026FFF	0x48026C00 - 0x48026FFF	DLYB2	1
	0x58026800 - 0x58026BFF	0x48026800 - 0x48026BFF	SDMMC2	1
	0x58026400 - 0x580267FF	0x48026400 - 0x480267FF	PSSI	1
	0x58026000 - 0x580263FF	0x48026000 - 0x480263FF	Reserved (delay block XSPI1)	1
	0x58025000 - 0x58025FFF	0x48025000 - 0x48025FFF	XSPI1	4

Table 3. Peripheral register boundary addresses (continued)

Bus	Secure boundary address	Nonsecure boundary address	Description	Size (Kbytes)
	0x58024000 - 0x58024FFF	0x48024000 - 0x48024FFF	FMC	4
AHB5	0x58023000 - 0x58023FFF	0x48023000 - 0x48023FFF	JPEG	4
	0x58022000 - 0x58022FFF	0x48022000 - 0x48022FFF	Reserved	4
	0x58021000 - 0x58021FFF	0x48021000 - 0x48021FFF	DMA2D	4
	0x58020000 - 0x58020FFF	0x48020000 - 0x48020FFF	HPDMA1	4
AHB4	0x5602A000 - 0x5602FFFF	0x4602A000 - 0x4602FFFF	Reserved (for AHB peripherals)	24
	0x56028000 - 0x56029FFF	0x46028000 - 0x46029FFF	RCC	8
	0x56025400 - 0x56027FFF	0x46025400 - 0x46027FFF	Reserved	11
	0x56025000 - 0x560253FF	0x46025000 - 0x460253FF	EXTI	1
	0x56024C00 - 0x56024FFF	0x46024C00 - 0x46024FFF	CRC	1
	0x56024800 - 0x56024BFF	0x46024800 - 0x46024BFF	PWR	1
	0x56024400 - 0x560247FF	0x46024400 - 0x460247FF	Reserved	1
	0x56024000 - 0x560243FF	0x46024000 - 0x460243FF	GPIO Q	1
	0x56023C00 - 0x56023FFF	0x46023C00 - 0x46023FFF	GPIO P	1
	0x56023800 - 0x56023BFF	0x46023800 - 0x46023BFF	GPIO O	1
	0x56023400 - 0x560237FF	0x46023400 - 0x460237FF	GPIO N	1
	0x56022000 - 0x560233FF	0x46022000 - 0x460233FF	Reserved	5
	0x56021C00 - 0x56021FFF	0x46021C00 - 0x46021FFF	GPIO H	1
	0x56021800 - 0x56021BFF	0x46021800 - 0x46021BFF	GPIO G	1
	0x56021400 - 0x560217FF	0x46021400 - 0x460217FF	GPIO F	1
	0x56021000 - 0x560213FF	0x46021000 - 0x460213FF	GPIO E	1
	0x56020C00 - 0x56020FFF	0x46020C00 - 0x46020FFF	GPIO D	1
	0x56020800 - 0x56020BFF	0x46020800 - 0x46020BFF	GPIO C	1
	0x56020400 - 0x560207FF	0x46020400 - 0x460207FF	GPIO B	1
	0x56020000 - 0x560203FF	0x46020000 - 0x460203FF	GPIO A	1

Table 3. Peripheral register boundary addresses (continued)

Bus	Secure boundary address	Nonsecure boundary address	Description	Size (Kbytes)
AHB3	0x54038000 - 0x5403FFFF	0x44038000 - 0x4403FFFF	Reserved (for AHB peripherals)	32
	0x54037000 - 0x54037FFF	0x44037000 - 0x44037FFF	RISAF23: backup RAM	4
	0x54036000 - 0x54036FFF	0x44036000 - 0x44036FFF	RISAF22: AHB RAM 1	4
	0x54035000 - 0x54035FFF	0x44035000 - 0x44035FFF	RISAF21: AHB RAM 0	4
	0x54034000 - 0x54034FFF	0x44034000 - 0x44034FFF	RISAF15: cache configuration port	4
	0x54033000 - 0x54033FFF	0x44033000 - 0x44033FFF	RISAF14: FMC2	4
	0x54032000 - 0x54032FFF	0x44032000 - 0x44032FFF	RISAF13: XSPI3	4
	0x54031000 - 0x54031FFF	0x44031000 - 0x44031FFF	RISAF12: XSPI2	4
	0x54030000 - 0x54030FFF	0x44030000 - 0x44030FFF	RISAF11: XSPI1	4
	0x5402F000 - 0x5402FFFF	0x4402F000 - 0x4402FFFF	Reserved	4
AHB3	0x5402E000 - 0x5402EFFF	0x4402E000 - 0x4402EFFF	RISAF9: VENCRAM/AXISRAM8	4
	0x5402D000 - 0x5402DFFF	0x4402D000 - 0x4402DFFF	RISAF8: CACHEAXI/AXISRAM7	4
	0x5402C000 - 0x5402CFFF	0x4402C000 - 0x4402CFFF	RISAF7: FLEXMEM	4
	0x5402B000 - 0x5402BFFF	0x4402B000 - 0x4402BFFF	RISAF6: CPU_MST	4
	0x5402A000 - 0x5402AFFF	0x4402A000 - 0x4402AFFF	RISAF5: NPU_MST1	4
	0x54029000 - 0x54029FFF	0x44029000 - 0x44029FFF	RISAF4: NPU_MST0	4
	0x54028000 - 0x54028FFF	0x44028000 - 0x44028FFF	RISAF3: AXISRAM2	4
	0x54027000 - 0x54027FFF	0x44027000 - 0x44027FFF	RISAF2: AXISRAM1	4
	0x54026000 - 0x54026FFF	0x44026000 - 0x44026FFF	RISAF1: TCMs	4
	0x54025400 - 0x54025FFF	0x44025400 - 0x44025FFF	Reserved	3
	0x54025000 - 0x540253FF	0x44025000 - 0x440253FF	IAC	1
	0x54024000 - 0x54024FFF	0x44024000 - 0x44024FFF	RIFSC	4
	0x54022000 - 0x54023FFF	0x44022000 - 0x44023FFF	PKA + RAM	8
	0x54021400 - 0x54021FFF	0x44021400 - 0x44021FFF	Reserved	3
	0x54021000 - 0x540213FF	0x44021000 - 0x440213FF	SAES	1
	0x54020C00 - 0x54020FFF	0x44020C00 - 0x44020FFF	Reserved	1
	0x54020800 - 0x54020BFF	0x44020800 - 0x44020BFF	CRYP1	1
	0x54020400 - 0x540207FF	0x44020400 - 0x440207FF	HASH	1
	0x54020000 - 0x540203FF	0x44020000 - 0x440203FF	RNG	1

Table 3. Peripheral register boundary addresses (continued)

Bus	Secure boundary address	Nonsecure boundary address	Description	Size (Kbytes)
AHB2	0x52027000 - 0x5203FFFF	0x42027000 - 0x4203FFFF	Reserved (for AHB peripherals)	100
	0x52026000 - 0x52026FFF	0x42026000 - 0x42026FFF	ADF1 (MDF2)	4
	0x52025000 - 0x52025FFF	0x42025000 - 0x42025FFF	MDF1	4
	0x52024000 - 0x52024FFF	0x42024000 - 0x42024FFF	Reserved	4
	0x52023000 - 0x52023FFF	0x42023000 - 0x42023FFF	RAMCFG	4
	0x52020000 - 0x52022FFF	0x42020000 - 0x42022FFF	Reserved	12
AHB1	0x50100000 - 0x50187FFF	0x40100000 - 0x40187FFF	Reserved (for AHB peripherals)	544
	0x50022400 - 0x500FFFFF	0x40022400 - 0x400FFFFF	Reserved	887
	0x50022000 - 0x500223FF	0x40022000 - 0x400223FF	ADC1/2	1
	0x50021000 - 0x50021FFF	0x40021000 - 0x40021FFF	GPDMA1	4
	0x50020000 - 0x50020FFF	0x40020000 - 0x40020FFF	Reserved	4
APB5	0x58008000 - 0x5800FFFF	0x48008000 - 0x4800FFFF	Reserved (for APB peripherals)	32
	0x58006000 - 0x58007FFF	0x48006000 - 0x48007FFF	CSI2 HOST wrapper	8
	0x58005000 - 0x58005FFF	0x48005000 - 0x48005FFF	VENC H264/JPEG encoder	4
	0x58004000 - 0x58004FFF	0x48004000 - 0x48004FFF	GFXTIM	4
APB5	0x58003000 - 0x58003FFF	0x48003000 - 0x48003FFF	Reserved	4
	0x58002000 - 0x58002FFF	0x48002000 - 0x48002FFF	DCMIPP	4
	0x58001000 - 0x58001FFF	0x48001000 - 0x48001FFF	LTDC	4
	0x58000000 - 0x58000FFF	0x48000000 - 0x48000FFF	Reserved	4

Table 3. Peripheral register boundary addresses (continued)

Bus	Secure boundary address	Nonsecure boundary address	Description	Size (Kbytes)
APB4	0x5600C000 - 0x5600FFFF	0x4600C000 - 0x4600FFFF	Reserved	16
	0x5600B000 - 0x5600BFFF	0x4600B000 - 0x4600BFFF	Reserved	4
	0x5600A000 - 0x5600AFFF	0x4600A000 - 0x4600AFFF	DTS (16FF version)	4
	0x56009000 - 0x56009FFF	0x46009000 - 0x46009FFF	BSEC	4
	0x56008000 - 0x56008FFF	0x46008000 - 0x46008FFF	SYSCFG	4
	0x56004C00 - 0x56007FFF	0x46004C00 - 0x46007FFF	Reserved(WDG_LS_D2)	13
	0x56004800 - 0x56004BFF	0x46004800 - 0x46004BFF	IWDG (WDG_LS_D1)	1
	0x56004400 - 0x560047FF	0x46004400 - 0x460047FF	TAMP	1
	0x56004000 - 0x560043FF	0x46004000 - 0x460043FF	RTC and backup registers	1
	0x56003C00 - 0x56003FFF	0x46003C00 - 0x46003FFF	VREFBUF	1
	0x56003400 - 0x56003BFF	0x46003400 - 0x46003BFF	Reserved	2
	0x56003000 - 0x560033FF	0x46003000 - 0x460033FF	LPTIMER5	1
	0x56002C00 - 0x56002FFF	0x46002C00 - 0x46002FFF	LPTIMER4	1
	0x56002800 - 0x56002BFF	0x46002800 - 0x46002BFF	LPTIMER3	1
	0x56002400 - 0x560027FF	0x46002400 - 0x460027FF	LPTIMER2	1
	0x56002000 - 0x560023FF	0x46002000 - 0x460023FF	Reserved	1
	0x56001C00 - 0x56001FFF	0x46001C00 - 0x46001FFF	I2C4	1
	0x56001800 - 0x56001BFF	0x46001800 - 0x46001BFF	Reserved	1
	0x56001400 - 0x560017FF	0x46001400 - 0x460017FF	SPI6/I2S	1
	0x56001000 - 0x560013FF	0x46001000 - 0x460013FF	Reserved	1
	0x56000C00 - 0x56000FFF	0x46000C00 - 0x46000FFF	LPUART1	1
	0x56000800 - 0x56000BFF	0x46000800 - 0x46000BFF	HDP	1
	0x56000000 - 0x460007FF	0x46000000 - 0x460007FF	Reserved	2
APB3	0x54002400 - 0x5400FFFF	0x44002400 - 0x4400FFFF	Reserved (for APB peripherals)	55
	0x54002000 - 0x540023FF	0x44002000 - 0x440023FF	DFT APB registers	1
	0x54001000 - 0x54001FFF	0x44001000 - 0x44001FFF	DBG_MCU	4
	0x54000000 - 0x54000FFF	0x44000000 - 0x44000FFF	DAP ROM table	4
APB2	0x52008000 - 0x5200FFFF	0x42008000 - 0x4200FFFF	Reserved (for APB peripherals)	32
	0x52006000 - 0x52007FFF	0x42006000 - 0x42007FFF	Reserved	7
	0x52005C00 - 0x52005FFF	0x42005C00 - 0x42005FFF	SAI2	1

Table 3. Peripheral register boundary addresses (continued)

Bus	Secure boundary address	Nonsecure boundary address	Description	Size (Kbytes)
APB2	0x52005800 - 0x52005BFF	0x42005800 - 0x42005BFF	SAI1	1
	0x52005400 - 0x520057FF	0x42005400 - 0x420057FF	Reserved	1
	0x52005000 - 0x520053FF	0x42005000 - 0x420053FF	SPI5	1
	0x52004C00 - 0x52004FFF	0x42004C00 - 0x42004FFF	TIMER9	1
	0x52004800 - 0x52004BFF	0x42004800 - 0x42004BFF	TIMER17	1
	0x52004400 - 0x520047FF	0x42004400 - 0x420047FF	TIMER16 (16-bit)	1
	0x52004000 - 0x520043FF	0x42004000 - 0x420043FF	TIMER15 (16-bit)	1
	0x52003C00 - 0x52003FFF	0x42003C00 - 0x42003FFF	TIMER18 (16-bit)	1
	0x52003800 - 0x52003BFF	0x42003800 - 0x42003BFF	Reserved	1
	0x52003400 - 0x520037FF	0x42003400 - 0x420037FF	SPI4	1
	0x52003000 - 0x520033FF	0x42003000 - 0x420033FF	SPI1/I2S1	1
	0x52002000 - 0x52002FFF	0x42002000 - 0x42002FFF	Reserved	4
	0x52001C00 - 0x52001FFF	0x42001C00 - 0x42001FFF	USART10	1
	0x52001800 - 0x52001BFF	0x42001800 - 0x42001BFF	UART9	1
	0x52001400 - 0x520017FF	0x42001400 - 0x420017FF	USART6	1
	0x52001000 - 0x520013FF	0x42001000 - 0x420013FF	USART1	1
	0x52000800 - 0x52000FFF	0x42000800 - 0x42000FFF	Reserved	2
	0x52000400 - 0x520007FF	0x42000400 - 0x420007FF	TIMER8 (16-bit)/PWM2	1
	0x52000000 - 0x520003FF	0x42000000 - 0x420003FF	TIMER1 (16-bit)/PWM1	1
APB1	0x50010000 - 0x5000FFFF	0x40010000 - 0x4000FFFF	Reserved (for APB peripherals)	0
	0x5000FC00 - 0x5000FFFF	0x4000FC00 - 0x4000FFFF	UCPD	1
	0x5000EC00 - 0x5000FBFF	0x4000EC00 - 0x4000FBFF	Reserved	4
	0x5000E800 - 0x5000EBFF	0x4000E800 - 0x4000EBFF	FDCAN3	1
	0x5000C000 - 0x5000E7FF	0x4000C000 - 0x4000E7FF	CAN memory	10
	0x5000AC00 - 0x5000BFFF	0x4000AC00 - 0x4000BFFF	Reserved	5
	0x5000A800 - 0x5000ABFF	0x4000A800 - 0x4000ABFF	CAN calibration unit	1
	0x5000A400 - 0x5000A7FF	0x4000A400 - 0x4000A7FF	FDCAN2	1
	0x5000A000 - 0x5000A3FF	0x4000A000 - 0x4000A3FF	FDCAN1	1
	0x50009800 - 0x50009FFF	0x40009800 - 0x40009FFF	Reserved	2
	0x50009400 - 0x500097FF	0x40009400 - 0x400097FF	MDIOS	5
	0x50008000 - 0x500093FF	0x40008000 - 0x400093FF	Reserved	1
	0x50007C00 - 0x50007FFF	0x40007C00 - 0x40007FFF	UART8	1
	0x50007800 - 0x50007BFF	0x40007800 - 0x40007BFF	UART7	4
	0x50006800 - 0x500077FF	0x40006800 - 0x400077FF	Reserved	1

Table 3. Peripheral register boundary addresses (continued)

Bus	Secure boundary address	Nonsecure boundary address	Description	Size (Kbytes)
APB1	0x50006400 - 0x500067FF	0x40006400 - 0x400067FF	I3C2	1
	0x50006000 - 0x500063FF	0x40006000 - 0x400063FF	I3C1	1
	0x50005C00 - 0x50005FFF	0x40005C00 - 0x40005FFF	I2C3	1
	0x50005800 - 0x50005BFF	0x40005800 - 0x40005BFF	I2C2	1
	0x50005400 - 0x500057FF	0x40005400 - 0x400057FF	I2C1	1
	0x50005000 - 0x500053FF	0x40005000 - 0x400053FF	UART5	1
	0x50004C00 - 0x50004FFF	0x40004C00 - 0x40004FFF	UART4	1
	0x50004800 - 0x50004BFF	0x40004800 - 0x40004BFF	USART3	1
	0x50004400 - 0x500047FF	0x40004400 - 0x400047FF	USART2	1
	0x50004000 - 0x500043FF	0x40004000 - 0x400043FF	SPDIFRX	1
	0x50003C00 - 0x50003FFF	0x40003C00 - 0x40003FFF	SPI3/I2S3	1
	0x50003800 - 0x50003BFF	0x40003800 - 0x40003BFF	SPI2/I2S2	1
	0x50003400 - 0x500037FF	0x40003400 - 0x400037FF	TIMER11 (basic)	1
	0x50003000 - 0x500033FF	0x40003000 - 0x400033FF	TIMER10 (basic)	1
	0x50002C00 - 0x50002FFF	0x40002C00 - 0x40002FFF	WWDG	1
	0x50002800 - 0x50002BFF	0x40002800 - 0x40002BFF	Reserved	1
	0x50002400 - 0x500027FF	0x40002400 - 0x400027FF	LPTIMER1	1
	0x50002000 - 0x500023FF	0x40002000 - 0x400023FF	TIMER14 (light)	1
	0x50001C00 - 0x50001FFF	0x40001C00 - 0x40001FFF	TIMER13 (light)	1
	0x50001800 - 0x50001BFF	0x40001800 - 0x40001BFF	TIMER12 (light)	1
	0x50001400 - 0x500017FF	0x40001400 - 0x400017FF	TIMER7 (basic)	1
	0x50001000 - 0x500013FF	0x40001000 - 0x400013FF	TIMER6 (basic)	1
	0x50000C00 - 0x50000FFF	0x40000C00 - 0x40000FFF	TIMER5 (32-bit)	1
	0x50000800 - 0x50000BFF	0x40000800 - 0x40000BFF	TIMER4 (32-bit)	1
	0x50000400 - 0x500007FF	0x40000400 - 0x400007FF	TIMER3 (32-bit)	1
	0x50000000 - 0x500003FF	0x40000000 - 0x400003FF	TIMER2 (32-bit)	1

3 System security

3.1 Key security features

- Security isolation using the TrustZone technology of the Arm Cortex-M55
- OS and dynamic user isolation using the Cortex-M55 MPU technology with firewalling
- Multi-tenancy support: configurable firewalls to allow a trusted OS to ensure software apps installed by different AI providers cannot use the NPU engine or DMA engines to access each other's data
- Secure provisioning of customer keys and firmware
- Secure boot-from-ROM authenticating and decrypting images loaded from an external flash memory or downloaded through UART, USB, I^2C, SPI, FDCAN, or JTAG
- Nonvolatile antifuse storage of, for example:
  - hardware-unique key (HUK) directly wired to DPA-resistant AES engine
  - private key used for secure provisioning of customer keys
  - customer-provisioned firmware signing and encryption keys
  - customer-provisioned key for managing return material analysis (RMA)
  - irreversible counters for key/certificates revocation, updates, factory resets
- Battery-backed volatile storage of, for example:
  - customer-provisioned secret boot hardware key (BHK), directly wired to DPA-resistant AES engine, automatically erased in case of tamper
  - anti-rollback counters
- Provisioning of AES keys in non-volatile storage (for example external flash memory), encrypted with unviewable device-unique keys derived from HUK: a provisioned AES key is never again viewable on the device, and is only used directly in AES hardware accelerators to encrypt and decrypt data
- General-purpose cryptographic acceleration:
  - Two AES 256-bit engine, supporting ECB, CBC, CTR, GCM, and CCM chaining modes: one engine DPA-resistant with side-channel counter-measures, and one fast-implementation engine without DPA-resistance
  - HASH processor, supporting SHA-2 secure hash
  - PKA (public key accelerator) for RSA/DH (up to 4096 bits) and ECC (up to 640 bits): implements side-channel counter measures and mitigations when manipulating secrets
  - RNG (true random number generator), NIST SP800-90B pre-certified
- Memory cipher engines for on-the-fly decryption of data in external memories: implements a key derivation function to increase protection against side-channel attacks (SCA)
- Low-latency encryption engines on NPU bus interfaces: allows AI data to be stored in encrypted format in both external and internal memories
- Arm PSA temporal isolation enforced by hardware using HDPL (hide protection level)
- Debug authentication scheme based on certificates: confers rights on the debugger at an authorized security level

- Active tamper and protection against temperature, voltage, and frequency attacks:
  - Up to seven (inputs, outputs) tamper pins, available in all power modes
  - Immediate erase of battery-backed volatile secrets on confirmed tamper detection

3.2 Secure boot

On an application reset, the Cortex-M55 always boots from the start of an on-chip ROM. This ROM code is used to authenticate and decrypt images loaded into the internal memory from an external flash memory, or through UART, USB, I^2C, SPI, FDCAN, or JTAG.

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

Lifecycle states can be partitioned into two underlying categories: BSEC-open or BSEC-closed (refer to *Section 4.3.7: Lifecycle management* for more details). This is determined by hardware on every system reset by examining the state of the lifecycle fuses.

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

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

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

On a BSEC-closed device, if the secure_boot fuse bit (see *Section 5: OTP mapping (OTP)*) is still unblown, then the boot ROM behavior depends on the DEV_BOOT pin:

- If the DEV_BOOT pin is set, the boot ROM disables the access to secrets, and opens the debug to allow customer development.
- If the DEV_BOOT pin is not set, the boot ROM executes the boot image regardless of whether it is correctly signed.

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

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

3.3 Secure provisioning

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

device-unique key pair ensures that this provisioning process is secure and counted. This protects against cloning.

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

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

3.4 Temporal isolation

The Arm PSA specification states:

1. The code in each stage must execute its tasks, complete, then terminate, and handover execution to the next stage.
2. The code in each stage must leave boot state to carry forward results and data from one stage to the next.
3. Private data for one stage (especially secrets) must not be directly accessible from later stages.

The STM32N6x5/x7xx devices define four HDPL:

- 0: Boot ROM
- 1: Updatable root of trust (uRoT)
- 2: Level 2
- 3: Level 3

The HDPL is represented with a monotonically increasing 2-bit counter register BSEC_HDPLSR (on the system reset), which does not wrap back to zero. Only a secure privileged software increments this counter (see *Section 4: Boot and security control (BSEC)*).

The SAES (secure AES engine) uses an HDPL input when performing a derivation of the fused HUK to create the DHUK (derived-HUK). The HDPL value seen by the SAES can be increased to a value higher than the current value, to provision data ready for the next temporal stage. However, the generated DHUK with a particular HDPL value cannot be re-generated during later temporal stages (see *Section 48: Secure AES coprocessor (SAES)*).

Temporal isolation can also be implemented in the fuse wrapper (BSEC) via sticky read locks and sticky programming locks. The BSEC hardware also ensures that only the boot ROM accesses certain fuse secrets.

3.5 Resource isolation

In STM32N6x5/x7xx, the runtime software isolation is based on TrustZone.

Each memory address is defined to exist either in the secure or the non-secure world. Some CPU resources (such as stack-pointer registers, MPU registers, interrupt vector tables, or timers), are physically banked between worlds. When the CPU executes a secure code (that is, code fetched from secure-world addresses), it uses the secure-world versions of these resources. When the CPU executes nonsecure code (that is, code fetched from nonsecure world addresses), it uses the nonsecure world versions of these resources. However, for the most part, secure and nonsecure codes use the same CPU hardware.

A nonsecure code is blocked from accessing secure world addresses. It can call only secure code (that is, move into the secure world) via SG (secure gate) instructions, which reside in special secure-world NSC (nonsecure callable) regions.

Trustzone ensures that the operation of a secure code and its data cannot be corrupted by nonsecure code.

3.5.1 Trustzone filtering using IDAU/SAU

The 256-Mbyte address ranges starting at the even addresses 0x00000000, 0x20000000, and 0x40000000, are physically aliased at ranges starting at the odd addresses 0x10000000, 0x30000000, and 0x50000000, respectively.

The intention is that the secure world accesses resources via the odd aliases, and the non-secure world accesses resources via the even aliases.

The address security (S for secure, NSC for nonsecure callable, or NS for non-secure, in descending security order) is determined by the device IDAU (implementation defined attribution unit), and the SAU (security attribution unit). Each of these two units proposes a security level for each address, the resulting level is the higher of the two proposals. The accesses to addresses determined to be S or NSC travel out on the bus marked as secure, those determined to be NS travel out on the bus marked as nonsecure.

The secure OS can define up to eight SAU regions, each of which can be configured as NS or NSC. Any address falling outside a defined SAU region remains secure, hence, at reset, the SAU unilaterally determines that the entire memory is secure.

The SAU is typically configured to open up NS regions for nonsecure code, data, and peripherals in the even address aliases. Locations from 0x10000000 to 0x1FFFFFFF (that contain SG instructions) must be covered by explicit SAU regions configured as NSC.

The IDAU determination of the security of each address is hardwired fixed.

Table 4 shows aliased targets, and the IDAU security assignment.

Table 4. Aliasing

Address region	IDAU assignment	Aliased targets
0x5-------	NSC (secure)	Peripherals
0x4-------	NS (nonsecure)	
0x3-------	NSC (secure)	Internal RAMs, DTCM
0x2-------	NS (nonsecure)	
0x1-------	NSC (secure)	Boot ROM, ITCM
0x0-------	NS (nonsecure)	

If the SAU is configured sensibly, the IDAU is normally irrelevant. The only potential effect of the IDAU is to prevent any attempt via the SAU to demote any of the three “odd” NSC regions listed above to be nonsecure.

A secure software configures the firewalls in front of physical resources (aliased in ranges listed in *Table 4*) to determine in which world the access destination resides. This blocks accesses that arrive marked with a mismatching security attribute. A nonsecure code cannot access destinations that reside in the secure world (as designated by the firewall). Destinations are then dedicated for use by the secure world.

External memories are mapped at addresses from 0x60000000 upwards (not aliased). The SAU can be configured to allocate portions of these memories to the non-secure world. These memories have RISAF firewalls in front of them, that can be configured to mirror SAU settings, and to ensure that DMAs are subject to the same restrictions as CPU code.

3.5.2 Privileged OS-controlled isolation

The Cortex-M55 MPU (memory protection unit) facilitates the isolation between a privileged operating system, and the user processes it schedules.

The MPU is banked between the secure and nonsecure worlds as follows:

- Secure-world accesses to MPU registers update physical registers in the MPU_S, which filters accesses that emerge from the SAU marked secure.
- Nonsecure world accesses to MPU registers update physical registers in the MPU_NS, which filters accesses emerging from the SAU marked nonsecure.

For each world, attributes are defined for 16 non-overlapping regions, and for the remaining background. The MPU blocks user code from accessing regions marked privileged.

If the MPU is configured to designate a memory region as normal, then all transactions initiated by the Cortex-55 on the AXI bus emerge as privileged (even if originated from a user process). Only the MPU prevents CPU user processes from accessing privileged normal data in SRAM and external flash memories.

When switching from one user process to another, the OS can re-program the MPU and peripheral firewalls to ensure each process can only access its own data. A user application may need to use a peripheral with an AXI initiator that continues to operate in background after the user has been descheduled. To avoid the peripheral being blocked by a modified MPU regime, the user process must ask the OS to configure the peripheral to operate in privilege mode on the user's behalf. It is up to the OS to sanity-check the requested operation. The HPDMA on the AXI bus is a special case (see *Section 3.5.4*).

The Cortex-M55 PAHBCR register must enable the P-AHB, and configure it to allow accesses to the peripheral region on the P-AHB bus. The MPU must be configured so that the peripheral region is execute-never device memory. Accesses to peripherals on the P-AHB carry the correct privilege level: to save wasting MPU regions, the MPU can be configured to make the whole peripheral region user-accessible, and to rely on the RISUP downstream filtering to restrict certain peripherals to OS control.

The Cortex-M55 introduces a PXN (privilege-execute-never) region attribute to prevent the OS from being tricked into jumping to user code.

3.5.3 RIF infrastructure

Thanks to hardware mechanisms in the bus, the secure OS ensures that resources and data are available only to agents that operate in the appropriate world, and at the appropriate privilege level. This infrastructure is named RIF (resource isolation framework).

The RIF consists of five components:

- RIMU: placed on AXI bus initiators to determine which attributes are attached to accesses
- RISUP: firewalls placed in front of peripheral bus targets to determine which security and privilege level is allowed to access that peripheral
- RIFSC: centralized block that contains the configuration registers controlling all RIMUs and RISUPs, and determining other general RIF parameters

- RISAF: firewalls placed in front of each memory target, with self-contained (local) configuration registers
- IAC (interrupt access controller): recording all security violation events from all firewalls around the device, in order to present a single interrupt to the CPU

Some peripherals have built-in firewalls, which are configured locally inside the peripheral. These peripherals are called RIF-aware.

The IAC has room for 256 interrupt sources, each corresponding to an IAC index from 0 to 255. Interrupts from the RISUPs occupy most of the first 128 indexes. Interrupts from RIF-aware peripherals and RISAFs occupy indexes from 128 onwards. RISUP and IAC indexes are aligned. The RISUP associated with the IAC index 32 x + y is programmed to be secure by setting RISC_SECCFGx[y]. It is programmed to be privileged by setting RISC_PRIVCFGx[y].

For all RIF firewalling, the security setting of a resource can be configured only by a secure privileged access: the secure OS can assign a RISAF base region to the nonsecure OS. The nonsecure OS determines the position, privilege, and readability of subregions within the base region.

If the resource is set to secure, nonsecure accesses to this resource are blocked. If the resource is set to nonsecure, secure accesses to memories are blocked (no restrictions for peripherals).

For all RIF firewalling, the privilege setting of a resource is only configurable by a privileged access: if the resource has already been configured to be secure, the access must also be secure. If a resource is set to privileged, only privileged accesses are allowed (no restriction if set to unprivileged).

By default, the reset condition for peripheral firewalls is nonsecure unprivileged, for memories is secure privileged. In any case, the Cortex-55 boots in secure mode: the secure software can immediately establish a security regime.

On detecting a violation, a RISAF records locally the attributes of the violating access, and sets an output error signal: its rising edge is captured in the IAC, which can then interrupt the CPU. The handler must first clear the associated flag in the IAC, then clear the error condition recorded in the RISAF.

An illegal access generates a bus error only if it is an instruction fetch.

3.5.4 Multi-tenancy

Several AI network providers may need to have their private NPU data on the device at the same time. This is referred to as multi-tenancy, where providers are considered as tenants on the device.

The secure OS needs to ensure that one tenant cannot snoop or corrupt the private NPU data of another tenant.

Note: Tenants must all trust the secure OS to implement the isolation.

To facilitate the isolation between the NPU networks of distinct tenants, and other CPU software, the RIF and bus infrastructure support the concept of compartments. An AXI bus transaction carries a 3-bit CID value that identifies the transaction initiator compartment. The secure OS configure the RISAF firewalls in front of memories on the AXI buses to filter accesses according to the carried CID.

RISAF treats a CID = 7 as a wildcard: if it allows an access with CID = n, it also allows it with CID = 7. The CID used by AXI transactions from the DAP is defined by DAPCID in RIFSC_RIMC_CR. This bitfield is reset to 7, but can be changed by a secure privileged access.

The CPU always issues transactions on the AXI bus with CID = 1. Only the CPU initiates transactions on the AHB bus (directly, or indirectly via the DMA). The AHB does not carry a CID bus: implicitly all transactions to peripherals occurring on the AHB bus have CID = 1.

In a multi-tenancy scenario, a distinct CID (values ≠ 1 and ≠7) is assigned to each tenant. The secure OS configures the RISAFs to ensure tenant data are only accessible by accesses carrying the tenant CID. Before allocating temporary exclusive NPU use to a particular tenant, the secure OS configures the RIFSC: the NPU RIMUs ensure AXI reads and writes are made with the CID of this tenant. This ensures the NPU cannot be used to snoop the data of a different tenant.

A mechanism on STM32N6x5/x7xx devices allows a secure tenant to use the HPDMA to move its own data, but no-one else's. The secure OS can allocate an HPDMA channel n to a tenant c by setting SCID = c in HPDMA_CnCIDCFGR. This ensures that transfers on this channel can be set up only by configuration writes that carry CID = c. Tenants operate as user processes, so such an allocated channel n must be configured with bit n = 0 in HPDMA_PRIVCFGR (unprivileged). The mechanism is available only for tenants operating in the secure world, so the secure OS must set bit n = 1 in HPDMA_SECCFGR. AXI transactions made by the channel carry security, privilege, and cid attributes that match what these registers are set to.

Just before scheduling a tenant's user process, the secure OS must write the tenant CID in SYSCFG_SEC_AIDCR. This ensures that all AHB secure user accesses which arrive at the HPDMA, appear to carry that CID. The tenant’s secure user process is then only able to use secure HPDMA channels allocated to its own CID. When descheduling the tenant’s secure user process, the secure OS resets SYSCFG_SEC_AIDCR to 1. The tenant’s HPDMA job continues to operate in the background. Other user processes cannot interfere with HPDMA channels belonging to that tenant.

The SYSCFG_SECPRIV_AIDCR register plays the same role for AHB secure privileged accesses which arrive at the HPDMA. The interrupt handler for a DMA channel allocated to a tenant user must configure this register in order to interrogate and clear the interrupt registers in this DMA channel.

SYSCFG_SEC_AIDCR and SYSCFG_SECPRIV_AIDCR affect only the CID of secure accesses. Nonsecure accesses always carry a CID of 1 (for example the CPU).

All other AXI initiators have RIMUs: what was described above for the NPU can be applied to other AXI initiators (but using the RISUP and RIMUs).

3.5.5 NPU timeslicing

If two AI networks come from different tenants who expect their data to be isolated from each other, a privileged software (trusted by both networks and running at a higher privilege or security level) must decide which tenant has exclusive use of the NPU at any given time. The OS only opens the RISUP on the NPU configuration port to user accesses while this particular tenant user process is running.

Before allocating exclusive NPU use to a tenant network, the secure OS arranges for the following:

- Reprogram all keys in the NPU.

- Clean the internal NPU state (registers and internal RAM).
- Program the NPU RIMC with a CID used in the RISAFs to protect the tenant data.

The RIMUs on the NPU initiator ports ensure that NPU accesses made on behalf of the tenant are made with the appropriate CID.

Note: Any software that accesses the NPU configuration port thereby accesses any data that the NPU accessed to. Each tenant trusts the secure OS to ensure that, while the NPU operates with a tenant CID, only the tenant software driver process accesses the NPU configuration port via the RISUP. When the secure OS switches to another tenant driver process, it must close the NPU RISUP: other processes cannot take control of the NPU while it is still processing the current tenant data.

3.5.6 NPU cache (CACHEAXI) management

The CID and security bits of an access that arrives at the CACHEAXI are treated exactly as if they were four extra address tag bits. This implies that cached data cannot be shared between two NPU networks with distinct CIDs: this may result in the same physical memory data appearing in two distinct cache lines (incoherency). The software must ensure that such shared data is uncacheable.

The privileged bit of an access is also stored in the cache line, and used during eviction. It is not treated as an address tag bit (not taken into account in deciding whether there is a HIT or MISS). The code must not attempt to lower the privilege level in the NPU RIMC without first flushing the cache: this may result in the user-level network reading data that had been left in the cache when the network was privileged.

With these mechanisms, it is not necessary to flush the whole CACHEAXI when the secure OS hands the NPU use from one tenant to another, at a time period boundary.

The CACHEAXI has a configuration port that allows the software to perform maintenance operations. For example, the user software may need to ensure dirty cache lines are cleaned (written back to memory) in order to restore system coherency with the CPU. Some dirty cache lines may must be invalidated without cleaning them (for example, when there is a problem with the downstream memory). Such an invalidate-only operation could be used by an NPU user to subvert the operation of another NPU user. The RISAF filter in front of each cache configuration port can be configured to ensure that invalidate-only operations are performed only by commands (written on the cache configuration port) coming from the secure OS (highest authority).

3.5.7 Memory firewalls

Each RISAF protecting a memory supports up to seven base regions: two subregions are associated with each base region.

Only the secure OS configures the permissions of a base region. Locations outside any enabled region belong to the secure OS.

If the secure OS configures a base region to reside in the nonsecure world, it can delegate control of the corresponding two subregions to the nonsecure OS. The nonsecure OS can then position subregion windows for tenants.

Note: The security level of a subregion cannot be raised relative to its encompassing base region. Although a base region can be made accessible by several CIDs, a subregion is dedicated to a single CID.

If the nonsecure OS needs to control more subregions (for example four), the secure OS can set up two identical base regions, both delegated to the nonsecure OS.

3.5.8 RIF-aware peripherals

Direct memory access controllers (GP/HPDMA)

The security attributes of each DMA channel can be configured separately. When a DMA channel is configured as secure, the source and destination transfers can be independently set as secure or nonsecure by a secure application: this is done with SSEC and DSEC in GP/HPDMA_CxTR1. Linked lists must be in the secure-world memory.

When a DMA channel is configured to be nonsecure, linked lists must be in the nonsecure world memory. SSEC and DSEC must also both be zero, otherwise the transfer does not proceed.

When a DMA channel is configured to be unprivileged, the linked lists, source, and destination must all be in unprivileged locations.

As mentioned in *Section 3.5.4*, HPDMA channels can be allocated to a compartment. The OS can allow a scheduled user process to use this channel. Accesses made by this channel (even after a user process descheduling) carry the compartment CID: this facilitates isolation between user processes.

See *Section 18: High-performance direct memory access controller (HPDMA)*. The software should leave HPDMA_CxCIDCFGR[SEM_EN] = 0, since a semaphore mechanism is intended to allow multiple concurrent CPUs to share a resource, and there is only one CPU.

Power control (PWR)

PWR registers are secured through PWR_SECCFGR and PWR_PRIVCFGR, individually protecting the following features:

- system supply configuration
- voltage scaling
- low-power mode
- wake-up (WKUP) pins
- voltage detection and monitoring
- backup domain control.

When a GPIOx is configured as secure (respectively privileged), the corresponding bits in PWR_IORETENRx and PWR_IORETRx (control the pin standby configuration) become secure (respectively privileged) (see *Section 13: Power control (PWR)* for details).

Secure clock and reset (RCC)

RIFSC security settings are transmitted to the RCC. If a peripheral RISUP is programmed as secure (respectively privileged), the peripheral clock and reset bits become secure (respectively privileged).

The following configurations can be made secure-only and/or privileged-only, using RCC_SECCFGR and RCC_PRIVCFGR:

- external clock (such as HSE or LSE), internal oscillator (such as HSI, MSI, or LSI)
- main PLL and AHB prescaler
- system clock-source selection

- MCO clock output
- reset flag
- automatic internal oscillator wake-up configuration

See *Section 14: Reset and clock control (RCC)* for details.

Real time clock (RTC)

The following RTC features can be individually made secure-only and/or privileged-only: alarm A, alarm B, calibration, initialization, timestamping, and wake-up (see *Section 61: Real-time clock (RTC)* for details).

Note: The RTC security configuration is not affected by a system reset.

Tamper and backup registers (TAMP)

The control of the main tamper functionality can be made secure-only and/or privileged-only.

The TAMP also contains a monotonic counter register for anti-rollback functions. Its use can be made secure-only and/or privileged-only.

The backup registers can be partitioned by a secure privileged software into three zones:

- protection zone 1: fully accessible for both secure and nonsecure
- protection zone 2: secure read/write, but read-only for nonsecure
- protection zone 3: secure-privileged

The privilege level of zones 1 and 2 can also be individually configured (see *Section 62: Tamper and backup registers (TAMP)* for details).

Liquid-crystal thin-film transistor display controller (LTDC)

The LTDC address space is split into three sections (common, layer1, and layer2), which are protected by a RISUP. Security and privilege settings of each region are determined by RIFSC registers.

The hardware ensures that, if both layers are configured to occupy the same virtual level, layer2 is displayed on top of layer1. When both secure and nonsecure softwares use a layer, the secure software must configure:

- layer1 as nonsecure
- layer2 as secure
- layer2 to occupy the top virtual layer (malicious nonsecure software never obscures it)

The common section can be set as secure or nonsecure.

General-purpose I/Os (GPIO)

Each GPIO port contains two 16-bit registers: GPIOx_SECCFGR and GPIOx_PRIVCFGR. These registers are only writable by the secure OS. They determine the security and privilege levels to configure each of the 16 pins of a GPIO port. Outputs of these registers are also wired to the PWR, where they determine permission to configure the retention behavior of each pin.

At reset, unlike all other peripherals, each GPIO pin is configured as secure by default. This simply an extra-physical-robustness feature: the CPU boots in secure-privileged mode, and can initialize all GPIOs as secure anyway.

When a digital AF is used (input/output mode), if the peripheral is configured as secure, the I/O pin must also be secure, otherwise input and output are zeroed. This stops non-secure software snooping or interfering with a secure I/O communication.

The ADC12 can control an analog switch in the GPIO. This switch is forced open when the ADC12 is nonsecure and the I/O is secure: this prevents secure analog inputs to be directed to a nonsecure peripheral.

Table 5 lists input/output signals that are not blocked when the I/O is set as secure and the associated peripheral is nonsecure.

Example: when a secure application sets PA0 as secure to be used as UART4_TX, if TAMP is nonsecure, it can be programmed to capture the USART input traffic through the TAMP_IN signal.

For each case described in the table below, the secure application must decide whether there is a potential effect on data integrity or confidentiality, and whether it is critical or not.

Table 5. Nonsecure peripheral functions that can be connected to secure I/Os

Peripheral	Signal(1)	Input	Output	Bitfield to set to make the peripheral secure
TAMP	TAMP_INx (x = 1 to 7)	X	-	TAMPSEC in TAMP_SECCFGR
	TAMP_OUTx (x = 1 to 6, 8)	-	X	
RTC	RTC_OUTx (x = 1, 2)	-	X	SEC in RTC_SECCFGR
	RTC_TS	X	-	TSSEC in RTC_SECCFGR
PWR	WKUPx (x = 1 to 8)	X	-	WUPxSEC in PWR_SECCFGR
RCC	LSCO	-	X	LSSEC in RCC_SECCFGR
EXTI	EXTIx (x=0 to 22)	X	-	SECx in EXTI_SECCFGR

1. To find the I/O corresponding to the signal/function on the package, refer to the product datasheet.

Refer to *Section 15: General-purpose I/Os (GPIO)* for more details.

Extended interrupts and event controller (EXTI)

The EXTI can be configured to make individual events secure-only and/or privileged-only (see *Section 25: Extended interrupts and event controller (EXTI)* for more details).

System configuration controller (SYSCFG)

All SYSCFG registers are secure-writable only (see *Section 16: System configuration controller (SYSCFG)* for more details).

Microcontroller debug unit (DBGMCU)

The MCU debug component (DBGMCU) contains registers to stop (freeze) a peripheral.

If a secure debug is not authorized, the DAP is only able to make nonsecure accesses on the bus. It is not possible to freeze secure peripherals. The stop bits are writable in accordance with the security level of the associated peripheral.

Refer to *Section 78: Debug support (DBG)* for more details

3.6 Secure storage

3.6.1 Battery backup registers

Secrets can be stored in backup registers. The secure-privileged software ensures that some or all registers are not accessible to lower security or privileged levels.
The 256-bit BHK is wired directly to a key derivation block in the SAES: it is used to derive further secret keys. The BHK can be made unreadable after being written.

Use of the BHK and access to backup registers and backup RAM, are blocked while a potential tamper is flagged.

The secrets are erased by a reset of the V_{BAT} battery-backup domain. Such a reset may be as a result of a legitimate “factory-reset” procedure, or because of a confirmed tamper. The secrets can subsequently be updated, if the device needs to be re-deployed.

3.6.2 Secure RAM and registers

The 16-Kbyte AHB SRAM2 aliased at 0x28004000 and 0x38004000 is called secure RAM: it is erased on reset and on a confirmed tamper, and blocked during a potential tamper (reads are zeroed, writes are ineffective). See *Section 62: Tamper and backup registers (TAMP)* for details of the tamper mechanism.

MCE key registers are also considered as security assets: they are immediately erased on a confirmed tamper.

Some SRAMs are erased by hardware on reset (see *Section 10.3: RAMCFG functional description*).

3.6.3 Fuse secrets

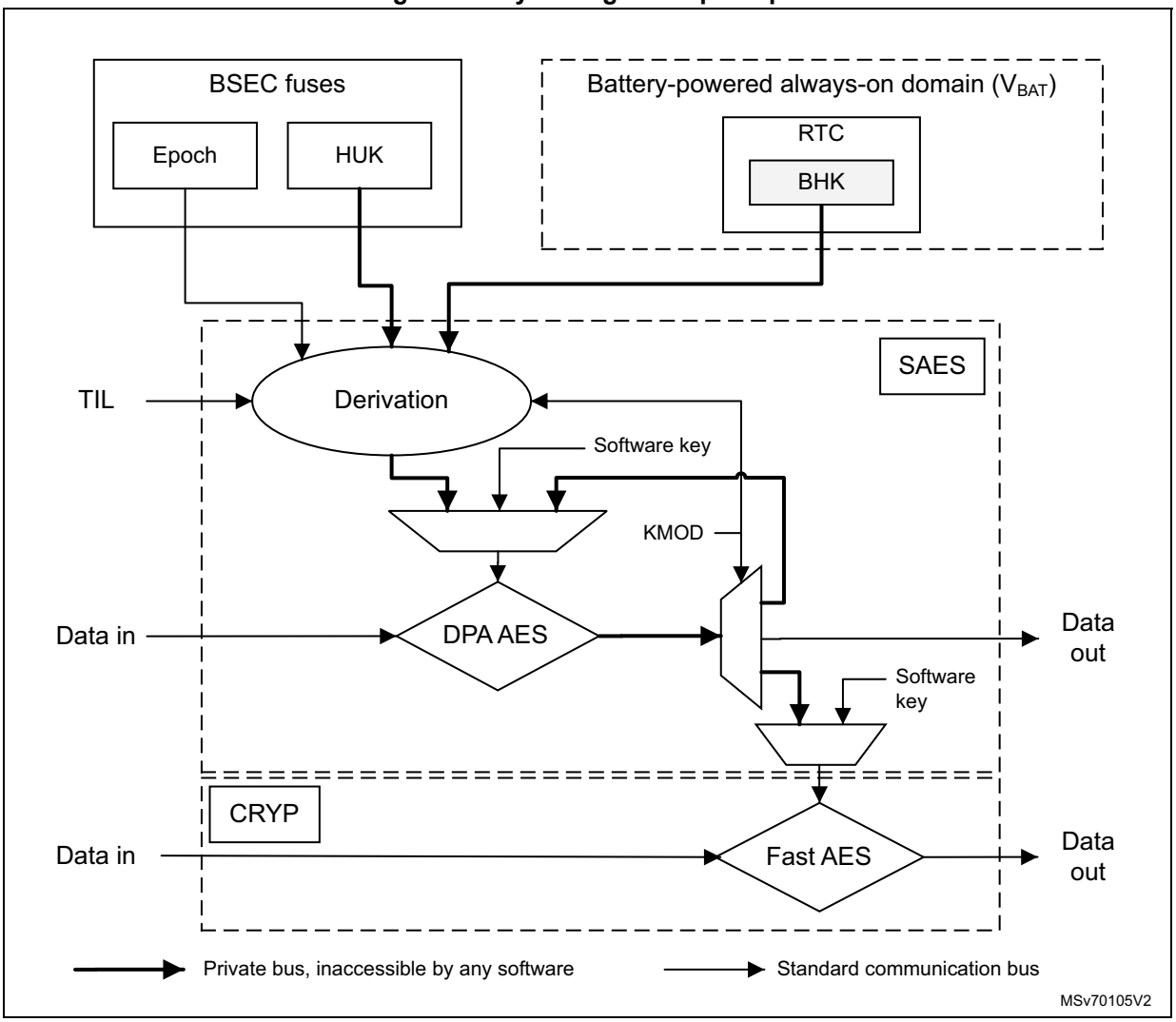
These are provisioned by ST, or by the customer. The secrets can only be read from the BSEC by secure-privileged software. The BSEC offers the possibility to sticky-read-lock fuse secrets: a secure software can hide a secret after it has been used (that is, implement HDPL). Fused secrets can never subsequently be updated.

The HUK is the fuse words 376 to 383 automatically provisioned with a random number during manufacturing. The fuse value is wired directly to the hardware SAES engine, and cannot be handled by software. It is unique to the device, and not readable by it. It constitutes a PU (physically unclonable function).

3.6.4 Hardware key management

Figure 3 shows how key management service applications use the DPA-resistant SAES engine to provision keys to the external flash memory, or fuses in encrypted form (encrypted with a device-unique key derived from HUK and/or BHK). The provisioned key can subsequently be used to encrypt/decrypt data, it cannot be directly handled by the software.

Figure 3. Key management principle



The mechanism is based on KMOD[1:0] and KEYSEL[2:0] in SAES_CR The decryption result is read out and handled by software only when KMOD = 0b00. If KMOD ≠ 0 and KEYSEL is selecting the output of the derivation function, the decryption result goes directly on dedicated buses to key registers in the SAES (when KMOD = 0b01), or (when KMOD = 0b10) to the fast CRYP1. KMOD is also an input to the derivation function that processes the device secret HUK and BHK

A key to be provisioned in the device can be encrypted with a device-unique key (derived from HUK and/or BHK) with KMOD = 0b01 or 0b10 in SAES_CR. This key is then stored somewhere in encrypted format. When this encrypted key needs to be used, it can only be successfully decrypted with the same KMOD setting. The key cannot now be output as clear text and handled by software: it can only be directly placed in unreadable key registers, and used to decrypt or encrypt data. Once the key has been provisioned, it can never again be viewed on the device in clear.

The derivation function depends on the epoch input from the BSEC: if epoch fuses are incremented, any keys provisioned during a previous epoch are never again usable.

The derivation function also depends on the 2-bit TIL input from the BSEC. The software can set this input value to something later than the current HDPL, but can never set it to anything earlier. Keys are always provisioned for use up to a particular HDPL stage. Once that stage has finished, this key is no longer usable.

The KEYSEL field in SAES_CR selects the source of the key used in the SAES.

- 0b001: derived key based only on the HUK
- 0b010: derived key based only on the BHK
- 0b100: derived key based on a combination of HUK and BHK
- 0b000: software loadable (but unreadable) key register

3.6.5 Unique ID

The device stores a 96-bit ID, unique to each device (see *Section 79: Device electronic signature*).

Application services use this unique identity key to identify the product in the cloud network, or to make it difficult for counterfeit devices or clones to inject untrusted data in the network.

Alternatively, the 256-bit device unique key (HUK) can be used (see *Section 3.8.1*).

3.7 Tamper detection and response

The STM32N6x5/x7xx devices include active protection of critical security assets against temperature, voltage, and frequency attacks. The detection of such attacks is managed in the RTC, which resides on V_{BAT} domain (see *Section 61: Real-time clock (RTC)* for details.

The devices support eight input/output tamper pins, which allow four independent active-tamper meshes (or up to seven if the same output pin is shared by several input pins). Four pins are in the V_{BAT} domain.

Active pins are clocked by the LSE, and are functional in all system operating modes (Run, Sleep, Stop, Standby, or Shutdown), and in V_{BAT} mode.

The detection time is programmable, and a digital filtering is available (tamper triggered after two false comparisons in four consecutive comparison samples).

Timestamps are automatically generated when a tamper event occurs.

Table 6. Internal tampers in TAMP

Tamper input	NOER bit number in TAMP_CR3	Tamper source
ITAMP1	0	RTC voltage domain (V_{SW}) monitoring
ITAMP2	1	Temperature monitoring
ITAMP3	2	LSE monitoring(1)
ITAMP4	3	HSE monitoring
ITAMP5	4	RTC calendar overflow (rtc_calovf)
ITAMP6	5	JTAG/SWD access when closed
ITAMP7	6	V_{CORE} monitoring
ITAMP8	7	Monotonic counter overflow

Table 6. Internal tampers in TAMP (continued)

Tamper input	NOER bit number in TAMP_CR3	Tamper source
ITAMP9	8	Fault detected in SAES, PKA, SAES, or RNG
ITAMP10	-	Reserved
ITAMP11	10	IWDG_reset and TAMP_potential_tamper

1. LSE missing or over frequency detection (> 2 MHz), glitch filter (> 2 MHz).

Each tamper source, whether internal or external, can be enabled or disabled.
When the event associated with an enabled tamper source occurs, it is called tamper event, and it results in the following:

- BHK and HUK (dedicated keybuses to SAES) are masked to zero and invalidated.
- PKA, SAES, CRYP, and HASH are reset, unless the boot ROM has set the SYSCFG_POTTAMPRSTCR register to indicate it safely controls them.
- The backup RAM and registers, and the secure RAM (16-Kbyte AHB SRAM2 aliased at 0x28004000 and 0x38004000) are inaccessible (read-as-zero).
- Fuse secrets are un-reloadable by anyone except the boot ROM.

Each tamper source can be further configured to generate an immediate erase of device secrets. An event on a tamper that is configured to do no erase, is called a potential tamper. The secure software must decide on the response.

Note: The boot ROM always sets SYSCONF_POTTAMPRSTCR to override any potential tamper, and to ensure it can access the firmware key, and use the crypto accelerators.

The boot ROM can load the FSBL even if a potential tamper occurs before or during boot.

Once the application analyzes the situation following a potential tamper event notification, there are two possible cases:

- The application launches secret erase with a software command by setting BKERASE in TAMP_CR2.
- The application just clears the flags to release secret blocking (false tamper).

An event on a tamper configured to perform immediate erase of device secrets is called a confirmed tamper. The device secrets consist of the backup registers, MCE key registers, the content of the PKA SRAM, and the secure AHB SRAM2 (erased as soon as V_{DD} is present). There is a configurable option that says that the content of the backup RAM must also be considered as device secrets, and erased on a confirmed tamper. It is optional because the expectation is that the backup RAM only ever contains secrets stored in a form encrypted by a key in backup registers, and when they need to be used, they are decrypted into the secure AHB SRAM2.

Each source of tamper can be configured to trigger the following events:

- Generate an interrupt, able to wake up the device from Stop and Standby modes.
- Generate an hardware trigger for the low-power timers.

The system includes a watchdog that the software needs to refresh regularly (and eventually fires if the software hangs). If the watchdog fires, and there is a potential tamper flagged (not yet analyzed by the software), ITAMP11 is set.

ITAMP11 must be configured as a confirmed tamper (performs an immediate erase of device secrets).

Note: SYSCONF_POTTAMPRSTCR only overrides a potential tamper. If a confirmed tamper occurs while the boot ROM operates, fuse secrets are made inaccessible until the next system reset. This may cause the authentication to fail, and a system reset is eventually triggered by the watchdog. By this time, all device secrets have been erased, and SYSCONF_POTTAMPRSTCR can override blocks that caused the flagging of a tamper event. The software can boot, and, for example, display a message indicating the occurrence of the confirmed tamper.

The effect of low-power modes on a tamper detection are summarized in the table below.

Table 7. Effect of low-power modes on TAMP

Mode	Description
Sleep	No effect on tamper detection features TAMP interrupts cause the device to exit Sleep mode.
Stop	No effect on tamper detection features, except for level detection with filtering and active tamper modes that remain active only when the clock source is LSE or LSI. Tamper events cause the device to exit Stop mode.
Standby	No effect on tamper detection features, except for level detection with filtering and active tamper modes which remain active only when the clock source is LSE or LSI. Tamper events cause the device to exit Standby mode.
Shutdown	No effect on tamper detection features, except for level detection with filtering and active tamper modes which remain active only when the clock source is LSE. Tamper events cause the device to exit Shutdown mode.

3.8 Crypto engines

3.8.1 Accelerators

The device implements cryptographic algorithms as recommended by national security agencies (such as NIST for the U.S.A, BSI for Germany, or ANSSI for France). These algorithms are used to support privacy, authentication, integrity, entropy, and identity attestation.

The embedded crypto engines enable lower processing times, and lower power consumption when performing cryptographic operations. This offloads these computations from the Cortex-M55. The SAES engine offers two important features, which cannot be implemented by software: pure hardware key handling, and DPA resistance.

For product certification purposes, ST provides certified device information on how these security functions are implemented and validated.

For more information on crypto engine processing times, refer to the dedicated section in the reference manual.

Table 8 lists the available accelerated cryptographic operations.

Table 8. Accelerated cryptographic operations

Operations	Algorithm	Specification	Key lengths (in bit)	Modes
Get entropy	RNG	NIST SP800-90B(1)	N/A	Software and hardware(2) modes running in parallel
Encryption, decryption	AES(3)	FIPS PUB 197 NIST SP800-38A	128, 256 (192 in CRYP only)	ECB, CBC, CTR
Authenticated encryption or decryption		NIST SP800-38C NIST SP800-38D		GCM, CCM
Cipher-based message authentication code		NIST SP800-38D		GMAC
Checksum	SHA-1	FIPS PUB 180-4	N/A	Digest 160-bit
Cryptographic hash	SHA-2			SHA-224, SHA-256, SHA-384, SHA-512
Keyed-hashing for message authentication	HMAC	FIPS PUB 198-1 IETF RFC 2104	Short, long	All supported algorithms
Encryption/decryption key-pair generation(4)	RSA	IETF RFC 8017 NIST SP800-56B	Up to 4160	RSAES-OAEP
Signature(4) with hashing Signature verification	RSA	IETF RFC 8017 FIPS PUB 186-4	Up to 4160	PKCS1-v1_5, PSS
	ECDSA	ANSI X9.62 IETF RFC 7027 FIPS PUB 186-4	Up to 640	Refer to the *Section 52: Public key accelerator (PKA)* for details
Key agreement	ECDH	ANSI X9.42		

1. Certifiable using STMicroelectronics reviewed documents.
2. Random number distribution to the SAES and the PKA, using a dedicated hardware bus.
3. ECB, CBC, CTR, GCM, and CCM chaining modes are protected against side-channel and timing attacks in SAES (see *Section 3.8.2*).
4. Private-key cryptography is protected against side-channel and timing attacks.

The PKA accelerates asymmetric crypto operations (such as key pair generation, ECC scalar multiplication, point on curve check). See *Section 52: Public key accelerator (PKA)* for more details.

The STM32N6x5/x7xx devices have two AES engines:

- The CRYP includes a fast AES engine with 11 cycles per 16-byte block (see *Section 49: Cryptographic processor (CRYP)*).
- The SAES contains a DPA-resistant AES engine, which implements counter-measures and mitigations against power and electromagnetic side-channel attacks. The SAES is much slower.

SAES and CRYP engines support the chaining modes mentioned in *Table 8*.

As shown in *Section 3.6*, the SAES can be used for extra-secure on-chip storage for sensitive information. For more information, refer to *Section 48: Secure AES coprocessor (SAES)*.

3.8.2 Memory cipher engine (MCE)

The MCE implements an on-the-fly encryption/decryption of data to/from external memories, which are more vulnerable to side-channel attack than internal memories (traffic on the interface is accessible). The MCE is then designed to make such attacks prohibitively expensive.

The MCE is instantiated in front of each external memory interface (memory-mapped at addresses above 0x60000000). The MCE is used to define four non-overlapping regions of target memory. Each region can be configured to be encrypted in a different way.

Secure software writes master keys into unreadable MCE registers, and then configure each region: define which master key is used for encrypting data in this region.

For each 32-byte memory block in a region, the MCE uses this block address to derive a specific key for encrypting data in this block. This is done on-the-fly, using the selected master-key. Side-channel attacks are then much more expensive, since an individual attack only uncovers 32 bytes of data.

The user chooses once, at application start, whether to use AES (128 or 256) or NOEKEON for the final data encryption (NOEKEON being faster).

Each master key is defined with an algorithm attribute that determines one of the following:

- It is input to a strong derivation for each block.
- It is input to a faster but weaker derivation.
- It is used directly as a very-fast but weak stream cipher (only suitable for read-only data).

Each key and its attributes are sticky-lockable. A key cannot be partially overwritten. If its attributes are changed, the key is zeroed.

When a tamper event is confirmed in the TAMP, all MCE keys are erased, encrypted regions are read as zero, and writes ignored, until the MCE is properly initialized again.

A typical MCE use is detailed in *Section 3.8.2*. For more details on the MCE programming, refer to *Section 51: Memory cipher engine (MCE).*

3.8.3 NPU encryption

The encryption that uses the MCE has a latency of at least 11 cycles, and a block size of 16 bytes. This imposes an heavy overhead on NPU memory transactions. Therefore the NPU is equipped with its own low-latency encryption engines on its 8-byte-wide AXI initiator ports.

The formula for creating a 64-bit ciphertext C_x, from plaintext P_x, which resides at an address $addr_x$, is:

$$C_x = P_x \oplus H(K_{128}, addr_x, ID_y)$$

where:

- H is the cryptographic function.
- K_{128} is a 128-bit key written to configuration registers in the NPU.

- ID_y is a value that software can modify when the NPU rewrites a new value to the same address.

The latency through the NPU encryption engine is 3 or 4 cycles (depending on a strength option selected by the software). See *Section 20: Neural-ART accelerator™ (NPU)*.

3.9 Access controlled debug

The BSEC and DBGMCU hardware support an Arm PSA-compliant debug authentication mechanism.

A secure debug is where the debugger arranges to make secure privileged accesses on its AXI initiator. A nonsecure debug is more restricted: the debugger can only make non-secure privileged accesses via its AXI port.

On power-up, regardless of whether or not the reset pin is still asserted, fuses are immediately read to determine whether the device is BSEC-open or BSEC-closed. If the device is BSEC-open, an attached debugger can perform a “chip discovery” (reading various CPU identification registers), even while holding the device in reset via the RESET pin.

On a BSEC-closed device, the debug access depends on the content of BSEC_DBGCR and BSEC_AP_UNLOCK, which are only writable once by secure-privileged code after the boot ROM has completed. The debug opens when both conditions are met:

- DBG_UNLOCK = 0xB4 in BSEC_DBGCR.
- UNLOCK = 0xB4 in BSEC_AP_UNLOCK.

The current HDPL (reported in BSEC_HDPLSR) must also be non-zero. It must reach the level defined by DBG_AUTH_TIL in BSEC_DBGCR. DBG_AUTH_SEC in BSEC_DBGCR determines whether it is secure or nonsecure debug (see *Section 4: Boot and security control (BSEC)* for more details).

In order to control the debug reopening, the uRoT can impose a debug authentication protocol. This protocol must implement a challenge response mechanism based on asymmetric cryptography to authenticate requests from the host (relying on an OEM debug public key stored in the device by the OEM).

The protocol can implement a bidirectional communication between the host and the device through a mailbox interface located in the DBGMCU. The host debugger writes to the mailbox via the JTAG/SWD interface, and expects to get responses and messages from the device via the same mailbox.

The debug authentication protocol is launched on a power-up of the device, when an open-request message is posted by the host, while the device is still under reset. Trusted DBG_AUTH code sees this, and generates a random challenge token. The debugger appends this token to a certificate chain that confers certain rights to the debugger, signs this "debug certificate" structure, and submits it back to the device. The DBG_AUTH code validates it with respect to a root public key provisioned in fuses by the OEM. The random token stops any later replay of the same interaction.

The debug certificate determines whether the debug is secure or nonsecure.

BSEC_DBGCR is only reset on power-up: it retains its value through warm resets. The debugger does not have to repeat the authentication procedure on system resets.

4 Boot and security control (BSEC)

4.1 Introduction

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

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

4.2 BSEC main features

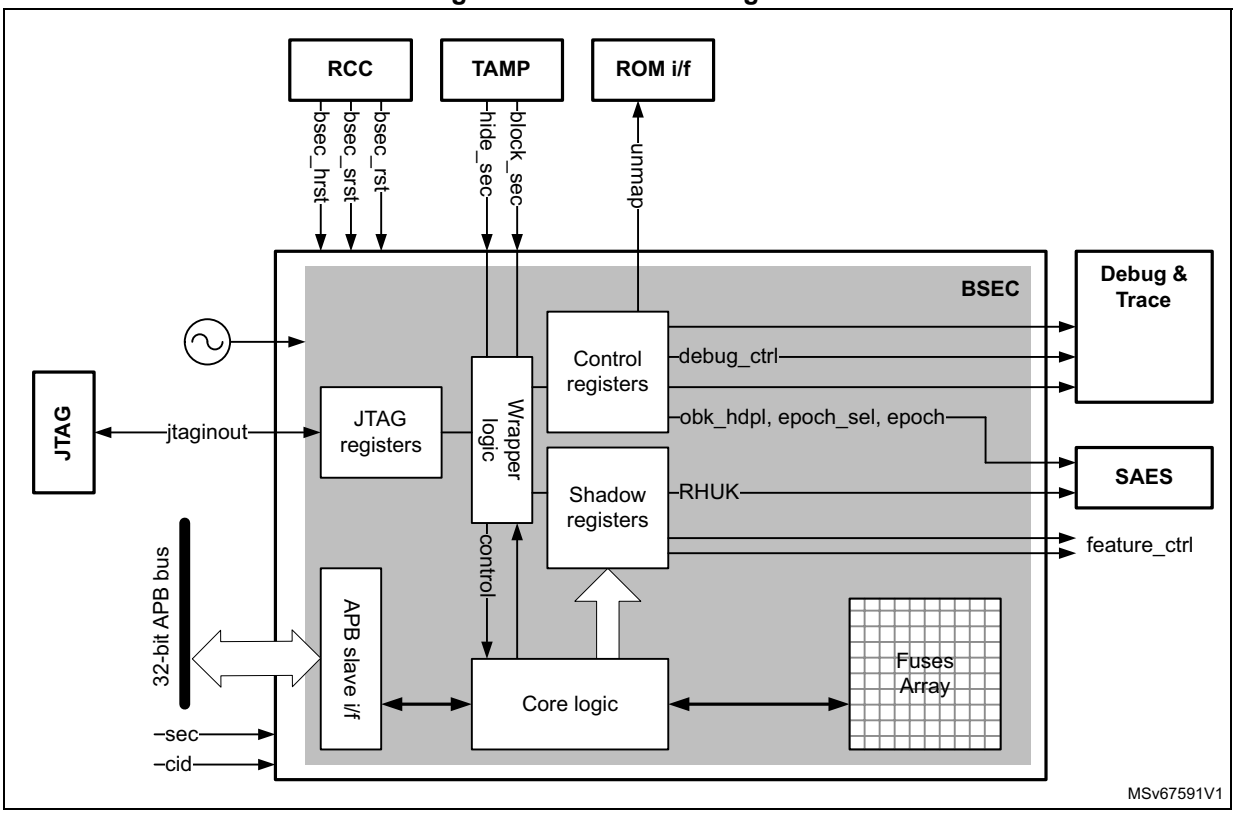
- APB peripheral, accessible through 32-bit word accesses only.
- 12 Kbits of OTP (effective)
  - 4096 lower OTP bits, bitwise (1-bit) programmable
  - 4096 middle OTP bits, bulk (32-bit) programmable
  - 4096 upper OTP bits, bulk (32-bit) programmable, only accessible when BSEC is in the BSEC-closed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

4.3 BSEC functional description

4.3.1 BSEC block diagram

Figure 4 shows the BSEC block diagram.

Figure 4. BSEC block diagram



4.3.2 BSEC internal signals

Table 9 describes the user relevant internal signals interfacing the BSEC peripheral.

Table 9. BSEC internal input/output signals

Signal name	Signal type	Description
bsec_rst	Digital input	BSEC cold/warm resets from RCC (see *Section 4.3.17*).
bsec_srst		
bsec_hrst		BSEC hot reset from RCC (see *Section 4.3.17*).
jtaginout	Digital input and output	Interface to JTAG, including clock and reset.
hide_sec	Digital input	Set by TAMP in case of confirmed tamper (see *Section 4.3.17*).
block_sec	Digital input	Set by TAMP in case of potential tamper (see *Section 4.3.17*).
rhuk	Digital output	256-bit root hardware unique key output to SAES peripheral.

Table 9. BSEC internal input/output signals (continued)

Signal name	Signal type	Description
obk_hdpl	Digital output	Control outputs to SAES peripheral to influence generation of DHUK.
epoch_sel epoch	Digital output	Control outputs to SAES peripheral to influence generation of DHUK.
debug_ctrl	Digital output	Device debug and trace control signals (see *Section 4.3.11*).
feature_ctrl	Digital output	Device feature control signals (see *Section 4.3.12*)
bsec_done	Digital output	End of BSEC cold/warm reset operation to RCC.

4.3.3 BSEC reset and clocks

BSEC is clocked by a device-embedded oscillator (RCBSEC). Because BSEC is clocked by an internal oscillator, an access to fuse array is robust against external fault injection on clocks.

The BSEC has three resets:

- a cold POR (power-on reset) directly driving *bsec_srst*.
- a warm system reset (*bsec_rst*) asserted when POR is asserted, but also by other sources of system reset, including a PAD reset.
- a hot reset input (*bsec_hrst*), driven directly by *bsec_done*.

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

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

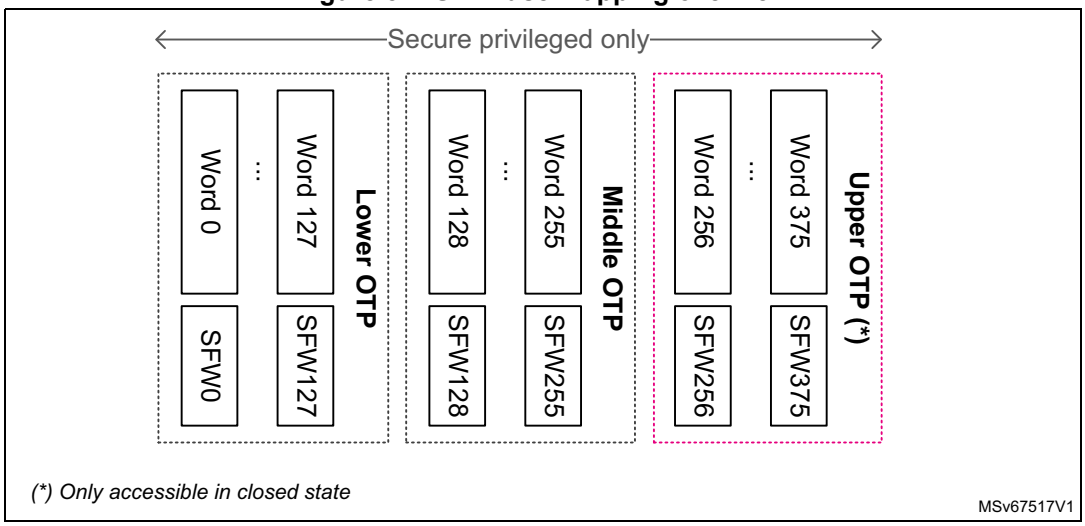
Some flops in the BSEC are on *bsec_hrst*, so they remain in reset until *bsec_done* is asserted. This reset is asserted while the Cortex-M is under reset.

4.3.4 Organization and use of fuses

Organization of fuses

One-time programmable fuse bits are organized in three groups of 32-bit words, as shown on *Figure 5*.

Figure 5. BSEC fuse mapping overview



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

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

The **upper** fuses region is similar to the middle region, except that it is hidden when in the BSEC-open state (see *Section 4.3.7*). Hence it is used to store secrets like symmetric keys or private asymmetric keys.

Each fuse word w is either “shadowed” or “unshadowed”, as defined by flag bit SFWw in corresponding BSEC_SFSRx register. A shadowed fuse word is loaded into flops at cold or warm reset, and can be read quickly by software. An unshadowed fuse word is not read out of the fuse memory at reset, then stored permanently in flops. It needs to be reloaded from the fuse memory each time it is needed. See *Reading fuses* in *Section 4.3.5* for details.

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

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

Details on fuse usage are described in *Section 5: OTP mapping (OTP)*.

4.3.5 Operations on fuses

Reading fuses

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

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

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

Each time the application reloads the content of a word stored in fuse memory using BSEC_OTPCR register with PROG bit cleared, BUSY bit and error status should be checked in BSEC_OTPSR (see *Section 4.3.6*).

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

Writing shadow registers

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

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

Programming fuses

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

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

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

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

Permanent program lock (PPLOCK)

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

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

There is no register in BSEC to store PPLOCK information. When the application needs to know if a fuse word programming is permanently locked, it has first to perform a reload of this word using BSEC_OTPCR and then check if the PPLF bit in BSEC_OTPSR register is set. See *Table 11* for details.

Fuse word locking features

In order to make fuse usage suitable for security, for example to implement the isolation scheme detailed in *Section 4.3.10*, BSEC has the following locking features:

Sticky program lock (SPLOCK)

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

Sticky write lock (SWLOCK)

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

Sticky reload lock (SRLOCK)

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

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

Upper fuses read and write locking (HIDEUP)

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

4.3.6 BSEC read and programming status reporting

After a cold and warm reset BSEC performs a first set of OTP reads, reporting the specific status structure defined in *Table 10*.

Table 10. BSEC initial status reporting structure

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

After any read or programming request, the BUSY bit is cleared, and BSEC reports the specific status structure defined in *Table 11*. When this structure is different than 0 the read or program operation might have failed, as indicated in the table.

Table 11. BSEC ad-hoc error/status reporting structure

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

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

4.3.7 Lifecycle management

BSEC state initialization

Following a cold or warm reset BSEC can be in the following possible security states, defined in *Table 12*.

Table 12. BSEC states definition

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

1. Defined in NVSTATE bitfield in BSEC_SR register (0x16: BSEC-open, 0x0D: BSEC-closed).
2. Derived hardware unique key in SAES peripheral uses RHUK information from BSEC (see *Section 4.3.15*).
3. Cannot be read or programmed by any means (functional or non-functional).
4. ST engineering modes are also available.
5. Root of trust application can unlock debug features unless any of the debug_lock bits (25/24/23/22) is blown in WORD18. ST engineering modes are disabled, by hardware, provided bit 20 of fuse word 124 is blown.
6. At the end of its execution the boot ROM writes 0xB9D8 FF1F to BSEC_UNMAPR, reverting boot ROM access to partial.

Errors following a BSEC initialization are detailed in *Section 4.3.6*. Details about reset management can be found in *Section 4.3.17*.

When neither BSEC-open nor BSEC-closed state (NVSTATE different than 0x16 or 0x0D), the security state is considered invalid, and the most conservative security configuration is applied: that is, fuse programming is disabled, upper fuses read as 0, debug is disabled, ROM is partially unmapped and shadow registers are read only. When NVSTATE = 0x23, a confirmed tamper is active in the device (see *Section 4.3.17*).

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

Opening BSEC

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

The recommended JTAG password programming sequence is:

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

The BSEC opening sequence is described below:

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

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

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

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

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

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

Closing BSEC

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

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

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

With the following conventions:

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

When the device is being closed, the application can limit re-opening cycles to only one attempt by blowing at least one bit in BSEC_FVR1[27:24]. To reduce re-opening cycles to zero, the application must blow at least one bit in BSEC_FVR1[31:28].

Note: For the device to become BSEC-closed, the application must perform a BSEC cold or warm reset.

4.3.8 Epoch registers

There are two 32-bit readable registers, BSEC_EPOCHR0 and BSEC_EPOCHR1, that are writable only by boot ROM accesses. It is the responsibility of the boot ROM to read bitwise fuses and populate these registers.

There is a 1-bit programmable register BSEC_EPOCHSELCR (output on *epoch_sel*) that, when programmed to *n*, ensures that BSEC_EPOCHR*n* is output on *epoch*. Both *epoch* and *epoch_sel* are wired to the SAES where they influence the generation of DHUK, derived from the RHUK.

Secure software can blow additional bits in the appropriate fuse word to increment an Epoch, thereby making impossible the use of all keys derived from the RHUK prior to that point.

4.3.9 BSEC local access control

Any access that is not both secure and privileged is blocked, with read data zeroed and write accesses ignored.

4.3.10 Temporal isolation in BSEC

In order to enforce isolation between boot stages the running application in each boot level must hide the secrets it was using before proceeding to the next level of code in the boot chain. Such protections must remain until the next boot sequence.

Four boot levels, with its corresponding hide protection level (HDPL), are defined:

- Level 0 (HDPL0) corresponds to hardware state machine or ROM code execution.
- Level 1 (HDPL1) corresponds to the execution of STMicroelectronics signed code or ROMed code. This code is the first stage of the secure boot.
- Level 2 (HDPL2) corresponds to the execution of the OEM signed code, stored in embedded SRAM. This code is the second stage of the secure boot.
- Level 3 (HDPL3) corresponds to the application run-time.

To help enforcing such isolation between boot stages, the ROM code and the application secure boot can use the following temporal isolation hardware capabilities of BSEC:

- When BSEC is in a closed state, ROM partially hides itself at the end of its execution in level 0 or level 1, writing 0xB9D8 FF1F to BSEC_UNMAPR register.
- When BSEC is in a closed state, fuse words 364 to 375 are reloadable only when BSEC_UNMAP=0xA1C0DE0D
- Any 32-bit OTP word can be made non-readable and/or non-writable until next cold or warm reset. It can also be made non-programmable, permanently. See Fuse word locking features and Permanent program lock (PPLOCK) methods in *Section 4.3.5*.

Note: Details on fuse usage are described in Section 5: OTP mapping (OTP).

The register BSEC_HDPLSR determines the current hide protection level of the device. The encoding offers physical robustness.

Table 13. Current HDPL coding

BSEC_HDPLSR[HDPL]	Level
0xB4	0
0x51	1
0x8A	2
0x6F	3
others	undefined

The register is reset to 0xB4 by *bsec_hrst*. The register is not directly writable. If less than 3, the current level can be incremented by writing a value of 0x60B166E7 to BSEC_HDPLCR.

The 2-bit INCR field of the BSEC_NEXTHDPLCR register is added to the level given by BSEC_HDPLSR (but clamped at a ceiling of 3), and the result is output on *obk_hdpl[7:0]* as follows.

Table 14. BSEC Next HDPL register usage

<table>
<tr><th colspan="2" rowspan="2">obk_hdpl [7:0]</th><th colspan="4">BSEC_NEXTHDPLCR [INCR]</th></tr>
<tr><th>0</th><th>1</th><th>2</th><th>3</th></tr>
<tr><td rowspan="5">BSEC_HDPLSR [HDPL]</td><td>0 (0xB4)</td><td>0xB4</td><td>0x51</td><td>0x8A</td><td rowspan="3">0x6F</td></tr>
<tr><td>1 (0x51)</td><td>0x51</td><td>0x8A</td><td rowspan="2">0x6F</td></tr>
<tr><td>2 (0x8A)</td><td>0x8A</td><td rowspan="2">0x6F</td></tr>
<tr><td>3 (0x6F)</td><td>0x6F</td><td colspan="2"></td></tr>
<tr><td>others (0xPQ)</td><td colspan="4">0xPQ</td></tr>
</table>

The *obk_hdpl* output goes to the SAES where it influences the generation of DHUK, derived from the RHUK. This mechanism allows keys for a later temporal level to be generated in advance.

4.3.11 BSEC debug and trace control

The wrapper contains a BSEC_DBGCR register, reset to zero on the scratch reset, and writeable only once per warm reset. The bytes of this register are named as follows

Table 15. BSEC debug register legal values

BSEC_DBGCR	field name	legal values
31:24	AUTH_SEC	0x00, 0xB4
23:16	AUTH_HDPL	0x00, 0xB4,0x51,0x8A,0x6F
15:8	UNLOCK	0x00, 0xB4
7:0	reserved	-

AUTH_HDPL encodes levels in the same way as HDPL field in BSEC_HDPLSR register.

Any attempt to write a non-legal byte value into these fields results in a 0x00 being written in that byte field.

The register is forced to zero (writes have no effect) whenever the device is BSEC-open.

The *dbg_unlocked* output, when taking the value of 0xB4, signifies that nonsecure debug is possible. The *dbg_unlocked_sec* output, when it also takes the value of 0xB4, signifies that secure debug is possible.

Both outputs are set to 0xB4 whenever the state is BSEC-open.

Whenever the BSEC state is not BSEC-open, then if HDPL = 0xB4, or AUTH_HDPL is undefined or indicates a level greater than that given by HDPL, they are both set to 0x00, otherwise *dbg_unlocked* takes the value of UNLOCK, and *dbg_unlocked_sec* takes the

value of AUTH_SEC. This implies that secure boot is enforced on a BSEC-closed device. It is the responsibility of the customer's signed code to decide on whether debug becomes unlocked at a certain level. This decision (i.e. the register write) only has to made once per power cycle, since the register is on the scratch reset and retains its value across warm resets.

Note: If a warm reset occurs during a debug session, then during the reset, the debugger becomes temporarily locked, since HDPL is reset to 0. On a BSEC-open device, it becomes unlocked again as soon as the BSEC has booted. On a BSEC-closed device, it becomes unlocked again as soon as HDPL is non-zero and matches or surpasses AUTH_HDPL.

The BSEC_AP_UNLOCK register is forced to zero (writes have no effect) whenever the state is BSEC-open. The *ap_unlocked* output, when taking the value 0xB4, indicates that the debugger can access the CM55 Debug Subsystem. It is set to 0xB4 whenever the state is BSEC-open, otherwise it is directly driven by the BSEC_AP_UNLOCK register, which is reset to 0 on a scratch reset.

A rising edge on *hide_sec* (a confirmed tamper) resets these registers to zero, and close debug. They are not writeable until the next cold or warm reset.

Note: These registers are on the scratch reset so that if software authorizes a debug, that authorization survives subsequent warm resets, and doesn't have to be re-authorized.

4.3.12 Device features control

Device features are controlled through read-only shadowed fuse word 9.See *Section 5: OTP mapping (OTP)* for details.

4.3.13 Scratch registers

The four BSEC_SCRATCHRx registers are only reset by a cold boot. They are therefore useful to store values that must remain persistent for the application across warm boots.

The eight BSEC_WOSCRx registers are only reset by a cold or warm boot. They are therefore useful to store values that must remain persistent for the application across hot boots.

Note: BSEC_WOSCRx registers are write once.

4.3.14 JTAG registers

The BSEC_JTAGINR and BSEC_JTAGOUTR registers are available to facilitate communication between the application (for example ROM) and the debug tools connected through the JTAG interface.

4.3.15 STM32 root hardware unique key (RHUK)

BSEC stores a root HUK that has been programmed by STMicroelectronics during the device manufacturing. A successful provisioning is indicated with HVALID bit set in BSEC_SR register.

The value of this key is obtained from the RNG peripheral of the device, without disclosure to the outside word. This key cannot be read or programmed by any secure application nor debugger.

When BSEC detects an integrity issue while reading the key information from non-volatile memory, or when application sets the sticky HKLOCK bit in BSEC_LOCKR, the key is indicated as invalid. When the key is invalid the SAES peripheral uses a default fixed value.

4.3.16 BSEC error management

Shadowed fuse word automatic readout errors

After a cold or warm reset, shadowed fuse words are automatically loaded into BSEC_FVR registers. At the end of this process, when INIT_DONE is set in BSEC_OTPSR, additional error information can be found in this register, as described in *Section 4.3.6*.

Individual fuse loading errors

Upon each fuse loading from OTP memory array, an 8-bit error structure in BSEC_OTPSR register is updated to tell the application if there was a problem with this particular fuse load. This structure is defined in *Section 4.3.6*.

4.3.17 BSEC tamper response mechanisms

This device uses tamper-detection and response mechanisms that can make it immediately inoperable, resulting in the automatic and immediate erasure of any sensitive data that may be stored in the device. The device also supports the concept of potential tamper event, where the device secrets are blocked while application software verifies that such tamper event is confirmed or not.

BSEC manages above mechanisms in the following way:

- When a tamper source triggers a potential tamper event in TAMP peripheral (NOERASE = 1), BSEC masks the RHUK value to SAES, and upper fuses read as 0 (unless BSEC_UNMAP = 0xA1C0 DE0D, meaning that the boot CPU is executing).
- When a tamper source triggers a confirmed tamper event in TAMP peripheral (NOERASE = 0), or a potential tamper becomes a confirmed tamper, fuses cannot be blown or reloaded, upper fuses read as 0, BSEC_DBGCR and BSEC_AP_UNLOCK registers are cleared, RHUK information is physically cleared to 0 & invalidated, and NVSTATE is set to 0x23 in BSEC_SR register. These limitations are deactivated after device secrets have been erased, and a SoC reset has been applied.

Above limitations are deactivated when the application clears the active tamper source(s) in TAMP peripheral.See *Section 62: Tamper and backup registers (TAMP)* for details.

Note: While tamper events are active the boot CPU can always execute the application firmware following an application reset or an exit from Standby for the boot CPU.

4.4 BSEC interrupts

BSEC does not support any interrupt.

4.5 BSEC registers

Those registers are accessible through the register interface of the BSEC peripheral.

4.5.1 BSEC fuse word w value register (BSEC_FVRw)

Address offset: 0x000 + 0x4 * w, (w = 0 to 375)

Reset value: 0xXXXX XXXX

Note: The reset value is 0x0000 0000 if the fuse word is unshadowed, defined by the fuse value if the fuse word is shadowed.

Secure privileged read and write only. Illegal reads return 0, illegal writes are ignored.

See the bitfield description for further details on fuse word accessibility.

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

Bits 31:0 **FV[31:0]**: fuse value

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

If fuse word {w} is accessible and shadowed, then:

– if SRLOCK{w} = 0, fuse word {w} can be reloaded by software into this register

– if SWLOCK{w} = 0, and {w} > 9, this register can be written to (otherwise, writes are ignored).

If fuse word {w} is accessible and unshadowed (SFW{w} bit cleared in corresponding BSEC_SFSRx register), writes are ignored and provided:

– the last operation on the fuse memory successfully reloaded word {w}

– no unshadowed fuse words have been read since that reload

then a read to this register returns the content of fuse word {w} (otherwise it returns 0).

If the fuse word {w} is inaccessible, then writes are ignored and reads return 0.

Unshadowed fuse word information stored in this register is cleared as soon as it is read.

When the state is BSEC-closed:

– Fuse words 364 to 375 are accessible only when BSEC_UNMAP=0xA1C0DE0D and when the access is from boot CPU.

– Fuse words 256 to 259 read as zero when permanent programming lock is set for those words.

4.5.2 BSEC sticky programming lock register x (BSEC_SPLOCKx)

Address offset: 0x800 + 0x4 * x (x = 0 to 11)

Reset value: 0x0000 0000

Secure privileged read and write only. Illegal reads return 0, illegal writes are ignored.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SPLOCK{i + 32 * x}															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPLOCK{i + 32 * x}															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:0 **SPLOCK{i + 32 * x}**: Sticky programming lock for word {i + 32 * x} (i = 0 to 31)

Setting this bit prevents permanent programming for the fuse word {i + 32 * x} until next cold or warm reset.

0: Fuse word {i + 32 * x} can be burnt in fuse memory array

1: Attempt to program fuse word {i + 32 * x} in OTP memory array is silently ignored

While BUSY is set in BSEC_OTPSR, writes have no effect, and reads return 0.

4.5.3 BSEC sticky write lock register x (BSEC_SWLOCKx)

Address offset: 0x840 +0x4 * x (x=0 to 11)

Reset value: 0x0000 0000

Secure privileged read and write only. Illegal reads return 0, illegal writes are ignored.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWLOCK{i + 32 * x}															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWLOCK{i + 32 * x}															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:0 **SWLOCK{i + 32 * x}**: sticky write lock for shadow register {i + 32 * x} (i = 0 to 31)

When fuse word {i + 32 * x} is shadowed, setting this bit prevents the writing of the shadow register of this fuse word until next cold or warm reset.

0: Write to shadow register BSEC_FVR{i + 32 * x} is allowed

1: Writes to shadow register BSEC_FVR{i + 32 * x} are silently ignored

While BUSY is set in BSEC_OTPSR, writes have no effect, and reads return 0.

4.5.4 BSEC sticky reload lock register x (BSEC_SRLOCKx)

Address offset: 0x880 + 0x4 * x (x = 0 to 11)

Reset value: 0x0000 0000

Secure privileged read and write only. Illegal reads return 0, illegal writes are ignored.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SRLOCK{i + 32 * x}															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRLOCK{i + 32 * x}															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:0 **SRLOCK{i + 32 * x}**: sticky reload lock for fuse word {i + 32 * x} (i = 0 to 31)

Setting this bit locks reloading of fuse word {i + 32 * x} from the OTP memory array until next cold or warm reset.

0: Fuse word {i + 32 * x} loading through BSEC_OTPCR is authorized.

1: Fuse word {i + 32 * x} loading through BSEC_OTPCR is denied until next cold or warm reset.

While BUSY is set in BSEC_OTPSR, writes have no effect, and reads return 0.

4.5.5 BSEC OTP valid register x (BSEC_OTPVLDRx)

Address offset: 0x8C0 + 0x4 * x (x = 0 to 11)

Reset value: 0x0000 0000

Secure privileged read only (return 0 otherwise).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
VLDF{i + 32 * x}															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VLDF{i + 32 * x}															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **VLDF{i + 32 * x}**: Valid flag for shadow register {i + 32 * x} (i = 0 to 31)

This bit represents the validity of the last reload of fuse word {i + 32 * x}.

0: An error occurred while fuse word {i + 32 * x} was last reloaded. The value read from BSEC_FVR{i + 32 * x} register cannot be relied on.

1: Last reload of fuse word {i + 32 * x} was done without error.

Each VLDF bit is updated when BSEC or the allowed application reloads corresponding fuse word from the OTP array.

4.5.6 BSEC shadowed fuses status register x (BSEC_SFSRx)

Address offset: 0x940 + 0x4 * x (x = 0 to 11)

Reset value: 0x0001 EBFF (SFSR0)

Reset value: 0xFF00 000F (SFSR3)

Reset value: 0x0000 000F (SFSR8)

Reset value: 0xFF00 0000 (SFSR11)

Reset value: 0x0000 0000 (others)

Secure privileged read-only (return 0 otherwise).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SFW{i + 32 * x}															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SFW{i + 32 * x}															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SFW{i + 32 * x}**: Shadowed fuse word {i + 32 * x} (i = 0 to 31)

This bit indicates whether fuse word {i + 32 * x} is shadowed or not.

0: Fuse word {i + 32 * x} is not shadowed. Fuse value must be reloaded using BSEC_OTPCR before reading BSEC_FVR{i + 32 * x} register.

1: Fuse word is shadowed in BSEC_FVR{i + 32 * x} register. Reloading the fuse value using BSEC_OTPCR is only required if the fuse value changed since the last BSEC cold or warm reset.

4.5.7 BSEC OTP control register (BSEC_OTPCR)

Address offset: 0xC04

Reset value: 0x0000 0000

Writing to this register causes an access to be made to the fuse memory.

Secure privileged read and write only. Illegal reads return 0, illegal writes are ignored.

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

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

Bits 21:19 **LASTCID[2:0]:** Last CID

This always returns the value 001.

Bits 18:15 Reserved, must be kept at reset value.

Bit 14 **PPLOCK:** Permanent programming lock

When this bit is set together with PROG bit, no further write to this fuse word is possible.

0: Fuse word at address ADDR[8:0] is programmed normally.

1: Fuse word at address ADDR[8:0] is locked, preventing further modifications.

Permanent programming lock mechanism is immutable, as it is committed in the fuse memory.

Bit 13 **PROG:** Fuse word programming

This bit determines whether BSEC operation is a read or a write. If a write is selected, the data stored in BSEC_WDR is programmed in the fuse word indexed by ADDR bitfield.

0: Fuse word read operation is required

1: Fuse word programming operation is required

While BUSY is set in BSEC_OTPSR, writes to PROG bit have no effect.

Bits 12:9 Reserved, must be kept at reset value.

Bits 8:0 **ADDR[8:0]:** Fuse word address

This bitfield represents the address of the fuse word in the OTP memory array.

When a lower fuse address is selected two fuse words are addressed by BSEC.

When fuse words 376 to 383 are selected, read or programming requests are denied.

When fuse words 368 to 375 are selected, programming requests are ignored.

4.5.8 BSEC write data register (BSEC_WDR)

Address offset: 0xC08

Reset value: 0x0000 0000

Secure privileged write only. Any reads return 0.

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

Bits 31:0 **WRDATA[31:0]:** OTP write data

The value written to this register can be written to fuse memory. This happens when the BSEC_OTPCR register is written with the PROG bit set.

When a lower fuse address is selected in BSEC_OTPCR (ADDR=0x0 to 0x3F), two fuse words are programmed by BSEC (one fuse word is programmed otherwise).

While BUSY is set in BSEC_OTPSR, writes to this register have no effect.

4.5.9 BSEC scratch register x (BSEC_SCRATCHRx)

Address offset: 0xE00 + 0x4 * x (x = 0 to 3)

Reset value: 0x0000 0000

This register is reset when *bsec_srst* is asserted.

Secure privileged read and write only. Illegal reads return 0, illegal writes are ignored.

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

Bits 31:0 **SDATA[31:0]**: Scratch data

This register can be used to store values that needs to be persistent across warm reset.

4.5.10 BSEC lock register (BSEC_LOCKR)

Address offset: 0xE10

Reset value: 0x0000 0000

Secure privileged read and write only. Illegal reads return 0, illegal writes are ignored.

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

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

Bit 2 **HKLOCK:** Hardware key lock

0: Derived hardware unique key (DHUK) in SAES peripheral is usable if HVALID is set in BSEC_SR

1: Derived hardware unique key (DHUK) in SAES peripheral is not usable.

Once set, this bit is cleared only by BSEC cold or warm reset.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **GWLOCK:** Global write lock

0: Writes to BSEC registers are allowed

1: Writes to BSEC registers are ignored (fuse programming and fuse reload disabled)

Once set, this bit is cleared only by BSEC cold or warm reset.

4.5.11 BSEC JTAG input register (BSEC_JTAGINR)

Address offset: 0xE14

Reset value: 0x0000 0000

Secure privileged read only (return 0 otherwise).

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

Bits 31:0 **JDATAIN[31:0]**: JTAG input data

Reads give the last input value from the JTAG interface

4.5.12 BSEC JTAG output register (BSEC_JTAGOUTR)

Address offset: 0xE18

Reset value: 0x0000 0000

Secure privileged write only (write ignored otherwise).

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

Bits 31:0 **JDATAOUT[31:0]**: JTAG output data

Application can write any value in this field to be output on the JTAG interface.

4.5.13 BSEC unmap register (BSEC_UNMAPR)

Address offset: 0xE24

Reset value: 0xXXXX XXXX

Note: The reset value is 0xA1C0 DE0D on a BSEC-closed device, 0xFFFF FFFF on a BSEC-open device.

Register bits cannot be cleared by application, they can only be set to 1.

Secure privileged read and write only. Illegal reads return 0, illegal writes are ignored. Only BootROM code can access.

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

Bits 31:0 **UNMAP[31:0]**: Unmap key
If this bitfield equals 0xA1C0 DE0D, the whole ROM is visible, else ROM is partially hidden.

4.5.14 BSEC status register (BSEC_SR)

Address offset: 0xE40

Reset value: 0x0000 0000

Secure privileged read only (return 0 otherwise).

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

Bits 31:26 **NVSTATE[5:0]**: Non-volatile state
0x16: BSEC-open state
0x0D: BSEC-closed state
0x7: BSEC-invalid state
0x23: BSEC-invalid state, with an active confirmed tamper triggered in the device
The BSEC state is evaluated each time INIT_DONE is set in BSEC_OTPSR.
For more information, refer to *Section 4.3.7: Lifecycle management*

Bits 25:17 Reserved, must be kept at reset value.

Bit 16 **DBGREQ**: debug request
0: Host debugger is not requesting debug
1: Host debugger is requesting debug

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

Bit 1 **HVALID**: Hardware key valid
0: Derived hardware unique key (DHUK) feature cannot be used in SAES peripheral.
1: Derived hardware unique key (DHUK) feature can be used in SAES peripheral.

Bit 0 Reserved, must be kept at reset value.

4.5.15 BSEC OTP status register (BSEC_OTPSR)

Address offset: 0xE44

Reset value: 0x0000 0000

This register is refreshed each time a fuse operation is triggered through BSEC_OTPCR.

Secure privileged read only (return 0 otherwise).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AMEF	PPLMF	PPLF	SECF	DEDF	DISTURBF	PROGFAIL
									r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTPSEC	OTPERR	OTPNVIR	Res.	HIDEUP	INIT_DONE	BUSY
									r	r	r		r	r	r

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **AMEF**: Addresses mismatch error flag

This bit is set when the address used by BSEC to read a fuse word is different from the programmed address (either by BSEC or by the application). Such event should never happen in normal BSEC operations.

Bit 21 **PPLMF**: Permanent programming lock mismatch flag

This bit is set when the two words constituting a lower fuse word are reporting different permanent programming lock statuses after BSEC reset or after the latest read.
For more details about permanent programming lock, refer to PPLOCK in BSEC_OTPCR.

Bit 20 **PPLF**: Permanent programming lock flag

This bit is set when the most recently read word is detected permanently locked.
For more details about permanent programming lock, refer to PPLOCK in BSEC_OTPCR.

Bit 19 **SECF**: Single error correction flag

This bit is set when a single-bit-error correction is detected after BSEC reset or after the latest read.

Bit 18 **DEDF**: Double error detection flag

This bit is set when a double error is detected after BSEC reset or after the latest read.
When DEDF bit is set after reading fuse word w, the value in BSEC_FVRw is the one read from OTP memory.

Bit 17 **DISTURBF**: Disturb flag

This bit is set when an unexpected error occurred after BSEC reset or after the latest read.
When DISTURBF bit is set after reading fuse word w, the value in BSEC_FVRw is the one read from OTP memory.

Bit 16 **PROGFAIL**: Programming failed

This bit is set when last OTP programming by BSEC failed.

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **OTPSEC**: OTP with single error correction

This bit is set when at least one single-error correction is detected during BSEC reset operations. In this case shadow registers are usable.

Bit 5 **OTPERR**: OTP with error

This bit is set when at least one error is detected during BSEC reset operations.

Bit 4 **OTPNVIR**: OTP not virgin

This bit is set when first fuse word 0 is detected non-0 after BSEC reset operations.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **HIDEUP**: Hide upper fuse words

This bit is set when application secret fuse words 256 to 383 becomes physically inaccessible, that is functionally hidden fuse words read as 0 and programming requests are ignored.

Bit 1 **INIT_DONE**: Initialization done

This bit is set when BSEC reset operations are completed. All shadowed fuses are readable in BSEC_FVRw registers, and status bits in BSEC_SR and BSEC_OTPSR are up to date.

Bit 0 **BUSY**: Busy flag

0: BSEC is idle.

1: BSEC is busy.

4.5.16 BSEC epoch register (BSEC_EPOCHRx)

Address offset: 0xE80 + 0x4 * x (x = 0 to 1)

Reset value: 0x0000 0000

Writeable by boot ROM only, readable by secure privileged trusted domain or BootROM code only. Illegal reads return 0, illegal writes are ignored.

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

Bits 31:0 **EPOCH[31:0]**: epoch

This value is wired out to the SAES peripheral, depending on the EPOCH_SEL bit in BSEC_EPOCHSELCR.

4.5.17 BSEC epoch selection control register (BSEC_EPOCHSELCR)

Address offset: 0xE88

Reset value: 0x0000 0000

Secure privileged read and write only. Illegal reads return 0, illegal writes are ignored.

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

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

Bit 0 **EPOCH_SEL**: Selection of the epoch value to be sent to the SAES
0: SAES peripheral uses BSEC_EPOCHR0 as EPOCH value.
1: SAES peripheral uses BSEC_EPOCHR1 as EPOCH value.

4.5.18 BSEC debug control register (BSEC_DBGCR)

Address offset: 0xE8C

Reset value: 0x0000 0000

Secure privileged read and write only. Writeable only on a BSEC-closed device, once per warm reset. Illegal reads return 0, illegal writes are ignored. Reset to zero on *bsec_srst* (cold reset).

The AUTH_HDPL byte field indicates a temporal isolation level, as follows: the value 0xB4 means level 0, the value 0x51 means level 1, the value 0x8A means level 2, and the value 0x6F means level 3. All other byte values are undefined.

When the state is BSEC-open, the *dbg_unlocked* and *dbg_unlocked_sec* outputs to the DBG_MCU block are set to 0xB4, and the register is zeroed.

When the state is not BSEC-open, both outputs are zeroed whenever either AUTH_HDPL or BSEC_HDPLSR[HDPL] are undefined, or BSEC_HDPLSR[HDPL] = 0xB4, or the level indicated by AUTH_HDPL is greater than that of BSEC_HDPLSR[HDPL]. Otherwise, the UNLOCK is placed on the *dbg_unlocked* output, and (UNLOCK and AUTH_SEC) is placed on the *dbg_unlocked_sec* output.

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

Bits 31:24 **AUTH_SEC[7:0]:**

0xB4: Secure debug authorized when UNLOCK = 0xB4 and the level indicated in BSEC_HDPLSR is greater than or equal to the level indicated by AUTH_HDPL

Others: Secure debug not authorized (provided state is not BSEC-open)

Bits 23:16 **AUTH_HDPL[7:0]:** level at which debug may be opened.

0xB4: Level 0

0x51: Level 1

0x8A: Level 2

0x6F: Level 3

Others: Undefined

Bits 15:8 **UNLOCK[7:0]:**

0xB4: Nonsecure debug authorized when the level indicated in BSEC_HDPLSR is greater than or equal to the level given by AUTH_HDPL

Others: Debug not authorized (provided state is not BSEC-open)

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

4.5.19 BSEC AP unlock (BSEC_AP_UNLOCK)

Address offset: 0xE90

Reset value: 0x0000 0000

Secure privileged read and write only. Writeable only on a BSEC-closed device, once per warm reset. Illegal reads return 0, illegal writes are ignored. Reset to zero on *bsec_srst* (cold reset). Bit 5 is reset to zero on every warm reset.

When the state is BSEC-open, the *ap_unlocked* output to the DBG_MCU block is set to 0xB4. and the register is zeroed. When the BSEC state is not BSEC-open, it takes the value of the 8-bit UNLOCK field.

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

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **UNLOCK[7:0]:**

0xB4: Unlock debug access port.

Others: Do not unlock.

4.5.20 BSEC HDPL status register (BSEC_HDPLSR)

Address offset: 0xE94

Reset value: 0x0000 00B4

Secure privileged read only. Illegal reads return 0, writes are ignored. Reset to 0xB4 on *bsec_hrst* (hot reset).

When coding a value of less than 3, it can be incremented by writing 0x60B166E7 to BSEC_HDPLCR.

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

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **HDPL[7:0]:** Current HDPL

0xB4: Level 0
0x51: Level 1
0x8A: Level 2
0x6F: Level 3
Others: Undefined

4.5.21 BSEC HDPL control register (BSEC_HDPLCR)

Address offset: 0xE98

Reset value: 0x0000 0000

Secure privileged write only. Illegal reads return 0, writes are ignored.

When BSEC_HDPLSR codes a value of less than 3, BSEC_HDPLSR can be incremented by writing 0x60B166E7 to BSEC_HDPLCR.

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

Bits 31:0 **INCR_HDPL[31:0]**: Increment HDPL

When HDPL is less than 3, writing 0x60B1 66E7 increments HDPL by one in BSEC_HDPLSR.
Otherwise, writes are ignored. Reads return zero.

4.5.22 BSEC next HDPL control register (BSEC_NEXTHDPLCR)

Address offset: 0xE9C

Reset value: 0x0000 0000

Secure privileged read and write only. Illegal reads return 0, writes are ignored.

This value determines the amount to add to the level indicated by BSEC_HDPLSR to output to the SAES on obk_hdpl.

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

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

Bits 1:0 **INCR[1:0]**: Increment
Amount to add to BSEC_HDPLSR to generate *obk_hdpl[7:0]* output (clamped at 3).

4.5.23 BSEC write once scratch register x (BSEC_WOSCRx)

Address offset: 0xF40 + 0x4 * x (x = 0 to 7)

Reset value: 0x0000 0000

This register is reset when *bsec_rst* is asserted.

Secure privileged read and write only. Illegal reads return 0, illegal writes are ignored.

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

Bits 31:0 **WOSDATA[31:0]**: Write once scratch data
ROM code can use this register to store values that needs to be persistent across hot resets. It is writable once, reset by either cold or warm BSEC reset.

4.5.24 BSEC hot reset count register (BSEC_HRCR)

Address offset: 0xFE8

Reset value: 0x0000 0000

This register is reset when *bsec_rst* is asserted.

Secure privileged read only (return 0 otherwise).

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

Bits 31:0 **HRC[31:0]**: Hot reset counter

This bitfield counts the number of hot resets since the last BSEC cold/warm reset.

See *Section 4.3.3: BSEC reset and clocks* for details about resets.

4.5.25 BSEC warm reset count register (BSEC_WRCR)

Address offset: 0xFEC

Reset value: 0x0000 0000

This register is reset when *bsec_srst* is asserted.

Secure privileged read only (return 0 otherwise).

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

Bits 31:0 **WRC[31:0]**: Warm reset counter

This bitfield counts the number of warm resets since the last BSEC cold reset.

See *Section 4.3.3: BSEC reset and clocks* for details about resets.

4.5.26 BSEC register map

Table 16. BSEC register map and reset values

Offset	Register name Reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000 + 0x4 * w, (w=0 to 375)	**BSEC_FVRw**	FV[31:0]																															
	Reset value(1)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0x5E0 - 0x7FC	Reserved	Res.																															

Table 16. BSEC register map and reset values (continued)

Offset	Register name Reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x800 + 0x4 * x, (x=0 to 11)	**BSEC_ SPLOCKx**	SPLOCK{i + 32 * x} (i = 31 to 0)																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x830 - 0x83C	Reserved	Res.																															
0x840 + 0x4 * x (x=0 to 11)	**BSEC_ SWLOCKx**	SWLOCK{i + 32 * x} (i = 31 to 0)																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x870 - 0x87C	Reserved	Res.																															
0x880 + 0x4 * x (x=0 to 11)	**BSEC_ SRLOCKx**	SRLOCK{i + 32 * x} (i = 31 to 0)																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x8B0 - 0x8BC	Reserved	Res.																															
0x8C0 + 0x4 * x (x=0 to 11)	**BSEC_ OTPVLDRx**	VLDF{i + 32 * x} (i = 31 to 0)																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x8F0 - 0x93C	Reserved	Res.																															
0x940 + 0x4 * x (x=0 to 11)	**BSEC_SFSRx**	SFW{i + 32 * x} (i = 31 to 0)																															
	Reset value[1]	x	x	x	x	x	x	x	x	0	0	0	0	0	0	0	x	x	x	x	0	x	0	x	x	x	x	x	x	x	x	x	x
0x970 - 0xC00	Reserved	Res.																															
0xC04	**BSEC_OTPCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LASTCID [2:0]			Res.	Res.	Res.	Res.	PPLOCK	PROG	Res.	Res.	Res.	Res.	ADDR[8:0]								
	Reset value											0	0	0					0	0					0	0	0	0	0	0	0	0	0
0xC08	**BSEC_WDR**	WRDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xC0C - 0xDFC	Reserved	Res.																															
0xE00 + 0x4 * x (x=0 to 3)	**BSEC_ SCRATCHRx**	SDATA[31:0]																															
	Reset value	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0xE10	**BSEC_LOCKR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HKLOCK	Res.	GWLOCK
	Reset value																														0		0
0xE14	**BSEC_ JTAGINR**	JDATAIN[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xE18	**BSEC_ JTAGOUTR**	JDATAOUT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xE1C	Reserved	Reserved																															
0xE20	Reserved	Res.																															
0xE24	**BSEC_ UNMAPR**	UNMAP[31:0]																															
	Reset value[1]	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
0xE28 - 0xE3C	Reserved	Res.																															
0xE40	**BSEC_SR**	NVSTATE[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBGREQ	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HVALID	Res.
	Reset value	0	0	0	0	0	0										0															0	

Table 16. BSEC register map and reset values (continued)

Offset	Register name Reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xE44	**BSEC_OTPSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AMEF	PPLMF	PPLF	SECF	DEDF	DISTURBF	PROGFAIL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTPSEC	OTPERR	OTPNVIR	Res.	HIDEUP	INIT_DONE	BUSY
	Reset value										0	0	0	0	0	0	0										0	0	0		0	0	0
0xE48 - 0xE7C	Reserved	Res.																															
0xE80 + 0x4 * x (x= 0 to 1)	**BSEC_ EPOCHRx**	EPOCH[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xE88	**BSEC_ EPOCHSELCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EPOCH_SEL
	Reset value																																0
0xE8C	**BSEC_DBGCR**	AUTH_SEC[7:0]								AUTH_HDPL[7:0]								UNLOCK[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								
0xE90	BSEC_ AP_UNLOCK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UNLOCK[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0xE94	BSEC_ HDPLSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPL[7:0]							
	Reset value																									0	0	0	0	0	0	B	4
0xE98	BSEC_ HDPLCR	INCR_HDPL[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xE9C	BSEC_NEXTHD PLCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INCR [1:0]	
	Reset value																															0	0
0xEA0 - 0xEA8	Reserved	Reserved																															
0xEAC - 0xEBC	Reserved	Reserved																															
0xEC0 - 0xF3C	Reserved	Reserved																															
0xF40 + 0x4*x (x=0 to 7)	BSEC_ WOSCRx	WOSDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xF60 - 0xFE4	Reserved	Res.																															
0xFE8	BSEC_HRCR	HRC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xFEC	BSEC_WRCR	WRC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. See register description.

Refer to *Section 2.3* for the register boundary addresses.

5 OTP mapping (OTP)

STM32N6x5/x7xx devices have 12032 OTP (one-time programmable) bits, which can be read-accessed in 376 words: BSEC_OTP_DATAx (x = 0 to 375).

OTP bits are read or programmed by the BSEC (boot, security, and OTP control),as detailed in *Section 4: Boot and security control (BSEC)*. They are organized in regions:

- Lower OTP region (OTP0 to OTP127)
- Mid OTP region (OTP128 to OTP255)
- Upper OTP region (OTP256 to OTP375)

Before being programmed, the OTP bits are at 0. The OTP words are listed in *Table 17*:

- *Description*: acronym and function. Words marked “Available to customer” have no predefined usage.
- *Shadowed*: loaded into a register after reset, can be read without a reload. A shadowed word may be sticky-write-locked by earlier software (such as BOOTROM). The first ten shadowed words are never writable.
- *Prog-locked by ST*: words locked during manufacturing.

Table 17. OTP mapping

Word	Description	Shadowed	Prog-locked by ST
Lower OTP region			
OTP0	OTP_HW_WORD0	Yes	Yes
OTP1	OTP_HW_WORD1	Yes	No
OTP2	OTP_HW_WORD2	Yes	No
OTP3	OTP_HW_WORD3	Yes	No
OTP4	OTP_HW_WORD4	Yes	Yes
OTP5	ID0	Yes	Yes
OTP6	ID1	Yes	Yes
OTP7	ID2	Yes	Yes
OTP8	OTP_RPN_OPTION	Yes	Yes
OTP9	OTP_RPN_CODING	Yes	Yes
OTP10	BOOTROM_CONFIG_1	No	No
OTP11	BOOTROM_CONFIG_2	Yes	No
OTP12	BOOTROM_CONFIG_3	No	No
OTP13	BOOTROM_CONFIG_4	Yes	No
OTP14	BOOTROM_CONFIG_5	Yes	No
OTP15	BOOTROM_CONFIG_6	Yes	No
OTP16	BOOTROM_CONFIG_7	Yes	No
OTP17	BOOTROM_CONFIG_8	No	No
OTP18	BOOTROM_CONFIG_9	No	No

Table 17. OTP mapping (continued)

Word	Description	Shadowed	Prog-locked by ST
OTP19	BOOTROM_CONFIG_10	No	No
OTP20	BOOTROM_CONFIG_11	No	No
OTP21	BOOTROM_CONFIG_12	No	No
OTP22	BOOTROM_CONFIG_13	No	No
OTP23	BOOTROM_CONFIG_14	No	Yes
OTP24	BOOT_TZ_EPOCH0	No	No
OTP25	BOOT_TZ_EPOCH1	No	No
OTP26	BOOT_TZ_EPOCH2	No	No
OTP27	BOOT_TZ_EPOCH3	No	No
OTP28	BOOT_TZ_EPOCH4	No	No
OTP29	BOOT_TZ_EPOCH5	No	No
OTP30	BOOT_TZ_EPOCH6	No	No
OTP31	BOOT_TZ_EPOCH7	No	No
OTP32	BOOT_NS_EPOCH0	No	No
OTP33	BOOT_NS_EPOCH1	No	No
OTP34	BOOT_NS_EPOCH2	No	No
OTP35	BOOT_NS_EPOCH3	No	No
OTP36	BOOT_NS_EPOCH4	No	No
OTP37	BOOT_NS_EPOCH5	No	No
OTP38	BOOT_NS_EPOCH6	No	No
OTP39	BOOT_NS_EPOCH7	No	No
OTP40 to OTP55	Available to customer	No	No
OTP56	TAMP_EN	No	No
OTP57	TAMP_CFM	No	No
OTP58	TAMP_CFG	No	No
OTP59 to OTP95	Available to customer	No	No
OTP96 to OTP99	Reserved	-	-
OTP100 to OTP103	Reserved	-	-
OTP104 to OTP123	Reserved	-	-
OTP124	HCONF1	Yes	No
OTP125 to OTP127	Reserved	-	-
Mid OTP region			
OTP128	STM32_CERTIF0	No	Yes
OTP129	STM32_CERTIF1	No	Yes
OTP130	STM32_CERTIF2	No	Yes

Table 17. OTP mapping (continued)

Word	Description	Shadowed	Prog-locked by ST
OTP131	STM32_CERTIF3	No	Yes
OTP132	STM32_CERTIF4	No	Yes
OTP133	STM32_CERTIF5	No	Yes
OTP134	STM32_CERTIF6	No	Yes
OTP135	STM32_CERTIF7	No	Yes
OTP136	STM32_CERTIF8	No	Yes
OTP137	STM32_CERTIF9	No	Yes
OTP138	STM32_CERTIF10	No	Yes
OTP139	STM32_CERTIF12	No	Yes
OTP140	STM32_CERTIF12	No	Yes
OTP141	STM32_CERTIF13	No	Yes
OTP142	STM32_CERTIF14	No	Yes
OTP143	STM32_CERTIF15	No	Yes
OTP144	STM32PUBKEY0	No	Yes
OTP145	STM32PUBKEY1	No	Yes
OTP146	STM32PUBKEY2	No	Yes
OTP147	STM32PUBKEY3	No	Yes
OTP148	STM32PUBKEY4	No	Yes
OTP149	STM32PUBKEY5	No	Yes
OTP150	STM32PUBKEY6	No	Yes
OTP151	STM32PUBKEY7	No	Yes
OTP152	STM32PUBKEY8	No	Yes
OTP153	STM32PUBKEY9	No	Yes
OTP154	STM32PUBKEY10	No	Yes
OTP155	STM32PUBKEY11	No	Yes
OTP156	STM32PUBKEY12	No	Yes
OTP157	STM32PUBKEY13	No	Yes
OTP158	STM32PUBKEY14	No	Yes
OTP159	STM32PUBKEY15	No	Yes
OTP160	OTP_ROT_HASH0	No	No
OTP161	OTP_ROT_HASH1	No	No
OTP162	OTP_ROT_HASH2	No	No
OTP163	OTP_ROT_HASH3	No	No
OTP164	OTP_ROT_HASH4	No	No
OTP165	OTP_ROT_HASH5	No	No

Table 17. OTP mapping (continued)

Word	Description	Shadowed	Prog-locked by ST
OTP166	OTP_ROT_HASH6	No	No
OTP167	OTP_ROT_HASH7	No	No
OTP168	ST_RSSE_EDMK_DERIV_CSTE_FUSE	No	Yes
OTP169	OTP_MAC1_ADDR_LOW	No	No
OTP170	OTP_MAC1_ADDR_HIGH	No	No
OTP171	OTP_MAC2_ADDR_LOW	No	No
OTP172	OTP_MAC2_ADDR_HIGH	No	No
OTP173 to OTP255	Available to customer	No	No
Upper OTP region			
OTP256	OTP_RMA_LOCK_PSWD0	Yes	No
OTP257	OTP_RMA_LOCK_PSWD1	Yes	No
OTP258	OTP_RMA_LOCK_PSWD2	Yes	No
OTP259	OTP_RMA_LOCK_PSWD3	Yes	No
OTP260 to OTP363	OEM secrets available to customer	No	No
OTP364	OEM_SECRET_FOR_CRYPTED_BOOT0	No	No
OTP365	OEM_SECRET_FOR_CRYPTED_BOOT1	No	No
OTP366	OEM_SECRET_FOR_CRYPTED_BOOT2	No	No
OTP367	OEM_SECRET_FOR_CRYPTED_BOOT3	No	No
OTP368	STM32PRVKEY0	No	Yes
OTP369	STM32PRVKEY1	No	Yes
OTP370	STM32PRVKEY2	No	Yes
OTP371	STM32PRVKEY3	No	Yes
OTP372	STM32PRVKEY4	No	Yes
OTP373	STM32PRVKEY5	No	Yes
OTP374	STM32PRVKEY6	No	Yes
OTP375	STM32PRVKEY7	No	Yes

The following OTP words are described in detail in *Table 18* or by BSEC:

- OTP0 to OTP4: used by BSEC for the security life cycle (see *Section 4: Boot and security control (BSEC)*)
- OTP9: used to code device part number (see *Section 79: Device electronic signature*)
- OTP10 to OTP22: used for the boot ROM configuration
- OTP56 to OTP58: used to record tamper configuration
- OTP96 to OTP99 and OTP104 to OTP 111: used for analog calibration parameters, set by ST
- OTP112 to OTP 119: used for engineering purposes, set by ST

- OTP120 to OTP123: used for settings such as analog trimming, set by ST
- OTP124: used for the product configuration by the user
- OTP125 to OTP 127: used for memory repair, set by ST.

Table 18. OTP fuse description (lower OTP region)

OTP word	OTP bits	Name or description	Detailed description
OTP0	-	OTP_HW_WORD0	OTP check word (virgin → non-virgin)
OTP1	-	OTP_HW_WORD1	OTP security word to close security state
OTP2	-	OTP_HW_WORD2	OTP word for reopening (close → open) via RMA password: RMA bits
OTP3	-	OTP_HW_WORD3	OTP word for reopening (close → open) via RMA password: RMA tries bits
OTP4	-	OTP_HW_WORD4	OTP word for TK retries (ECIES) and retention cell disabling
OTP5	-	ID0	96-bit unique ID for engineering purpose
OTP6	-	ID1	
OTP7	-	ID2	
OTP8	-	OTP_RPN_CODING	Reserved
OTP9	-	RPN_CODING	See *Section 79: Device electronic signature*
OTP10	-	BOOTROM_CONFIG_1	Boot source configuration word
	[0]	stkeyprov_ecies_attempted	Status of ECIES ST key provisioning when it was attempted: – 0 (failed): ECIES ST key provisioning last attempt failed – 1 (successful): ECIES ST key provisioning last attempt successful
	[1]	stkeyprov_ecies_ok	– 0 (no): HWKEY not provisioned – 1 (yes): HWKEY provisioned
	[6:2]	Reserved	Reserved
	[14:7]	security_counter	Security counter involved in product ID for chip certificate verification by HSM-OEM in SFI context
	[18:15]	st_pub_key_id	ST ECDSA public key ID (ST key instance fuse part) involved in product ID for chip certificate verification by HSM-OEM in SFI context
	[26:19]	rssefw_active_signing_key	[1-256] → [1-8]: Value of monotonic counter is X, where X is the position of the most significant bit at 1. Eight possible ST public keys (ST key revocation feature for RSSe_FW authentication)
	[31:27]	Reserved	Reserved
OTP11	-	BOOTROM_CONFIG_2	-
	[0]	no_data_cache	– 0 (enabled): DCACHE used by boot ROM – 1 (disabled): DCACHE not used by boot ROM
	[1]	no_cpu_pll	– 0 (enabled): PLLs for CPU/AXI enabled for cold boot – 1 (disabled): PLLs for CPU/AXI not enabled for cold boot

Table 18. OTP fuse description (lower OTP region) (continued)

OTP word	OTP bits	Name or description	Detailed description
OTP11	[2]	sdmmc1_not_default_af	– 0 (no): SDMMC1 uses default hard coded AFmux – 1 (yes): SDMMC1 uses AFmux defined in OTP
	[3]	sdmmc2_not_default_af	– 0 (no): SDMMC2 uses default hard coded AFmux – 1 (yes): SDMMC2 uses AFmux defined in OTP
	[4]	Reserved	Reserved
	[8:5]	flash_boot_source	If different from 0, identifies the flash memory used to boot: – 1 (sdcard): SD-Card SDMMC1 – 2 (emmc): *e*.MMC SDMMC1 – 3 (snor): XSPI NOR – 4 (snand): XSPI NAND – 5 (hflash): XSPI HyperFlash – 6 (pnand): FMC pNAND – 7 (sdcard): SD-Card SDMMC2 – 8 (emmc): *e*.MMC SDMMC2 – others: invalid
	[16:9]	boot_source_disable	Each bit disables a boot source (default to UART if all disabled): – 0x01 (usb): USB boot source disabled – 0x02 (uart): UART boot source disabled – 0x04 (fdcan): FDCAN boot source disabled – 0x08 (spi): SPI boot source disabled – 0x10 (i2c): I2C boot source disabled
	[19:17]	spi_instance_disable	– 0b001: SPI1 disabled – 0b010: SPI2 disabled – 0b100: SPI3 disabled
	[22:20]	uart_instance_disable	– 0b001: USART1 disabled – 0b010: USART2 disabled – 0b100: USART3 disabled
	[25:23]	fdcan_instance_disable	– 0b001: disable FDCAN1 disabled – 0b010: disable FDCAN2 disabled – 0b100: disable FDCAN3 disabled
	[28:26]	i2c_instance_disable	– 0b001: I2C1 disabled – 0b010: I2C2 disabled – 0b100: I2C3 disabled
	[29]	dont_boot_on_cfm_tamper	– 0: boot on confirmed tamper – 1: do not boot on confirmed tamper
	[30]	tamp_boot_cfg_glob_enable	Enable the configuration of tampers in boot ROM before boot: – 0: configuration of tampers is disabled – 1: configuration of tampers is enabled
	[31]	Reserved	Reserved

Table 18. OTP fuse description (lower OTP region) (continued)

OTP word	OTP bits	Name or description	Detailed description
OTP12	-	BOOTROM_CONFIG_3	-
	[31:0]	st_fsbl_monotonic_counter	[1-0xFFFF] → [1-32]: Value of monotonic counter is X, where X is the position of the most significant bit at 1
OTP13	-	BOOTROM_CONFIG_4	Boot source configuration word
	[3:0]	mode0	– 0 (af_nopull_ls): AF; no pull; low speed – 1 (af_nopull_ms): AF; no pull; medium speed – 2 (af_nopull_hs): AF; no pull; high speed – 3 (af_pullup_ls): AF; pull up; low speed – 4 (af_pullup_ms): AF; pull up; medium speed – 5 (af_pullup_hs): AF; pull up; high speed – 6 (af_pulldown_ls): AF; pull down; low speed – 7 (af_pulldown_ms): AF; pull down; medium speed – 8 (af_pulldown_hs): AF; pull down; high speed – 9 (gpio_out_high): GPIO output high – 10 (gpio_out_low): GPIO output low – 11 (gpio_in): GPIO input – 12 (gpio_open_nopull): GPIO open drain; No pull – 13 (gpio_open_pullup): GPIO open drain; pull up – 14 (gpio_open_pulldown): GPIO open drain; pull down – 15 (gpio_analog): GPIO analog mode
	[7:4]	afmux0	Values between 0 and 15
	[11:8]	pin0	– [0-15]: pin ID between 0 and 15 for GPIOA to GPIOG and GPIOP – [0-12]: pin ID between 0 and 12 for GPION – [0-8]: pin ID between 0 and 8 for GPIOH and GPIOQ – [0-5]: pin ID between 0 and 5 for GPIOO
	[15:12]	port0	– 0: reserved – 1 (PA): Bank A – 2 (PB): Bank B – 3 (PC): Bank C – 4 (PD): Bank D – 5 (PE): Bank E – 6 (PF): Bank F – 7 (PG): Bank G – 8 (PH): Bank H – 9 (PN): Bank N – 10 (PO): Bank O – 11 (PP): Bank P – 12 (PQ): Bank Q – 0b1111: Invalid configuration
	[19:16]	mode1	idem BOOTROM_CONFIG_4.mode0
	[23:20]	afmux1	idem BOOTROM_CONFIG_4.afmux0

Table 18. OTP fuse description (lower OTP region) (continued)

OTP word	OTP bits	Name or description	Detailed description
OTP13	[27:24]	pin1	idem BOOTROM_CONFIG_4.pin0
	[31:28]	port1	idem BOOTROM_CONFIG_4.port0
OTP14	-	BOOTROM_CONFIG_5	-
	[3:0]	mode0	idem BOOTROM_CONFIG_4.mode0
	[7:4]	afmux0	idem BOOTROM_CONFIG_4.afmux0
	[11:8]	pin0	idem BOOTROM_CONFIG_4.pin0
	[15:12]	port0	idem BOOTROM_CONFIG_4.port0
	[19:16]	mode1	idem BOOTROM_CONFIG_4.mode0
	[23:20]	afmux1	idem BOOTROM_CONFIG_4.afmux0
	[27:24]	pin1	idem BOOTROM_CONFIG_4.pin0
	[31:28]	port1	idem BOOTROM_CONFIG_4.port0
OTP15	-	BOOTROM_CONFIG_6	-
	[3:0]	mode0	idem BOOTROM_CONFIG_4.mode0
	[7:4]	afmux0	idem BOOTROM_CONFIG_4.afmux0
	[11:8]	pin0	idem BOOTROM_CONFIG_4.pin0
	[15:12]	port0	idem BOOTROM_CONFIG_4.port0
	[19:16]	mode1	idem BOOTROM_CONFIG_4.mode0
	[23:20]	afmux1	idem BOOTROM_CONFIG_4.afmux0
	[27:24]	pin1	idem BOOTROM_CONFIG_4.pin0
	[31:28]	port1	idem BOOTROM_CONFIG_4.port0
OTP16	-	BOOTROM_CONFIG_7	-
	[0]	disable_traces	– 0 (no): boot ROM traces disabled – 1 (yes): boot ROM traces enabled
	[1]	disable_hse_freq_detect	– 0 (no): HSE frequency auto-detection disabled – 1 (yes): HSE frequency auto-detection enabled
	[2]	disable_hse_bypass_detect	– 0 (no): HSE bypass detection disabled – 1 (yes): HSE bypass detection enabled
	[4:3]	Reserved	Reserved
	[5]	fmc_force_sw_reset	-
	[6]	emergency_debug_req	– 0 (no): emergency debug not requested – 1 (yes): emergency debug requested
	[7]	emmc_128k_boot_partition	– 0 (no): BootROM does not support e.MMC with 128-Kbyte boot partition – 1 (yes): BootROM supports e.MMC with 128-Kbyte boot partition

Table 18. OTP fuse description (lower OTP region) (continued)

OTP word	OTP bits	Name or description	Detailed description
OTP16	[8]	Reserved	Reserved
	[9]	iomgr_port	Reserved
	[10]	iomgr_muxen	Reserved
	[13:11]	HSE_value	– 0b000 (auto): HSE value is auto-detected among 10, 12, 14, 16, 20, 24, 28, 32, 36, 40, and 48 MHz. – 0b001: HSE = 19.2 MHz – 0b010: HSE = 20 MHz – 0b011: HSE = 24 MHz – 0b100: HSE = 38.4 MHz – 0b101: HSE = 40 MHz – 0b110: HSE = 48 MHz – 0b111: Reserved
	[14]	snand_need_plane_select_1	– 0 (no): serial NAND plane select not needed – 1 (yes): serial NAND plane select needed
	[17:15]	pnand_number_of_ecc_bits_1	– 0 (unset): ECC unset – 1 (hamming): ECC 1 bit (Hamming) – 2 (bch4): ECC 4 bits (BCH4) – 3 (bch8): ECC 8 bits (BCH8) – 4 (ondie): on-die ECC
	[18]	pnand_bus_width_1	– 0 (8 bits): data width is 8 bits – 1 (16 bits): data width is 16 bits
	[26:19]	nand_nb_of_blocks_1	[1-256]: number of block = 256 × value
	[28:27]	nand_block_size_1	[1-4]: block size in number of pages
	[30:29]	nand_page_size_1	– 0 (64): 64 pages per block – 1 (128): 128 pages per block – 2 (256): 256 pages per block
	[31]	pnand_param_stored_in_otp	– 0 (no): BootROM uses ONFI parameter table to get parallel NAND parameters – 1 (yes): parallel NAND parameters are defined in bank1 or bank2, depending on nand_config_distribution value
OTP17	-	BOOTROM_CONFIG_8	-
	[7:0]	oem_ecdsa_active_key	[1-256] → [1-8]: Value of monotonic counter is X, where X is the position of the most significant bit at 1. Eight possible OEM public keys (OEM key revocation feature for OEM-FSBL authentication)
	[31:8]	Reserved	Reserved

Table 18. OTP fuse description (lower OTP region) (continued)

OTP word	OTP bits	Name or description	Detailed description
OTP18	-	BOOTROM_CONFIG_9	-
	[3:0]	secure_boot	– 0 (closed_unlocked): device in CLOSED_UNLOCKED state. Secure boot is not enforced (OEM FSBL authentication is not mandatory). – 1 (closed_locked): device in CLOSED_LOCKED state. Secure boot is enforced (OEM FSBL authentication is mandatory).
	[4]	fsbl_decrypt_prio	– 0 (cryp): BootROM uses CRYP to decrypt FSBL – 1 (saes): BootROM uses SAES to decrypt FSBL
	[8:5]	prov_done	– 0 (no): provisioning not done or not finished successfully. Device is CLOSED_LOCKED_UNPROVD and accepts only ST-BootExtension FSBL. – 1 (yes): provisioning successfully completed. Device is CLOSED_LOCKED_PROVD and accepts only OEM FSBL.
	[12:9]	enable_fingerprint	– 0x0: fingerprint feature is disabled – others: fingerprint feature is enabled
	[15:13]	Reserved	Reserved
	[21:16]	nb_added_secrets	Number of OTP words located in upper area [360 -nb_added_stsecrets ... 359] provisioned (in encrypted mode) with ST secrets. These will be decoded and used by RSSE fw. Coding up to 64 ST secrets to provision in EWS (with DEV_BOOT)
	[25:22]	debug_lock	– 0: do not lock debug enabling – [1-64]: lock debug enabling
	[26]	ns_epoch_enable	– 0: the BootROM only sets bsec_epoch0 – 1: the BootROM set both bsec_epoch0 and bsec_epoch1
	[31:27]	Reserved	Reserved

Table 18. OTP fuse description (lower OTP region) (continued)

OTP word	OTP bits	Name or description	Detailed description
OTP19	-	BOOTROM_CONFIG_10	-
	[17:0]	Reserved	Reserved
	[20:18]	rng_htcr_value	– 1: 0xA2B3 – 2: 0xAA74 – 3: 0xA6BA – 4: 0x9AAE – 5: 0x72AC – 6: 0xAAC7 – others: RNG HTCR not modified
	[24:21]	dev_boot_port	– 0: reserved – 1 (PA): Bank A – 2 (PB): Bank B – 3 (PC): Bank C – 4 (PD): Bank D – 5 (PE): Bank E – 6 (PF): Bank F – 7 (PG): Bank G – 8 (PH): Bank H – 9 (PN): Bank N – 10 (PO): Bank O – 11 (PP): Bank P – 12 (PQ): Bank Q
	[28:25]	dev_boot_pin	– [0-15]: pin ID between 0 and 15 for GPIOA to GPIOG, and GPIOP – [0-12]: pin ID between 0 and 12 for GPION – [0-8]: pin ID between 0 and 8 for GPIOH and GPIOQ – [0-5]: pin ID between 0 and 5 for GPIOO
	[31:29]	Reserved	Reserved
OTP20	-	BOOTROM_CONFIG_11	-
	[31:0]	oem_fsbl_monotonic_counter	[1-0xFFFF] → [1-32]: Value of monotonic counter is X, where X is the position of the most significant bit at 1
OTP21	-	BOOTROM_CONFIG_12	-
	[31:0]	oem_fsbl_monotonic_counter	[1-0xFFFF] → [33-64]: Value of monotonic counter is 32 + X, where X is the position of the most significant bit at 1
OTP22	-	BOOTROM_CONFIG_13	Reserved
OTP23	-	BOOTROM_CONFIG_14	Reserved

Table 18. OTP fuse description (lower OTP region) (continued)

OTP word	OTP bits	Name or description	Detailed description
OTP24	-	BOOTROM_TZ_EPOCH0	If the highest blown bit is the nth of these 256 bits, the boot ROM sets BSEC3_EPOCH_TZ = n
OTP25	-	BOOTROM_TZ_EPOCH1	
OTP26	-	BOOTROM_TZ_EPOCH2	
OTP27	-	BOOTROM_TZ_EPOCH3	
OTP28	-	BOOTROM_TZ_EPOCH4	
OTP29	-	BOOTROM_TZ_EPOCH5	
OTP30	-	BOOTROM_TZ_EPOCH6	
OTP31	-	BOOTROM_TZ_EPOCH7	
OTP32	-	BOOTROM_NS_EPOCH0	If the highest blown bit is the nth of these 256 bits, the boot ROM sets BSEC3_EPOCH_NS = n
OTP33	-	BOOTROM_NS_EPOCH1	
OTP34	-	BOOTROM_NS_EPOCH2	
OTP35	-	BOOTROM_NS_EPOCH3	
OTP36	-	BOOTROM_NS_EPOCH4	
OTP37	-	BOOTROM_NS_EPOCH5	
OTP38	-	BOOTROM_NS_EPOCH6	
OTP39	-	BOOTROM_NS_EPOCH7	
OTP40 to OTP55	-	Customer zone	Customer values

Table 18. OTP fuse description (lower OTP region) (continued)

OTP word	OTP bits	Name or description	Detailed description
OTP56	-	TAMP_EN	-
	[0]	tamp1_enable	0: disabled, 1: enabled
	[1]	tamp2_enable	0: disabled, 1: enabled
	[2]	tamp3_enable	0: disabled, 1: enabled
	[3]	tamp4_enable	0: disabled, 1: enabled
	[4]	tamp5_enable	0: disabled, 1: enabled
	[5]	tamp6_enable	0: disabled, 1: enabled
	[6]	tamp7_enable	0: disabled, 1: enabled
	[7]	tamp8_enable	0: disabled, 1: enabled
	[8]	itamp1_enable	0: disabled, 1: enabled
	[9]	itamp2_enable	0: disabled, 1: enabled
	[10]	itamp3_enable	0: disabled, 1: enabled
	[11]	itamp4_enable	0: disabled, 1: enabled
	[12]	itamp5_enable	0: disabled, 1: enabled
	[13]	itamp6_enable	0: disabled, 1: enabled
	[14]	itamp7_enable	0: disabled, 1: enabled
	[15]	itamp8_enable	0: disabled, 1: enabled
	[16]	itamp9_enable	0: disabled, 1: enabled
	[17]	Reserved	Reserved
	[18]	itamp11_enable	0: disabled, 1: enabled
	[31:19]	Reserved	Reserved

Table 18. OTP fuse description (lower OTP region) (continued)

OTP word	OTP bits	Name or description	Detailed description
OTP57	-	TAMP_CFM	-
	[0]	tamp1_confirmed	0: potential tamper, 1: confirmed tamper
	[1]	tamp2_confirmed	0: potential tamper, 1: confirmed tamper
	[2]	tamp3_confirmed	0: potential tamper, 1: confirmed tamper
	[3]	tamp4_confirmed	0: potential tamper, 1: confirmed tamper
	[4]	tamp5_confirmed	0: potential tamper, 1: confirmed tamper
	[5]	tamp6_confirmed	0: potential tamper, 1: confirmed tamper
	[6]	tamp7_confirmed	0: potential tamper, 1: confirmed tamper
	[7]	tamp8_confirmed	0: potential tamper, 1: confirmed tamper
	[8]	itamp1_confirmed	0: potential tamper, 1: confirmed tamper
	[9]	itamp2_confirmed	0: potential tamper, 1: confirmed tamper
	[10]	itamp3_confirmed	0: potential tamper, 1: confirmed tamper
	[11]	itamp4_confirmed	0: potential tamper, 1: confirmed tamper
	[12]	itamp5_confirmed	0: potential tamper, 1: confirmed tamper
	[13]	itamp6_confirmed	0: potential tamper, 1: confirmed tamper
	[14]	itamp7_confirmed	0: potential tamper, 1: confirmed tamper
	[15]	itamp8_confirmed	0: potential tamper, 1: confirmed tamper
	[16]	itamp9_confirmed	0: potential tamper, 1: confirmed tamper
	[17]	Reserved	Reserved
	[18]	itamp11_confirmed	0: potential tamper, 1: confirmed tamper
	[31:19]	Reserved	Reserved
OTP58	-	TAMP_CFG	-
	[1:0]	tamp1_cfg	00: level low, 01: level high, 10: edge falling, 11: edge rising
	[3:2]	tamp2_cfg	00: level low, 01: level high, 10: edge falling, 11: edge rising
	[5:4]	tamp3_cfg	00: level low, 01: level high, 10: edge falling, 11: edge rising
	[7:6]	tamp4_cfg	00: level low, 01: level high, 10: edge falling, 11: edge rising
	[9:8]	tamp5_cfg	00: level low, 01: level high, 10: edge falling, 11: edge rising
	[11:10]	tamp6_cfg	00: level low, 01: level high, 10: edge falling, 11: edge rising
	[13:12]	tamp7_cfg	00: level low, 01: level high, 10: edge falling, 11: edge rising
	[15:14]	tamp8_cfg	00: level low, 01: level high, 10: edge falling, 11: edge rising
	[31:16]	Reserved	Reserved
OTP59 to OTP95	-	Reserved	Reserved

Table 18. OTP fuse description (lower OTP region) (continued)

OTP word	OTP bits	Name or description	Detailed description
OTP96 to OTP99	-	Reserved	Reserved
OTP100 to OTP102	-	Reserved	Reserved
OTP103	-	Reserved	Reserved
OTP104 to OTP123	-	Reserved	Reserved
OTP124	-	HCONF1	-
	[0]	IWDG1_HW	IWDG1 start on reset
	[1]	IWDG1_FZ_STOP	IWDG1 freeze in Stop mode
	[2]	IWDG1_FZ_STANDBY	IWDG1 freeze in Standby mode
	[9:3]	Reserved	Reserved
	[10]	RST_STOP	Reset caused if the device is put in Stop mode
	[11]	RST_STDBY	Reset caused if the device is put in Standby mode
	[12]	SELINBORH	– 0: BOR disabled – 1: BOR = 2.7 V
	[13]	HSLV_VDDIO5	VDDIO5 I/O segment below 2.5 V for I/O mode. The I/O segment is used by SDMMC2 port.
	[14]	HSLV_VDDIO4	VDDIO4 I/O segment below 2.5 V for I/O mode (I/O segment used by SDMMC1 port)
	[15]	HSLV_VDDIO3	VDDIO3 I/O segment below 2.5 V for I/O mode (I/O segment used by XSPIM port 2)
	[16]	HSLV_VDDIO2	VDDIO2 I/O segment below 2.5 V for I/O mode (I/O segment used by XSPIM port 1)
	[17]	HSLV_VDD	Main I/O segment below 2.5 V for I/O mode
	[19:18]	Reserved	Reserved
	[20]	DFT_DISABLE	– 0: scan and bist available – 1: scan and bist only available on an OPEN part
	[31:21]	Reserved	Reserved
OTP125 to OTP127	-	Reserved	Reserved

Table 19. OTP fuse description (mid OTP region)

OTP words	Name or description	Detailed description
OTP128	STM32CERTIF0	STM32 device certificate (signature of the public key)
OTP129	STM32CERTIF1	
OTP130	STM32CERTIF2	
OTP131	STM32CERTIF3	
OTP132	STM32CERTIF4	
OTP133	STM32CERTIF5	
OTP134	STM32CERTIF6	
OTP135	STM32CERTIF7	
OTP136	STM32CERTIF8	
OTP137	STM32CERTIF9	
OTP138	STM32CERTIF10	
OTP139	STM32CERTIF11	
OTP140	STM32CERTIF12	
OTP141	STM32CERTIF13	
OTP142	STM32CERTIF14	
OTP143	STM32CERTIF15	
OTP144	STM32PUBKEY0	STM32 device public key
OTP145	STM32PUBKEY1	
OTP146	STM32PUBKEY2	
OTP147	STM32PUBKEY3	
OTP148	STM32PUBKEY4	
OTP149	STM32PUBKEY5	
OTP150	STM32PUBKEY6	
OTP151	STM32PUBKEY7	
OTP152	STM32PUBKEY8	
OTP153	STM32PUBKEY9	
OTP154	STM32PUBKEY10	
OTP155	STM32PUBKEY11	
OTP156	STM32PUBKEY12	
OTP157	STM32PUBKEY13	
OTP158	STM32PUBKEY14	
OTP159	STM32PUBKEY15	

Table 19. OTP fuse description (mid OTP region) (continued)

OTP words	Name or description	Detailed description
OTP160	OTP_ROT_HASH0	Hash of Table of hashes of OEM public keys
OTP161	OTP_ROT_HASH1	
OTP162	OTP_ROT_HASH2	
OTP163	OTP_ROT_HASH3	
OTP164	OTP_ROT_HASH4	
OTP165	OTP_ROT_HASH5	
OTP166	OTP_ROT_HASH6	
OTP167	OTP_ROT_HASH7	
OTP168	ST_RSSE_EDMK_DERIV_CSTE_FUSE	ST Encryption Decryption Master Key Derivation constant
OTP169	OTP_MAC1_ADDR_LOW	MAC_ADDR1
OTP170	OTP_MAC1_ADDR_HIGH	
OTP171	OTP_MAC2_ADDR_LOW	MAC_ADDR2
OTP172	OTP_MAC2_ADDR_HIGH	
OTP173 to OTP255	Available to customer	-
OTP256	OTP_RMA_LOCK_PSWD	RMA password
OTP257	OTP_RMA_LOCK_PSWD	
OTP258	OTP_RMA_LOCK_PSWD	
OTP259	OTP_RMA_LOCK_PSWD	
OTP260 to OTP363	OEM secrets available to customer	-
OTP364	OEM_SECRET_FOR_CRYPTED_BOOT0	OEM secret used to derive FSBL decryption key
OTP365	OEM_SECRET_FOR_CRYPTED_BOOT1	
OTP366	OEM_SECRET_FOR_CRYPTED_BOOT2	
OTP367	OEM_SECRET_FOR_CRYPTED_BOOT3	
OTP368	STM32PRVKEY0	STM32 device private key (ST)
OTP369	STM32PRVKEY1	
OTP370	STM32PRVKEY2	
OTP371	STM32PRVKEY3	
OTP372	STM32PRVKEY4	
OTP373	STM32PRVKEY5	
OTP374	STM32PRVKEY6	
OTP375	STM32PRVKEY7	

6 Resource isolation framework security controller (RIFSC)

6.1 Introduction

Resource isolation framework (RIF) is a set of hardware blocks designed to enforce and manage isolation of STM32 hardware resources like memory and peripherals. Some resources (such as GP/HPDMA) manage their own security configuration internally (they are configured locally). Such resources are called “RIF-aware”. Most resources are “non-RIF-aware”. The RIFSC centralizes the security configuration of such non-RIF-aware resources.

6.2 RIFSC main features

- **RISC** registers associated with **RISUP** logic (resource isolation slave unit for peripherals) sitting in front of the configuration port of non-RIF-aware peripherals, and with RCC peripheral clock control logic, assigning security and privilege levels to peripheral access and peripheral clock control.
- **RIMC** registers associated with **RIMU** logic (resource isolation master unit), assigning non-RIF-aware bus masters the secure, privilege and compartment (CID) information they drive onto the system bus when initiating AXI transactions.
- Secure guard, where an AXI bus master port is forced to be nonsecure when its associated configuration port is programmed nonsecure
- Support for a special debug domain, giving greater access for the debugger

6.3 RIFSC functional description

6.3.1 RIFSC reset and clocks

The RIFSC has a single reset. After a system reset, all non-RIF-aware peripherals become non-secure and unprivileged, reachable by any CID.

6.3.2 RISUP

RISUP blocks are instantiated in front of the AHB configuration port of non-RIF-aware peripherals to filter configuration accesses.

Each non-RIF-aware peripheral is assigned a unique “RISUP index” p < 128. Whenever the RISUP in front of peripheral p blocks an illegal access, it sends an “illegal access event” pulse to the IAC. The IAC records this event in the IAFp bit of the relevant IAC_ISRx register. Thus, the RISUP index is aligned with the IAC index.

For x < 4 and i < 32, the filtering is determined as follows.

- If RIFSC_RISC_SECCFGx[i] = 1, then nonsecure accesses to peripheral p = i + 32x are blocked. Also, nonsecure world attempts to program RCC registers to turn off the clock of (or reset) peripheral p are blocked.

- If RIFSC_RISC_PRIVCFGRx[i] = 1, then user accesses to peripheral p = i + 32x are blocked. Also, attempts by a user process to program RCC registers to turn off the clock of (or reset) peripheral p are blocked.

Whenever the RCC detects an illegal attempt to turn off a clock of (or reset) a peripheral, it sends an event pulse to the IAC (recorded in the IAF136 bit of the relevant IAC_ISRx register).

Note: There is no record of which peripheral was the object of this illegal attempt.

Table 20 identifies the RISUP index of each non-RIF-aware peripheral.

Table 20. RISUP indexes

RISUPs 31 to 0 (see RIFSC_RISC_PRIVCFGR0 / SECCFGR0 / RCFGLOCKR0)																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIM5	TIM4	TIM3	TIM2	TIM1	FDCAN1/2/3	LPUART1	USART10	UART9	UART8	UART7	USART6	UART5	UART4	USART3	USART2	USART1	I3C2	I3C1	I2C4	I2C3	I2C2	I2C1	SAI2	-	SAI1	SPI6/I2S6	SPI5	SPI4	SPI3/I2S3	SPI2/I2S2	SPI1/I2S1
:																															
RISUPs 63 to 32 (see RIFSC_RISC_PRIVCFGR1 / SECCFGR1 / RCFGLOCKR1)																															
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
-	SYSCFG	SPDIFRX	ETH1	-	UCPD1	OTG2_HS	OTG1_HS	MDIOS	SDMMC2	SDMMC1	MDF1	ADF1	LPTIM5	LPTIM4	LPTIM3	LPTIM2	LPTIM1	GFXTIM	TIM18	TIM17	TIM16	TIM15	TIM14	TIM13	TIM12	TIM11	TIM10	TIM9	TIM8	TIM7	TIM6
:																															
RISUPs 95 to 64 (see RIFSC_RISC_PRIVCFGR2 / SECCFGR2 / RCFGLOCKR2)																															
95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80	79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
-	DCMI	DCMIPP	CSI2HOST	-	FMC	XSPIM	XSPI3	XSPI2	XSPI1	-	MCE4	MCE3	MCE2	MCE1	CRYP1	HASH	SAES	PKA	RNG	-	-	-	-	-	-	WWDG	IWDG	CRC	-	VREFBUF	ADC12
:																															
RISUPs 127 to 96 (see RIFSC_RISC_PRIVCFGR3 / SECCFGR3 / RCFGLOCKR3)																															
127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	NPU	-	LTDC_L2	LTDC_L1	LTDC_CMN	DMA2D	GFXMMU	GPU	ICACHE	VENC	JPEG

The RISUP generates a bus error if an instruction fetch arrives at the peripheral.

6.3.3 RCC security settings for RIF-aware peripherals and RAMs

Table 21 shows the use made of the RIFSC to configure the security and privilege access rights for resetting or gating the clock of RIF-aware peripherals and RAMs (resources that

do not have RISUPs in front of them). Just as for all the non-RIF-aware peripherals with RISUPs mentioned above, the outputs of the registers below are also wired to the RCC.

Table 21. RISC indexes purely for RCC security control

RCC security controls 159 to 128 (RIFSC_RISC_PRIVCFGR4 / SECCFGR4 / RCFGLOCKR4)																															
159	158	157	156	155	154	153	152	151	150	149	148	147	146	145	144	143	142	141	140	139	138	137	136	135	134	133	132	131	130	129	128
GPIOA	-	BKPSRAM	AHBRAM2	AHBRAM1	-	NPU config	-	-	-	-	-	VENCRAM	CACHEAXI RAM	FLEXRAM	-	-	-	CPU RAM1	CPU RAM0	-	-	-	-	-	-	RTC	-	HPDMA1	GPDMA1	-	-
⋮																															
RCC security controls 191 to 160 (RIFSC_RISC_PRIVCFGR5 / SECCFGR5 / RCFGLOCKR5)																															
191	190	189	188	187	186	185	184	183	182	181	180	179	178	177	176	175	174	173	172	171	170	169	168	167	166	165	164	163	162	161	160
-	RAMCFG	-	-	HDP	-	-	XSPIPHY2	XSPIPHY1	XSPIPHYCOMP	-	NPU_RAM3	NPU_RAM2	NPU_RAM1	NPU_RAM0	-	MCO2	MCO1	DTS	-	GPIOQ	GPIOP	GPIOO	GPION	-	GPIOH	GPIOG	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB

Note: For the RIF-aware peripherals, it is the responsibility of the trusted domain software to make sure this security configuration of the control of the clock and reset of the peripheral is consistent with the internal security configuration of this peripheral.

6.3.4 RIMU

The security attributes (CID, security, privilege) of accesses made by a non-RIF-aware peripheral that is an AXI bus master, can be configured in RIMC_ATTRx registers, where x is the “RIMU index” of the peripheral. *Table 22* gives the index of each of these AXI bus master peripherals, and the index of the related RISUP protecting the configuration port of this peripheral. If nonsecure world software is permitted to configure the peripheral, then the RIMU_ATTRx[MSEC] setting is ignored, and all AXI accesses initiated by the peripheral are forced to be nonsecure too. This override mechanism is referred to as “secure guard”.

Table 22. RIMU resource assignment

Master index	Master name	Secure guard RISUP index
0	Trace (ETR)	No RISUP
1	NPU	106
2	SDMMC1	53
3	SDMMC2	54
4	OTG1	56
5	OTG2	57
6	ETH1	60

Table 22. RIMU resource assignment (continued)

Master index	Master name	Secure guard RISUP index
7	GPU	99
8	DMA2D	101
9	DCMIPP	93
10	LTDC_L1	103
11	LTDC_L2	104
12	VENC	97

RIMC registers must be initialized by the secure privileged software after the device reset. Optionally, this software can set the GLOCK bit in RIMC_CR, preventing further writes to all RIMC registers. This bit can only be cleared by a power-on reset.

The RISC_RIMC_CR register contains a DAPCID field that gives the CID value used by debugger (DAP) accesses onto the AXI bus. The reset value of this register is 0x7.
All RISAFs treat a 0x7 as a legal CID value, regardless of their configuration. The secure privileged master can reprogram the DAPCID to allow the DAP to mimic another compartment (for example, to debug the security configuration).

6.4 RIFSC registers

These registers are accessible through the register interface of the RIFSC peripheral.

6.4.1 RIFSC RISC slave configuration register x (RIFSC_RISC_CR)

Address offset: 0x000

Reset value: 0x0000 0000

Secure privileged write access only. Any read access is allowed on this register. Writes are ignored if GLOCK is set in this register.

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

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

Bit 0 **GLOCK**: Global lock

This bit locks the configuration of RIFSC RISC registers until next reset. This bit is cleared by default and, once set, it cannot be reset until global RIFSC reset.

0: RIFSC RISC registers are writable.

1: All writes to RIFSC RISC registers are ignored.

6.4.2 RIFSC RISC slave security configuration register x (RIFSC_RISC_SECCFGRx)

Address offset: 0x010 + 0x4 * x, (x = 0 to 5)

Reset value: 0x0000 0000

Nonsecure or unprivileged writes to this register are ignored, while any read is allowed. Writes are ignored if GLOCK is set in RIFSC_RISC_CR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEC{i + 32 * x}															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC{i + 32 * x}															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SEC{i + 32 * x}**: security configuration for peripheral {i + 32 * x} (i = 0 to 31)
0: Secure and nonsecure data access are granted to the peripheral {i + 32 * x}.
1: Secure data access only are granted to the peripheral {i + 32 * x}.

6.4.3 RIFSC RISFC slave privileged register x (RIFSC_RISC_PRIVCFGRx)

Address offset: 0x030 + 0x4 * x, (x = 0 to 5)

Reset value: 0x0000 0000

Unprivileged writes to this register are ignored, while any read is allowed. Writes are ignored if GLOCK is set in RIFSC_RISC_CR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRIV{i + 32 * x}															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIV{i + 32 * x}															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PRIV{i + 32 * x}**: privileged-only access permission for peripheral {i + 32 * x} (i = 0 to 31)
0: Privileged and unprivileged data access are granted to the peripheral {i + 32 * x}.
1: Privileged data access only are granted to the peripheral {i + 32 * x}.

*Note: If corresponding SEC{i + 32 * x} bit is set in RIFSC_RISC_SECCFGRx, this bit can only be written by a secure privileged application.*

6.4.4 RIFSC RISC slave resource configuration lock register x (RIFSC_RISC_RCFGLOCKRx)

Address offset: 0x050 + 0x4 * x, (x = 0 to 5)

Reset value: 0x0000 0000

Secure privileged write access only. Any read access is allowed on this register. Writes are ignored if GLOCK is set in this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RLOCK{i + 32 * x}															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RLOCK{i + 32 * x}															
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:0 **RLOCK{i + 32 * x}**: resource lock for peripheral {i + 32 * x} (i = 0 to 31)
This bit is set to lock the peripheral resource {i + 32 * x}. It is cleared by default and, once set, this bit cannot be cleared until the RIFSC RISC peripheral is reset.
0: SEC{i + 32 * x} in RIFSC_RISC_SECCFGRx and PRIV{i + 32 * x} in RIFSC_RISC_PRIVCFGRx are writable.
1: Writes to SEC{i + 32 * x} and PRIV{i + 32 * x} are ignored.

6.4.5 RIFSC RIMC master configuration register (RIFSC_RIMC_CR)

Address offset: 0xC00

Reset value: 0x0000 0710

Secure privileged write access only. Any read access is allowed on this register. Writes are ignored if GLOCK is set in RIFSC_RISC_CR.

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

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:8 **DAPCID[2:0]**: debug access port compartment ID
This bitfield defines the CID of the DAP.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **GLOCK**: global lock
This bit is used to lock the configuration of RIFSC RIMC registers until next reset. This bit is cleared by default and, once set, it cannot be reset until global RIFSC reset.
0: RIFSC RIMC registers are writable.
1: All writes to RIFSC RIMC registers are ignored.

6.4.6 RIFSC RIMC master attribute register x (RIFSC_RIMC_ATTRx)

Address offset: 0xC10 + x * 0x4 (x = 0 to 12)

Reset value: 0x0000 0000

Secure privileged write access only. Any read access is allowed on this register.

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

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

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

Bit 9 **MPRIV**: master privileged
Value of the privileged flag on the interconnect for this master.
0: This master is unprivileged.
1: This master is privileged.

Bit 8 **MSEC**: master secure
Value of the secure flag on the interconnect for this master.
0: This master is nonsecure.
1: This master is secure.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **MCID[2:0]**: master CID
Value of the CID flag on the interconnect for this master. This bitfield cannot be written with a value of 0x7 (write to MCID[2:0] is ignored).

Bits 3:0 Reserved, must be kept at reset value.

6.4.7 RIFSC peripheral protection status register 0 (RIFSC_PPSR0)

Address offset: 0xFB0

Reset value: 0xFFFF FF7F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PPEN{i}															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PPEN{i}															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PPEN{i}**: peripheral protection enable {i} (i = 0 to 31)
0: SEC{i}, PRIV{i}, and RLOCK{i} bits are not present.
1: SEC{i}, PRIV{i}, and RLOCK{i} bits are present.

6.4.8 RIFSC peripheral protection status register 1 (RIFSC_PPSR1)

Address offset: 0xFB4

Reset value: 0x77FF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PPEN{i + 32}															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PPEN{i + 32}															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PPEN{i + 32}**: peripheral protection enable {i + 32} (i = 0 to 31)
0: SEC{i + 32}, PRIV{i + 32}, and RLOCK{i + 32} bits are not present.
1: SEC{i + 32}, PRIV{i + 32}, and RLOCK{i + 32} bits are present.

6.4.9 RIFSC peripheral protection status register 2 (RIFSC_PPSR2)

Address offset: 0xFB8

Reset value: 0xF7DF F03B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PPEN{i + 64}															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PPEN{i + 64}															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PPEN{i + 64}**: peripheral protection enable {i + 64} (i = 0 to 31)
0: SEC{i + 64}, PRIV{i + 64}, and RLOCK{i + 64} bits are not present.
1: SEC{i + 64}, PRIV{i + 64}, and RLOCK{i + 64} bits are present.

6.4.10 RIFSC peripheral protection status register 3 (RIFSC_PPSR3)

Address offset: 0xFBC

Reset value: 0x0000 05FF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PPEN{i + 96}															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PPEN{i + 96}															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PPEN{i + 96}**: peripheral protection enable {i + 96} (i = 0 to 31)
0: SEC{i + 96}, PRIV{i + 96}, and RLOCK{i + 96} bits are not present.
1: SEC{i + 96}, PRIV{i + 96}, and RLOCK{i + 96} bits are present.

6.4.11 RIFSC peripheral protection status register 4 (RIFSC_PPSR4)

Address offset: 0xFC0

Reset value: 0xBBEF FFEF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PPEN{i + 128}															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PPEN{i + 128}															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PPEN{i + 128}**: peripheral protection enable {i + 128} (i = 0 to 31)
0: SEC{i + 128}, PRIV{i + 128}, and RLOCK{i + 128} bits are not present.
1: SEC{i + 128}, PRIV{i + 128}, and RLOCK{i + 128} bits are present.

6.4.12 RIFSC peripheral protection status register 5 (RIFSC_PPSR5)

Address offset: 0xFC4

Reset value: 0x7DDE EF7F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PPEN{i + 160}															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PPEN{i + 160}															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PPEN{i + 160}**: peripheral protection enable {i + 160} (i = 0 to 31)
0: SEC{i + 160}, PRIV{i + 160}, and RLOCK{i + 160} bits are not present.
1: SEC{i + 160}, PRIV{i + 160}, and RLOCK{i + 160} bits are present.

6.4.13 RIFSC register map

Table 23. RIFSC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	RIFSC_RISC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GLOCK
	Reset value																																0
0x010 + 0x4 * x (x = 0 to 5) Last address: 0x024	RIFSC_RISC_SECCFGRx	SEC{i + 32 * x}																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028-0x02C	Reserved	Reserved																															
0x030 + 0x4 * x (x = 0 to 5) Last address: 0x044	RIFSC_RISC_PRIVCFGRx	PRIV{i + 32 * x}																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048-0x04C	Reserved	Reserved																															
0x050 + 0x4 * x (x = 0 to 5) Last address: 0x064	RIFSC_RISC_RCFGLOCKRx	RLOCK{i + 32 * x}																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x68-0xBFC	Reserved	Reserved																															
0xC00	RIFSC_RIMC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DAPCID [2:0]				Res.	Res.	Res.	Res.	Res.	Res.	GLOCK
	Reset value																						1	1	1								0
0xC04-0xC0C	Reserved	Reserved																															
0xC10 + 0x4 * x (x = 0 to 12) Last address: 0xC40	RIFSC_RIMC_ATTRx	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MPRIV	MSEC	Res.	MCID[2:0]			Res.	Res.	Res.	Res.
	Reset value																							0	0		0	0	0				
0xC44 to 0xFAC	Reserved	Reserved																															
0xFB0	RIFSC_PPSR0	PPEN{i}																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
0xFB4	RIFSC_PPSR1	PPEN{i + 32}																															
	Reset value	0	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0xFB8	RIFSC_PPSR2	PPEN{i + 64}																															
	Reset value	1	1	1	1	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	1	1	1	0	1	1
0xFBC	RIFSC_PPSR3	PPEN{i + 96}																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	1	1	1	1	1	1
0xFC0	RIFSC_PPSR4	PPEN{i + 128}																															
	Reset value	1	0	1	1	1	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1
0xFC4	RIFSC_PPSR5	PPEN{i + 160}																															
	Reset value	0	1	1	1	1	1	0	1	1	1	0	1	1	1	1	0	1	1	1	0	1	1	1	1	0	1	1	1	1	1	1	1

Refer to *Section 2.3* for the register boundary addresses.

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

7.1 RISAF introduction

The RIF (resource isolation framework) is a comprehensive set of hardware blocks designed to enforce and manage isolation of STM32 hardware resources, like memory and peripherals.

Through RISAF registers, a secure privileged software application or the application to whom the configuration has been delegated, assigns memory regions and subregions to one or more security domains (secure, privileged, compartment).

7.2 RISAF main features

- Assign variable sized address space regions and subregions to security domains.
- Make memory regions:
  - Restricted to one or more than one hardware compartments
  - Secure-only or nonsecure-only
  - Privileged-only or privileged/unprivileged
  - Read-only, write-only, or read/write
- The secure privileged software can delegate configuration and positioning of a subregion (within a region) to another security domain.
- The secure privileged software can delegate configuration and positioning of a subregion (within a nonsecure region) to a nonsecure privileged software.

7.3 RISAF implementation

The following table describes the characteristics of the available RISAFs, including watermark programming details (such as granularity and protected address space).

Table 24. RISAF resource assignment

Protected memory	Resource name	Target bus	IAC index	Number of configurable base regions	Granularity	Address space size and alignment	CID management
TCM	RISAF1	AXI	139	7	4 Kbytes	1 Gbyte	Programmable CID filtering on system bus (CID=0 to 7). To delegate to nonsecure set DCCID = 1.
AXISRAM1	RISAF2		140			1 Mbyte	
AXISRAM2	RISAF3		141				
NPU master 0	RISAF4		142	11		4 Gbytes	
NPU master 1	RISAF5		143				
CPU master	RISAF6		144				
FLEXRAM	RISAF7	AXI	145	7	4 Kbytes	400 Kbytes	
CACHEAXI/ AXISRAM7	RISAF8		146			256 Kbytes	
VENCRAM/ AXISRAM8	RISAF9		147			128 Kbytes	
XSPI1	RISAF11		149			256 Mbytes	
XSPI2	RISAF12		150				
XSPI3	RISAF13		151				
FMC	RISAF14		152				
CACHEAXI configuration port	RISAF15	AHB	153	2	4 bytes	4 Kbytes	Fixed CID0 filtering on system bus. To delegate to nonsecure set DCCID = 0.
AHB RAM1	RISAF21		154	7	512 bytes	16 Kbytes	
AHB RAM2	RISAF22		155				
Backup RAM	RISAF23		156	3		4 Kbytes	

RISAF4/5/6 are used to partition the 2-Mbyte NPU RAM. This memory is implemented as four RAM cuts, each with its own AXI target port, but using an interleaved (re-mapped) addressing scheme. To retain the software normal view of memory addressing, the RISAFs protecting these RAMs are placed on the three entry ports of the NPU interconnect. The user must program an identical partitioning into each of the three RISAFs. These RISAFs are furnished with two extra regions that the secure privileged software must configure to be transparent (permissive) to all accesses either side of the NPU RAMs.

As summarized in the CID management column of *Table 24*, the AHB bus does not carry CID information (it is fixed at 0). The RISAFs on the AHB bus only filter on security and privilege, in read or in write (using RDENC0 or WRENC0 bits, respectively). All accesses are effectively assumed to be owned by the CPU. When a secure privileged application delegates to the nonsecure world the access control configuration of subregions in a nonsecure base region, DCEN must be set, and DCCID and SRCID must stay at 0 for this specific base region.

As summarized in *Table 24* the AXI bus carries CID information. All AHB configuration accesses to AXI RISAF registers are assumed to come from CID1. When a secure privileged application delegates to the nonsecure world the access control configuration of subregions in a nonsecure base region, DCEN must be set and DCCID must be 0x1.

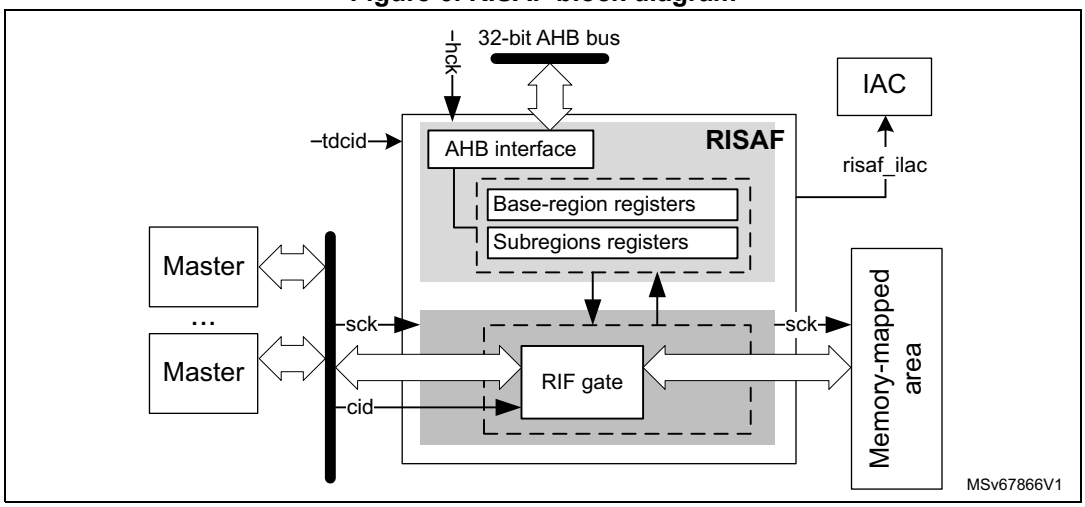
For STM32N6x5xx devices the configuration port of the CACHEAXI is not functional and must not be addressed. For STM32N6x7xx devices the AHB configuration port of the CACHEAXI is protected by a small RISAF that offers two base regions. These must be configured to cover the register address spaces, each side of the registers controlling cache invalidation. The cache invalidation operation can thus be reserved for use only by the secure privileged software, because the controlling registers are left in the default region 0.

7.4 RISAF functional description

7.4.1 RISAF block diagram

The following figure shows the block diagram of the RISAF associated with the IAC, and optionally, a memory controller.

Figure 6. RISAF block diagram



1. The tdcid input value is equal to CID = 1.

7.4.2 RISAF internal signals

Table 25 describes internal signals available at RISAF level, not at product level (on pins).

Table 25. RISAF internal input/output signals

Name	Type	Description
risaf_sck	Digital input	System bus clock, also used by the protected memory controller
risaf_hclk		AHB bus clock
risaf_ilac	Digital output	Illegal access signal to IAC

7.4.3 RISAF reset and clocks

The RISAF peripheral has two clocks, one for its AHB register interface, and one for its system bus interfaces. The system bus clock is shared with the system bus memory controller target.

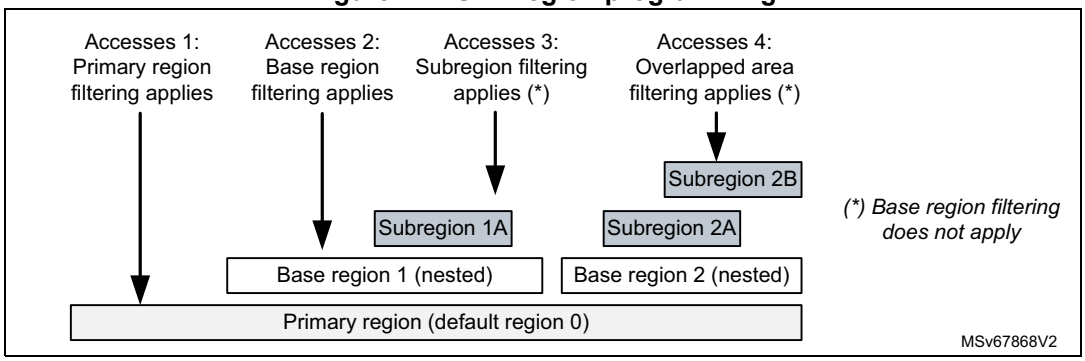
The RISAF peripheral has a single reset input.

Address spaces protected by the RISAF are by default secure, privileged, and CID = 1 only.

7.4.4 RISAF address space management

Through RISAF registers, a secure privileged software application, or the application to whom the configuration has been delegated, assigns memory regions to one or more security domains (secure, privilege, compartment ID). *Figure 7* illustrates the principle of this allocation.

Figure 7. RISAF region programming



As shown in *Figure 7*, there are multiple types of access filtered by RISAF:

- Accesses 1 - Default region: access to an area not belonging to any enabled base region. Filtering for such access is fixed to secure, privileged and CID = 1 only.
- Accesses 2 - Application region > 0: access to an area in an enabled base region, not belonging to any enabled subregion. Filtering for such access is defined in the base region, where one or more compartments can be granted specific accesses.
- Accesses 3 - Non overlapped subregion: access to an area in an enabled nested subregion. Filtering for such access is defined in the subregion, with only one compartment that can be granted access.
- Accesses 4 - Overlapped subregion: accesses to an area overlapping two enabled nested subregions. In this case, filtering rules for these subregions are used by the RISAF as follows:
  - If both subregions are secure (resp. nonsecure), the overlap is secure (resp. nonsecure).
  - If a subregion is nonsecure (resp. unprivileged), the overlap is nonsecure (resp. unprivileged).
  - If a subregion grants access to a compartment, the access can go through. Up to two compartments can be granted access to an overlapped subregion.

Note: When a base or subregion is disabled, the associated filtering is disabled as well. For access types 3 or 4 (detailed above), base region filtering rules do not apply. When two enabled regions overlap, if one region grants access, the access goes through.

7.4.5 RISAF programming

Only the secure privileged software can set up the access configuration to base region x through RISAF_REGx_CFGR, RISAF_REGx_CIDCFGR, RISAF_REGx_STARTR, and RISAF_REGx_ENDR.

The secure privileged software, or the application to which the subregion z configuration has been delegated in RISAF_REGx_zNESTR, can set up the access configuration to subregion z of base region x through RISAF_REGx_zCFGR, RISAF_REGx_zSTARTR, and RISAF_REGx_zENDR (z = A or B).

As shown in *Table 26*, subregion right options depend upon the base region privileged and security rights.

Table 26. RISAF subregion security setup matrix

Base region rights	Nested subregion rights			
	Secure, privileged	**Secure, unprivileged**	**Nonsecure, privileged**	**Nonsecure, unprivileged**
Secure, privileged	OK(1)	OK	OK	OK
Secure, unprivileged	Not allowed	OK(1)	Not allowed	OK
Nonsecure, privileged	Not allowed	Not allowed	OK(1)	OK
Nonsecure, unprivileged	Not allowed	Not allowed	Not allowed	OK(1)

1. Default configuration

Configuring base regions in RISAF

The number of regions, the region watermark granularity, and the address space size depend upon the RISAF instance (see *Section 7.3*).

If the secure privileged software must use a base region x (for example at boot time), the following steps are recommended.

1. The secure privileged software sets start and end addresses for each of the required base region x, using RISAF_REGx_STARTR and RISAF_REGx_ENDR:
   - These registers are initialized with 32-bit physical addresses when the protected address space size is 4 Gbytes. For different sizes, refer to the register description.
   - Base regions must be defined within their primary region. The hardware enforces this rule if the application does not follow it.
2. For each of the above region x, the secure privileged software programs RISAF_REGx_CIDCFGR, granting read and/or write access to any compartment.
3. For each of the above region x, the secure privileged software finalizes the programming by configuring RISAF_REGx_CFGR, setting up security parameters.
4. After a final check on the above configuration, the secure privileged software enables each required base region x, setting the BREN bit in RISAF_REGx_CFGR.

At any time the secure privileged software can lock the RISAF configuration until the next reset, by setting the GLOCK bit in RISAF_CR. Subregion registers are the exception to this rule, as explained in *Configuring subregions in RISAF*.

Configuring subregions in RISAF

Note: The number of subregions, the subregion watermark granularity, and the primary region address space size depend on the RISAF resource described in Section 7.3.

Important notes on access control programming and delegation are in Section 7.3.

When an application needs to use the subregion z defined in the base region x, there are two configuration options in RISAF_REGx_zNESTR:

- The secure privileged software programs the RIF configuration for this subregion. In this case DCEN is cleared.
- The secure privileged software delegates the RIF configuration to a single CID, defined with DCCID. In this case DCEN must also be set.

Configuring a subregion z consists in the following steps.

1. Define the start and end address of the subregion using RISAF_REGx_zSTARTR and RISAF_REGx_zENDR registers.
   - When programming subregion watermarks, use similar rules as for base regions (see *Configuring base regions in RISAF*).
   - Subregions must be defined within their base region. The RISAF enforces this rule if the application does not follow it.
2. Modify WREN and RDEN in RISAF_REGx_zCFGR to modify the access right of allowed compartment (defined with SCID bits).
3. In the same registers, modify the security and privileged access rights for the allowed compartment, if needed (see *Table 26* for applicable restrictions).
4. After a final check on the above configuration, the subregion must be enabled by setting SREN bit in RISAF_REGx_zCFGR.

After a secure privileged software has locked all base regions (using GLOCK bit), subregion owners can lock their configuration using the RLOCK bit in RISAF_REGx_zCFGR.

Note: Only a secure application can change SEC in RISAF_REGx_zCFGR. If the base region is secure, only a secure application can configure subregion registers.

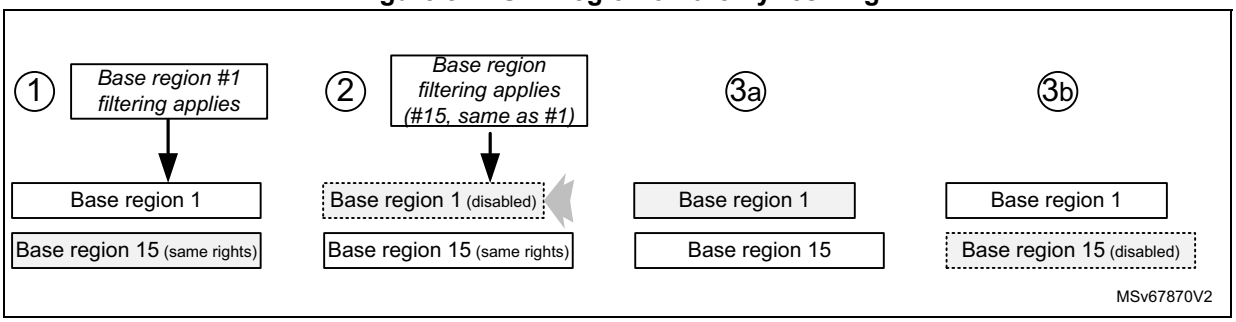
7.4.6 RISAF runtime modification of region or subregion configuration

The RISAF supports the on-the-fly update of the base region and subregion programming, except for the start- and end-address boundaries:

- To change RISAF_REGx_STARTR and RISAF_REGx_ENDR, the base region must be disabled (BREN = 0). In this case, the default primary region access rules apply immediately (secure, privileged, CID = 1 only).
- To change RISAF_REGx_zSTARTR and RISAF_REGx_zENDR (z = A or B), the subregion must be disabled (SREN = 0). In this case, the base region access rules apply immediately.

If the first solution introduces too many constraints, a workaround is shown in *Figure 8*.

Figure 8. RISAF region on-the-fly resizing



The recommended on-the-fly resizing of base region 1 becomes:

1. Define a spare base region (for example 15), exactly as current region 1.
2. Disable region 1 by clearing BREN.The region 15 access control (identical to region 1) immediately applies. No effect is experienced at system level. The region 1 can be safely resized.
3. Enable region 1, then disable region 15. The new region 1 access control immediately applies. Optionally, subregions can be added on top.

7.4.7 Managing illegal accesses

As a RIF component, the RISAF does not use the standard interrupt lines toward NVIC or GIC concentrators. The IAC captures all RIF interrupt events, called illegal access events.

Refer to *Section 8: Illegal access controller (IAC)* for more information on how illegal accesses are managed.

7.5 RISAF registers

The registers are accessible through the interface of the RISAF. Only word (32-bit) accesses are supported.

7.5.1 RISAF configuration register (RISAF_CR)

Address offset: 0x000

Reset value: 0x0000 0000

Secure, privileged, trusted domain CID = 1, write access only. Any read access is allowed, writes are ignored if GLOCK is set.

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

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

Bit 0 **GLOCK**: Global lock

This bit locks the. configuration of RISAF registers until next reset. It is cleared by default, and, once set, cannot be reset until global RISAF reset.

0: RISAF registers are writable.

1: All writes to RISAF registers are ignored, except RISAF_IACR, RISAF_REGx_zCFGR, RISAF_REGx_zSTARTR, and RISAF_REGx_zENDR (z = A to B).

7.5.2 RISAF illegal access status register (RISAF_IASR)

Address offset: 0x008

Reset value: 0x0000 0000

Secure, privileged, trusted domain CID = 1, read access only.

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

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

Bit 1 **IAEF**: Illegal access error flag

This bit is set when an illegal access is detected on the system bus. More details on the error can be found in RISAF_IAESR and RISAF_IADDR.

This bit is cleared by setting corresponding bit in RISAF_IACR.

Bit 0 **CAEF**: Configuration access error flag

This bit is set when an illegal access to any RISAF configuration register is detected. It is cleared by setting the corresponding bit in RISAF_IACR.

7.5.3 RISAF illegal access clear register (RISAF_IACR)

Address offset: 0x00C

Reset value: 0x0000 0000

Secure, privileged, trusted domain CID = 1, write access only.

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

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

Bit 1 **IAEF**: Illegal access error flag

Set this bit to clear IAEF in RISAF_IASR, allowing the capture of a new error information from RIF in RISAF_IAESR and RISAF_IADDR registers.

Note: Clearing this bit does not clear RISAF_IADDR.

Bit 0 **CAEF**: Configuration access error flag

Set this bit to clear CAEF in RISAF_IASR.

7.5.4 RISAF illegal access error status register (RISAF_IAESR)

Address offset: 0x020

Reset value: 0x0000 0000

Secure, privileged, trusted domain CID = 1, read access only.

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

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **IANRW**: Illegal access read/write

When IAEF = 1 in RISAF_IASR, IANRW captures the access type of the illegal access that has been detected.

0: Illegal access was a data read or an instruction fetch

1: Illegal access was a data write

Bit 6 Reserved, must be kept at reset value.

Bit 5 **IASEC**: Illegal access security
When IAEF = 1 in RISAF_IASR, IASEC captures the security type of the illegal access that has been detected.
0: Illegal access was nonsecure
1: Illegal access was secure

Bit 4 **IAPRIV**: Illegal access privileged
When IAEF = 1 in RISAF_IASR, IAPRIV captures the privilege state of the master that issued the illegal access detected.
0: Illegal access was unprivileged
1: Illegal access was privileged

Bit 3 Reserved, must be kept at reset value.

Bits **2**:0 **IACID[2:0]**: Illegal access compartment ID
When IAEF = 1 in RISAF_IASR, IACID[2:0] captures the compartment ID of the master that issued the illegal access detected.

7.5.5 RISAF illegal address register (RISAF_IADDR)

Address offset: 0x024

Reset value: 0x0000 0000

Secure, privileged, trusted domain CID = 1, read access only.

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

Bits 31:0 **IADD[31:0]**: Illegal address
When IAEF = 1 in RISAF_IASR, IADD[31:0] captures the address of the erroneous access. Additional information can be found in RISAF_IAESR. Captured address information is the byte offset from the base of the protected address space.

7.5.6 RISAF region x configuration register (RISAF_REGx_CFGR)

Address offset: 0x040 + 0x40 * (x - 1) (x = 1 to 15)

Reset value: 0x0000 0000

Secure, privileged, trusted domain CID = 1, write access only. Any read access is allowed on this register. Writes are ignored if GLOCK is set in RISAF_CR.

Refer to *Section 7.3* to verify the number of base regions supported by this RISAF instance.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIVC7	PRIVC6	PRIVC5	PRIVC4	PRIVC3	PRIVC2	PRIVC1	PRIVC0
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BREN
							rw								rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **PRIVCy**: Privileged access for compartment y (y = 7 to 0)

This bit is taken into account only if BREN = 1.

0: Application running in compartment y can access to region x in privileged and unprivileged mode.

1: Application running in compartment y can access to region x in privileged mode only.

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **SEC**: Secure region

This bit is taken into account only if BREN = 1.

0: Only nonsecure requests can access region x. SEC value in RISAF_REGx_zCFGR is internally treated as 0-ed by RISAF (only nonsecure requests to subregions).

1: Only secure requests can access region x, and optional subregions can be defined as secure or nonsecure.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **BREN**: Base region enable

0: Base region x is disabled. Access control of primary region 0 (secure, privileged, trusted domain CID = 1) applies to any access between region x start and end addresses. Associated subregions are also disabled.

1: Base region x is enabled. Access controls defined in region x apply to any access between region x start and end addresses.

Note: The filtering starts from start address RISAF_REGx_STARTR (included) and ends at end address RISAF_REGx_ENDR (included).

7.5.7 RISAF region x start-address register (RISAF_REGx_STARTR)

Address offset: 0x044 + 0x40 * (x - 1) (x = 1 to 15)

Reset value: 0x0000 0000

Secure, privileged, trusted domain CID = 1, write access only. Any read access is allowed on this register. Writes are ignored if GLOCK is set in RISAF_CR, or if BREN is set in RISAF_REGx_CFGR.

Refer to *Section 7.3* to verify the number of base regions supported by this RISAF instance, and for details on protected address space size and watermark granularity.

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

Bits 31:0 **BADDSTART[31:0]**: Base region address start

This bitfield defines the byte-address offset (relative to the base address of the protected address space) of the lowest byte-address in base region x.

Address bits less than the granularity and greater than the protected address space size cannot be written to, and remain 0.

7.5.8 RISAF region x end-address register (RISAF_REGx_ENDR)

Address offset: 0x048 + 0x40 * (x - 1) (x = 1 to 15)

Reset value: 0x0000 0FFF

The reset value given here applies when the granularity is 4 Kbytes. If the granularity is 2^n, the reset value is 2^n - 1.

Secure, privileged, trusted domain CID = 1, write access only. Any read access is allowed on this register. Writes are ignored if GLOCK is set in RISAF_CR, or if BREN is set in RISAF_REGx_CFGR.

Refer to *Section 7.3* to verify the number of base regions supported by this RISAF instance, and for details on protected address space size and watermark granularity.

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

Bits 31:0 **BADDEND[31:0]**: Base region address end

This bitfield defines the byte-address offset (relative to the base address of the protected address space) of the highest byte-address in base region x.

Address bits less than the granularity cannot be modified, and remain 1.

Address bits greater than the protected address space size cannot be written, and remain 0.

7.5.9 RISAF region x CID configuration register (RISAF_REGx_CIDCFGR)

Address offset: 0x04C + 0x40 * (x - 1) (x = 1 to 15)

Reset value: 0x0000 0000

Secure, privileged, trusted domain CID = 1, write access only. Any read access is allowed on this register. Writes are ignored if GLOCK is set in RISAF_CR.

Refer to *Section 7.3* to verify the number of base regions supported by this RISAF instance.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRENC7	WRENC6	WRENC5	WRENC4	WRENC3	WRENC2	WRENC1	WRENC0
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDENC7	RDENC6	RDENC5	RDENC4	RDENC3	RDENC2	RDENC1	RDENC0
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **WRENCy**: Write enable for compartment y (y = 7 to 0)

This bit is taken into account only if BREN = 1 in RISAF_REGx_CFGR.

0: Application running in compartment y cannot write to region x (write ignored).

1: Application running in compartment y can write to region x. Additional control defined by SEC and PRIVCy in RISAF_REGx_CFGR, also applies.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **RDENCy**: Read enable for compartment y (y = 7 to 0)

This bit is taken into account only if BREN = 1 in RISAF_REGx_CFGR.

0: Application running in compartment y cannot read region x (read as 0).

1: Application running in compartment y can read region x. Additional control defined by SEC and PRIVCy in RISAF_REGx_CFGR, also applies.

7.5.10 RISAF region x subregion A configuration register (RISAF_REGx_ACFGR)

Address offset: 0x050 + 0x40 * (x - 1) (x = 1 to 15)

Reset value: 0x0000 0000

Unprivileged writes are ignored. Nonsecure writes are ignored if SEC = 1 in RISAF_REGx_CFGR, or if DCEN = 0 in RISAF_REGx_ANESTR. Any read access is allowed to this register. Writes are ignored if RLOCK = 1 in this register.

When the delegated configuration is enabled (DCEN = 1 in RISAF_REGx_ANESTR), only the compartment DCCID defined in RISAF_REGx_ANESTR can write to this register.

When the delegation is disabled (DCEN = 0), only the secure privileged software can program this register.

Refer to *Section 7.3* to verify the number of base regions and subregions supported by this RISAF instance.

Caution: Check *Section 7.3* before programming SRCID[2:0] bitfield.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	WREN	RDEN	Res.	Res.	PRIV	SEC	Res.	SRCID[2]	SRCID[1]	SRCID[0]	Res.	Res.	RLOCK	SREN
		rw	rw			rw	rw		rw	rw	rw			rs	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **WREN**: Write enable

This bit is taken into account only if SREN = 1 in this register.

0: Any write to subregion A of region x is ignored

1: Subregion A of region x can be written

Bit 12 **RDEN**: Read enable

This bit is taken into account only if SREN = 1 in this register.

0: Any read to subregion A of region x returns 0

1: Subregion A of region x can be read

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **PRIV**: Privileged subregion

This bit is taken into account only if SREN = 1 in this register.

0: Privileged and unprivileged accesses are granted in subregion A of region x

1: Only privileged accesses are granted in subregion A of region x

Note: When base region x is defined as unprivileged for the SCID compartment (PRIVCy = 0 in RISAF_REGx_CFGR, with SRCID = Cy), unprivileged accesses to subregion A are always granted.

Bit 8 **SEC**: Secure subregion

This bit is taken into account only if SREN = 1 in this register. Nonsecure modification to it are discarded.

0: Only nonsecure requests can access subregion A of region x

1: Only secure requests can access subregion A of region x

Note: When base region x is defined as nonsecure (SEC = 0 in RISAF_REGx_CFGR), only nonsecure accesses to subregion A are granted.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **SRCID[2:0]**: Subregion CID

When SREN = 1 in this register, SRCID[2:0] defines which compartment has access to subregion A of RISAF region x.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **RLOCK**: Resource lock

When this bit is set, the subregion resource is locked.

0: RISAF_REGx_ACFGR, RISAF_REGx_ASTARTR, and RISAF_REGx_AENDR are writable

1: Writes to RISAF_REGx_ACFGR, RISAF_REGx_ASTARTR, and RISAF_REGx_AENDR are ignored

Note: This bit is set once, cleared only by RISAF reset. It cannot be set if GLOCK = 0 in RISAF_CR.

Bit 0 **SREN**: Subregion enable

0: Subregion A is disabled. Access control of base region x applies to any access between this subregion start and end addresses.

1: Subregion A of region x is enabled. Access control defined in subregion A applies to any access between subregion A start and end addresses.

Note: The filtering starts from address RISAF_REGx_ASTARTR (included), and ends at address RISAF_REGx_AENDR (included).

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

Address offset: 0x054 + 0x40 * (x - 1) (x = 1 to 15)

Reset value: 0x0000 0000

Unprivileged writes are ignored. Nonsecure writes are ignored if SEC = 1 in RISAF_REGx_CFGR, or if DCEN = 0 in RISAF_REGx_ANESTR. Any read access is allowed to this register. Writes are ignored if RLOCK = 1 or SREN = 1 in RISAF_REGx_ACFGR.

When the delegated configuration is enabled (DCEN = 1 in RISAF_REGx_ANESTR), only the compartment DCCID defined in RISAF_REGx_ANESTR can write to this register. When the delegation is disabled (DCEN = 0), only the secure privileged software can program this register.

Refer to *Section 7.3* to verify the number of base regions and subregions supported by this RISAF instance, and for details on protected address space size and watermark granularity.

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

Bits 31:0 **SADDSTART[31:0]**: Subregion address start

This bitfield defines the byte-address offset (relative to the base address of the protected address space) of the lowest byte-address in subregion A in base region x.

Address bits less than the granularity and greater than the protected address space size cannot be written, and remain 0.

The effective subregion is the intersection of the subregion and its base region (there is no point in programming SADDSTART below the associated BADDSTART).

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

Address offset: 0x058 + 0x40 * (x - 1) (x = 1 to 15)

Reset value: 0x0000 0FFF

This reset value applies when the granularity is 4 Kbytes. More in general, if the granularity is 2^n, the reset value is 2^n - 1.

Unprivileged writes are ignored. Nonsecure writes are ignored if SEC = 1 in RISAF_REGx_CFGR, or if DCEN = 0 in RISAF_REGx_ANESTR. Any read access is allowed to this register. Writes are ignored if RLOCK = 1 or SREN = 1 in RISAF_REGx_ACFGR.

When the delegated configuration is enabled (DCEN = 1 in RISAF_REGx_ANESTR), only the compartment DCCID defined in RISAF_REGx_ANESTR can write to this register. When the delegation is disabled (DCEN = 0), only the secure privileged software can program this register.

Refer to *Section 7.3* to verify the number of base regions and subregions supported by this RISAF instance, and for details on protected address space size and watermark granularity.

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

Bits 31:0 **SADDEND[31:0]**: Subregion address end

This bitfield defines the byte-address offset (relative to the base address of the protected address space) of the highest byte-address of subregion A in base region x.

Address bits less than the granularity cannot be modified, and remain 1.

Address bits greater than the protected address space size cannot be written, and remain 0.

The effective subregion is the intersection of the subregion and its base region (there is no point in programming SADDEND higher than the associated BADDEND).

7.5.13 RISAF region x subregion A nested mode register (RISAF_REGx_ANESTR)

Address offset: 0x05C + 0x40 * (x - 1) (x = 1 to 15)

Reset value: 0x0000 0000

Secure, privileged, trusted domain CID = 1, write access only. Any read access is allowed on this register. Writes are ignored if GLOCK is set in RISAF_CR.

Refer to *Section 7.3* to verify the number of base regions and subregions supported by this RISAF instance.

Caution: Check *Section 7.3* before setting DCEN bit.

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

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 **DCCID[2:0]**: Delegated configuration CID

The secure privileged software can use this bitfield to statically define which compartment can program the RIF configuration of subregion A if DCEN = 1 in this register.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **DCEN**: Delegated configuration enable

0: RIF configuration for subregion A can be written only by the secure privileged software.

1: RIF configuration for subregion A can be written only by compartment DCCID, defined in this register. If base region x is defined as secure (SEC = 1), nonsecure write accesses to this subregion RIF configuration by compartment DCCID are ignored.

Note: RIF configuration of subregion A is defined as RISAF_REGx_ACFGR, RISAF_REGx_ASTARTR, and RISAF_REGx_AENDR.

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

7.5.14 RISAF region x subregion B configuration register (RISAF_REGx_BCFGR)

Address offset: 0x060 + 0x40 * (x - 1) (x = 1 to 15)

Reset value: 0x0000 0000

Unprivileged writes are ignored. Nonsecure writes are ignored if SEC = 1 in RISAF_REGx_CFGR, or if DCEN = 0 in RISAF_REGx_BNESTR. Any read access is allowed to this register. Writes are ignored if RLOCK = 1 or SREN = 1 in RISAF_REGx_BCFGR.

When the delegated configuration is enabled (DCEN = 1 in RISAF_REGx_BNESTR), only the compartment DCCID defined in RISAF_REGx_BNESTR can write to this register. When the delegation is disabled (DCEN = 0), only the secure privileged software can program this register.

Refer to *Section 7.3* to verify the number of base regions and subregions supported by this RISAF instance.

Caution: Check *Section 7.3* before programming SRCID[2:0] bitfield.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	WREN	RDEN	Res.	Res.	PRIV	SEC	Res.	SRCID [2]	SRCID [1]	SRCID [0]	Res.	Res.	RLOCK	SREN
		rw	rw			rw	rw		rw	rw	rw			rs	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **WREN**: Write enable

This bit is taken into account only if SREN = 1 in this register.

0: Any write to subregion B of region x is ignored

1: Subregion B of region x can be written

Bit 12 **RDEN**: Read enable

This bit is taken into account only if SREN = 1 in this register.

0: Any read to subregion B of region x returns 0

1: Subregion B of region x can be read

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **PRIV**: Privileged subregion

This bit is taken into account only if SREN = 1 in this register.

0: Privileged and unprivileged accesses are granted in subregion B of region x

1: Only privileged accesses are granted in subregion B of region x

Note: When base region x is defined as unprivileged for the SCID compartment (PRIVCy = 0 in RISAF_REGx_CFGR, with SRCID = Cy), unprivileged accesses to subregion B are always granted.

Bit 8 **SEC**: Secure subregion

This bit is taken into account only if SREN = 1 in this register. Nonsecure modification to it are discarded.

0: Only nonsecure requests can access subregion B of region x

1: Only secure requests can access subregion B of region x

Note: When base region x is defined as nonsecure (SEC = 0 in RISAF_REGx_CFGR), only nonsecure accesses to subregion B are granted.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **SRCID[2:0]**: Subregion CID

When SREN = 1 in this register, SRCID[2:0] defines which compartment has access to subregion B of RISAF region x.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **RLOCK**: Resource lock

When this bit is set, the subregion resource is locked.

0: RISAF_REGx_BCFGR, RISAF_REGx_BSTARTR, and RISAF_REGx_BENDR are writable

1: Writes to RISAF_REGx_BCFGR, RISAF_REGx_BSTARTR, and RISAF_REGx_BENDR are ignored

Note: This bit is set once, cleared only by RISAF reset. It cannot be set if GLOCK = 0 in RISAF_CR.

Bit 0 **SREN**: Subregion enable

0: Subregion B of region x is disabled. Access control of base region x applies to any access between this subregion start and end addresses.

1: Subregion B of region x is enabled. Access control defined in subregion B applies to any access between subregion B start and end addresses.

Note: The filtering starts from address RISAF_REGx_BSTARTR (included), and ends at address RISAF_REGx_BENDR (included).

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

Address offset: 0x064 + 0x40 * (x - 1) (x = 1 to 15)

Reset value: 0x0000 0000

Unprivileged writes are ignored. Nonsecure writes are ignored if SEC = 1 in RISAF_REGx_CFGR, or if DCEN = 0 in RISAF_REGx_BNESTR. Any read access is allowed to this register. Writes are ignored if RLOCK = 1 or SREN = 1 in RISAF_REGx_BCFGR.

When the delegated configuration is enabled (DCEN = 1 in RISAF_REGx_BNESTR), only the compartment DCCID defined in RISAF_REGx_BNESTR can write to this register. When the delegation is disabled (DCEN = 0), only the secure privileged software can program this register.

Refer to *Section 7.3* to verify the number of base regions and subregions supported by this RISAF instance, and for details on protected address space size and watermark granularity.

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

Bits 31:0 **SADDSTART[31:0]**: Subregion address start

This bitfield defines the byte-address offset (relative to the base address of the protected address space) of the lowest byte-address in subregion B in base region x.

Address bits less than the granularity and greater than the protected address space size cannot be written, and remain 0.

The effective subregion is the intersection of the subregion and its base region (there is no point in programming SADDSTART below the associated BADDSTART).

7.5.16 RISAF region x subregion B end-address register (RISAF_REGx_BENDR)

Address offset: 0x068 + 0x40 * (x - 1) (x = 1 to 15)

Reset value: 0x0000 0FFF

This reset value applies when the granularity is 4 Kbytes. More in general, if the granularity is 2^n, the reset value is 2^n - 1.

Unprivileged writes are ignored. Nonsecure writes are ignored if SEC = 1 in RISAF_REGx_CFGR, or if DCEN = 0 in RISAF_REGx_BNESTR. Any read access is allowed to this register. Writes are ignored if RLOCK = 1 or SREN = 1 in RISAF_REGx_BCFGR.

When the delegated configuration is enabled (DCEN = 1 in RISAF_REGx_BNESTR), only the compartment DCCID defined in RISAF_REGx_BNESTR can write to this register. When the delegation is disabled (DCEN = 0), only the secure privileged software can program this register.

Refer to *Section 7.3* to verify the number of base regions and subregions supported by this RISAF instance, and for details on protected address space size and watermark granularity.

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

Bits 31:0 **SADDEND[31:0]**: Subregion address end

This bitfield defines the byte-address offset (relative to the base address of the protected address space) of the highest byte-address of subregion B in base region x.

Address bits less than the granularity cannot be modified, and remain 1.

Address bits greater than the protected address space size cannot be written, and remain 0.

The effective subregion is the intersection of the subregion and its base region (there is no point in programming SADDEND higher than the associated BADDEND).

7.5.17 RISAF region x subregion B nested mode register (RISAF_REGx_BNESTR)

Address offset: 0x06C + 0x40 * (x - 1) (x = 1 to 15)

Reset value: 0x0000 0000

Secure, privileged, trusted domain CID = 1, write access only. Any read access is allowed on this register. Writes are ignored if GLOCK is set in RISAF_CR.

Refer to *Section 7.3* to verify the number of base regions and subregions supported by this RISAF instance.

Caution: Check *Section 7.3* before setting DCEN bit.

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

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 **DCCID[2:0]**: Delegated configuration CID

The secure privileged software can use this bitfield to statically define which compartment can program the RIF configuration of subregion B if DCEN = 1 in this register.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **DCEN**: Delegated configuration enable

0: RIF configuration for subregion B can be written only by the secure privileged software.

1: RIF configuration for subregion B can be written only by compartment DCCID, defined in this register. If base region x is defined as secure (SEC = 1), nonsecure write accesses to this subregion RIF configuration by compartment DCCID are ignored.

Note: RIF configuration of subregion B is defined as RISAF_REGx_BCFGR, RISAF_REGx_BSTARTR, and RISAF_REGx_BENDR.

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

7.5.18 RISAF register map

Table 27. RISAF register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	RISAF_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GLOCK
	Reset value																																0
0x004	Reserved	Reserved																															
0x008	RISAF_IASR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IAEF	CAEF
	Reset value																															0	0
0x00C	RISAF_IACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AEF	CAEF
	Reset value																															0	0
0x010-0x01C	Reserved	Reserved																															
0x020	RISAF_IAESR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IANRW	Res.	IASEC	IAPRIV	Res.	IACID [2:0]		
	Reset value																									0		0	0		0	0	0
0x024	RISAF_IADDR	IADD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028-0x03C	Reserved	Reserved																															
0x040 + 0x040*(x-1) (x=1 to 15) Last address: 0x3C0	RISAF_REGx_CFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIVC7	PRIVC6	PRIVC5	PRIVC4	PRIVC3	PRIVC2	PRIVC1	PRIVC0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BREN
	Reset value									0	0	0	0	0	0	0	0								0								0
0x044 + 0x40*(x-1) (x=1 to 15) Last address: 0x3C4	RISAF_REGx_STARTR	BADDSTART[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048 + 0x40*(x-1) (x=1 to 15) Last address: 0x3C8	RISAF_REGx_ENDR	BADDEND[31:0]																															
	Reset value(1)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x04C + 0x40*(x-1) (x=1 to 15) Last address: 0x3CC	RISAF_REGx_CIDCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRENC7	WRENC6	WRENC5	WRENC4	WRENC3	WRENC2	WRENC1	WRENC0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDENC7	RDENC6	RDENC5	RDENC4	RDENC3	RDENC2	RDENC1	RDENC0
	Reset value									0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x050 + 0x40*(x-1) (x=1 to 15) Last address: 0x3D0	RISAF_REGx_ACFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WREN	RDEN	Res.	Res.	PRIV	SEC	Res.	SRCID [2:0]			Res.	Res.	RLOCK	SREN
	Reset value																			0	0			0	0		0	0	0			0	0
0x054 + 0x40*(x-1) (x=1 to 15) Last address: 0x3D4	RISAF_REGx_ASTARTR	SADDSTART[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 27. RISAF register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x058 + 0x40*(x-1) (x=1 to 15) Last address: 0x3D8	RISAF_REGx_AENDR	SADDEND[31:0]																															
	Reset value(1)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x05C + 0x40*(x-1) (x=1 to 15) Last address: 0x3DC	RISAF_REGx_ANESTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCCID [2:0]			Res.	DCEN	Res.	Res.
	Reset value																										0	0	0		0		
0x060 + 0x40*(x-1) (x=1 to 15) Last address: 0x3E0	RISAF_REGx_BCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WREN	RDEN	Res.	Res.	PRIV	SEC	Res.	SRCID [2:0]			Res.	Res.	RLOCK	SREN
	Reset value																			0	0			0	0		0	0	0			0	0
0x064 + 0x40*(x-1) (x=1 to 15) Last address: 0x3E4	RISAF_REGx_BSTARTR	SADDSTART[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x068 + 0x040*(x-1) (x=1 to 15) Last address: 0x3E8	RISAF_REGx_BENDR	SADDEND[31:0]																															
	Reset value(1)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x06C + 0x40*(x-1) (x=1 to 15) Last address: 0x3EC	RISAF_REGx_BNESTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCCID [2:0]			Res.	DCEN	Res.	Res.
	Reset value																										0	0	0		0		

1. Reset value is 0x0000 0FFF when the granularity is 4 Kbytes.

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

8 Illegal access controller (IAC)

8.1 IAC introduction

The RIF (resource isolation framework) is a comprehensive set of hardware blocks designed to enforce and manage the isolation of STM32 hardware resources, such as memory and peripherals.

The RIF uses the IAC (illegal access controller) to centralize the detection of RIF-related illegal accesses, managed by a secure application.

8.2 IAC main features

- Centralized detection of RIF-related illegal accesses, managed by a secure application. Supported illegal accesses are: secure, privileged.
- Only accessible by secure privileged CPU

8.3 IAC implementation

The index of the peripherals managed by the IAC in this device is defined in *Table 28*. Indexes 0 to 127 are also used in the RIFSC to configure the proper access control using RISUP blocks.

Table 28. Peripheral indexes in IAC

Peripherals 31 to 0																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIM5	TIM4	TIM3	TIM2	TIM1	FDCAN1/2/3	LPUART1	USART10	UART9	UART8	UART7	USART6	UART5	UART4	USART3	USART2	USART1	I3C2	I3C1	I2C4	I2C3	I2C2	I2C1	SAI2	-	SAI1	SPI6/I2S6	SPI5	SPI4	SPI3/I2S3	SPI2/I2S2	SPI1/I2S1
Peripherals 63 to 32																															
63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
-	SYSCONF	SPDIFRX	GB-ETH	-	UCPD	OTG1_FS	OTG1_HS	MDIOS	SDMMC2	SDMMC1	MDF1	ADF1	LPTIM5	LPTIM4	LPTIM3	LPTIM2	LPTIM1	GFXTIM	TIM18	TIM17	TIM16	TIM15	TIM14	TIM13	TIM12	TIM11	TIM10	TIM9	TIM8	TIM7	TIM6
Peripherals 95 to 64																															
95	94	93	92	91	90	89	88	87	86	85	84	83	82	81	80	79	78	77	76	75	74	73	72	71	70	69	68	67	66	65	64
-	DCMI	DCMIPP	CSI2HOST	-	FMC	XSPIM	XSPI3	XSPI2	XSPI1	-	MCE4	MCE3	MCE2	MCE1	CRYP1	HASH	SAES	PKA	RNG	-	-	-	-	-	-	WWDG	IWDG	CRC	-	VREFBUF	ADC12

Table 28. Peripheral indexes in IAC (continued)

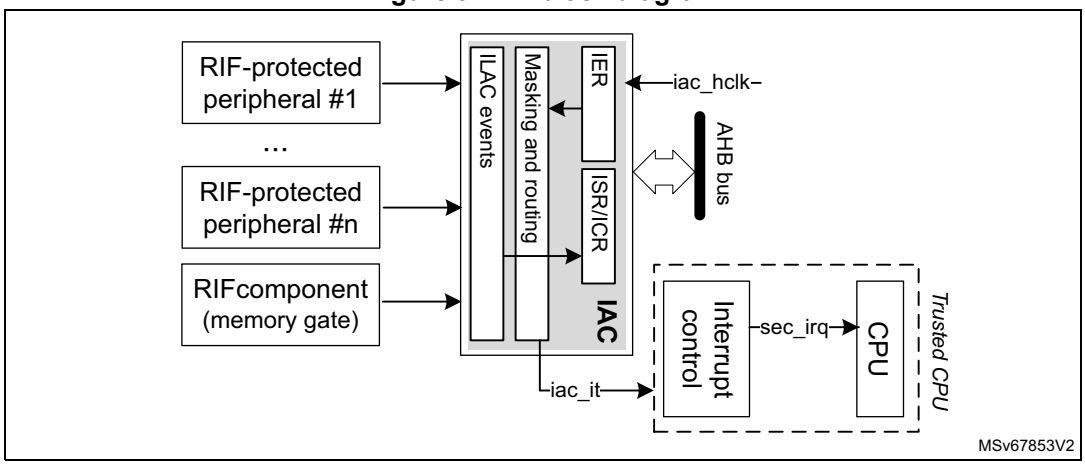
Peripherals 127 to 96																															
127	126	125	124	123	122	121	120	119	118	117	116	115	114	113	112	111	110	109	108	107	106	105	104	103	102	101	100	99	98	97	96
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	NPU - Neural ART	-	LTDC_L2	LTDC_L1	LTDC_CMN	DMA2D	GFXMMU	GPU	ICACHE	VENC	JPEG
Peripherals 159 to 128																															
159	158	157	156	155	154	153	152	151	150	149	148	147	146	145	144	143	142	141	140	139	138	137	136	135	134	133	132	131	130	129	128
-	RIFSC	RISAF23 (BKPSRAM)	RISAF22 (AHB RAM2)	RISAF21 (AHB RAM1)	-	RISAF15 (Cache config)	RISAF14 (FMC)	RISAF13 (XSPI3)	RISAF12 (XSPI2)	RISAF11 (XSPI1)	-	RISAF9 (VENCRAM)	RISAF8 (CACHEAXI)	RISAF7 (FLEXRAM)	RISAF6 (CPU_MST)	RISAF5 (NPU_MST1)	RISAF4 (NPU_MST0)	RISAF3 (AXISRAM2)	RISAF2 (AXISRAM1)	RISAF1 (TCM)	IAC	PWR_CTRL	RCC	BSEC	TAMP	RTC	-	HPDMA1	GPDMA1	EXTI	CM55

8.4 IAC functional description

8.4.1 IAC block diagram

Figure 9 shows the IAC block diagram, in context.

Figure 9. IAC block diagram



8.4.2 IAC internal signals

Table 29 lists the internal signals available at IAC level, not at product level.

Table 29. IAC internal signals

Name	Type	Description
iac_hclk	Digital input	AHB bus clock
iac_it	Digital output	Global interrupt request

8.4.3 IAC reset and clocks

The IAC peripheral has a dedicated clock and reset control in the RCC.

The IAC requires its clock in order to trigger an interrupt request event.

8.4.4 IAC use in RIF

The IAC gathers illegal events generated within the system when an illegal access is detected. The IAC can then generate a secure interrupt towards the secure CPU, if needed. By default, all events in the IAC are masked.

The IAC has room for 160 interrupt sources. Each source corresponds to an IAC index from 0 to 159. The interrupt sources from non-RIF-aware peripherals occupy most of the first 128 indexes. The interrupt sources from RIF-aware peripherals and RIF firewalls occupy indexes from 128 onwards. All these sources are defined in *Section 8.3*.

8.4.5 IAC management by trusted application

For each interrupt source of index <i>, an illegal access sets the IAFi flag in IAC_ISRx. A secure privileged application can clear this flag using the corresponding bit in IAC_ICRx. An IAC interrupt is triggered when IAFi is set while IAIEi is set in IAC_IERx.

Indexes between 0 to 127 are assigned to non-RIF-aware peripherals, with a RISUP block instantiated in front of its configuration port to filter configuration accesses. The programming of these RISUP blocks (in the RIFSC) uses the same index as the IAC.

Indexes 128 and above are assigned to RIF components and RIF-aware peripherals. These peripherals can block illegal accesses to their configuration registers (without RISUP).

For RIF-aware peripherals protecting a memory (RISAF), a secure privileged application can obtain, in the peripheral, more details about the cause of the illegal access event:

- The illegal access status register (xx_IASR) indicates if it is a configuration access error (CAEF= 1), or/and a system bus access error (IAEF = 1).
- The illegal access error status register (xx_IAESR) and the illegal address register (xx_IADDR) give detailed information on the system bus access error.

The IAC interrupt handler must clear IAFi in IAC_ISR before clearing the error flag(s) in the firewall peripheral xx (using xx_IACR register). Otherwise, an illegal access can occur between clearing the flags in the firewall and clearing the flag in the IAC, and the application is unaware of this error.

8.5 IAC interrupts

The RIF interrupt management in the IAC is summarized in *Table 30*.

Note: Only secure privileged application can enable, read, and clear illegal access interrupts.

If there is an illegal access to the IAC itself, the status flag corresponding to the IAC index in Section 8.3 is set.

Table 30. IAC interrupt request

Acronym	Interrupt event	Event flag in IAC_ISR	Enable control bit in IAC_IER	Interrupt clear method
IAC	Illegal access error	IAF	IAIE	Set IAF bit in IAC_ICR

8.6 IAC registers

All IAC registers are accessed only by words (32 bits).

8.6.1 IAC interrupt enable register x (IAC_IERx)

Address offset: 0x000 + 0x4 * x (x = 0 to 5)

Reset value: 0x0000 0000

Secure privileged, read and write only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IAIE{i + 32 * x}															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IAIE{i + 32 * x}															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IAIE{i + 32 * x}**: Illegal access interrupt enable for peripheral {i + 32 * x} (i = 0 to 31)
Each bit is set to unmask illegal access events from peripheral {i + 32 * x}.
0: Illegal access event from peripheral does not generate interrupt (masked).
1: Illegal access event from peripheral can generate interrupts (unmasked).

8.6.2 IAC interrupt status register x (IAC_ISRx)

Address offset: 0x080 + 0x4 * x (x = 0 to 5)

Reset value: 0x0000 0000

Secure privileged, read only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IAF{i + 32 * x}															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IAF{i + 32 * x}															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **IAF{i + 32 * x}**: Illegal access interrupt enable for peripheral {i + 32 * x} (i = 0 to 31)

Each bit is set when an illegal access event occurs in the peripheral {i + 32 * x} (see *Section 8.3* for details). This bit is cleared when the corresponding IAF bit is set in IAC_ICRx.

0: No illegal access event detected for the peripheral (since reset or the last time this bit was cleared).

1: At least one illegal access event has been detected for the peripheral (since the last time this bit was cleared).

8.6.3 IAC interrupt clear register x (IAC_ICRx)

Address offset: 0x100 + 0x4 * x (x = 0 to 5)

Reset value: 0x0000 0000

Secure, privileged write only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IAF{i + 32 * x}															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IAF{i + 32 * x}															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **IAF{i + 32 * x}**: Illegal access flag clear for peripheral {i + 32 * x} (i = 0 to 31)

Setting each bit clears the status flag of the illegal access event {i + 32 * x} in IAC_ISRx.

0: IAF {i + 32 * x} flag status not affected

1: IAF {i + 32 * x} flag status cleared in IAC_ISRx

8.6.4 IAC ILAC input status register x (IAC_IISRx)

Address offset: 0x36C + 0x4 * x (x = 0 to 5)

Reset value: 0xFFFF FF7F, 0x77FF FFFF, 0x77DF F03B, 0x0000 05FF, 0x7BEF FFEF, 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ILACIN{i + 32 * x}															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ILACIN{i + 32 * x}															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ILACIN{i + 32 * x}**: Illegal access input {i + 32 * x} (i = 0 to 31)

0: ILAC input {i + 32 * x} to IAC not present

1: ILAC input {i + 32 * x} to IAC present

8.6.5 IAC register map

Table 31. IAC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000 +0x4*x (x = 0 to 5) Last address: 0x014	IAC_IERx	IAIE{i + 32 * x}																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018-0x07C	Reserved	Reserved																															
0x080 +0x4*x (x = 0 to 5) Last address: 0x094	IAC_ISRx	IAF{i + 32 * x}																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x098-0x0FC	Reserved	Reserved																															
0x100 +0x4*x (x = 0 to 5) Last address: 0x114	IAC_ICRx	IAF{i + 32 * x}																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x118-0x368	Reserved	Reserved																															
0x36C +0x4*x (x = 0 to 5) Last address: 0x384	IAC_IISRx	ILACIN{i + 32 * x}																															
	Reset value	0xFFFF FF7F, 0x77FF FFFF, 0x77DF F03B, 0x0000 05FF, 0x7BEF FFEF, 0x0000 0000																															

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

9 Boot modes

9.1 Boot after system reset

The ROM code is the first one executed after any system reset.

The boot mode is determined by BOOT0 and BOOT1 pins, and one OTP word.

Table 32. Boot modes

BOOT0	BOOT1	Boot source #1
-	1	Development boot
0	0	Flash boot
1	0	Serial boot

BOOT0 is a dedicated pin latched upon reset release.

The BOOT1 is a non dedicated boot pin. The BOOT1 value comes from BOOT1 pin (default pin), or any other pin defined by BSEC BOOTROM_CONFIG10 - OTP_WORD19[28:21] = dev_boot_port + dev_boot_pins, see *Table 18: OTP fuse description (lower OTP region)*.

9.1.1 Flash boot

If a flash boot is selected, the firmware is loaded from an external flash memory. The flash source selection is done through BOOTROM_CONFIG2 - OTP_WORD11[8:5] = boot_source (4 bits). The possible sources are listed below:

- XSPI serial NOR (in SPI mode, single)
- XSPI HyperFlash™ (8-bit)
- *e*.MMC™ SDMMC1 or *e*.MMC™ SDMMC2 (up to JEDEC v5.1)
- SD-Card SDMMC1 (up to SD standard v6.0)

See *Table 18: OTP fuse description (lower OTP region)*.

If no boot source is programmed in OTP, default source is serial NOR.

9.1.2 Serial boot

If serial boot is selected, the image is loaded from a serial interface. Serial interfaces and serial instances can be disabled by BOOTROM_CONFIG2 - OTP_WORD11[16:9] = boot_source_disable (8 bits).

Possible sources are:

- USB boot: USB 2.0 OTG HS
- UART boot: USART1, USART2, and UART4

See *Table 18: OTP fuse description (lower OTP region)*.

9.1.3 Development boot

If BOOT1 is selected, the BootROM code finishes in an endless loop after having reopened debug in a secure way.

This boot mode is available only when the device is in development.

Secure installation

The ROM code is the root-of-trust of secure firmware installation.

9.2 Boot from a low-power mode

In case of boot from a low-power mode (such as Standby mode with SRAM retention), the bootROM is not executed, and the CPU starts from the boot address defined in *Section 16.1: SYSCFG registers*).

10 SRAM configuration controller (RAMCFG)

10.1 RAMCFG introduction

The RAMCFG configures the features of the internal SRAMs: AHBSRAM1/2, AXISRAM1 to 6, BKPSRAM, FLEXRAM, and VENCRAM.

10.2 RAMCFG main features

The internal SRAM supports the following features, configured in the RAMCFG:

- Error code correction (ECC):
  - Single-error detection and correction with interrupt generation
  - Double-error detection with interrupt generation
  - Status with failing address
- SRAM hardware erase: on reset or dedicated event, the RAM content is automatically erased (written as 0).
- SRAM software erase: the software can trigger an SRAM erase through RAMCFG registers.

10.3 RAMCFG functional description

10.3.1 Internal SRAM features

The following SRAMs are embedded in the devices, each with specific features:

- AXISRAM1 to 6 are the main SRAMs. AHBSRAM1/2 are preferably for DMA-controlled peripheral-to-memory data flow.

  AXISRAM2 to 6 can be shut down when the application is in Run mode.
- FLEXMEM can be allocated as Cortex-M55 TCM, or as system RAM (FLEXRAM).

  80 Kbytes of the FLEXMEM are retained in Standby mode, either allocated as extended ITCM (64 +16-Kbyte ECC), or allocated as FLEXRAM.
- The BKPSRAM content is retained in low-power modes, even when V_{DD} is off in V_{BAT} mode. On a tamper-event detection, the BKPSRAM content is erased.
- The AXISRAM8/VENCRAM implements hardware and software erases. It can be used at system level when the VENC is not used by the application. In this case, the AXISRAM8 is addressed.
- The RAMCFG embeds registers that are related to the internal SRAMs ECC, shutdown control, and software erase.
- SRAM7 (256 KB) is used as default at the system level.
  - On STM32N6x5 devices it is used only as internal RAM at system level.
  - On STM32N6x7 devices, after configuration (CACHEAXI_CR1 register, bit 0 EN: enable) this area is used as cache memory dedicated to Neural ART and is no longer accessible at system level. Due to this capacity, SRAM7 is controlled directly in the CACHEAXI registers, and is not under RAMCFG.

Note: The baseline I/D-TCM content can be retained in Standby mode (as BKPSRAM and 80 Kbytes of FLEXMEM), but this is not under RAMCFG control.

The CACHEAXI internal RAM can be used at system level when the NPU does not use its cache (as the VENCRAM). This is controlled directly in the CACHEAXI, and is not under RAMCFG.

Table 33 summarizes the features supported by each internal SRAM.

Table 33. Internal SRAM features

SRAM feature	AXISRAM1	AXISRAM2	FLEXRAM	AXISRAM3	AXISRAM4	AXISRAM5	AXISRAM6	AXISRAM7	AHBSRAM1	AHBSRAM2	VENCRAM/ AXISRAM8	BKPSRAM
Size (Kbytes)	624	1024	Up to 400	448	448	448	448	256	16	16	128	8
Word size	64	64	64	64	64	64	64	64	32	32	64	39(1)
Retention in Standby mode	-	-	X(2)	-	-	-	-	-	-	-	-	X
Retention in V_{BAT} mode	-	-	-	-	-	-	-	-	-	-	-	X
Block on potential tamper, erase on confirmed tamper	-	-	-	-	-	-	-	-	-	X	-	X
Hardware erase on reset	X	X	-	-	-	-	-	-	-	X	-	-
Software erase	X	X	X	X	X	X	X	X	X	X	X	X
ECC	-	-	X(3)	-	-	-	-	-	-	-	-	X

1. 32 bits of effective data, and 7 bits of embedded ECC.
2. Only the 80 Kbytes that correspond to the first portion of the I-TCM extension can be retained.
3. The ECC concerns only the FLEXMEM when used as extended TCM. It is under control of the Cortex-M55 TCM interface.

Table 34 details the conditions for the hardware erase.

Table 34. Hardware-erase conditions for internal SRAMs

SRAM	Conditional hardware erase	Condition
AXISRAM1	No	Erase on system reset
AXISRAM2		
AHBSRAM2	No	Erase on system reset and on confirmed tamper

10.3.2 FLEXRAM control

STM32N6x5/x7xx devices include a Cortex-M55 core that exposes a TCM interface. This TCM interface is spread in one 39-bit (32 effective data bits + 7 ECC protection bits) I-TCM (instruction TCM interface), and four parallel 39-bit D-TCMs (data TCM interfaces).

STM32N6x5/x7xx devices also embed 4096 Kbytes of RAM accessible on the AXI interconnect. The first half (FLEXRAM, AXISRAM1/2) is in the CPU frequency domain, and is preferably used for CPU storage. The second half of this memory (AXISRAM3/4/5/6/7) is in the NPU frequency domain, and is preferably used for the AI computing storage.

To let customers tailor the SRAM use to their needs, the FLEXMEM offers the possibility to allocate part of the CPU side SRAM (FLEXRAM part) as extended TCM, both on I-TCM and D-TCM interfaces. *Table 35* lists the supported allocations.

Table 35. FLEXRAM supported configurations

Config I-TCM	Config D-TCM	I-TCM fix	I-TCM flex (+ ECC)	I-TCM total	D-TCM fix	D-TCM flex (+ ECC)	D-TCM total	AXI-side FLEXRAM
0	0	1 x 64	0	64	4 × 32	0	128	400
0	1	1 x 64	0	64	4 × 32	4 × 32 (+ 8)	256	240
1	0	1 x 64	1 x 64 (+16)	128	4 × 32	0	128	320
1	1	1 x 64	1 x 64 (+16)	128	4 × 32	4 × 32 (+ 8)	256	160
2	0	1 x 64	1 x 192 (+ 48)	256	4 × 32	0	128	160
2	1	1 x 64	1 x 192 (+ 48)	256	4 × 32	4 × 32 (+ 8)	256	0

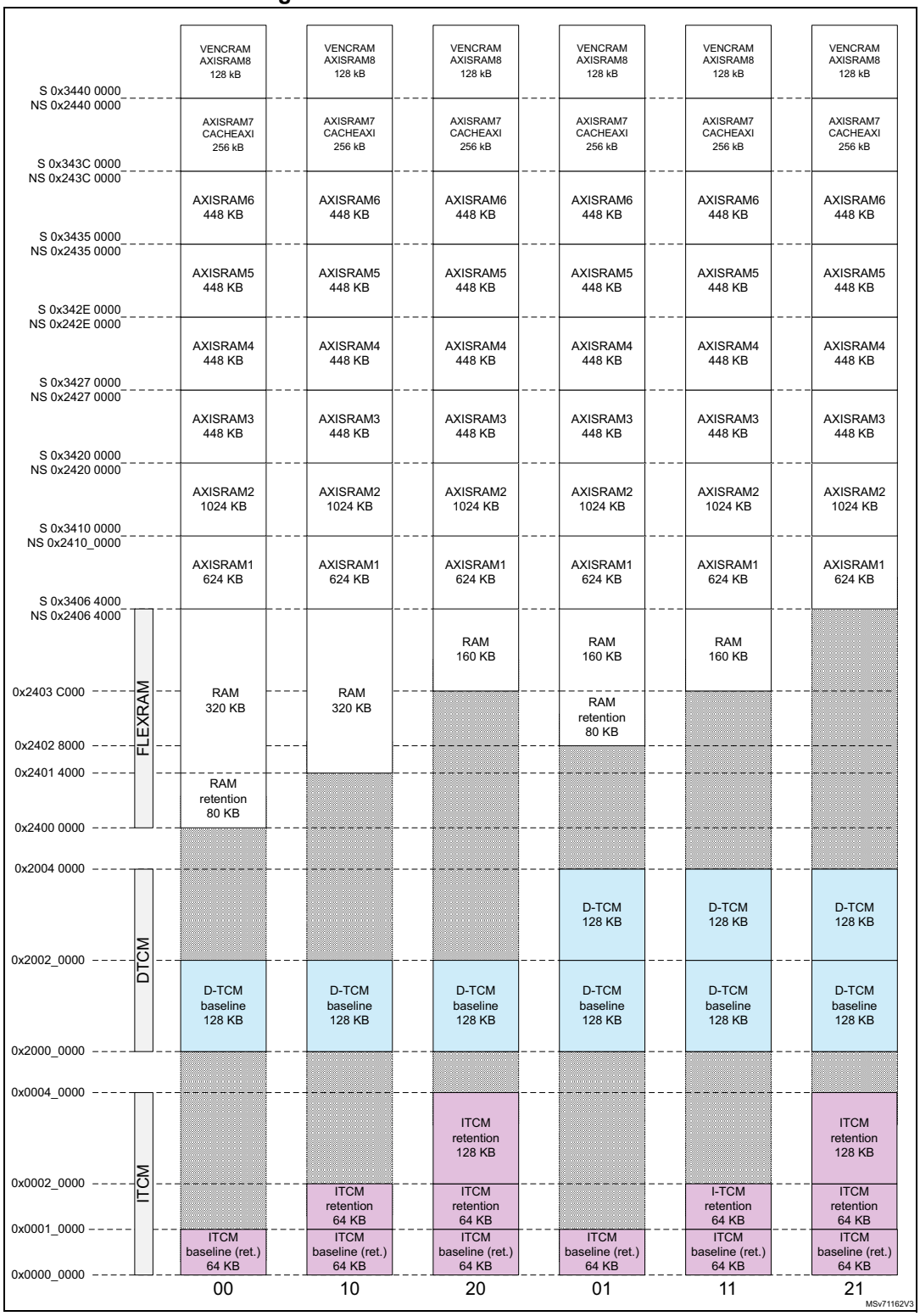
The retention regions are detailed in *Table 36* and *Figure 10*, sharing the same color code.

Table 36. FLEXMEM versus retention

Configuration	FLEXRAM base address(1)	I-TCM extension	D-TCM extension	Retention region
00	0x2400 0000	0 KB	0 KB	RAM region: 0x2400 0000 to 0x2401 3FFF
10	0x2401 4000	64 KB (+16 KB ECC)		I-TCM extension: first 64 KB (+16 KB ECC)
20	0x2403 C000	192 KB (+48 KB ECC)		
01	0x2402 8000	0 KB	128 KB (+32 KB ECC)	RAM region: 0x2402 8000 to 0x2403 BFFF
11	0x2403 C000	64 KB (+16 KB ECC)		I-TCM extension: first 64 KB (+16 KB ECC)
21	No RAM	192 KB (+48 KB ECC)		

1. All addresses considered as their non secure aliases.

Figure 10. FLEXMEM versus retention



Note: AXISRAM1 start address is 0x2406 4000 (aliased at 0x3406 4000 in secure boundary), whatever the FLEXMEM configuration.

The FLEXMEM configuration depends upon CFGDTCMSZ[3:0] and CFGITCMSZ[3:0] in SYSCFG_CM55TCMCR. Its configuration cannot be changed at runtime, a reboot is needed.

Use in low-power modes

The I-TCM and D-TCM baselines can be kept in retention when the device is in Standby mode. This allows the CPU to restart from the TCM content.

The first 64 Kbytes (+16 Kbytes for ECC) of extended I-TCM through the FLEXMEM can also be kept in retention. This allows a maximum of 128 Kbytes of I-TCM in retention.

When only the I-TCM baseline is used for the Cortex-M55, 80 Kbytes of the AXI RAM can be kept in retention.

The total amount of TCM that can be kept in retention is 320 Kbytes:

- 64 Kbytes of I-TCM baseline (+ 16 Kbytes for ECC)
- 64 Kbytes of I-TCM extended (+ 16 Kbytes for ECC)
- 4 x 32 Kbytes of D-TCM baseline (+ 4 x 8 Kbytes for ECC)

10.3.3 ECC (BKPSRAM)

The ECC is supported by the BKPSRAM. Seven ECC bits are added per 32 bits of SRAM: this allows a 2-bit error detection, and a 1-bit error correction on memory read access.

As the ECC is calculated and checked for a 32-bit word, byte and half-word write accesses are managed by the SRAM interface: it first reads the whole word, then writes the word again with the new byte/half-word value. ECC double errors are also detected during these byte or half-word AHB write accesses (read/modify/write done by the interface). The byte or half-word write access latency is two AHB clock cycles.

Caution: In case of a byte or half-word write on the SRAM with ECC, the read/modify/write operation is done in a buffer. The buffer content is written into the SRAM two AHB clock cycles after the SRAM AHB is released (when the SRAM is no more accessed).

The ECC is also available on the Cortex-M55 I-TCM and D-TCM.

When the FLEXMEM is used as I/D-TCM, it supports ECC through specific memory cut/word organization. When the FLEXMEM is used as FLEXRAM, it does not support ECC, even for the part that can be set in retention

Single and double ECC errors

When a single error is detected, it is automatically corrected, and SEDC (single error detected and corrected) is set in RAMCFG_BKPSRAMISR.

An interrupt is generated if enabled by SEIE in RAMCFG_BKPSRAMIER. The failing address is stored in RAMCFG_BKPSRAMESEAR, if ALE is set in RAMCFG_BKPRAMCR.

Caution: Single errors are not reported if SEDC is not cleared in time when a new error happens.

When a double error is detected, DED is set in RAMCFG_BKPSRAMISR. An interrupt is generated if enabled by DEIE in RAMCFG_BKPSRAMIER. The failing address is stored in RAMCFG_BKPSRAMEDEAR if ALE is set in RAMCFG_BKPSRAMCR.

Caution: Double errors are not reported if DED is not cleared. Double errors are not corrected, and must be managed by exception.

The ECC can be activated/deactivated by executing the following software sequence:

1. Write 0xAE in RAMCFG_BKSRAMECCKEYR.
2. Write 0x75 in RAMCFG_BKSRAMECCKEYR.
3. Write 1 to ECCE in RAMCFG_BKPSRAMCR to activate the ECC (write 0 to this bit to deactivate the ECC).

Fault injection

The RAMCFG supports a software loop for runtime fault injection in the BKPSRAM when the ECC is activated:

1. With the ECC enabled, the application writes at a given address.
2. The application disables the ECC writing.
3. The application writes a different data at the same address as previously. Data are updated but the matching ECC word is not modified.
4. The application enables ECC writing.
5. The application reads at the same address as previously. An interrupt is expected due to either one error corrected, or several errors detected. The application can check that the desired address appears in RAMCFG_BKPSRAMSEAR or _BKPSRAMDEAR.

10.4 RAMCFG low-power modes

After a reset, by default, the RAM is controlled by the RAM clock through the RCC embedded memories enable register (RCC_MEMENR). The following configurations apply:

- AXISRAM1 and AXISRAM2 are enabled.
- AXISRAM3 to AXISRAM6 are in power-down (shutdown) mode.
- AXISRAM7 operates as internal RAM at system level, and AXISRAM8 is in power-down (shutdown) mode.
- BKPSRAM is enabled.

When RAMs are in shutdown mode, their content is not retained. The following controls and behaviors apply:

- SRAMSD in AXISRAMx_CR (x = 2, 3, 4, 5, or 6) determines whether the RAM is in shutdown mode.
- When a RAM is in shutdown mode:
  - Writing to it has no effect.
  - Reading from it returns 0.
  - Upon exiting the shutdown state, the memory is not immediately accessible. A safe period of 40 ns must elapse before the memory becomes functional.

Sequences to safely enter or exit shutdown mode:

- To enter shutdown:
  a) Disable the RAM clock through the RCC (RCC_MEMENR)
  b) Shut the RAM down by writing 1 to SRAMSD in RAMCFG_AXISRAMxCR
- To exit shutdown:
  a) Power the RAM up by writing 0 to SRAMSD in RAMCFG_AXISRAMxCR
  b) Read back the value from the RAMCFG (alternatively, wait 40 ns)
  c) Enable the RAM clock through the RCC. (RCC_MEMENR)

d) The RAM is now ready for use.

Note: AXISRAM3 to AXISRAM6 are interleaved at the system level under software control through SYSCFG_NPU_ICNCR. Interleaving must be deactivated before shutting down these memories, especially if only part of them is set to shutdown.

Low power mode:

- Run mode: AXISRAM2 to AXISRAM6 can be set to power down (shutdown) under software control, AXISRAM7 and AXISRAM8 cannot
- Stop mode: all RAMs are set to low-power mode under the control of PWR control register
- Standby mode: all RAMs are off except BKPSRAM, BKPSRAM and 80 Kbytes of the FLEXRAM are set to retention under software control

10.5 RAMCFG interrupts

Two interrupts (maskable by software) are generated internally, and a global interrupt request is exported to the interrupt controller. *Table 37* lists RAMCFG interrupt requests.

Table 37. RAMCFG interrupt requests

Interrupt acronym	Memory	Interrupt event	Event flag(1)	Enable control bit	Interrupt clear method
RAMCFG	BKPSRAM	ECC single error detection and correction	SEDC	SEIE	Write 1 in CSEDC
		ECC double error detection	DED	DEIE	Write 1 in CDED

1. All these bits are in RAMCFG_BKPSRAMISR.

10.6 RAMCFG registers

The RAMCFG registers can be accessed in word, half-word, and byte format, unless differently specified.

10.6.1 RAMCFG AXISRAM1 control register (RAMCFG_AXISRAM1CR)

Address offset: 0x000

Reset value: 0x0000 0000

Reset on any system reset.

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

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **SRAMER**: SRAM erase

This bit can be set by software only after writing the unlock sequence in ERASEKEY of RAMCFG_AXISRAM1ERKEYR. Setting this bit starts the SRAM erase. This bit is automatically cleared by hardware at the end of the erase operation.

0: No erase operation ongoing

1: Erase operation ongoing

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

10.6.2 RAMCFG AXISRAMx interrupt status register (RAMCFG_AXISRAMxISR)

Address offset: 0x008 + 0x80 * (x - 1), (x = 1 to 6)

Reset value: 0x0000 0000

Reset on any system reset.

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

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **SRAMBUSY**: SRAM busy with erase operation
Depending on the SRAM, the erase operation can be performed due to software request, system reset if the option is not disabled by software and tamper detection (see *Table 33*).
0: No erase operation ongoing
1: Erase operation ongoing

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

10.6.3 RAMCFG AXISRAMx erase key register (RAMCFG_AXISRAMxERKEYR)

Address offset: 0x028 + 0x80 * (x - 1), (x = 1 to 6)

Reset value: 0x0000 0000

Reset on any system reset.

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

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **ERASEKEY[7:0]**: Erase write protection key
The following steps are required to unlock the write protection of SRAMER in RAMCFG_AXISRAMxCR.
a) Write 0xCA into this field.
b) Write 0x53 into this field.
Note: Writing a wrong key reactivates the write protection.

10.6.4 RAMCFG AXISRAMx control register (RAMCFG_AXISRAMxCR)

Address offset: 0x080 + 0x80 * (x - 2), (x = 2 to 6)

Reset value: 0x0000 0000, 0x0010 0000, 0x0010 0000, 0x0010 0000, 0x0010 0000

Reset on any system reset.

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

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **SRAMSD**: Shutdown AXISRAMx
0: AXISRAMx memory is powered.
1: AXISRAMx memory is in shutdown, and its content is not retained.

Bits 19:9 Reserved, must be kept at reset value.

Bit 8 **SRAMER**: SRAM erase
This bit can be set by software only after writing the unlock sequence in ERASEKEY of RAMCFG_AXISRAMxERKEYR. Setting this bit starts the SRAM erase. This bit is automatically cleared by hardware at the end of the erase operation.
0: No erase operation ongoing
1: Erase operation ongoing

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

10.6.5 RAMCFG AHBSRAMx control register (RAMCFG_AHBSRAMxCR)

Address offset: 0x300 + 0x80 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

Reset on any system reset.

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

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **SRAMER**: SRAM erase
This bit can be set by software only after writing the unlock sequence in ERASEKEY of RAMCFG_AHBSRAMxERKEYR. Setting this bit starts the SRAM erase. This bit is automatically cleared by hardware at the end of the erase operation.
0: No erase operation ongoing
1: Erase operation ongoing

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

10.6.6 RAMCFG AHBSRAMx interrupt status register (RAMCFG_AHBSRAMxISR)

Address offset: 0x308 + 0x80 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

Reset on any system reset.

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

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **SRAMBUSY**: SRAM busy with erase operation
Depending on the SRAM, the erase operation can be performed due to software request, system reset if the option is not disabled by software and tamper detection (see *Table 33*).
0: No erase operation ongoing
1: Erase operation ongoing

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

10.6.7 RAMCFG AHBSRAMx erase key register (RAMCFG_AHBSRAMxERKEYR)

Address offset: 0x328 + 0x80 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

Reset on any system reset.

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

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **ERASEKEY[7:0]**: Erase write protection key
To unlock the write protection of SRAMER in RAMCFG_AHBSRAMxCR.
a) Write 0xCA into this field.
b) Write 0x53 into this field.
Note: Writing a wrong key reactivates the write protection.

10.6.8 RAMCFG VENCRAM control register (RAMCFG_VENCRAMCR)

Address offset: 0x400

Reset value: 0x0000 0000

Reset on any system reset.

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

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **SRAMER**: SRAM erase

This bit can be set by software only after writing the unlock sequence in ERASEKEY of RAMCFG_VENCRAMERKEYR. Setting this bit starts the SRAM erase. This bit is automatically cleared by hardware at the end of the erase operation.

0: No erase operation ongoing

1: Erase operation ongoing

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

10.6.9 RAMCFG VENCRAM interrupt status register (RAMCFG_VENCRAMISR)

Address offset: 0x408

Reset value: 0x0000 0000

Reset on any system reset.

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

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **SRAMBUSY**: SRAM busy with erase operation

Depending on the SRAM, the erase operation can be performed due to software request, system reset if the option is not disabled by software and tamper detection (see *Table 33*).

0: No erase operation ongoing

1: Erase operation ongoing

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

10.6.10 RAMCFG VENCRAM erase key register (RAMCFG_VENCRAMERKEYR)

Address offset: 0x428

Reset value: 0x0000 0000

Reset on any system reset.

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

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **ERASEKEY[7:0]**: Erase write protection key

To unlock the write protection of SRAMER in RAMCFG_VENCRAMCR.

a) Write 0xCA into this field.

b) Write 0x53 into this field.

Note: Writing a wrong key reactivates the write protection.

10.6.11 RAMCFG BKPSRAM control register (RAMCFG_BKPSRAMCR)

Address offset: 0x480

Reset value: 0x0000 0000

Reset on any system reset.

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

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **SRAMER**: SRAM erase

This bit can be set by software only after writing the unlock sequence in ERASEKEY of RAMCFG_BKPSRAMERKEYR. Setting this bit starts the SRAM erase. This bit is automatically cleared by hardware at the end of the erase operation.

0: No erase operation ongoing

1: Erase operation ongoing

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **ALE**: Address latch enable

0: Failing address not stored in RAMCFG_BKPSRAMESEAR or _BKPSRAMEDEAR

1: Failing address stored in RAMCFG_BKPSRAMESEAR or _BKPSRAMEDEAR

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **ECCE**: ECC enable

When set, this bit can be cleared by software only after writing the unlock sequence in RAMCFG_BKPSRAMECCKEYR.

0: ECC disabled

1: ECC enabled

10.6.12 RAMCFG BKPSRAM interrupt enable register (RAMCFG_BKPSRAMIER)

Address offset: 0x484

Reset value: 0x0000 0000

Reset on any system reset.

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

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

Bit 1 **DEIE**: ECC double error interrupt enable
0: Double error interrupt disabled
1: Double error interrupt enabled

Bit 0 **SEIE**: ECC single error interrupt enable
0: Single error interrupt disabled
1: Single error interrupt enabled

10.6.13 RAMCFG BKPSRAM interrupt status register (RAMCFG_BKPSRAMISR)

Address offset: 0x488

Reset value: 0x0000 0000

Reset on any system reset.

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

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **SRAMBUSY**: SRAM busy with erase operation
Depending on the SRAM, the erase operation can be performed due to software request, system reset if the option is not disabled by software and tamper detection (see *Table 33*).
0: No erase operation ongoing
1: Erase operation ongoing

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

Bit 1 **DED**: ECC double-error detected
0: No double error
1: Double error detected

Bit 0 **SEDC**: ECC single error detected and corrected
0: No single error detected
1: Single error detected and corrected

10.6.14 RAMCFG BKPSRAM single error address register (RAMCFG_BKPSRAMESEAR)

Address offset: 0x48C

Reset value: 0x0000 0000

Reset on any system reset.

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

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:0 **ESEA[10:0]**: ECC single error address
When ALE is set in RAMCFG_BKPSRAMCR, this field is updated with the word address corresponding to the ECC single error.

10.6.15 RAMCFG BKPSRAM double error address register (RAMCFG_BKPSRAMEDEAR)

Address offset: 0x490

Reset value: 0x0000 0000

Reset on any system reset.

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

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:0 **EDEA[10:0]**: ECC double error address
When ALE is set in RAMCFG_BKPSRAMCR, this field is updated with the word address corresponding to the ECC double error.

10.6.16 RAMCFG BKPSRAM interrupt clear register (RAMCFG_BKPSRAMICR)

Address offset: 0x494

Reset value: 0x0000 0000

Reset on any system reset.

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

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

Bit 1 **CDED**: Clear ECC double-error interrupt
Writing 1 to this flag clears DED in RAMCFG_BKPSRAMISR.

Bit 0 **CSEDC**: Clear ECC single-error interrupt
Writing 1 to this flag clears SEDC in RAMCFG_BKPSRAMISR.

10.6.17 RAMCFG BKPSRAM ECC key register (RAMCFG_BKPSRAMECCKEYR)

Address offset: 0x4A4

Reset value: 0x0000 0000

Reset on any system reset.

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

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **ECCKEY[7:0]**: ECC write protection key
The following steps are required to unlock the write protection of ECCE in RAMCFG_BKPSRAMCR.
a) Write 0xAE into this field.
b) Write 0x75 into this field.

Note: Writing a wrong key reactivates the write protection.

10.6.18 RAMCFG BKPSRAM erase key register (RAMCFG_BKPSRAMERKEYR)

Address offset: 0x4A8

Reset value: 0x0000 0000

Reset on any system reset.

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

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **ERASEKEY[7:0]**: Erase write protection key

The following steps are required to unlock the write protection of SRAMER in RAMCFG_BKPSRAMCR.

a) Write 0xCA into this field.

b) Write 0x53 into this field.

Note: Writing a wrong key reactivates the write protection.

10.6.19 RAMCFG FLEXRAM control register (RAMCFG_FLEXRAMCR)

Address offset: 0x500

Reset value: 0x0000 0000

Reset on any system reset.

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

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **SRAMER**: SRAM erase

This bit can be set by software only after writing the unlock sequence in ERASEKEY of RAMCFG_FLEXRAMERKEYR. Setting this bit starts the SRAM erase. This bit is automatically cleared by hardware at the end of the erase operation.

0: No erase operation ongoing

1: Erase operation ongoing

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

10.6.20 RAMCFG FLEXRAM interrupt status register (RAMCFG_FLEXRAMISR)

Address offset: 0x508

Reset value: 0x0000 0000

Reset on any system reset.

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

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **SRAMBUSY**: SRAM busy with erase operation

Depending on the SRAM, the erase operation can be performed due to software request, system reset if the option is not disabled by software and tamper detection (see *Table 33*).

0: No erase operation ongoing

1: Erase operation ongoing

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

10.6.21 RAMCFG FLEXRAM erase key register (RAMCFG_FLEXRAMERKEYR)

Address offset: 0x528

Reset value: 0x0000 0000

Reset on any system reset.

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

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **ERASEKEY[7:0]**: Erase write protection key

The following steps are required to unlock the write protection of SRAMER in RAMCFG_FLEXRAMCR.

a) Write 0xCA into this field.

b) Write 0x53 into this field.

Note: Writing a wrong key reactivates the write protection.

10.6.22 RAMCFG register map

Table 38. RAMCFG register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	RAMCFG_AXISRAM1CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																								0								
0x004	Reserved	Reserved																															
0x008 + 0x80 * (x - 1) (x = 1 to 6)	RAMCFG_AXISRAMxISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMBUSY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																								0								
0x028 + 0x80 * (x - 1) (x = 1 to 6)	RAMCFG_ AXISRAMxERKEYR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERASEKEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x080 + 0x80 * (x - 2) (x = 2 to 6)	RAMCFG_AXISRAMxCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMSD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value												0												0								
0x300 + 0x80 * (x - 1) (x = 1 to 2)	RAMCFG_ AHBSRAMxCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																								0								
0x308 + 0x80 * (x - 1) (x = 1 to 2) Last address: 0x388	RAMCFG_ AHBSRAMxISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMBUSY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																								0								
0x328 + 0x80 * (x - 1) (x = 1 to 2) Last address: 0x3A8	RAMCFG_ AHBSRAMxERKEYR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERASEKEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x3AC - 0x3FC	Reserved	Reserved																															
0x400	RAMCFG_VENCRAMCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																								0								
0x404	Reserved	Reserved																															
0x408	RAMCFG_VENCRAMISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMBUSY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																								0								
0x40C - 0x424	Reserved	Reserved																															
0x428	RAMCFG_ VENCRAMERKEYR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERASEKEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

Table 38. RAMCFG register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x42C - 0x47C	Reserved	Reserved																															
0x480	RAMCFG_BKPSRAMCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCE
	Reset value																								0								0
0x484	RAMCFG_BKPSRAMIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEIE	SEIE
	Reset value																															0	0
0x488	RAMCFG_BKPSRAMISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMBUSY	Res.	Res.	Res.	Res.	Res.	Res.	DED	SEDC
	Reset value																								0							0	0
0x48C	RAMCFG_ BKPSRAMESEAR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ESEA[10:0]										
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x490	RAMCFG_ BKPSRAMEDEAR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EDEA[10:0]										
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x494	RAMCFG_BKPSRAMICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CDED	CSEDC
	Reset value																															0	0
0x498 - 0x4A0	Reserved	Reserved																															
0x4A4	RAMCFG_ BKPSRAMECCKEYR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECCKEY[7:0]								
	Reset value																									0	0	0	0	0	0	0	0
0x4A8	RAMCFG_ BKPSRAMERKEYR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERASEKEY[7:0]								
	Reset value																									0	0	0	0	0	0	0	0
0x4AC	Reserved	Reserved																															
0x500	RAMCFG_FLEXRAMCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																								0								
0x504	Reserved	Reserved																															
0x508	RAMCFG_FLEXRAMISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRAMBUSY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																								0								
0x50C - 0x524	Reserved	Reserved																															
0x528	RAMCFG_ FLEXRAMERKEYR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERASEKEY[7:0]								
	Reset value																									0	0	0	0	0	0	0	0

Refer to *Section 2.3* for the register boundary addresses.

11 Texture cache (ICACHE)

11.1 ICACHE introduction

The texture cache (ICACHE) is introduced on the AXI read-only texture port of the GPU to improve performance when reading texture data from internal and external memories.

The texture cache is an assembly of ICACHE (a peripheral with AHB ports) and an AXI-to-AHB bus bridge plugged on ICACHE AHB slave port, providing an AXI interface on the texture cache slave port.

The following sections only describe the AHB ICACHE peripheral itself.

Some specific features, like hit-under-miss, and critical-word-first refill policy, result in close to zero-wait-state performance in most use cases.

11.2 ICACHE main features

The main features of ICACHE are listed below:

- Bus interface
  - One 64-bit AHB slave port, the read port (input from GPU texture read interface)
  - One 64-bit AHB master port: master1 port (output to texture bus of main AXI/AHB bus matrix)
  - One 32-bit AHB slave port for control (input from AHB peripherals interconnect, for ICACHE registers access)
- Cache access
  - 0 wait-state on hits
  - Hit-under-miss capability: ability to serve processor requests (access to cached data) during an ongoing line refill due to a previous cache miss
  - Optimal cache line refill thanks to WRAPw bursts of the size of the cache line (32-bit word size, **w**, aligned on cache line size)
  - n-way set-associative default configuration with possibility to configure as 1-way, means direct mapped cache, for applications needing very-low-power consumption profile
- Replacement and refill
  - pLRU-t (pseudo-least-recently-used, based on binary tree) replacement policy, algorithm with best complexity/performance balance
  - Critical-word-first refill policy, minimizing processor stalls
- Performance counters

  The ICACHE implements two performance counters:
  - Hit monitor counter (32-bit)
  - Miss monitor counter (16-bit)
- Error management
  - Possibility to detect an unexpected cacheable write access, to flag an error, and optionally, to raise an interrupt
- Maintenance operation
  - Cache invalidate: full cache invalidation, fast command, noninterruptible

11.3 ICACHE implementation

Table 39. ICACHE features

Feature	ICACHE
Number of ways	4
Cache size	32 Kbytes
Cache line width	32 bytes
Number of regions to remap	0
Data size of AHB slave interface	64 bits
Data size of AHB fast master1 interface	64 bits
Data size of AHB slow master2 interface	0

11.4 ICACHE functional description

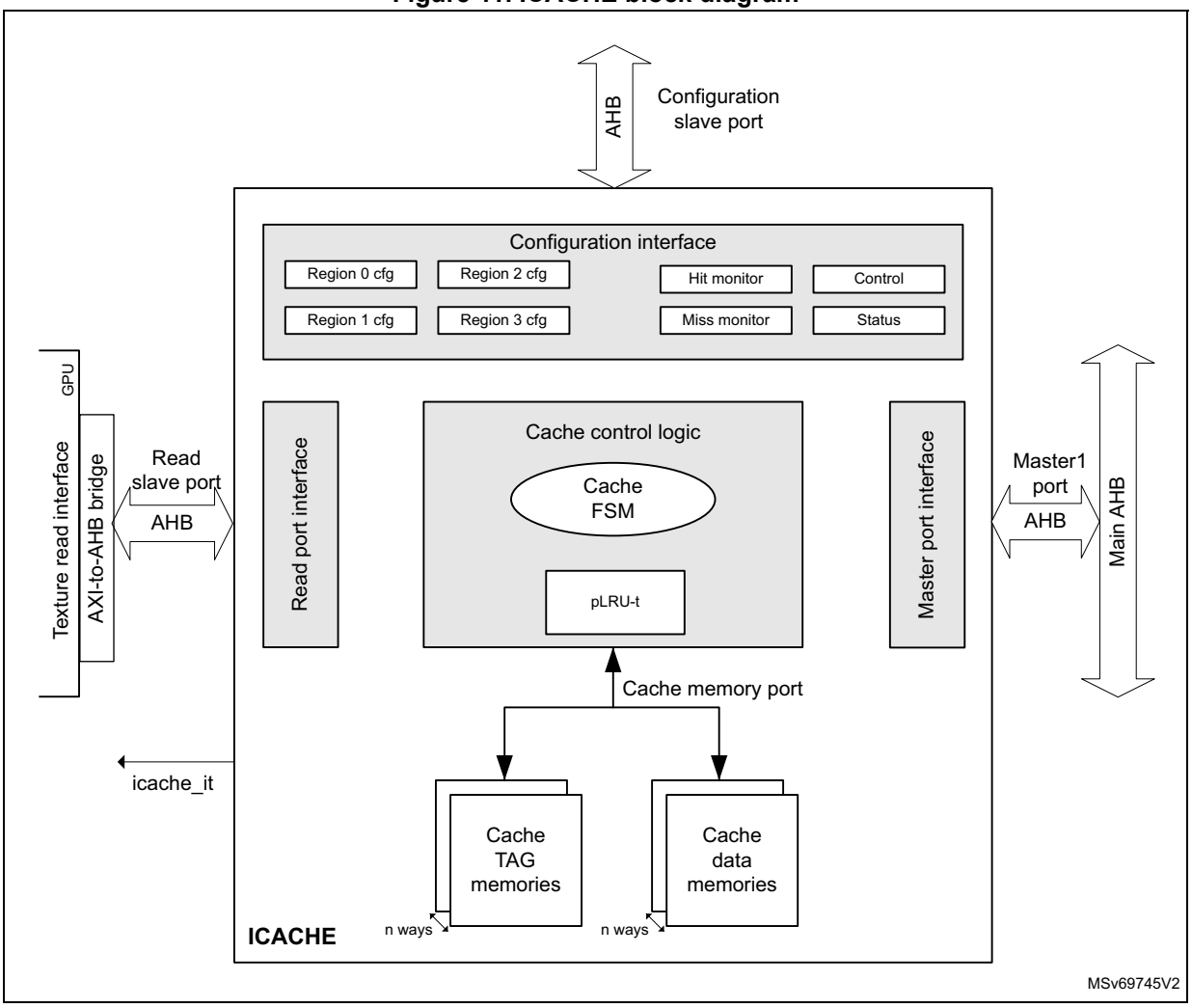
The purpose of the texture cache is to cache GPU read accesses to texture data. As such, the ICACHE manages only cacheable read transactions, and does not manage cacheable write transactions.

The noncacheable transactions (both read and write) bypass the ICACHE.

For the error management purpose, if a write cacheable transaction is presented (this happens only in case of bad software programming), the ICACHE sets an error flag and, if enabled, raises an interrupt to the processor.

11.4.1 ICACHE block diagram

Figure 11. ICACHE block diagram



11.4.2 ICACHE reset and clocks

The ICACHE is clocked on the texture AHB bus clock.

When the ICACHE reset signal is released, a cache invalidate procedure is automatically launched, making the ICACHE busy (ICACHE_SR = 0x0000 0001).

When this procedure is finished:

- the ICACHE is invalidated: “cold cache”, with all cache line valid bits = 0 (ICACHE must be filled up)
- ICACHE_SR = 0x0000 0002 (reflecting the cache is no longer busy)
- the ICACHE is disabled: the EN bit in ICACHE_CR holds its reset state (= 0).

Note: When disabled, the ICACHE is bypassed: slave input requests are forwarded to the master port.

11.4.3 ICACHE TAG memory

The ICACHE TAG memory contains:

- address tags indicating which data are contained in the cache data memories
- validity bits

There is one valid bit per cache line (per way).

The valid bit is set when a cache line is refilled (after a miss).

Valid bits are reset in any of the below cases:

- after the ICACHE reset is released
- when the cache is disabled, by setting EN = 0 in ICACHE_CR (by software)
- when executing an ICACHE invalidate command, by setting CACHEINV = 1 in ICACHE_CR (by software)

When a cacheable transaction is received at the execution input port, its AHB address (HADDR_in) is split into the following fields (see *Table 40* for B and W definitions):

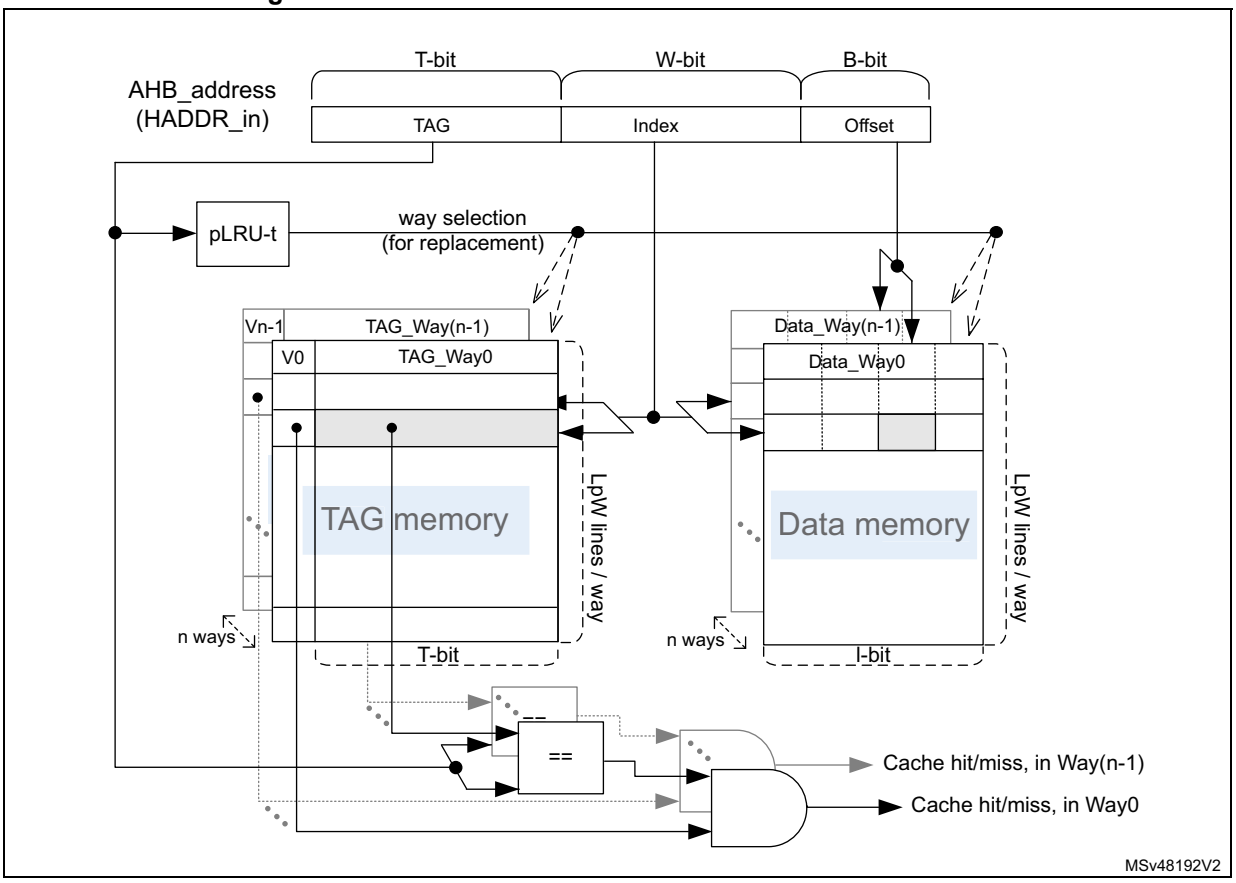
- HADDR_in[B-1:0]: address byte offset, indicates which byte to select inside a cache line.
- HADDR_in[B+W-1:B]: address way index, indicates which cache line to select inside each way.
- HADDR_in[31:B+W]: tag address, to be compared to the TAG memory address to check if the requested data is already available (meaning valid) inside the ICACHE.

The following table gives a summary of the ICACHE main parameters for TAG memory dimensioning. *Figure 12* shows the functional view of TAG and data memories, for an n-way set associative ICACHE.

Table 40. TAG memory dimensioning parameters for n-way set associative operating mode (default)

Parameter	Value	Example
Cache size	S Kbytes = s bytes (s = 1024 x S)	8 Kbytes = 8192 bytes
Cache number of ways	n	2
Cache line size	L-byte = l-bit (l = 8 x L)	16-byte = 128-bit
Number of cache lines	LpW = s / (n x L) lines / way	256 lines / way
Address byte offset size	B = $\log_2$(L) bit	4-bit
Address way index size	W = $\log_2$(LpW) bit	8-bit
TAG address size	T = (32 - W - B) bit	20-bit

Figure 12. ICACHE TAG and data memories functional view



11.4.4 Direct-mapped ICACHE (1-way cache)

The default configuration (at reset) is an n-way set associative cache (WAYSEL = 1 in ICACHE_CR), but the user can configure the ICACHE as direct mapped by writing WAYSEL = 0 (only possible when the cache is disabled, EN = 0 in ICACHE_CR).

The following table gives a summary of ICACHE main parameters for TAG memory when the direct-mapped cache operating mode is selected.

Table 41. TAG memory dimensioning parameters for direct-mapped cache mode

Parameter	Value	Example
Cache size	S Kbytes = s bytes (s = 1024 x S)	8 Kbytes = 8192 bytes
Cache number of ways	1	1
Cache line size	L-byte = l-bit (l = 8 x L)	16-byte = 128-bit
Number of cache lines	LpW = s / L lines	512 lines
Address byte offset size	B = $\log_2$(L) bit	4-bit
Address way index size	W = $\log_2$(LpW) bit	9-bit
TAG address size	T = (32 - W - B) bit	19-bit

All cache operations (such as read, refill, invalidation) remain the same in the direct-mapped configuration. The only difference is the absence of a replacement algorithm in case of line eviction (as explained in *Section 11.4.7*): only one way (the unique one) is possible for any data refill.

11.4.5 ICACHE enable

To activate the ICACHE, the EN bit in ICACHE_CR must be set to 1.

When the ICACHE is disabled, it is bypassed and all transactions are copied from the slave to the master port in the same clock cycle.

It is recommended to initialize or modify the main memory content (region to be later cached) with the ICACHE disabled, and to enable the ICACHE only when this region remains unchanged (an enabled ICACHE detects cacheable write transactions as errors).

To ensure performance determinism, it is recommended to wait for the end of a potential cache invalidate procedure before enabling the ICACHE. This procedure occurs when the hardware reset signal is released, when CACHEINV is set, or when EN is cleared in ICACHE_CR. During the procedure, BUSYF is set in ICACHE_SR, and once finished, BUSYF is cleared and BSYENDF is set in the same register (raising the ICACHE interrupt if enabled on such a busy end condition).

The software must test BUSYF and/or BSYENDF values before enabling the ICACHE. Else, if the ICACHE is enabled before the end of an invalidate procedure, any cache access (while BUSYF = 1) is treated as noncacheable, and its performance depends on the main memory access time.

The ICACHE is, by default, disabled at boot.

11.4.6 Cacheable and noncacheable traffic

The ICACHE is placed on the GPU texture bus, and thus caches all internal and external memory regions (ranging from address 0x0000 0000 to 0x3FFF FFFF, respectively 0x6000 0000 to 0x9FFF FFFF, of the memory map).

An incoming memory request to the ICACHE is defined as cacheable according to its AHB transaction memory lookup attribute, as shown in *Table 42*. This AHB attribute depends on the GPU setting for the addressed region.

Table 42. ICACHE cacheability for AHB transaction

AHB lookup attribute	Cacheability
1	Cacheable
0	Noncacheable

In the case of a noncacheable access (read or write), the ICACHE is bypassed. The AHB transaction is propagated unchanged to the master output port.

The bypass does not increase the latency of the access to the targeted memory.

In the case of a cacheable access, the ICACHE behaves as explained in *Section 11.4.7*.

11.4.7 Cacheable accesses

When the ICACHE receives a cacheable transaction from the GPU, it checks if the address requested is present in its TAG memory, and if the corresponding cache line is valid.

There are then three alternatives:

- The address is present inside the TAG memory, the cache line is valid: **cache hit**, the data is read from the cache and provided to the processor in the same cycle.
- The address is not present in the TAG memory: **cache miss**, the data is read from the main memory and provided to the processor, and a cache line refill is performed.

  The critical-word-first policy ensures minimum wait cycles for the processor, since read data can be provided while the cache still performs a cache line refill (associated latency is the latency of fetching one word from the main memory).

  The burst generated on the ICACHE master bus is WRAPw (w being the cache line width, in words).

  The AHB transaction attributes are also propagated to the main AHB bus matrix on the master port.
- The address is not present in TAG memory, but belongs to the refill burst from the main memory currently ongoing: **cache hit** (hit-under-miss feature).

  This happens during cache-line refill. The ICACHE can provide the requested data as soon as data are available at its master interface, thus avoiding a miss (fetching data from the main memory).

In the case of cache refill (due to cache miss), the ICACHE selects which cache line is written with the refill data:

- In direct map (1-way) mode, only one line can be used to store the refill data: the line pointed by the index of the input address.
- In n-way set associative mode, one line among n can be used (the line pointed by the address index, in each of the n ways). The way selection is based on a pLRU-t replacement algorithm that points, for each index, on the way candidate for the next refill.

If the cache line where the refill data must be written is already valid, the targeted cache line must be invalidated first. This is true whatever the direct map or n-way set associative cache mode.

11.4.8 ICACHE maintenance

The software can invalidate the whole content of the ICACHE by programming CACHEINV in the ICACHE_CR register.

When CACHEINV = 1, the ICACHE control logic sets the BUSYF flag in ICACHE_SR and launches the invalidate cache operation, resetting each TAG valid bit to 0 (one valid bit per cache line). CACHEINV is automatically cleared.

Once the invalidate operation is finished (all valid bits reset to 0), the ICACHE automatically clears BUSYF, and sets BSYENDF in the ICACHE_SR register.

If enabled on this flag condition (BSYENDIE = 1 in ICACHE_IER), the ICACHE interrupt is raised. Then, the (empty) cache is available again.

11.4.9 ICACHE performance monitoring

The ICACHE provides the following monitors for performance analysis:

- The 32-bit hit monitor counts the cacheable AHB-transactions on the slave cache port that hits the ICACHE content.

  It also takes into account all accesses whose address is present in the TAG memory or in the refill buffer (due to a previous miss, and whose data is coming, or is soon to come, from the cache master port) (see *Section 11.4.7*).
- The 16-bit miss monitor counts the cacheable AHB-transactions on the slave cache port that misses the ICACHE content.

  It also takes into account all accesses whose address is not present neither in the TAG memory nor in the refill buffer.

Upon reaching their maximum values, these monitors do not wrap over.

Hit and miss monitors can be enabled and reset by software allowing the analysis of specific pieces of code.

The software can perform the following tasks:

- Enable/stop the hit monitor through HITMEN in ICACHE_CR.
- Reset the hit monitor by setting HITMRST in ICACHE_CR.
- Enable/stop the miss monitor through MISSMEN in ICACHE_CR.
- Reset the miss monitor by setting MISSMRST in ICACHE_CR.

To reduce power consumption, these monitors are disabled (stopped) by default.

11.4.10 ICACHE boot

The ICACHE is disabled (EN = 0 in ICACHE_CR) at boot.

Once the boot is finished, the ICACHE can be enabled (software setting EN = 1 in CACHE_CR).

11.5 ICACHE low-power modes

At device level, using the ICACHE reduces the power consumption by reading textures from the internal ICACHE most of the time, rather than from the bigger and then more power consuming main memories. This reduction is even higher if the cached main memories are external.

Applications with lower performance profile (in terms of hit ratio) and stringent power consumption constraints may benefit from the lower power consumption of an ICACHE configured as direct mapped. This single-way cache configuration is obtained by programming WAYSEL = 0 in ICACHE_CR (see *Figure 12*). The power consumption is reduced by accessing, for each request, only the necessary cut of TAG and data memories. The cache effect still improves memory access performance, even if for most texture accesses, it is less efficient than with an n-way set associative cache mode.

11.6 ICACHE error management and interrupts

If an unsupported cacheable write request is detected (functional error), the ICACHE generates an error by setting the ERRF flag in ICACHE_SR. An interrupt is generated if the corresponding interrupt enable bit is set (ERRIE = 1 in ICACHE_IER).

The other possible interrupt generation is at the end of a cache invalidation operation. When the cache-busy state is finished, the ICACHE sets the BSYENDF flag in ICACHE_SR. An interrupt is generated if the corresponding interrupt enable bit is set (BSYENDIE = 1 in ICACHE_IER).

All ICACHE interrupt sources raise the same and unique interrupt signal, icache_it, and then use the same interrupt vector.

Table 43. ICACHE interrupts

Interrupt vector	Interrupt event	Event flag	Enable control bit	Interrupt clear method
ICACHE	Functional error	ERRF in ICACHE_SR	ERRIE in ICACHE_IER	Set CERRF to 1 in ICACHE_FCR
	End of busy state (invalidate finished)	BSYENDF in ICACHE_SR	BSYENDIE in ICACHE_IER	Set CBSYENDF to 1 in ICACHE_FCR

The ICACHE also propagates all AHB bus errors (such as address decoding issues) from the master1 port back to the slave read port.

11.7 ICACHE registers

11.7.1 ICACHE control register (ICACHE_CR)

Address offset: 0x000

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MISSMRST	HITMRST	MISSMEN	HITMEN
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WAYSEL	CACHEINV	EN
													rw	w	rw

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **MISSMRST**: miss monitor reset
0: release the cache miss monitor reset (needed to enable the counting)
1: reset cache miss monitor

Bit 18 **HITMRST**: hit monitor reset
0: release the cache miss monitor reset (needed to enable the counting)
1: reset cache hit monitor

Bit 17 **MISSMEN**: miss monitor enable
0: cache miss monitor switched off. Stopping the monitor does not reset it.
1: cache miss monitor enabled

Bit 16 **HITMEN**: hit monitor enable
0: cache hit monitor switched off. Stopping the monitor does not reset it.
1: cache hit monitor enabled

Bits 15:3 Reserved, must be kept at reset value.

Bit 2 **WAYSEL**: cache associativity mode selection
This bit allows user to choose ICACHE set-associativity. It can be written by software only when cache is disabled (EN = 0).
0: direct mapped cache (1-way cache)
1: n-way set associative cache (reset value)

Bit 1 **CACHEINV**: cache invalidation
Set by software and cleared by hardware when the BUSYF flag is set (during cache maintenance operation). Writing 0 has no effect.
0: no effect
1: invalidate entire cache (all cache lines valid bit = 0)

Bit 0 **EN**: enable
0: cache disabled
1: cache enabled

11.7.2 ICACHE status register (ICACHE_SR)

Address offset: 0x004

Reset value: 0x0000 0001

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

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

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

Bit 2 **ERRF**: cache error flag
0: no error
1: an error occurred during the operation (cacheable write)

Bit 1 **BSYENDF**: busy end flag
0: cache busy
1: full invalidate CACHEINV operation finished

Bit 0 **BUSYF**: busy flag
0: cache not busy on a CACHEINV operation
1: cache executing a full invalidate CACHEINV operation

11.7.3 ICACHE interrupt enable register (ICACHE_IER)

Address offset: 0x008

Reset value: 0x0000 0000

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

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

Bit 2 **ERRIE**: interrupt enable on cache error

Set by software to enable an interrupt generation in case of cache functional error (cacheable write access)

0: interrupt disabled on error

1: interrupt enabled on error

Bit 1 **BSYENDIE**: interrupt enable on busy end

Set by software to enable an interrupt generation at the end of a cache invalidate operation.

0: interrupt disabled on busy end

1: interrupt enabled on busy end

Bit 0 Reserved, must be kept at reset value.

11.7.4 ICACHE flag clear register (ICACHE_FCR)

Address offset: 0x00C

Reset value: 0x0000 0000

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

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

Bit 2 **CERRF**: clear cache error flag

Set by software.

0: no effect

1: clears ERRF flag in ICACHE_SR

Bit 1 **CBSYENDF**: clear busy end flag

Set by software.

0: no effect

1: clears BSYENDF flag in ICACHE_SR.

Bit 0 Reserved, must be kept at reset value.

11.7.5 ICACHE hit monitor register (ICACHE_HMONR)

Address offset: 0x010

Reset value: 0x0000 0000

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

Bits 31:0 **HITMON[31:0]**: cache hit monitor counter

11.7.6 ICACHE miss monitor register (ICACHE_MMONR)

Address offset: 0x014

Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MISSMON[15:0]**: cache miss monitor counter

11.7.7 ICACHE register map

Table 44. ICACHE register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	ICACHE_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MISSMRST	HITMRST	MISSMEN	HITMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WAYSEL	CACHEINV	EN
	Reset value													0	0	0	0														1	0	0
0x004	ICACHE_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERRF	BSYENDF	BUSYF
	Reset value																														0	0	1
0x008	ICACHE_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERRIE	BSYENDIE	Res.
	Reset value																														0	0	

Table 44. ICACHE register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00C	ICACHE_FCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CERRF	CBSYENDF	Res.
	Reset value																														0	0	
0x010	ICACHE_HMONR	HITMON[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	ICACHE_MMONR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MISSMON[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

12 AXI cache (CACHEAXI)

12.1 CACHEAXI introduction

The AXI cache (CACHEAXI) is introduced on the AXI interconnect driven by the NeuralNetwork peripheral (NPU), to improve the performance of data traffic, by caching the NPU data accessed in the external memories. This feature is available only on STM32N6x7xx devices.

When configured as an SRAM, the CACHEAXI can be accessed by the NPU, and also by the Cortex-M55 processor or by any AXI master peripheral.

12.2 CACHEAXI main features

The main features of the CACHEAXI are described below:

- Bus interface
  - 64-bit AXI slave port (from AXI interconnect, receiving master requests), with independent read and write channels (read address and data channels; write address, data and response channels)
  - 64-bit AXI master port (to AXI interconnect, accessing main memories), with independent read and write channels (read address and data channels; write address, data, and response channels)
  - 32-bit AHB slave port for control (input from AHB peripherals interconnect, for CACHEAXI registers access)
- The CACHEAXI can be configured to behave as an SRAM:
  - Reuse all data SRAM banks (so same size as in cache mode)
  - Dedicated 64-bit AXI slave SRAM port, to access the CACHEAXI in SRAM mode, with independent read and write channels (read address and data channels; write address, data, and response channels)
  - Pipelined AXI accesses with a deep acceptance capability of up to six requests on the read AXI channel and up to five requests on the write AXI channel
- Cache access
  - One read or write access per cycle (priority to write access in case of simultaneous read and write requests)
  - Pipelined AXI accesses with a deep acceptance capability of up to 7 requests on both read and write AXI channels
  - Throughput: 1 read or write hit access served per AXI clock cycle
  - N-ways set-associative cache
  - Supports both write-back and write-through policies (selectable with AXI transaction WT/WB attribute)
  - Supports both allocate and nonallocate policies (selectable with AXI transaction allocate attribute):

    - Read can allocate or not

    - Write-back always allocate

    - Write-through always nonallocate (write-around)
  - Supports byte (8-bit), half-word (16-bit), word (32-bit), or double-word (64-bit) read or write transfer beat sizes
- Replacement and refill
  - pLRU-t replacement policy (pseudo-least-recently-used, based on binary tree), algorithm with best complexity/performance balance
  - Allocation size: on cache miss, refill request size set as cache line size
  - Prefetch hint forwarded to master port to inform downstream logic/memory to prefetch data
  - Two eviction buffers, for missing requests with cache line refill (allocate) not to be blocked by potential cache line eviction operation
- System compartments support:
  - Cache lines tagged by a 3-bit compartment identifier that reflects compartment attribute of the input request on slave port, and that is used on eviction or clean

operation to set compartment attribute at the output master port

- TrustZone security support
- Maintenance operations
  - Cache invalidate: full cache invalidation (fast command, noninterruptible)
  - a Cache clean range: cleans cache lines whose address belongs to programmed range (background task, interruptible)
  - Cache clean and invalidate range: cleans and invalidates cache lines whose address belongs to a programmed range (background task, interruptible)
- Performance counters
  - 2 hit monitor counters (32-bit): number of read hits, number of write hits
  - 2 miss monitor counters (32-bit): number of read misses, number of write misses
  - 2 allocation monitor counters (32-bit): number of read-allocate misses, number of write-allocate misses
  - Write-through monitor counter: number of write-through transactions
  - Eviction monitor counter: number of dirty cache line evictions (on read-allocate misses or on write-allocate misses).
- Error management
  - Propagates the AXI bus error on the AXI master cache port back to the corresponding AXI slave cache port (on noncacheable request, or on refill of cacheable request with allocation that misses).
  - Detects error for master port request initiated by the CACHEAXI itself (a dirty cache line written back into the main memory, because of an eviction or a clean operation), flags this error, and optionally raises an interrupt

12.3 CACHEAXI implementation

In STM32N6x7 devices, the CACHEAXI is a read/write cache placed on the AXI interconnect driven by two master ports of the NeuralNetwork peripheral (NPU).

The CACHEAXI caches the external memory regions accessed through XSPI manager port 1 and port 2, or FMC interfaces.

By placing an interconnect demultiplexing node in front of its slave cache port, the CACHEAXI receives only NPU memory requests that address the external memories (address in range 0x6000 0000:0xDFFF FFFF), and that are cacheable. Noncacheable memory requests are routed away from the slave cache port and address directly the external memories.

Table 45. CACHEAXI features

Features	CACHEAXI1
RW cache	1 (read-write)
SRAM port	1 (support SRAM port)
Cache size	256 Kbytes
Number of ways	8
Cache line width	64 bytes

Table 45. CACHEAXI features (continued)

Features	CACHEAXI1
Data size of AXI Slave and Master interfaces	64 bits
No Privilege	0 (Privilege bit in TAG)
No CID	0 (CID bits in TAG)

12.4 CACHEAXI functional description

The purpose of the CACHEAXI is to cache external memories data, when accessed by a master peripheral through an AXI interconnect.

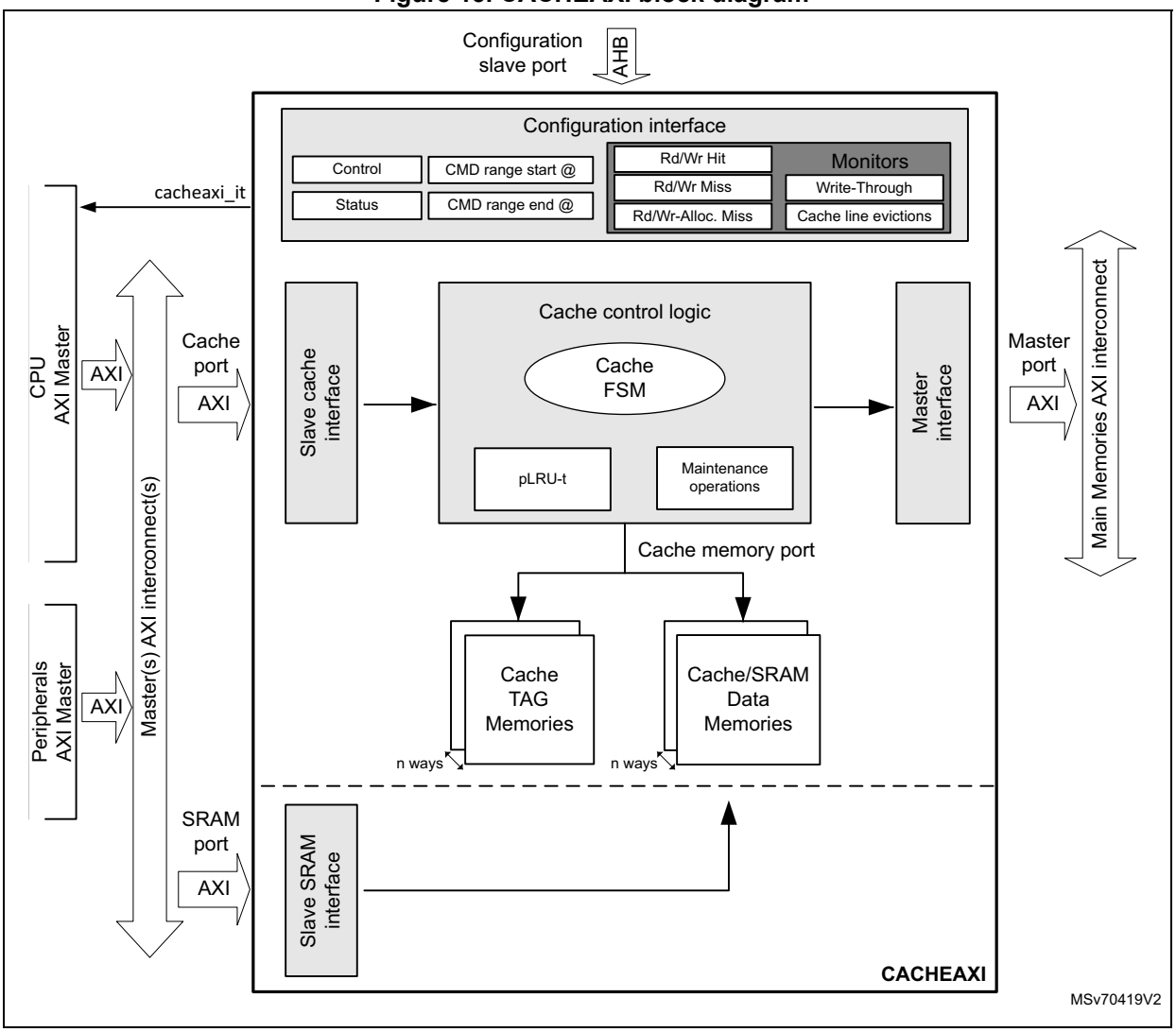
By default (at reset), EN = 0 in CACHEAXI_CR1, and the CACHEAXI behaves as an SRAM.

When its hardware configuration is read-write cache, the CACHEAXI manages both read and write transactions (received on its AXI read slave port, AXI write slave port, respectively).

When configured (by software) to behave as an SRAM, the CACHEAXI manages memory request received on its slave SRAM port (CACHEAXI internal data memories are mapped as a dedicated SRAM region in the system global memory map). Both read and write transactions are supported in SRAM mode.

12.4.1 CACHEAXI block diagram

Figure 13. CACHEAXI block diagram



12.4.2 CACHEAXI reset and clocks

At reset, cache is disabled (EN = 0 in CACHEAXI_CR1), which means that cache mode is disabled.

CACHEAXI transactions on slave cache port are clocked on the (Master) AXI interconnect clock received on slave cache port. The CACHEAXI master port clock is always a copy of the slave cache port clock, in any of the following conditions:

- The cache mode is disabled, and input requests received on slave cache port are just copied to the CACHEAXI master port (including the AXI clock).
- The cache mode is enabled, and the CACHEAXI master port drives the refill requests or the noncacheable transactions (that are forwarded from the slave cache port).

When the CACHEAXI hardware configuration supports the optional SRAM mode and logic, and when the CACHEAXI is disabled, it is configured in SRAM mode:

- The CACHEAXI is clocked on the Master AXI interconnect clock received on slave SRAM port.
- CACHEAXI data memories can be accessed by data request on slave SRAM port.
- The cache mode is disabled, but cache TAG memories are initialized (see invalidate procedure below).

When the CACHEAXI reset signal is released, a cache invalidate procedure is automatically launched, making the CACHEAXI busy (CACHEAXI_SR = 0x0000 0001).

When this procedure is finished:

- The cache control logic and TAG memories are initialized: all cache line valid, dirty, compartment ID and privilege TAG bits = 0.
- CACHEAXI_SR = 0x0000 0002 (reflecting the cache control logic is no longer busy).
- The cache mode is still disabled (EN bit in CACHEAXI_CR1 holds its reset state = 0), but CACHEAXI is ready to be switched in cache mode.

Then, when cache mode is enabled (EN = 1 in CACHEAXI_CR1):

- The CACHEAXI is clocked on the (Master) AXI interconnect clock received on its slave cache port.
- The CACHEAXI is in “cold cache” state, and can serve input requests received on its slave cache port.

When the cache mode is enabled, the SRAM mode is disabled, and input requests received on slave SRAM port are not served: write requests are ignored, and read requests are responded with 0s.

Access to CACHEAXI registers is always clocked by the clock of the AHB configuration slave port.

12.4.3 CACHEAXI TAG memory

The CACHEAXI TAG memory contains:

- Address tags that indicate which data are contained in the cache data memories
- Valid bits
- Dirty bits
- Privilege bits
- Compartment ID (CID) bits

There is one valid bit (V), one privilege bit (P), one 3-bit CID (C[2:0]) and one dirty bit (D) per cache line (per way).

The valid bit enables/disables access to the data cache line: if the line is not valid, the data access (read or write) is performed in the main memory.

The valid bit is set when the cache line is written (allocated by either a read-allocate miss or a write-back miss).

The valid bits are reset in any of the below cases:

- After CACHEAXI reset is released
- When cache mode is disabled (by setting EN low in CACHEAXI_CR1, by software)
- When executing the full invalidate command, when software sets CACHEINV = 1 in CACHEAXI_CR1
- When executing the clean and invalidate range command, when software sets CACHECMD = 0b11 and STARTCMD = 1 in CACHEAXI_CR2.

See *Section 12.4.10: CACHEAXI maintenance* for details on cache maintenance commands.

The dirty bit indicates that the cache line has up-to-date values with respect to the main memory content (in other words, cache has last right value, main memory is not up to date).

The dirty bit is set when the cache line is written by a slave cache port write transaction (only in case of an access with write-back attribute).

The dirty bits are reset in any of the below cases:

- After CACHEAXI reset is released
- When a line refill is performed on a read miss (on a write-back miss, the refilled cache line is modified by the written data and dirty bit = 1)
- When cache invalidation is performed
- When executing one of the CACHEAXI clean operations (cache line written back to the main memory), setting by software CACHECMD = 0b01 or 0b011 in CACHEAXI_CR2

The privilege bit indicates if the data is managed by a privileged entity. It is assigned according to the value of AXI privileged attribute at slave cache port, for the first access to this line (it is written only during a line allocation). The privilege bit holds same polarity as the privileged attribute: 1 for privileged access, 0 for unprivileged access.

The privilege bits are reset:

- After CACHEAXI reset is released
- When the cache is invalidated.

The 3-bit CID indicates to which processing compartment a data belongs to. It is assigned according to the value of AXI user-defined CID attribute at slave cache port for the first access to this line (it is written only during a line allocation).

The CID bits (C[2:0]) are reset:

- After CACHEAXI reset is released
- When the cache is invalidated.

When a cacheable transaction is received at the slave cache port, its AXI transfer address (AxADDR_in[31:0]; either ARADDR_in if on the AXI read channel, or AWADDR_in if on AXI write channel) is split into the following fields (see the table below for B and W values):

- AxADDR_in[B-1:0]: address byte offset, indicates which byte to select inside a cache line
- AxADDR_in[B+W-1:B]: address way index indicates which cache line to select inside each way
- AxADDR_in[31:B+W]: tag address, to be compared to TAG memory address to check if the requested data is already available (meaning valid) inside CACHEAXI

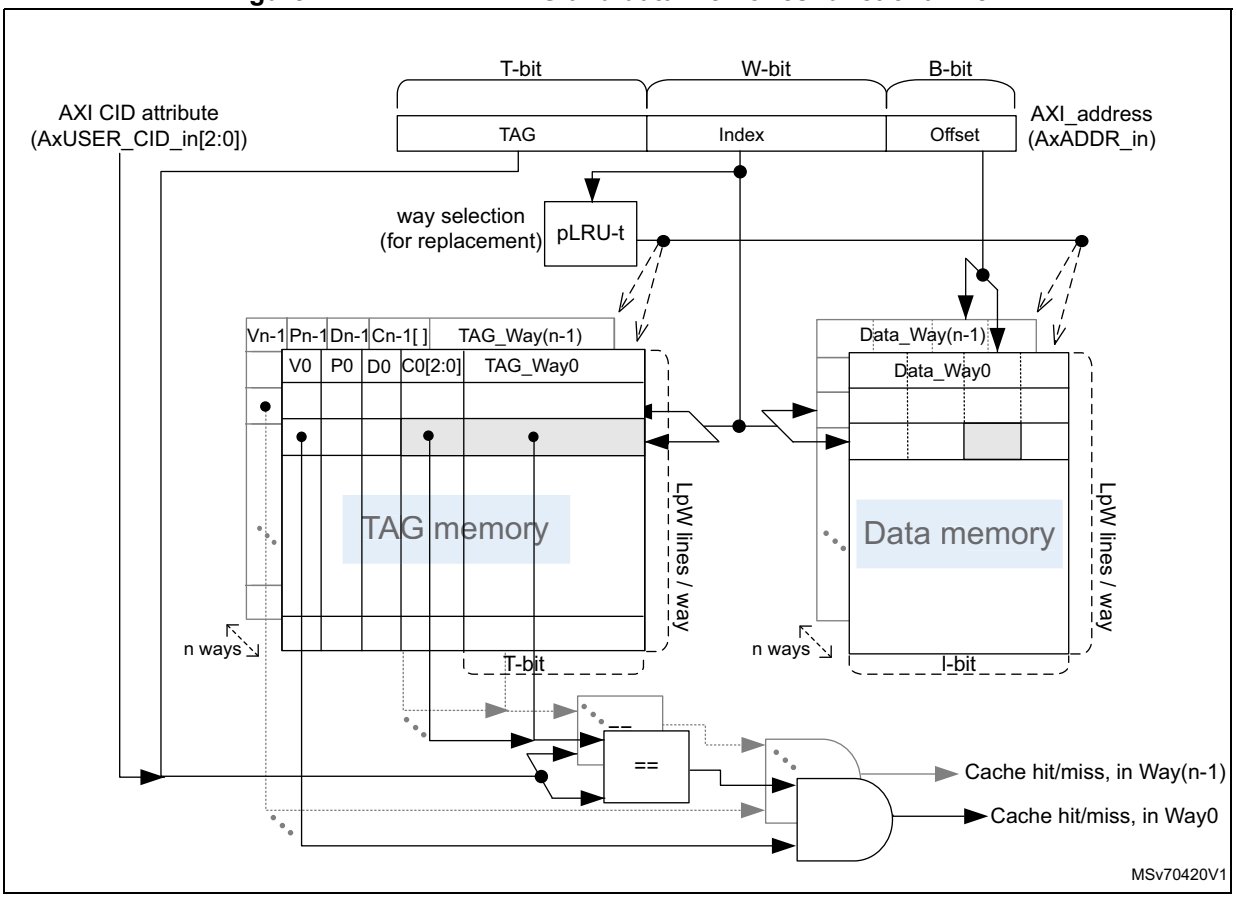
The table below gives the main CACHEAXI parameters for TAG memory dimensioning.

Table 46. TAG memory dimensioning parameters

Parameter	Value	Example
Cache size	S Kbytes = s bytes (s = 1024 x S)	256 Kbytes = 262144 bytes
Cache number of ways	n	8
Cache line size	L-byte = l-bit (l = 8 x L)	64-byte = 512-bit
Number of cache lines (per way)	LpW = s / (n x L) lines/way	512 lines/way
Address byte offset size	B = log2(L) bit	6-bit
Address way index size	W = log2(LpW) bit	9-bit
TAG address size	T = (32 - W - B) bit	17-bit

Figure 14. shows the functional view of TAG and data memories, for an n-way set associative CACHEAXI.

Figure 14. CACHEAXI TAG and data memories functional view



12.4.4 CACHEAXI enable

To activate the CACHEAXI functioning as a cache, EN must be set in CACHEAXI_CR1.

When the CACHEAXI is disabled, it is in SRAM mode, and then bypassed by input requests received on the slave cache port (all these transactions are copied from slave cache port to master port in the same clock cycle, and no comparison is performed with TAG memory).

CACHEAXI is in SRAM mode at boot time (cache mode disabled by default) and can be addressed by any memory request received on slave SRAM port.

12.4.5 Cacheable and noncacheable AXI traffic to slave cache port

The CACHEAXI is developed for AXI bus master peripheral and, when in cache mode, it caches the memory regions addressable by the AXI bus connected to its master port.

The AXI bus traffic to the main memory regions can also be cacheable or noncacheable. An incoming memory request to CACHEAXI slave cache port is defined as cacheable according to its AXI 4-bit memory attribute (cache attribute).

The CACHEAXI supports the allocate or nonallocate policy for both read or write missing transactions. In case of write transaction, the CACHEAXI write policy can be defined as write-through or write-back. Allocation and write policies are defined by the AXI 4-bit memory attribute. This attribute is driven by the master peripheral that initiates the memory request.

Table 47. CACHEAXI supported AXI 4-bit memory/cache attribute

Read or write AXI channel	AXI memory attribute	CACHEAXI behavior
Read	ARCACHE[3:0] = **00**10	Read, noncacheable nonbufferable: no look-up, read from memory (cache bypassed, read forwarded to master port)
	ARCACHE[3:0] = **11**11	Read, cacheable allocate: hit read from cache, miss read from memory, with allocation
	ARCACHE[3:0] = **10**10	Read, cacheable nonallocate: hit read from cache, miss read from memory, without allocation
Write	AWCACHE[3:0] = **00**10	Write, noncacheable nonbufferable: no look-up, write to memory (cache bypassed, write forwarded to master port)
	AWCACHE[3:0] = **11**11	Write-back, cacheable allocate: hit write to cache, miss read-allocate-modify (write data only into allocated cache line)
	AWCACHE[3:0] = **01**10	Write-through, cacheable nonallocate: hit write to cache and to memory, miss write to memory only, without allocation.

In case of noncacheable access, the CACHEAXI is bypassed: the AXI transaction is propagated unchanged from the slave cache port to the master output port, including all the standard AXI4 attributes (except the cache type ones, which are useless for an AXI request to a main memory), and also the user-defined attributes.

- A noncacheable read transaction on the AXI read channel (of slave cache port) is forwarded to the AXI read channel (of master port).
- A noncacheable write transaction on the AXI write channel (of slave cache port) is forwarded to the AXI write channel (of master port).

The bypass does not increase the latency of the access to the targeted memory.

AXI attribute signals are set by the master peripheral that initiates the AXI memory request.

In case of cacheable access on slave cache port, the CACHEAXI behaves as explained in the next section.

12.4.6 Cacheable accesses

When the CACHEAXI receives a cacheable transaction from the AXI master peripheral on its slave cache port, it checks if the requested address and the compartment ID are present in its TAG memory, and if the corresponding cache line is valid. Regarding the hit/miss check, the compartment ID can be seen as a 3-bit extension of the address. In the next paragraphs, the address considered for the comparison with the TAG addresses is made of both the address itself and of the CID: the address matches if both the actual address and the CID match; the address mismatches if one of them mismatches.

For **read transaction**, the following alternatives exist:

- The address matches the one of a valid cache line TAG: **cache read hit**, the data is read from the cache and provided to the master peripheral.
- The address does not match a valid cache line TAG: **cache read miss**, the data is read from the main memory and provided to the master peripheral.

  Depending on the read transaction allocate attribute, a cache line refill is performed, or not. If the allocation must be done, the read request to main memory has the cache line size. If not, the read request to main memory has simply the size of the input request (propagated from slave cache port to master port).

  The AXI protection attribute and CID/prefetch user-defined attributes are also propagated from the slave cache port to the master port, for the (nonallocated) simple data access or cache line refill request.

  Note: The first read transaction that misses, blocks the cache slave port: no further cacheable transaction makes a lookup in cache lines TAG until missing read is served by the CACHEAXI (data read, and if a read-allocate, cache line refilled).

For **write-back transaction** (write transaction, with cacheable write-back allocate attributes, the following alternatives exist:

- The address matches the one of a valid cache line TAG: **cache write-back hit**, the data is written in cache (in the matching cache line).
- The address does not match a valid cache line TAG: **cache write-back miss**.

  A line allocation is performed by reading the entire cache line data from main memory.

  The AXI protection attribute and the user-defined attribute for the cache line refill request are propagated from the slave cache port to the master port.

  Once the refilled line has been written in the CACHEAXI, the initial data provided on slave cache port is written in the CACHEAXI (it overwrites a part of the cache line that was refilled just before).

  Note: The first write-back transaction that misses, blocks the cache slave port: no further cacheable transaction makes a lookup in cache lines TAG until missing write-back is served by the CACHEAXI (cache line refilled and data written).

For **write-through transaction** (write transaction, with cacheable write-through nonallocate attributes, the following alternatives exist:

- Address matches the one of a valid cache line TAG: **cache write-through hit**, the data is written both in cache and in main memory (through master port).
- Address does not match a valid cache line TAG: **cache write-through miss**, the data incoming at slave cache port is written only in main memory (unlike the write-back miss, there is no line allocation and data is not written in cache).

In case of cache refill (due to cache miss), the CACHEAXI selects which cache line is written with the refill data: as an n-way set associative cache, one line among n can be used (the line pointed by the address index, in each of the n ways). The way selection is based on a pLRU-t replacement algorithm. This algorithm points, for each index, on the way candidate for the next refill.

If ever the cache line where the refill data must be written is already valid, the targeted cache line must be evicted first:

- If the dirty tag of this line equals 0 (a clean cache line), the line is simply invalidated.
- If the dirty tag of this line equals 1 (a dirty cache line), the evicted cache line is written into an eviction buffer, before to be written back in the main memory (see *Section 12.4.7*).

  The CACHEAXI generates an AXI burst write transaction on its master port, with burst type set to INCR and with AXI memory transaction attribute signals set as detailed below:

  – privileged = evicted line TAG privilege bit (P)
  – data (not instruction)
  – CID[2:0] = evicted line TAG CID bits (C[2:0])

  These AXI attributes cannot be propagated from the slave port (as it is the case for all other transactions emitted on CACHEAXI master port) because the evicting transaction has no relation with the initial missing transaction (that made the refill of the cache line to evict).

12.4.7 AXI traffic to master port

To improve the throughput of the refill traffic generated by the CACHEAXI on its master port (to improve global CACHEAXI performance), the CACHEAXI implements the following feature:

- Prefetch hint forwarded from slave read cache port to master port to inform downstream logic/memory to prefetch extra data from main memory (in addition to the ones requested by this master port refill request)

In order to improve CACHEAXI eviction performance, the following feature is implemented:

- An eviction buffer receives the AXI write transaction that corresponds to the dirty cache line evicted from cache. When either a cache read-with-allocation, or a cache write-back request misses, it initiates a cache line refill operation, that potentially implies the eviction of a dirty cache line (when the n ways of the cache line index to be used for the refill are already all valid).

  The eviction buffer allows the CACHEAXI not to block a missing request beyond the occurrence of the refill operation, and not to wait for the evicted cache line operation (write to main memory) to be performed on the CACHEAXI master port. The eviction operation is just sent to the eviction buffer, that initiates it on master port in the background (when the master port is available for that).

12.4.8 AXI traffic to slave SRAM port

When the CACHEAXI is in SRAM mode it addresses 256 Kbytes SRAM area (SRAM7). This region is mapped in the system global memory map, and the SRAM size equals the CACHEAXI (cache) size.

When in SRAM mode (CACHEAXI_CR1.EN = 0), the CACHEAXI SRAM can be addressed by the Cortex core or other AXI bus master peripheral on its dedicated slave SRAM port (hardware configuration option).

This AXI slave SRAM port has independent read and write channels (read address and data channels; write address, data and response channels), as the AXI slave cache port.

The read or write AXI transaction received on AXI slave SRAM port has the following characteristics:

- The relevant address field is restricted to the CACHEAXI size (for instance, only the 17 last significant bits of the address are relevant to address a 128-Kbyte CACHEAXI).
- Byte (8-bit), half-word (16-bit), word (32-bit) or double-word (64-bit) transfer beat size.
- No cache type attribute (useless to access an SRAM).
- No support of AXI standard protection attributes, and no support of AXI user-defined attributes (since they are filtered, upstream of AXI slave SRAM port).

12.4.9 CACHEAXI security

The CACHEAXI implements an Armv8-M TrustZone.

CACHEAXI configuration registers are protected at system level.

12.4.10 CACHEAXI maintenance

The CACHEAXI features several maintenance operations that the software can program in CACHEAXI_CR1 and CACHEAXI_CR2 control registers:

- **Full invalidate**: invalidates the whole cache, noninterruptible task.

  The software can invalidate the whole CACHEAXI content by programming CACHEINV in CACHEAXI_CR1.

  When CACHEINV is set, the CACHEAXI control logic sets the BUSYF flag in CACHEAXI_SR, and performs the operation of cache invalidation, in each cache line TAG, reseting:

  – the valid bit (V = 0)
  – the dirty bit (D = 0)
  – the privilege bit (P = 0)
  – the compartment ID bits (C[2:0] = 000)

  to prevent unknown values at next cache line validation.

  CACHEINV in CACHEAXI_CR1 is automatically cleared.

  Once the full invalidate operation is finished, the CACHEAXI automatically clears BUSYF and sets BSYENDF in CACHEAXI_SR.

  If enabled on this flag condition (BSYENDIE = 1 in CACHEAXI_IER), the CACHEAXI interrupt is raised, on **cacheaxi_it** signal. The (empty) cache is available again.

  This full invalidate operation is not interruptible, meaning that the cache does not treat any cacheable request while BUSYF is set (the incoming request is blocked until the full invalidate is finished). However, the noncacheable traffic is treated (since the

request address is not compared to TAG ones) as the CACHEAXI is bypassed in the same clock cycle (same behavior as when CACHEAXI cache mode is disabled).

- **Clean range**: cleans a certain range of addresses in the cache, background task (interruptible).

  Cleaning a cache line means making sure that main memory content is up-to-date with the data which may have been modified in cache. The clean operation consists in performing the write-back in main memory of the cache lines that are tagged as “dirty” (the ones with TAG dirty bit set).

  The software can clean a given data region in the CACHEAXI by programming STARTCMD = 1, and CACHECMD = 0b01 in CACHEAXI_CR2, after the address range was programmed into CACHEAXI_CMDRSADDRR (range start address) and CACHEAXI_CMDREADDRR (range end address).

  The CACHEAXI control logic then parses the whole TAG memory. If the read line address (TAG address + line index) falls in the programmed address range (CACHEAXI_CMDRSADDRR ≤ Line Addr ≤ CACHEAXI_CMDREADDRR) and the corresponding line is dirty, this line is cleaned: the whole cache line is written-back in memory through the CACHEAXI master port, and its TAG dirty bit is cleared.

  When STARTCMD is set, the CACHEAXI control logic sets BUSYCMDF in CACHEAXI_SR, and launches the clean range operation. STARTCMD in CACHEAXI_CR2 is also automatically cleared.

  Once the operation is finished (all TAG memory parsed), the CACHEAXI automatically clears BUSYCMDF and sets CMDENDF in CACHEAXI_SR.

  If enabled on this flag condition (CMDENDIE = 1 in CACHEAXI_IER), the CACHEAXI interrupt is raised, on **cacheaxi_it** signal.

  During this clean range operation, the CACHEAXI is interruptible: it can accept new incoming requests that take higher priority than the cleaning process. The TAG memory is accessed for clean range operation only if not already accessed by an external cache request. This implies that clean range execution is usually not performed in one go, but can be interrupted.

  It is under the software responsibility that no bus initiator attempts to change the content of the region being cleaned until the clean range is completed. The software must take advantage of BUSYCMDF in CACHEAXI_SR, and polls this flag to prevent any spurious access to the area being cleaned. Alternatively the software can also rely on the command end flag (CMDENDF) or on the CACHEAXI interrupt to detect the end of the clean range execution.

- **Clean and invalidate range**: cleans and invalidates a certain range of addresses in the cache, background task (interruptible).

  This operation cleans the “dirty” cache lines that belong to the operation address range (the same as clean range operation), and also invalidates all the (valid) cache lines that belong to this address range (whether they are dirty or not).

  Note: When a cache line is invalidated, the pLRU-t pointer (for this cache line index) is updated to point to the way that was just invalidated (best candidate for the next allocation).

  The software can launch this clean and invalidate range operation, by programming STARTCMD = 1, and CACHECMD = 0b11 in CACHEAXI_CR2, after the address range was programmed into CACHEAXI_CMDRSADDRR (range start address) and CACHEAXI_CMDREADDRR (range end address).

  This sets and clears the same flags, and potentially the same interrupt as clean range operation.

12.4.11 CACHEAXI performance monitoring

The CACHEAXI provides the following monitors for performance analysis:

- The read-hit monitor counts the cacheable AXI read transactions on the slave cache port that hit the CACHEAXI content.
- The write-hit monitor counts the cacheable AXI write transactions on slave cache port that hit the CACHEAXI content.
- The read-miss monitor counts the cacheable AXI read transactions on slave cache port that miss the CACHEAXI content.
- The write-miss monitor counts the cacheable AXI write transactions on slave cache port that miss the CACHEAXI content.

  The hit and miss monitors above actually count the number of lookups (in the CACHEAXI TAG memory) implied by the AXI transactions:

  – when an AXI burst transaction results in several data transfers (or beats) that target the same cache line address range, only one lookup occurs, and the hit or miss monitor only increments by 1.

  – when the data transfers (or beats) of a given AXI burst target several cache line address ranges, the hit/miss monitor increment globally by the number of looked-up cache lines (some can hit, others can miss).
- The read-allocate-miss monitor counts the cache line refill operations (main memory reads) implied by cacheable AXI read-allocate transactions that miss.
- The write-allocate-miss monitor counts the cache line refill operations (main memory reads) implied by cacheable AXI write-back transactions, that miss (and then allocate).
- The write-through monitor counts the total amount of main memory write operations implied by cacheable AXI write-through transactions (whether they hit or miss).
- The eviction monitor counts:

  – the total amount of (dirty) cache lines evicted by the refill operations resulting from either read-allocate misses or write-allocate (write-back) misses,

  – and also, the amount of cache lines ‘evicted’ by a clean operation, meaning the number of dirty lines with an address included in the address range specified by a given 'clean range' or 'clean and invalidate range' maintenance operation, and that are dirty.

  Only dirty cache lines written back to main memory are counted; eviction of non dirty cache lines, that are simply invalidated, are not counted.

The monitors count is effective only when the CACHEAXI is enabled (in cache mode).

Noncacheable read or write transactions are not counted.

Upon reaching their maximum values, the monitors do not wrap over.

The software can perform the following tasks:

- Enable/stop the read (write) hit monitor, through R(W)HITMEN in CACHEAXI_CR1.
- Reset the read (write) hit monitor, by setting R(W)HITMRST in CACHEAXI_CR1.
- Enable/stop the read (write) miss monitor, through R(W)MISSMEN in CACHEAXI_CR1.
- Reset the read (write) miss monitor, by setting R(W)MISSMRST in CACHEAXI_CR1.
- Enable/stop the read-allocate (write-allocate) miss monitor, through R(W)AMMEN in CACHEAXI_CR1.
- Reset the read-allocate (write-allocate) miss monitor, by setting R(W)AMMRST in CACHEAXI_CR1.
- Enable/stop the write-through monitor, through WTMEN in CACHEAXI_CR1.
- Reset the write-through monitor, by setting WTMRST in CACHEAXI_CR1.
- Enable/stop the eviction monitor, through EVIMEN in CACHEAXI_CR1.
- Reset the eviction monitor, by setting EVIMRST in CACHEAXI_CR1.

To reduce power consumption, these monitors are disabled (stopped) by default.

12.4.12 CACHEAXI boot

The CACHEAXI is disabled (EN = 0 in CACHEAXI_CR1) at boot: cache mode is disabled and the CACHEAXI behaves as an SRAM.

Once the boot is finished, the CACHEAXI cache mode can be enabled (by setting EN in CACHEAXI_CR1).

12.5 CACHEAXI low-power modes

Using the CACHEAXI reduces the power consumption at device level by, most of the time, loading/storing data from/to the internal CACHEAXI rather than from the bigger main memories (more power consuming). This reduction is even much higher when the cached main memories are external.

At CACHEAXI level, the access to a given cache line is pipelined: the TAG memory is accessed first (for lookup operation to determine if the access misses all the ways, or hits a given way). In case of hit, the data memory is accessed in a second cycle, and only the data memory cut holding the hitting way is accessed. Accessing only one way among all the n ways available reduces CACHEAXI power consumption to the minimum.

12.6 CACHEAXI error management and interrupts

A transaction initiated on the CACHEAXI master port may return an error (a write attempt into a read-only memory, a security issue or an address decoding issue, for instance).

If the master port request was propagated from a slave cache port request (non-cacheable request, or cacheable request with allocation that misses), the CACHEAXI propagates the AXI bus error from the AXI master cache port back to the corresponding AXI slave cache port.

If ever the master port request is initiated by the CACHEAXI itself (dirty cache line written-back into the main memory because of an eviction or a clean operation), the CACHEAXI

receives this functional error, and flags it internally by setting ERRF in CACHEAXI_SR. And an interrupt is generated if the corresponding interrupt enable bit is set (ERRIE = 1 in CACHEAXI_IER).

Another case of interrupt generation is at the end of a full invalidate operation: when the cache busy state is finished, the CACHEAXI sets BSYENDF in CACHEAXI_SR. An interrupt is then generated if the corresponding interrupt enable bit is set (BSYENDIE = 1 in CACHEAXI_IER).

The last case is at the end of a maintenance range operation (clean/invalidate range): when the command busy state is finished, the CACHEAXI sets CMDENDF in CACHEAXI_SR. An interrupt is generated if the corresponding interrupt enable bit is set (CMDENDIE = 1 in CACHEAXI_IER).

The CACHEAXI has a unique interrupt signal, **cacheaxi_it** (and then use the same interrupt vector whatever the interrupt source).

Table 48. CACHEAXI interrupts

Interrupt vector	Interrupt event	Event flag	Enable control bit	Interrupt clear method
CACHEAXI	Functional error	ERRF flag in CACHEAXI_SR	ERRIE bit in CACHEAXI_IER	Set CERRF bit to 1 in CACHEAXI_FCR
CACHEAXI	End of busy state (full invalidate finished)	BSYENDF flag in CACHEAXI_SR	BSYENDIE bit in CACHEAXI_IER	Set CBSYENDF bit to 1 in CACHEAXI_FCR
CACHEAXI	End of cache range operation (address range based)	CMDENDF flag in CACHEAXI_SR	CMDENDIE bit in CACHEAXI_IER	Set CCMDENDF bit to 1 in CACHEAXI_FCR

12.7 CACHEAXI registers

12.7.1 CACHEAXI control register 1 (CACHEAXI_CR1)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EVIMRST	WTMRST	EVIMEN	WTMEN	WAMMRST	RAMMRST	WAMMEN	RAMMEN	WMISSMRST	WHITMRST	WMISSMEN	WHITMEN	RMISSMRST	RHITMRST	RMISSMEN	RHITMEN
w	w	rw	rw	w	w	rw	rw	w	w	rw	rw	w	w	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CACHEINV	EN
														w	rw

Bit 31 **EVIMRST:** eviction monitor reset
0: no effect
1: reset cache eviction monitor

Bit 30 **WTMRST:** write-through monitor reset
0: no effect
1: reset cache write-through monitor

Bit 29 **EVIMEN:** eviction monitor enable
0: cache eviction monitor switched off. Stopping the monitor does not reset it.
1: cache eviction monitor enabled

Bit 28 **WTMEN:** write-through monitor enable
0: cache write-through monitor switched off. Stopping the monitor does not reset it.
1: cache write-through monitor enabled

Bit 27 **WAMMRST:** write-allocate miss monitor reset
0: no effect
1: reset cache write-allocate miss monitor

Bit 26 **RAMMRST:** read-allocate miss monitor reset
0: no effect
1: reset cache read-allocate miss monitor

Bit 25 **WAMMEN:** write-allocate miss monitor enable
0: cache write-allocate miss monitor switched off. Stopping the monitor does not reset it.
1: cache write-allocate miss monitor enabled

Bit 24 **RAMMEN:** read-allocate miss monitor enable
0: cache read-allocate miss monitor switched off. Stopping the monitor does not reset it.
1: cache read-allocate miss monitor enabled

Bit 23 **WMISSMRST:** write-miss monitor reset
0: no effect
1: reset cache write-miss monitor

Bit 22 **WHITMRST:** write-hit monitor reset
0: no effect
1: reset cache write-hit monitor

Bit 21 **WMISSMEN:** write-miss monitor enable
0: cache write-miss monitor switched off. Stopping the monitor does not reset it.
1: cache write-miss monitor enabled

Bit 20 **WHITMEN:** write-hit monitor enable
0: cache write-hit monitor switched off. Stopping the monitor does not reset it.
1: cache write-hit monitor enabled

Bit 19 **RMISSMRST:** read-miss monitor reset
0: no effect
1: reset cache read-miss monitor

Bit 18 **RHITMRST:** read-hit monitor reset
0: no effect
1: reset cache read-hit monitor

Bit 17 **RMISSMEN:** read-miss monitor enable
0: cache read-miss monitor switched off. Stopping the monitor does not reset it.
1: cache read-miss monitor enabled

Bit 16 **RHITMEN:** read-hit monitor enable
0: cache read-hit monitor switched off. Stopping the monitor does not reset it.
1: cache read-hit monitor enabled

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

Bit 1 **CACHEINV:** full cache invalidation
Can be set by software, only when EN = 1.
Cleared by hardware when the BUSYF flag is set (during full cache invalidation operation).
Writing 0 has no effect.
0: no effect
1: invalidate entire cache (all cache lines valid bit = 0)

Bit 0 **EN:** enable
0: cache mode disabled (cache port bypassed and SRAM7 port active)
1: cache mode enabled (SRAM7 port not active)

12.7.2 CACHEAXI status register (CACHEAXI_SR)

Address offset: 0x004

Reset value: 0x0000 0001

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

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

Bit 4 **CMDENDF:** command end flag
Cleared by writing CACHEAXI_FCR.CCMDENDF = 1.
0: cache busy or in idle
1: CACHECMD command finished

Bit 3 **BUSYCMDF:** command busy flag
0: cache not busy on a CACHECMD command
1: cache busy on a CACHECMD command (clean or clean-and-invalidate an address range)

Bit 2 **ERRF:** cache error flag
Cleared by writing CACHEAXI_FCR.CERRF = 1.
0: no error
1: an error occurred during the operation (eviction or clean operation write-back error).

Bit 1 **BSYENDF:** full invalidate busy end flag
Cleared by writing CACHEAXI_FCR.CBSYENDF = 1.
0: cache busy or in idle
1: full invalidate CACHEINV operation finished

Bit 0 **BUSYF:** full invalidate busy flag
0: cache not busy on a CACHEINV operation
1: cache executing a full invalidate CACHEINV operation

12.7.3 CACHEAXI interrupt enable register (CACHEAXI_IER)

Address offset: 0x008

Reset value: 0x0000 0000

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

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

Bit 4 **CMDENDIE**: interrupt enable on command end
Set by software to enable an interrupt generation at the end of a cache command (clean or clean-and-invalidate an address range)
0: interrupt disabled on command end
1: interrupt enabled on command end

Bit 3 Reserved, must be kept at reset value.

Bit 2 **ERRIE:** interrupt enable on cache error
Set by software to enable an interrupt generation in case of cache functional error (eviction or clean operation write-back error)
0: interrupt disabled on error
1: interrupt enabled on error

Bit 1 **BSYENDIE:** interrupt enable on busy end
Set by SW to enable an interrupt generation at the end of a cache full invalidate operation.
0: Interrupt disabled on busy end
1: Interrupt enabled on busy end

Bit 0 Reserved, must be kept at reset value.

12.7.4 CACHEAXI flag clear register (CACHEAXI_FCR)

Address offset: 0x00C

Reset value: 0x0000 0000

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

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

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

Bit 4 **CCMDENDF:** clear command end flag
Set by software.
0: no effect
1: clears CMDENDF flag in CACHEAXI_SR.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CERRF:** clear cache error flag
Set by software.
0: no effect
1: clears ERRF flag in CACHEAXI_SR.

Bit 1 **CBSYENDF:** clear full invalidate busy end flag
Set by software.
0: no effect
1: clears BSYENDF flag in CACHEAXI_SR.

Bit 0 Reserved, must be kept at reset value.

12.7.5 CACHEAXI read-hit monitor register (CACHEAXI_RHMONR)

Address offset: 0x010

Reset value: 0x0000 0000

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

Bits 31:0 **RHITMON[31:0]:** cache read-hit monitor counter

12.7.6 CACHEAXI read-miss monitor register (CACHEAXI_RMMONR)

Address offset: 0x014

Reset value: 0x0000 0000

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

Bits 31:0 **RMISSMON[31:0]:** cache read-miss monitor counter

12.7.7 CACHEAXI read-allocate miss monitor register (CACHEAXI_RAMMONR)

Address offset: 0x018

Reset value: 0x0000 0000

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

Bits 31:0 **RAMMON[31:0]:** cache read-allocate miss monitor counter

12.7.8 CACHEAXI eviction monitor register (CACHEAXI_EVIMONR)

Address offset: 0x01C

Reset value: 0x0000 0000

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

Bits 31:0 **EVIMON[31:0]:** cache eviction monitor counter

12.7.9 CACHEAXI write-hit monitor register (CACHEAXI_WHMONR)

Address offset: 0x020

Reset value: 0x0000 0000

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

Bits 31:0 **WHITMON[31:0]:** cache write-hit monitor counter

12.7.10 CACHEAXI write-miss monitor register (CACHEAXI_WMMONR)

Address offset: 0x024

Reset value: 0x0000 0000

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

Bits 31:0 **WMISSMON[31:0]:** cache write-miss monitor counter

12.7.11 CACHEAXI write-allocate miss monitor register (CACHEAXI_WAMMONR)

Address offset: 0x028

Reset value: 0x0000 0000

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

Bits 31:0 **WAMMON[31:0]:** cache write-allocate miss monitor counter

12.7.12 CACHEAXI write-through monitor register (CACHEAXI_WTMONR)

Address offset: 0x02C

Reset value: 0x0000 0000

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

Bits 31:0 **WTMON[31:0]:** cache write-through monitor counter

12.7.13 CACHEAXI control register 2 (CACHEAXI_CR2)

Address offset: 0x100

Reset value: 0x0000 0000

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

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

Bits 2:1 **CACHECMD[1:0]:** cache command maintenance operation (clean or clean-and-invalidate an address range)

This field can be set and cleared by software, only when no maintenance command is ongoing (BUSYCMDF = 0).

00: no operation

01: clean range

11: clean and invalidate range

others: reserved

Bit 0 **STARTCMD:** starts maintenance range command (maintenance operation defined in CACHECMD).

This bit can be set by software, only when EN = 1, BUSYCMDF = 0, BUSYF = 0, and CACHECMD[1:0] = 0b01 or 0b11.

It is cleared by hardware when BUSYCMDF is set (during cache maintenance operation). Writing 0 to this bit has no effect.

0: command operation (cache maintenance) finished

1: start maintenance command (cache maintenance)

12.7.14 CACHEAXI command range start address register (CACHEAXI_CMDRSADDRR)

Address offset: 0x104

Reset value: 0x0000 0000

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

Bits 31:6 **CMDSTARTADDR[31:6]:** start address of range to which the cache maintenance command specified in CACHEAXI_CR2.CACHECMD field applies

This field must be set before CACHEAXI_CR2.CACHECMD is written.

Bits 5:0 Reserved, must be kept at reset value.

12.7.15 CACHEAXI command range end address register (CACHEAXI_CMDREADDRR)

Address offset: 0x108

Reset value: 0x0000 0000

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

Bits 31:6 **CMDENDADDR[31:6]:** end address of range to which the cache maintenance command specified in CACHEAXI_CR2.CACHECMD field applies

This field must be set before CACHEAXI_CR2.CACHECMD is written.

Bits 5:0 Reserved, must be kept at reset value.

12.7.16 CACHEAXI register map

Table 49. CACHEAXI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	CACHEAXI_CR1	EVIMRST	WTMRST	EVIMEN	WTMEN	WAMMRST	RAMMRST	WAMMEN	RAMMEN	WMISSMRST	WHITMRST	WMISSMEN	WHITMEN	RMISSMRST	RHITMRST	RMISSMEN	RHITMEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CACHEINV	EN
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0															0	0
0x004	CACHEAXI_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMDENDF	BUSYCMDF	ERRF	BSYENDF	BUSYF
	Reset value																												0	0	0	0	1
0x008	CACHEAXI_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMDENDIE	Res.	ERRIE	BSYENDIE	Res.
	Reset value																												0		0	0	
0x00C	CACHEAXI_FCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCMDENDF	Res.	CERRF	CBSYENDF	Res.
	Reset value																												0		0	0	
0x010	CACHEAXI_RHMONR	RHITMON[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	CACHEAXI_RMMONR	RMISSMON[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	CACHEAXI_RAMMONR	RAMMON[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 49. CACHEAXI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x01C	CACHEAXI_ EVIMONR	EVIMON[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	CACHEAXI_ WHMONR	WHITMON[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	CACHEAXI_ WMMONR	WMISSMON[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	CACHEAXI_ WAMMONR	WAMMON[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	CACHEAXI_ WTMONR	WTMON[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030-0x0FC	Reserved	Reserved																															
0x100	CACHEAXI_ CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CACHECMD [1:0]		STARTCMD
	Reset value																														0	0	0
0x104	CACHEAXI_ CMDRS ADDRR	CMDSTARTADDR[31:5](1)																											Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
0x108	CACHEAXI_ CMDRE ADDRR	CMDENDADDR[31:5](1)																											Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					

1. Bit 5 is reserved (=0) for CACHEAXI configuration with 64-bytes cache line width.

Refer to *Section 2.3* for the register boundary addresses.

13 Power control (PWR)

13.1 PWR introduction

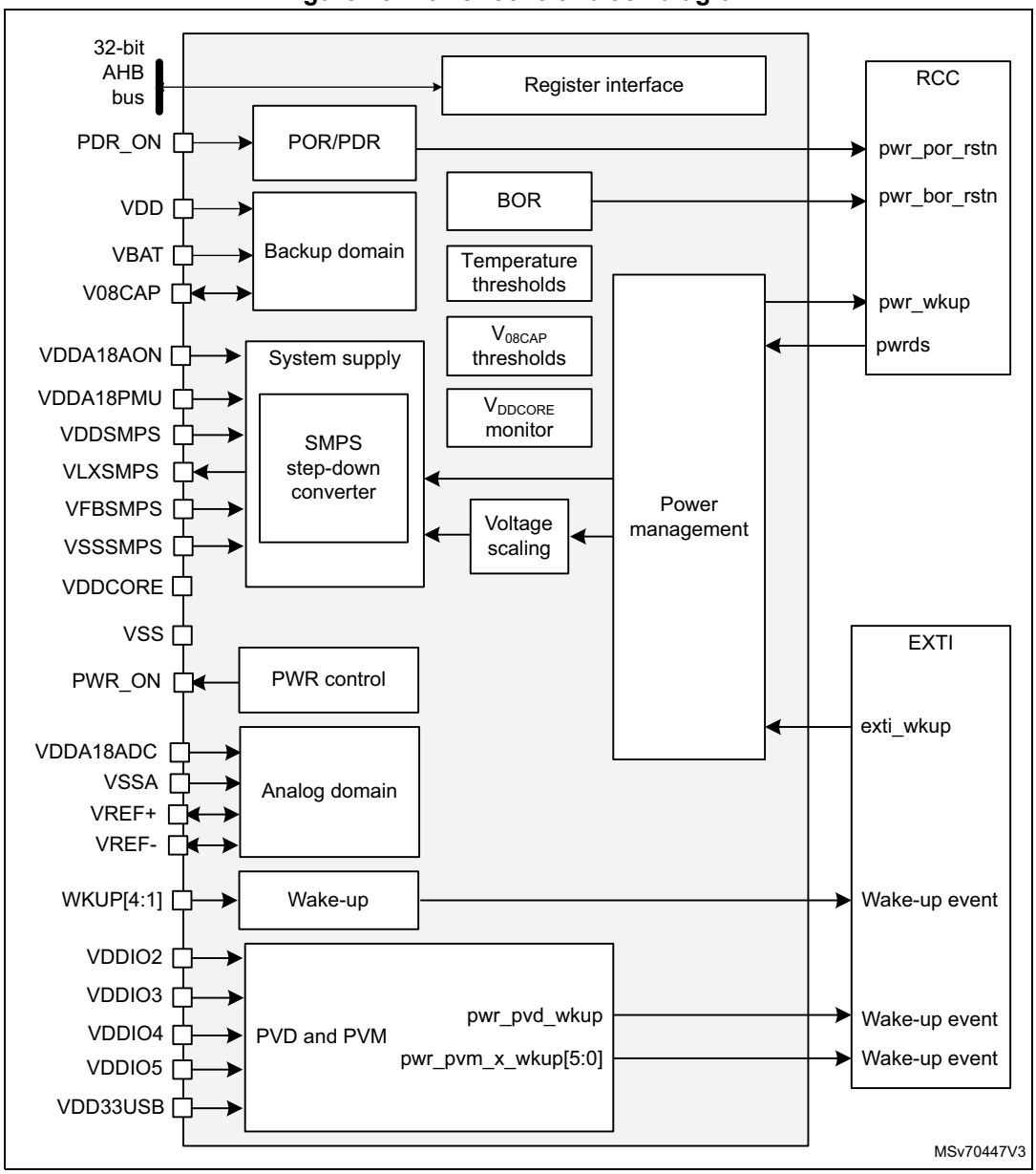
This section gives an overview of the supply architecture for the different power domains and of the supply configuration controller. It also describes the features of the power-supply supervisors, and explains how the V_{CORE} supply domain is configured, depending upon the operating modes, the selected performance (clock frequency), and the voltage scaling.

13.2 PWR main features

- Power supplies and supply domains
  - Core domains (V_{CORE} = V_{DDCORE})
  - V_{DD} domain (V_{RET})
  - Backup domain (V_{SW}, V_{BKP})
  - Analog domain ($V_{DDA18ADC}$)
- System supply voltage regulation
  - Switched-mode power supply power-efficient voltage down-converter (SMPS step-down converter)
  - 0v8 backup regulator (external to the PWR)
- Power supply supervision
  - POR/PDR monitor
  - BOR monitor
  - VDDA18PMU monitor
  - PVD monitor
  - PVM monitor (V_{DDIO2}, V_{DDIO3}, V_{DDIO4}, V_{DDIO5}, $V_{DD33USB}$, $V_{DDA18ADC}$)
  - V_{08CAP} thresholds
  - Temperature thresholds
  - V_{DDCORE} monitor
- Power management
  - Operating modes
  - Voltage scaling control
  - Low-power modes

13.3 PWR block diagram

Figure 15. Power control block diagram



13.3.1 PWR pins and internal signals

Table 50 lists the PWR inputs and output signals connected to package pins or balls, while *Table 51* shows the internal PWR signals.

Table 50. PWR input/output signals connected to package pins or balls

Pin name	Signal type	Description
VDD	Input	Main I/O and V_{DD} domain supply input
VDDA18AON	Input	Main analog supply input
VBAT	Input	Backup battery supply input
VDDA18PMU	Input	Step-down converter analog supply input
VDDSMPS	Input	Step-down converter supply input
VLXSMPS	Output	Step-down converter supply output
VFBSMPS	Input	Step-down converter feedback voltage sense
VSSSMPS	Input	Step-down converter ground
V08CAP	Input/output	Digital backup domain supply
VDDCORE	Input	Core domain supply
VDDA18ADC	Input	ADC analog supply
VREF+, VREF-	Input/output	External reference voltage for ADCs
VDDIO2	Input	Independent I/O supply 2 (PO[5:0] and PP[15:0]), usually for XSPI1M_P1 (XSPI)
VDDIO3	Input	Independent I/O supply 3 (PN[12:0])., usually for XSPI1M_P2 (XSPI)
VDDIO4	Input	Independent I/O supply 4 (PC[1], PC[12:6] and PH[2,9]), usually for eMMC
VDDIO5	Input	Independent I/O supply 5 (PC[0], PC[5:2] and PE[4]), usually for SD-Card.
VDD33USB	Input	USB HS PHYs and USB Type C PHY 3V3 supply input
VDDA18USB	Input	USB HS PHYs analog supply input
VDDA18PLL	Input	PLL analog supply
VDDCSI	Input	CSI PHY digital supply input
VDDA18CSI	Input	CSI PHY analog 1v8 supply input
VSS	Input	Main ground
VSSA	Input	Analog ground
PWR_ON	Output	Step-down converter or core supply enable output
NRST(1)	Reset	System reset, can be used to provide reset to external devices
PDR_ON	Input	Power-down reset enable
WKUPx	Input	Wake-up pins
PWR_CSLEEP	Output	MCU in Sleep mode
PWR_CSTOP	Output	MCU in Stop modes

1. Bidirectional reset pin with embedded weak pull-up resistor.

Table 51. PWR internal input/output signals

Signal name	Signal type	Description
AHB	Input/output	AHB register interface
pwr_pvd_wkup	Output	Programmable voltage detector output

Table 51. PWR internal input/output signals (continued)

Signal name	Signal type	Description
pwr_pvm_x_wkup[5:0]	Output	Peripheral voltage monitor output
pwr_por_rstn	Output	Power-on reset
pwr_bor_rstn	Output	Brownout reset
exti_wkup	Input	Wake-up request
pwr_wkup	Output	Bus matrix clock wake-up request (system clock)

Each wake-up event (WKUPx) can be generated from four pins or internal events.

Table 52. Wake-up source selection

Port	Wake-up event
PA0	WKUP1
PA2	WKUP2
PC13	WKUP3
PD2	WKUP4

13.4 Power supplies

The system requires supply on VDD, VDDA18AON, VDDA18PMU, VDDSMPS, and VDDCORE to start, and allows independent supplies for VDDA18ADC, VBAT, VDD33USB, VDDIO2, VDDIO3, VDDIO4, and VDDIO5.

- V_{DD}: external power supply for I/Os (can be 1.8 V or 3.3 V typical)
- V_{BAT}: optional external power supply for backup domain when V_{DD} is not present (V_{BAT} mode)

  This power supply must be connected to V_{DD} when no battery is used.
- $V_{DDA18AON}$: external power supply for system analog such as reset, power management, oscillators, and OTP
- V_{DDSMPS}: external power supply for the SMPS step-down converter

  This power supply must be tied to V_{SS} when the SMPS is not used.
- $V_{DDA18PMU}$: external analog power supply for the SMPS step-down converter

  This power supply can be connected to V_{DDSMPS} through an inductor-based filter. It must be tied to V_{SS} when the SMPS is not used.
- V_{LXSMPS}: step-down converter supply output
- V_{FBSMPS}: step-down converter sense feedback
- V_{SSSMPS}: separate step-down converter ground
- V_{DDCORE}: digital core domain supply, dependent on V_{DD} supply

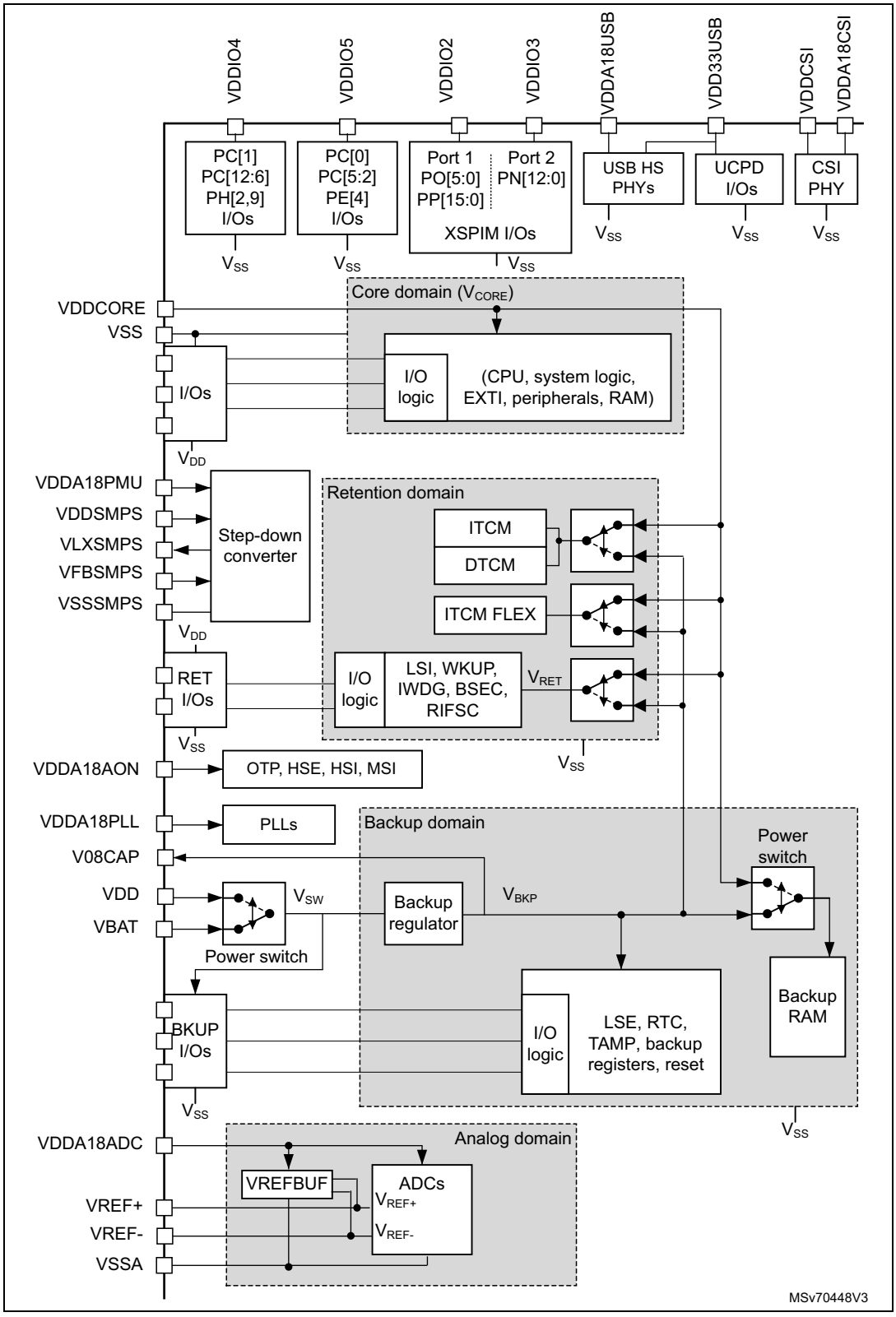
  V_{DD} must be present before V_{DDCORE}. The V_{CORE} is delivered by an external power supply through VDDCORE pin, or by the SMPS step-down converter.

  V_{DDCSI} must usually be connected to V_{DDCORE}.
- $V_{DDA18ADC}$: external analog power supply for ADCs and voltage reference buffers, independent from any other supply

- V_{REF+}: external reference voltage for ADCs, independent from any other supply
  - When the voltage reference buffer is enabled, V_{REF+} and V_{REF-} are delivered by the internal voltage reference buffer.
  - When the voltage reference buffer is disabled, V_{REF+} is delivered by an independent external reference supply.
- V_{SSA}: separate analog and reference voltage ground
- V_{DDIO2}: external power supply for 22 I/Os (PO[5:0] and PP[15:0]), independent from any other supply
- V_{DDIO3}: external power supply for 13 I/Os (PN[12:0]), independent from any other supply
- V_{DDIO4}: external power supply for 10 I/Os (PC[1], PC[12:6], and PH[9,2]), independent from any other supply
- V_{DDIO5}: external power supply for six I/Os (PC[0], PC[5:2], and PE[4]), independent from any other supply
- $V_{DD33USB}$: external power supply for USB2 HS PHYs and USB Type-C® (CC1 and CC2 pins), independent from any other supply
- $V_{DDA18USB}$: external analog power supply for USB2 HS PHYs
- $V_{DDA18CSI}$: external analog power supply for CSI D-PHY
- $V_{DDA18PLL}$: external analog power supplies for PLLs
- V_{SS}: common ground for all supplies except for step-down converter and analog peripherals.

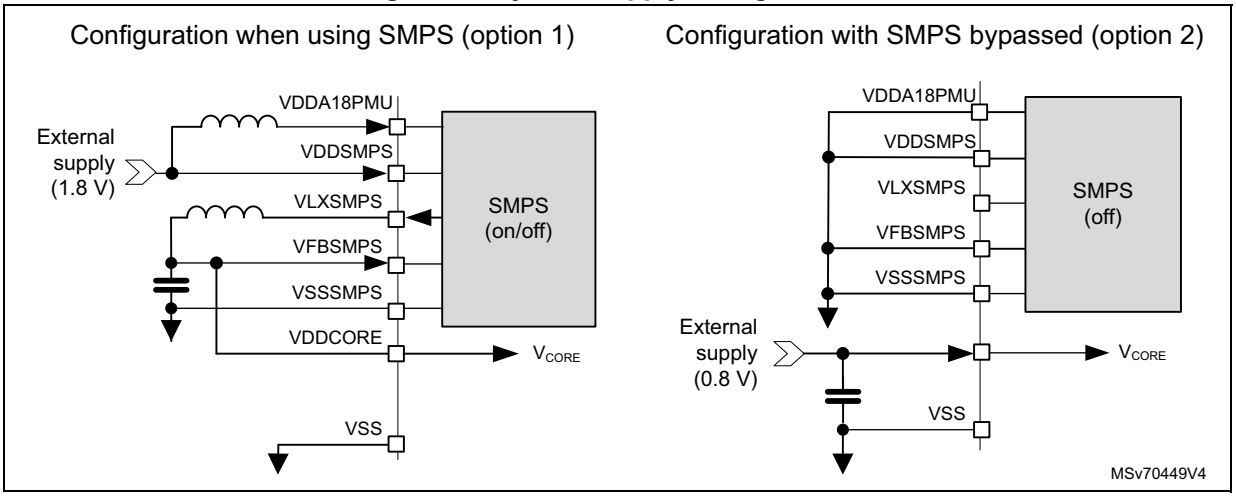
Note: Depending upon the operating power supply range, some peripherals can be used with limited features and performance. For more details, refer to General operating conditions in the datasheet.

Figure 16. Power supply overview



By configuring the SMPS step-down converter, the supply configurations shown in *Figure 17* are supported for the V_{CORE} domain.

Figure 17. System supply configurations



The different supply configurations are controlled through SDEN in PWR_CR1, according to *Table 53*. When the internal SMPS step-down converter is disabled, write SDEN bit in PWR_CR1 to 0 as soon as possible

Table 53. Supply configuration control

Supply configuration	SDEN	Description
Startup configuration	1	V_{CORE} power domains are supplied from external source. SMPS step-down converter enabled at 0.8 V can be used to supply the V_{CORE}.
SMPS step-down converter supply	1	V_{CORE} power domains are supplied from SMPS step-down converter according to VOS. SMPS step-down converter power mode (MR, LP, off) follows system low-power modes.
SMPS step-down converter disabled	0	V_{CORE} supplied from external source. SMPS step-down converter disabled, voltage monitoring still active.

13.4.1 System supply startup

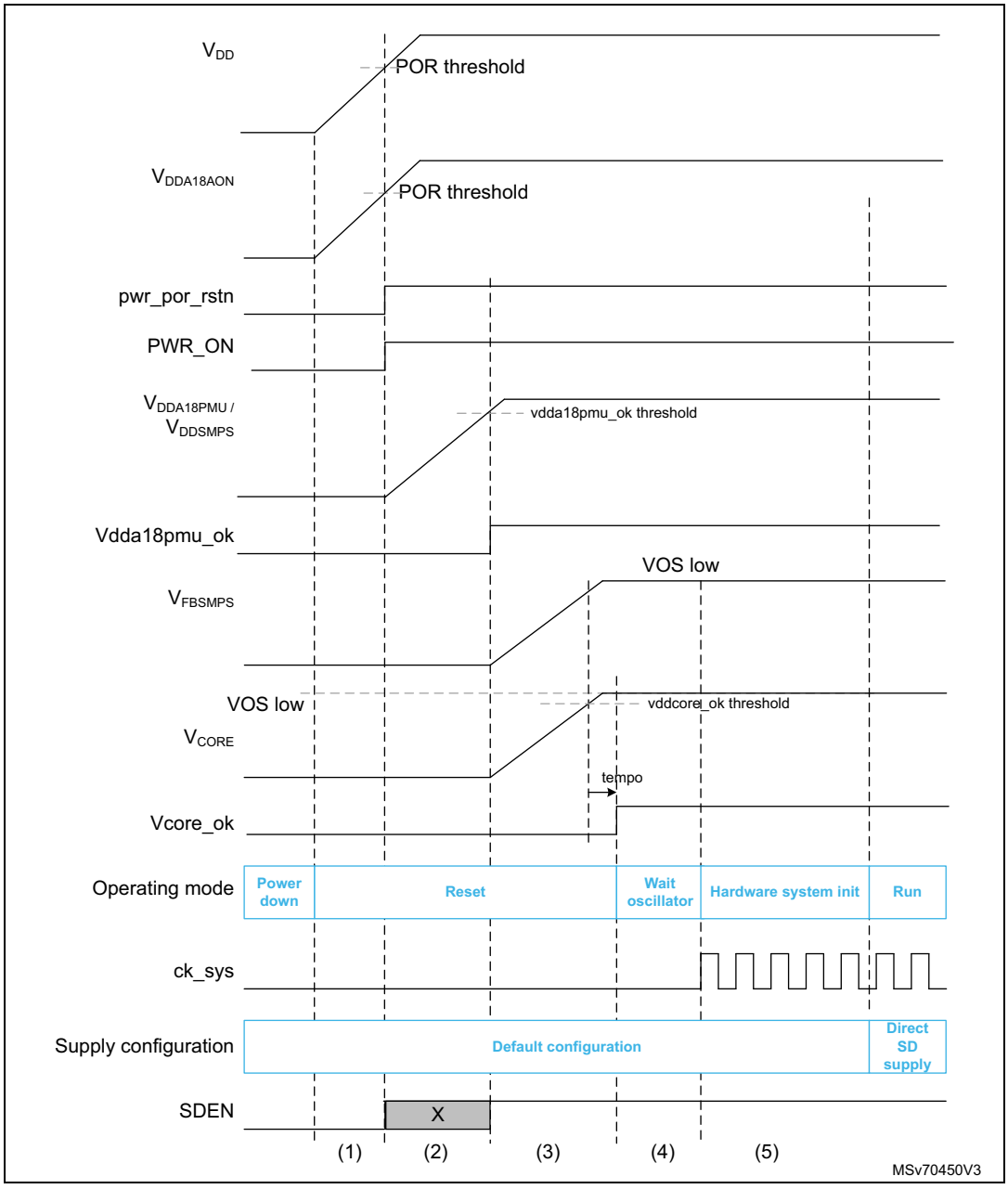
The system startup sequence from power-on in different supply configurations is the following (see *Figure 17* for direct SMPS supply and external supply, respectively):

V_{CORE} directly supplied from the SMPS step-down converter

1. When the system is powered on, the POR monitors V_{DD} and $V_{DDA18AON}$ supplies. Once the supplies are above the POR threshold level, the external voltage regulator providing $V_{DDA18PMU}$ and V_{DDSMPS} supplies is enabled via the PWR_ON signal.
2. The SMPS step-down converter is kept in reset as long as $V_{DDA18PMU}$ and V_{DDSMPS} are not stable.
3. Once $V_{DDA18PMU}$ and V_{DDSMPS} supplies are above the Vdda18pmu_ok threshold level, the SMPS step-down converter is taken out of reset, and the output level is set by default at 0.8 V (VOS low). The system is kept in reset mode as long as V_{DDCORE} is stable.

4. Once V_{DDCORE} supply is above the Vddcore_ok threshold level, the system is taken out of reset, and the HSI oscillator is enabled.
5. Once the oscillator is stable, the system is initialized: option bytes are loaded, and the CPU starts in Run mode.

Figure 18. Device startup (V_{CORE} supplied directly from SMPS step-down converter)



When exiting Standby mode, the supply configuration is known by the system, as the PWR_CR1 content is retained.

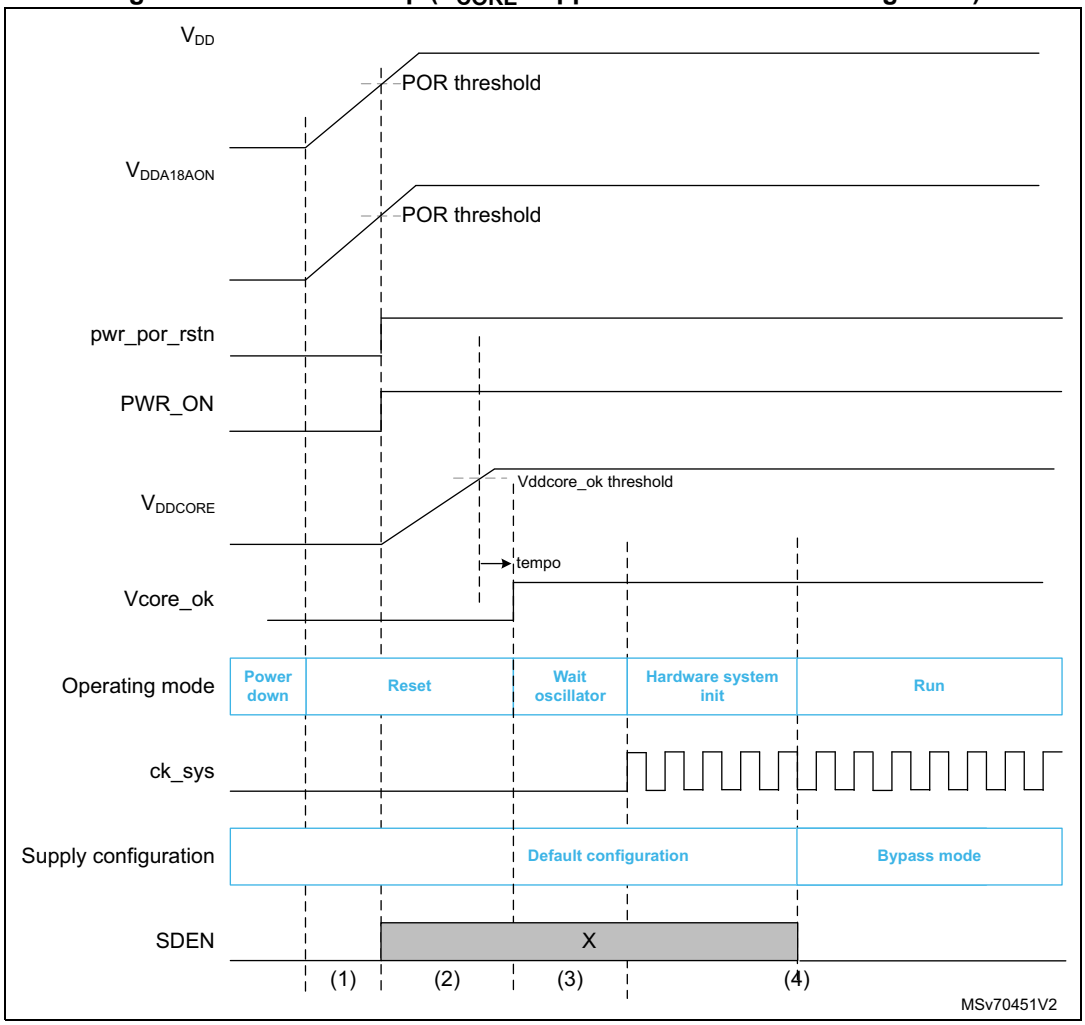
V_{CORE} supplied in bypass mode (SMPS off)

The devices that feature the SMPS can be used in bypass mode.

1. When the system is powered on, the POR monitors V_{DD} and $V_{DDA18AON}$ supplies. Once the supplies are above the POR threshold level, the external voltage regulator providing the V_{DDCORE} supply is enabled via the PWR_ON signal.
2. The system is kept in reset mode as long as V_{DDCORE} is not stable.
3. Once V_{DDCORE} supply is above the Vddcore_ok threshold level, the system is taken out of reset, and the HSI oscillator is enabled.
4. Once the oscillator is stable, the system is initialized: option bytes are loaded, and the CPU starts in Run mode. The software must disable SMPS bit clearing SDEN in PWR_CR1.

Note: In SMPS off mode or bypass mode, reading SDEN returns 0. The SW must write 0 to SDEN in PWR_CR1 register for the low power features to work properly.

Figure 19. Device startup (V_{CORE} supplied from an external regulator)



13.4.2 Core domain

The V_{CORE} core domain supply can be provided by the SMPS step-down converter, or by an external supply (V_{DDCORE}). V_{CORE} supplies all the digital circuitries, except for the backup domain and the retention domain in Standby mode. When a system reset occurs,

the SMPS step-down converter is enabled to deliver 0.8 V. This allows the system to start up in any supply configuration (see *Figure 17*).

After a power-on reset, the software must configure the used supply configuration in PWR_CR1 before changing VOS in PWR_VOSCR, or the RCC sys_ck frequency. The different system supply configurations are controlled as shown in *Table 53*.

SMPS step-down converter regulator

The SMPS requires an external coil to be connected between the VLXSMPS pin and (via a capacitor) VSS. The converter can be used to supply directly the V_{CORE} domain. It works in three different power modes: main (MR), low-power (LP), or off.

The converter operating modes depend upon the system modes (Run, Stop, or Standby mode), and are configured through the associated VOS and SVOS levels:

- Run mode

  The SMPS step-down converter operates in MR mode, and provides full power to the V_{CORE} domain (core, memories, and digital peripherals). PWR_ON is set high. The regulator output voltage can be scaled by software to different voltage levels (VOS low and VOS high) that are configured through VOS in PWR_VOSCR . The VOS voltage scaling is used to optimize the power consumption when the system is clocked below the maximum frequency. By default, VOS low is selected after a system reset. VOS can be changed on-the-fly to adapt to the required system performance.

- Stop mode

  The SMPS step-down converter supplies the V_{CORE} domain to retain the content of registers and internal memories, and must be set in low-power mode. The voltage regulator mode is selected through SVOS in PWR_CPUCR. In low-power mode, only SVOS low and SVOS high scaling are allowed. Due to a lower voltage level for the SVOS low, the Stop mode consumption can be further reduced.

- Standby mode

  The SMPS step-down converter is off, and V_{CORE} domains are powered down. The content of registers and memories are lost except for the retention domain, and the backup domain.

  The PWR_ON signal is set low. The external regulator supplying $V_{DDA18PMU}$ and V_{DDSMPS} can be switched off.

13.4.3 PWR external supply

When V_{CORE} is supplied from an external source (bypass mode), different operating modes can be used, depending upon the system operating modes:

- In Run mode

  The external source supplies full power to the V_{CORE} domain. PWR_ON is set high. The external source output voltage is scalable through different voltage levels (VOS low and VOS high). Setting VOS in PWR_VOSCR has no effect on the external supply. The application can use the register value to track the status of the external supplied voltage, it is responsible to set properly the external supply voltage. Other PWR_VOSCR register fields have default settings, and are "don't care" in external supply.

- In Stop mode

  The external source supplies V_{CORE} domain to retain the content of registers and internal memories. In low-power mode, only SVOS high scaling is allowed.

- In Standby mode

  The PWR_ON signal is set low, the external regulator is switched off, and the V_{CORE} domain is powered down. The content of registers and memories is lost except for the retention domain and the backup domain.

13.4.4 Backup domain

To retain the content of the backup domain (RTC, backup registers, and backup RAM) when V_{DD} is turned off, the VBAT pin can be connected to an optional voltage supplied from a battery or from another source.

The switching to V_{BAT} is controlled by the power-down reset (PDR) embedded in the reset block that monitors the V_{DD} supply.

Warning: **During $t_{RSTTEMPO}$ (temporization at V_{DD} startup), or after a PDR is detected, the power switch between V_{BAT} and V_{DD} remains connected to VBAT.**
During the startup phase, if V_{DD} is established in less than $t_{RSTTEMPO}$ (see the datasheet for the value of $t_{RSTTEMPO}$), and $V_{DD} > V_{BAT} + 0.6$ V, a current may be injected into VBAT through an internal diode connected between V_{DD} and the power switch (V_{BAT}).
If the power supply/battery connected to the VBAT pin cannot support this current injection, it is strongly recommended to connect an external low-drop diode between this power supply and the VBAT pin.

When the V_{DD} supply is present, the backup domain is supplied from V_{DD}, to save V_{BAT} power supply battery life time. If no external battery is used in the application, it is recommended to connect VBAT externally to VDD, and add a 100 nF external ceramic capacitor between VBAT and VSS.

When the backup domain is supplied by VBAT (analog switch connected to VBAT), the following pins are available:

- PC13, PC14, and PC15, which can be configured by RTC or LSE (see *Section 61.3: RTC functional description*).
- PC13, PQ7 (TAMP_IN/OUT), and PD8, PH4 (only TAMP_IN) when they are configured by the TAMP peripheral as tamper pins.

Accessing the backup domain

After reset, the backup domain (RCC_BDCR, RCC_RTC, TAMP registers, backup registers, and backup RAM) is protected against possible unwanted write accesses. To enable access to the backup domain, set DBP in PWR_DBPCR.

For more detail on RTC and backup RAM access, refer to *Section 14: Reset and clock control (RCC)*.

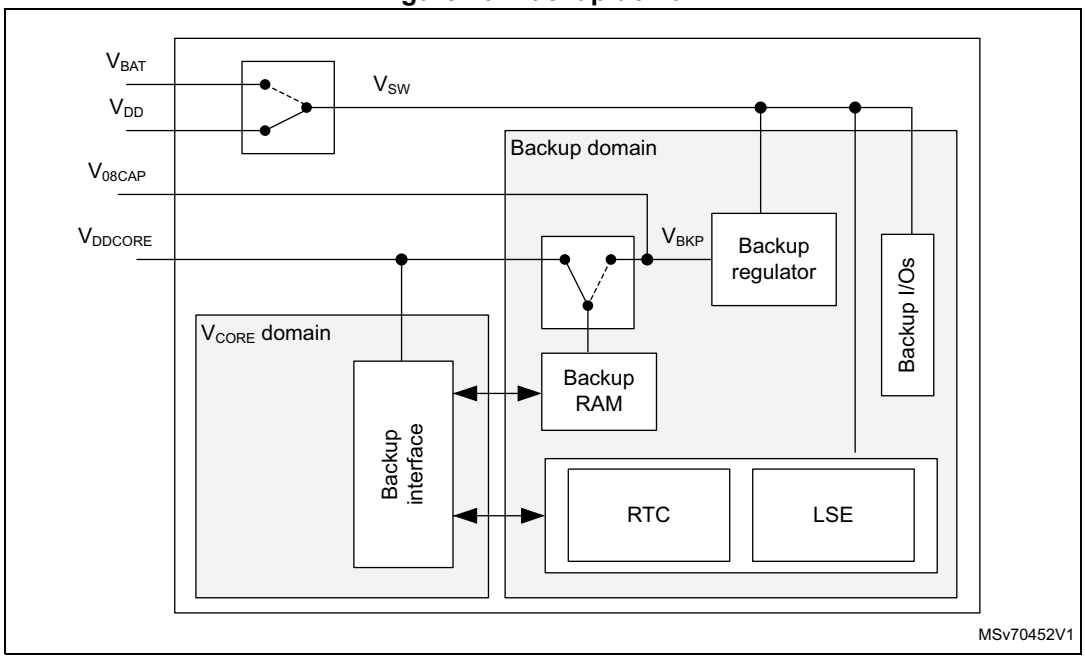
Backup RAM (BKPSRAM)

The backup domain includes 8 Kbytes of BKPSRAM accessible in 32-, 16-, or 8-bit data mode.

- In Run and Stop (SVOS high) modes, the BKPSRAM is supplied from V_{CORE} supply.
- In Stop (SVOS low) mode, the BKPSRAM is supplied through the backup regulator in V_{SW} domain.
- In Standby and V_{BAT} modes, the BKPSRAM can be supplied through the backup regulator in V_{SW} domain.

When the backup supply is enabled through BKPRBSEN in PWR_BDCR2, the BKPSRAM content is retained even in Standby and/or V_{BAT} mode.

Figure 20. Backup domain



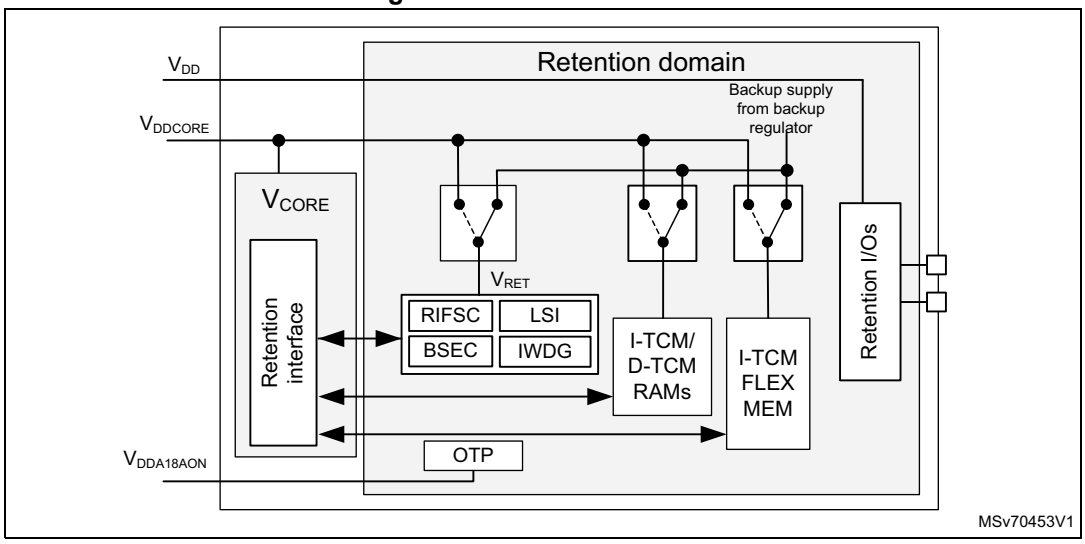
13.4.5 Retention domain

The retention domain includes boot, OTP controller (BSEC), independent watchdog (IWDG), resource isolation framework controller (RIFSC), I-TCM, D-TCM, and I-TCM FLEXRAMs.

- In Run and Stop (SVOS high) modes, the retention domain is supplied from the V_{CORE} supply.
- In Stop (SVOS low) and Standby modes, the retention domain is supplied through the backup regulator.

The content of the domain is retained in Standby mode (not powered in V_{BAT} mode).

Figure 21. Retention domain



I-TCM and D-TCM RAMs

The retention domain includes 64 + 128 K bytes (instruction/data) of TCM for Cortex-M55.

- In Run and Stop (SVOS high) modes, I-TCM, and D-TCM memories are supplied from the V_{CORE} supply.
- In Stop (SVOS low) mode, I-TCM, and D-TCM memories are supplied through the backup regulator in the V_{SW} domain.
- In Standby mode, I-TCM and D-TCM memories can be supplied through the backup regulator in the V_{SW} domain.

When the backup supply is enabled by TCMRBSEN in PWR_CR4, the content of these memories is retained even in Standby mode.

I-TCM FLEXMEM

The retention domain includes 64 Kbytes of extended TCMs for the Cortex-M55 (the I-TCM FLEXMEM extension). This memory can be allocated to I-TCM or to system AXI RAM (see *Section 10: SRAM configuration controller (RAMCFG)* for details

- In Run and Stop (SVOS high) modes, the I-TCM FLEXMEM is supplied from the V_{CORE} supply.
- In Stop (SVOS low) mode, the I-TCM FLEXMEM is supplied through the backup regulator in the V_{SW} domain.
- In Standby mode, the I-TCM FLEXMEM can be supplied through the backup regulator in the V_{SW} domain.

When the backup supply is enabled by TCMFLXRBSEN in PWR_CR4, the content of these memories is retained even in Standby mode.

13.4.6 Analog supply

Separate $V_{DDA18ADC}$ analog supply

The analog supply domain is powered by dedicated VDDA18ADC and VSSA pins that allow the supply to be filtered and shielded from noise on the PCB, thus improving ADC conversion accuracy:

- The analog supply voltage input is available on a separate VDDA18ADC pin.
- An isolated supply ground connection is provided on VSSA pin.

Analog reference voltage V_{REF+}/V_{REF-}

To achieve better accuracy low-voltage signals, the ADC has a separate reference voltage, available on VREF+ pin. The user can connect a separate external reference voltage on V_{REF+}.

V_{REF+} controls the highest voltage, represented by the full scale value. The lower voltage reference (V_{REF-}) must be connected to V_{SSA}.

When enabled by ENVR in VREFBUF_CSR, V_{REF+} is provided from the internal voltage reference buffer, which can also deliver a reference voltage to external components through VREF+/VREF- pins.

When the internal voltage reference buffer is disabled by ENVR, V_{REF+} is delivered by an independent external reference supply voltage.

13.5 Power supply supervision

Power supply level monitoring is available on the following supplies:

- V_{DD}/$V_{DDA18AON}$ via POR/PDR (see *Section 13.5.1*), BOR (see *Section 13.5.2*), and PVD monitor (see *Section 13.5.6*)
  - The POR/PDR monitoring flag is available from PORRSTF in the RCC.
  - The BOR monitoring flag is available from BORRSTF in the RCC.
  - The PVD monitoring flag is available from PVDO in PWR_CR2. In addition, an interrupt and wake-up can be generated via the EXTI.
- $V_{DDA18PMU}$ via Vdda18pmu_ok, which keeps in reset the SMPS, as long as the level is not correct.
  - The $V_{DDA18PMU}$ monitoring has no flags.
- V_{BAT} via V_{BAT} threshold (see *Section 13.5.8*)
  - The V_{08CAP} monitoring flags are available from V08CAPH and V08CAPL in PWR_BDCR1. In addition, an interrupt and wake-up can be generated via EXTI.
- V_{DDCORE} via Vcore_ok, which keeps the V_{CORE} domain in reset as long as the level is not correct (see *Section 13.5.3*).
  - The V_{DDCORE} ok monitoring flag is available from SBF in PWR_CPUCR when exiting the Standby mode.
- V_{DDCORE} via V_{CORE} threshold (see *Section 13.5.5*)
  - The V_{DDCORE} monitoring flags are available from VCOREH and VCOREL in PWR_CR3. In addition, an interrupt and wake-up can be generated via EXTI.
- V_{DDIO2}, V_{DDIO3}, V_{DDIO4}, V_{DDIO5}, $V_{DD33USB}$, $V_{DDA18ADC}$ via peripheral voltage monitor (see *Section 13.5.7*)

- V_{SW} via rst_vsw, which keeps V_{SW} domain in reset mode as long as the level is not the correct one
  - The V_{SW} monitoring has no flag. V_{SW} registers reset value can be used.
- V_{FBSMPS} can be monitored via VOSRDY in PWR_VOSCR.

13.5.1 Power-on reset (POR)/power-down reset (PDR)

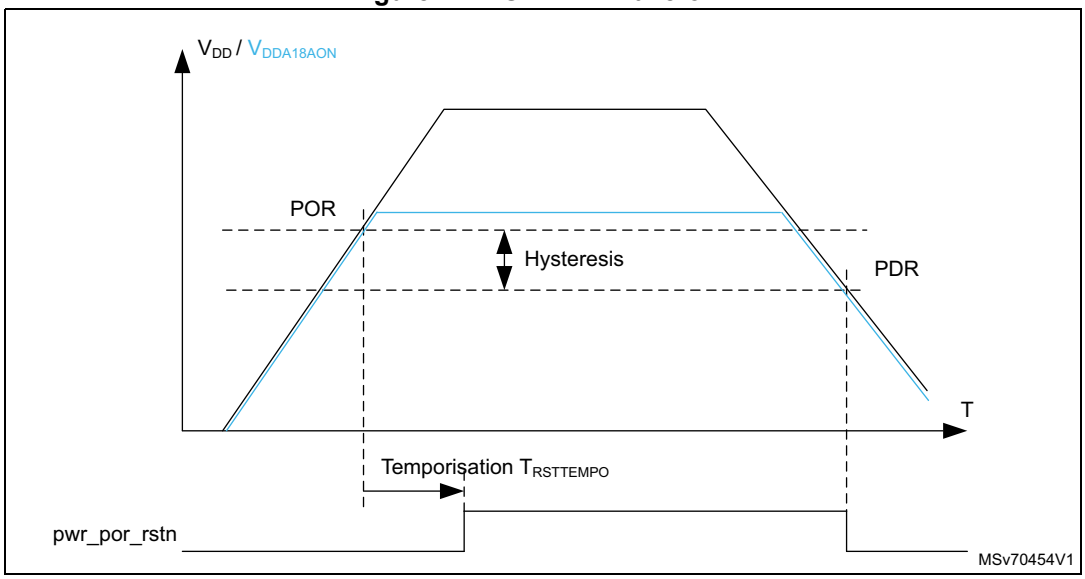
The system has an integrated POR/PDR circuitry that ensures proper start-up operation.

The system remains in reset mode when V_{DD} and $V_{DDA18AON}$ are below a specified V_{POR} threshold, without the need for an external reset circuit. Once these supply levels are above the V_{POR} threshold, the system is taken out of reset (see *Figure 22*). For more details concerning these thresholds, refer to the electrical characteristics in the datasheet.

The POR/PDR generate also an application reset (NRST).

The PDR can be enabled/disabled by the device PDR_ON input pin.

Figure 22. POR/PDR waveform



1. For thresholds and hysteresis values, refer to the datasheets.

13.5.2 Brownout reset (BOR)

During power-on, the brownout reset (BOR) keeps the system under reset until the V_{DD} supply voltage reaches the specified V_{BOR} threshold.

The V_{BOR} threshold is configured through system option bytes. By default, the BOR is off. The selection of another BOR threshold takes effect only after the device has loaded the option bytes. The following programmable V_{BOR} thresholds can be selected:

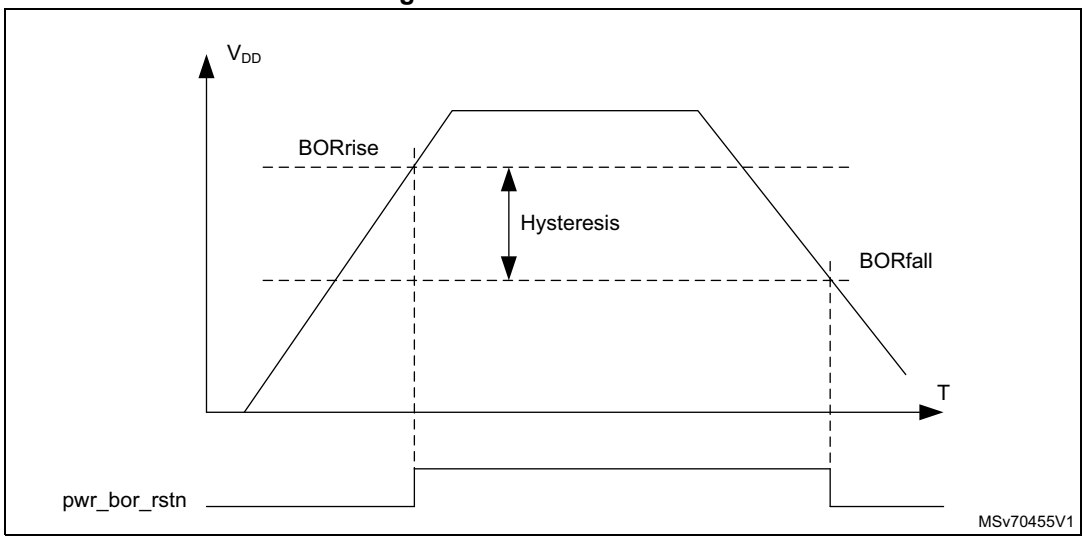
- BOR off (V_{BOR0}): reset threshold level above 1.67 V
- BOR level 1 (V_{BOR1}): reset threshold level above 2.7 V

For more details on these thresholds, refer to the electrical characteristics in the datasheet.

When the BOR is enabled and the V_{DD} supply voltage drops below the selected V_{BOR} threshold, an application reset (NRST) is generated.

The BOR can be disabled by programming the system option bytes. To disable it, V_{DD} must have been higher than V_{BOR0} to start the system option byte programming sequence. The power-down is then monitored by the PDR (see *Section 13.5.1*).

Figure 23. BOR thresholds



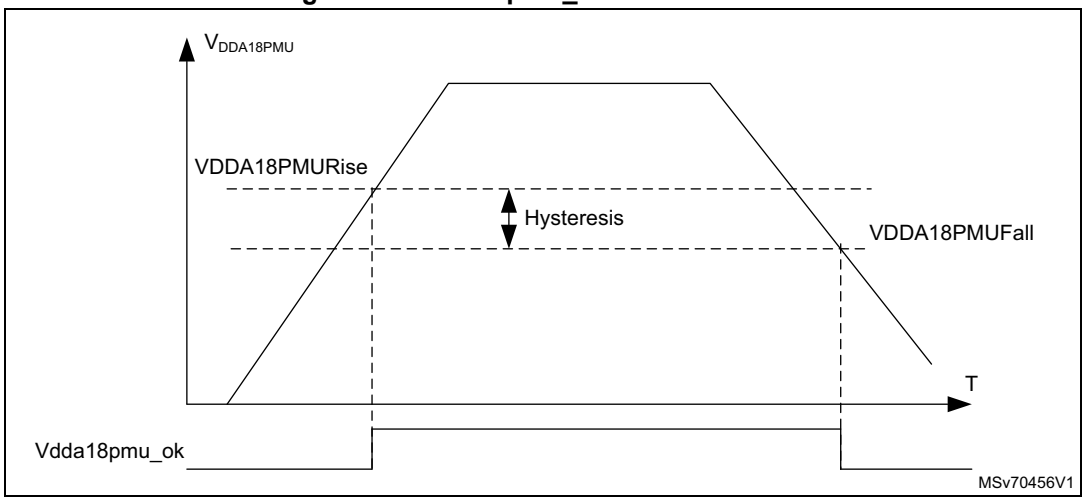
1. For thresholds and hysteresis values, refer to the datasheets.

13.5.3 Vdda18pmu_ok reset

The system integrates a circuit for proper startup of the integrated SMPS.

The SMPS remains in reset mode when $V_{DDA18PMU}$ is below the threshold Vdda18pmu_ok. Once the $V_{DDA18PMU}$ supply level is above Vdda18pmu_ok, the SMPS is taken out of reset, and the output level is set by default at 0.8 V (VOS low). For more details concerning this reset threshold, refer to the electrical characteristics in the datasheet.

Figure 24. Vdda18pmu_ok reset threshold



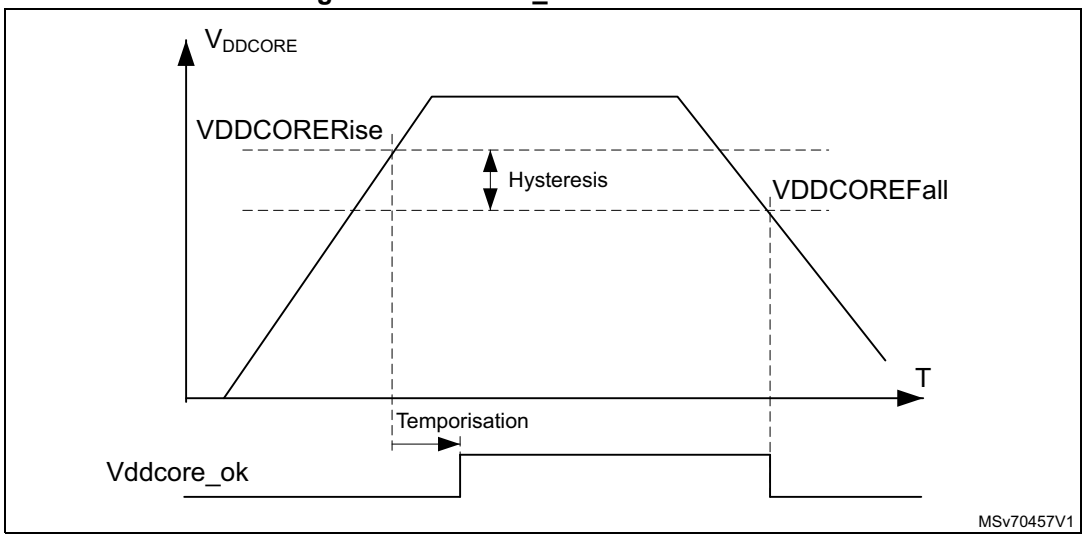
13.5.4 Vddcore_ok reset

The system has in integrated circuit for proper startup of the V_{CORE} domain.

The V_{CORE} domain remains in reset mode when V_{DDCORE} is below the operation threshold Vddcore_ok. Once the V_{DDCORE} supply level is above Vddcore_ok, the V_{CORE} domain is taken out of reset. For more details concerning this reset threshold, refer to the electrical characteristics in the datasheet.

When SVOS is cleared in PWR_CPUCR, the V_{DDCORE} supply level can be lowered in Stop mode (SVOS low).

Figure 25. Vddcore_ok reset thresholds



13.5.5 V_{DDCORE} monitoring

The system has a detection circuitry that detects if the V_{DDCORE} supply voltage is below or above the operating range. The detection is done by comparing the V_{DDCORE} with two thresholds (high and low threshold). The level of the low threshold can be selected by VCORELLS in PWR_CR3. The high level is fixed.

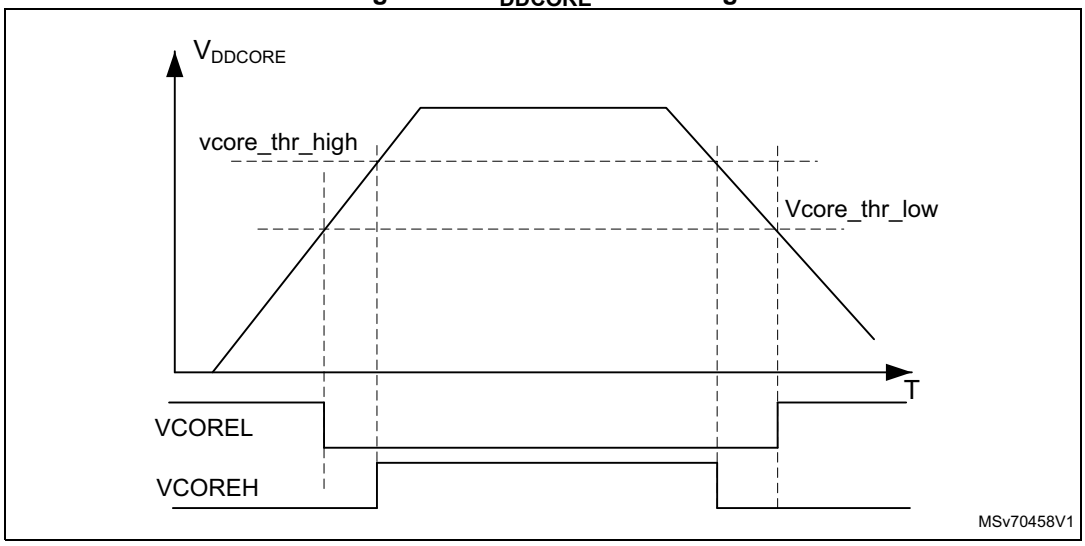
VCOREH and VCOREL flags are available in PWR_CR3, to indicate if V_{DDCORE} is higher or lower than the thresholds. VCOREH and VCOREL flags are available on tamper signals but also connected to the EXTI, and can generate an interrupt if enabled through EXTI registers.

The detection is enabled by setting VCOREMONEN in PWR_CR3.

The V_{DDCORE} voltage thresholds, when enabled, are not available in Stop, Standby, and V_{BAT} modes.

Note: The low threshold of V_{DDCORE} monitoring is inaccurate. To ensure the value of voltage threshold use ADC.

Figure 26. V_{DDCORE} monitoring



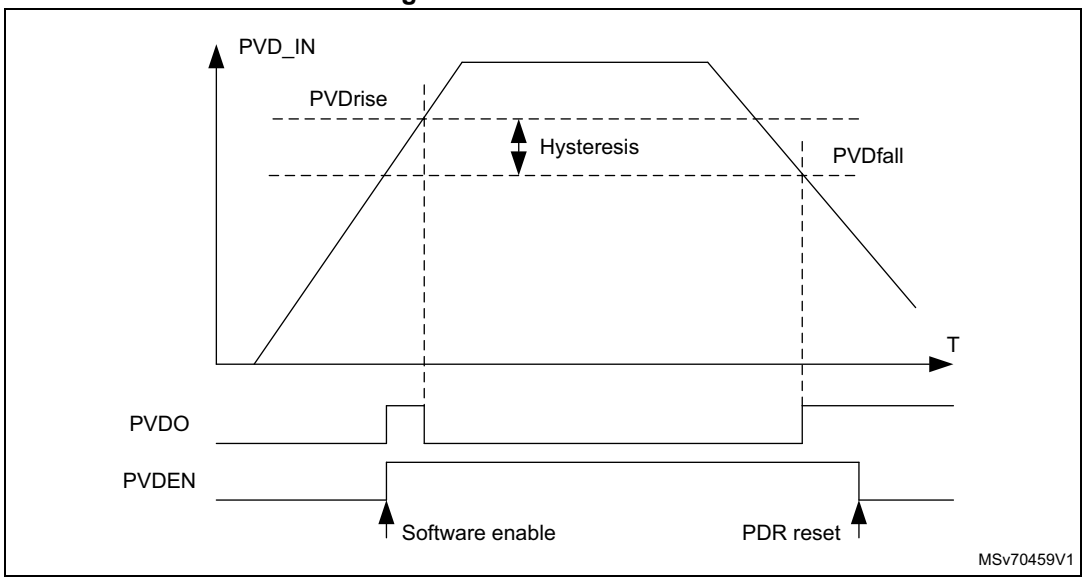
13.5.6 Programmable voltage detector (PVD)

The PVD can be used to monitor a voltage level on PVD_IN pin. The voltage level on PVD_IN is compared to the internal VREFINT level. The PVD is enabled by setting PVDEN in PWR_CR2. The PVD is not available in Standby mode.

A PVDO flag is available in PWR_CR2 to indicate if the voltage level on PVD_IN is higher or lower than the PVD threshold. This event is internally connected to the EXTI, and can generate an interrupt if enabled through the EXTI registers.

The PVDO output interrupt can be generated when the voltage level on PVD_IN drops below the PVD threshold, and/or when the voltage level on PVD_IN rises above the PVD threshold depending on EXTI rising/falling edge configuration. As an example, the service routine can perform emergency shutdown tasks.

Figure 27. PVD thresholds



1. For thresholds and hysteresis values, refer to the datasheets.

13.5.7 Peripheral voltage monitoring (PVM)

Only V_{DD} is monitored by default, as it is the only supply required for all system-related functions. The other supplies (V_{DDIO2}, V_{DDIO3}, V_{DDIO4}, V_{DDIO5}, VDD33USB, and $V_{DDA18ADC}$) can be independent from V_{DD}, and can be monitored with six peripheral voltages (PVM):

- VDDIO2VM monitors the PO[5:0] and PP[15:0] I/Os supply V_{DDIO2}. VDDIO2RDY indicates if V_{DDIO2} is higher or lower than the VIO2VM threshold.
- VDDIO3VM monitors the PN[12:0] I/Os supply V_{DDIO3}. VDDIO3RDY indicates if V_{DDIO3} is higher or lower than the VIO3VM threshold.
- VDDIO4VM monitors PC[1], PC[12:6], and PH[9,2] I/Os supply V_{DDIO4}. VDDIO4RDY indicates if V_{DDIO4} is higher or lower than the VIO4VM threshold.
- VDDIO5VM monitors the PC[0], PC[5:2] and PE[4] I/Os supply V_{DDIO5}. VDDIO5RDY indicates if V_{DDIO5} is higher or lower than the VIO5VM threshold.
- USB33VM monitors the USB supply $V_{DD33USB}$. USB33RDY indicates if $V_{DD33USB}$ is higher or lower than the VUSB33VM threshold.
- AVM monitors the ADC supply $V_{DDA18ADC}$. ARDY indicates if $V_{DDA18ADC}$ is higher or lower than the VAVM threshold.

For thresholds and hysteresis, refer to the datasheet.

Each PVM output is connected to an EXTI line, and can generate an interrupt if enabled through the EXTI registers. The EXTI_PVM_x output interrupt is generated when the independent power supply drops below the PVM threshold, and/or when it rises above the PVM threshold, depending on EXTI line rising/falling edge configuration.

The PVM is not available in Standby mode.

The independent supplies (V_{DDIO2}, V_{DDIO3}, V_{DDIO4}, V_{DDIO5}, $V_{DD33USB}$, and $V_{DDA18ADC}$) are not considered as present by default, and a logical and electrical isolation is applied to ignore any information coming from the peripherals supplied by these dedicated supplies.

- If these supplies are shorted externally to V_{DD}, the application must assume they are available without enabling any peripheral voltage monitoring, and the power isolation can be removed by setting the corresponding supply valid bits.
- If these supplies are independent from V_{DD}, the PVM can be enabled to confirm whether the supply is present or not.

The following sequence must be done before using the USB HS PHYs:

1. If $V_{DD33USB}$ is independent from V_{DD}:
   a) Enable the USB33VM by setting USB33VMEN in PWR_SVMCR3.
   b) Wait for the USB33VM wake-up time.
   c) Wait until USB33RDY is set in PWR_SVMCR3.
   d) Optional: Disable the USB33VM for consumption saving.
2. Set USB33SV in PWR_SVMCR3 to remove the $V_{DD33USB}$ power isolation.

The following sequence must be done before using the analog-to-digital converters:

1. If $V_{DDA18ADC}$ is independent from V_{DD}:
   a) Enable the AVM by setting AVMEN in PWR_SVMCR3.
   b) Wait for the AVM wake-up time.
   c) Wait until ARDY is set in PWR_SVMCR3.
   d) Optional: Disable the AVM for consumption saving.

2. Set ASV in PWR_SVMCR3 to remove the $V_{DDA18ADC}$ power isolation.

GPIOs can work in the 1.8 V or 3.3 V power supply ranges. It is needed to select the supply range of the I/Os before using them (the 3.3 V range is selected by default). The voltage range configuration for V_{DD}, V_{DDIO2}, and V_{DDIO3} is retained in Standby mode.

The configuration can be retained for V_{DDIO4} and V_{DDIO5} in Standby mode when VDDIO4VRSTBY is set in PWR_SVMCR1(and VDDIO5VRSTBY in PWR_SVMCR2). The voltage range must not be retained if the supply range may change when exiting Standby mode (SD card).

The following sequence must be done before using VDDIO4 I/Os:

1. If V_{DDIO4} is independent from V_{DD}:
   a) Enable the VDDIO4VM by setting VDDIO4VMEN in PWR_SVMCR1.
   b) Wait for the VDDIO4VM wake-up time.
   c) Wait until VDDIO4RDY is set in PWR_SVMCR1.
   d) Optional: Disable the VDDIO4VM for consumption saving.
   e) If V_{DDIO4} is in 1.8 V range: Set the V_{DDIO4} voltage range by setting VDDIO4VRSEL in PWR_SVMCR1.
   f) Optional: Retain VDDIO4VRSEL configuration by setting VDDIO4VRSTBY in PWR_SVMCR1.
2. Set VDDIO4SV in PWR_SVMCR1 to remove the V_{DDIO4} power isolation.

Same sequence must be done for VDDIO5 I/Os.

The following sequence must be done before using VDDIO2 I/Os:

1. If V_{DDIO2} is independent from V_{DD}:
   a) Enable the VDDIO2VM by setting VDDIO2VMEN in PWR_SVMCR3.
   b) Wait for the VDDIO2VM wake-up time.
   c) Wait until VDDIO2RDY is set in PWR_SVMCR3.
   d) Optional: Disable the VDDIO2VM for consumption saving.
   e) If V_{DDIO2} is in 1.8 V range: Set the V_{DDIO2} voltage range by setting VDDIO2VRSEL in PWR_SVMCR3.
2. Set VDDIO2SV in PWR_SVMCR3 to remove the V_{DDIO2} power isolation.

Same sequence must be done for the VDDIO3 I/Os.

13.5.8 Battery voltage thresholds

The battery voltage V_{BAT} supply can be monitored by comparing it with two threshold levels: $V_{08CAPhigh}$ and $V_{08CAPlow}$. V08CAPH and V08CAPL flags in PWR_BDCR1 indicate if V_{SW} is higher or lower than the thresholds.

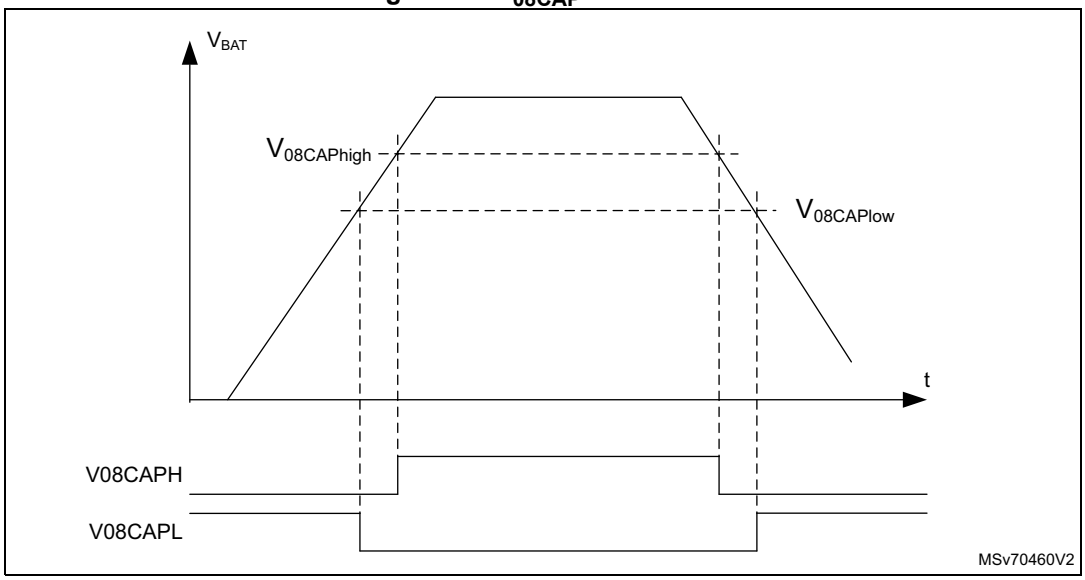
The V_{BAT} supply monitoring can be enabled/disabled via MONEN in the PWR_BDCR1. When V_{08CAP} monitoring is enabled, the battery voltage thresholds increase the power consumption. As an example, V_{SW} levels monitoring can be used to trigger a tamper event for an over or under voltage of the RTC power supply domain (available in V_{BAT} mode).

The battery voltage thresholds, when enabled, are also available in Standby and V_{BAT} modes.

V08CAPH and V08CAPL are connected to RTC tamper signals (see *Section 61: Real-time clock (RTC)*.

Note: When the device does not operate in V_{BAT} mode, the battery voltage monitoring checks the V_{DD} level. When V_{DD} is available, V_{SW} is connected to V_{DD} through the internal power switch (see Section 13.4.4).

Figure 28. V_{08CAP} thresholds



1. For thresholds and hysteresis values, refer to the datasheets.

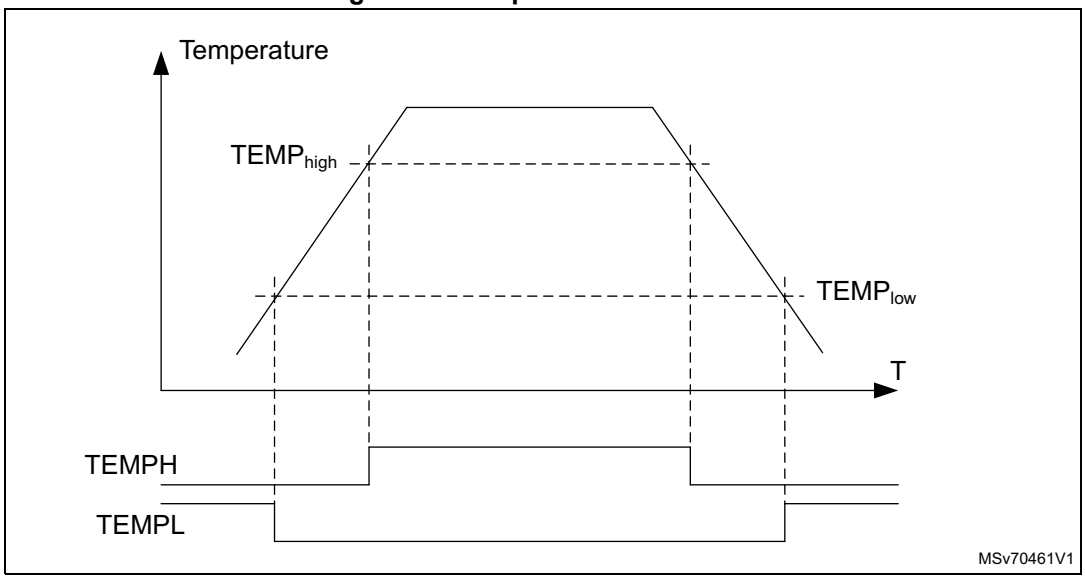
13.5.9 Temperature thresholds

The junction temperature can be monitored by comparing it with two threshold levels, $TEMP_{high}$ and $TEMP_{low}$. TEMPH and TEMPL flags in PWR_BDRC1 indicate whether the device temperature is higher or lower than the thresholds. The temperature monitoring can be enabled/disabled via MONEN in PWR_BDRC1. When the temperature monitoring is enabled, the temperature thresholds increase the power consumption. As an example, threshold levels can be used to trigger a routine to perform temperature control tasks.

The temperature thresholds, when the monitoring is enabled, are also available in Standby and V_{BAT} modes.

TEMPH and TEMPL wake-up interrupts are available on the RTC tamper signals (see *Section 61: Real-time clock (RTC)*).

Figure 29. Temperature thresholds



1. For thresholds and hysteresis values, refer to the datasheets.

13.6 Power management

The power management block controls the V_{CORE} supply in accordance with the system operation modes (see *Section 13.6.1*).

The device power domains can operate in one of the following operating modes:

- Run (power on, clock on)
- Sleep (power on, core clock stopped, peripherals kept running)
- Stop (power on, clock off)
- Standby (power off, clock off)

The V_{CORE} supply level follows the system operating mode (Run, Stop, or Standby mode).

The following voltage scaling features allow the power to be controlled with respect to the required system performance (see *Table 54*):

- To obtain a given system performance, the corresponding voltage scaling must be set in accordance with the system clock frequency. To do this, configure VOS bits to the Run mode voltage scaling.
- To obtain the best trade-off between power consumption and latency when exiting Stop mode, configure SVOS bit to Stop mode voltage scaling.

13.6.1 Operating modes

Several system operating modes are available to tune the system according to the performance required (when the CPU does not need to execute code, and waits for an external event). The user must select the operating mode that gives the best compromise between low power consumption, short start-up time, and available wake-up sources.

The operating modes are used to control the clock distribution to the different system blocks, and to power them.

In Run mode, the power consumption can be reduced by one of the following means:

- Lower the system performance by slowing down the system clocks, and reducing the V_{CORE} supply level through VOS voltage scaling bit.
- Gate the clocks to APBx and AHBx peripherals when they are not used, through PERxEN bits.

Table 54. Operating mode summary

System	Entry	Wake-up	System oscillator	System clock	Peripheral clock	CPU clock	Voltage regulator	PWR_ON
Run	-	-	ON	ON	ON	ON(1)	ON (VOS low/high)	1
Sleep	WFI or return from ISR or WFE(2)	See *Table 55*			ON/OFF(3)			
Stop SVOS high	SVOS + SLEEPDEEP + WFI or return from ISR, WFE, or wake-up source cleared(4)		ON/OFF(5)	OFF		OFF	ON (SVOS high)	
Stop SVOS low	SLEEPDEEP + WFI or return from ISR, WFE, or wake-up source cleared(3)		OFF		OFF		ON (SVOS low)	
Standby	PDDS + SLEEPDEEP + WFI or return from ISR, WFE, or wake-up source cleared(3)	WKUP pins rising or falling edge, RTC alarm (alarm A or alarm B), RTC wake-up event, RTC tamper events, RTC timestamp event, external reset in NRST pin, IWDG reset	OFF	OFF	OFF	OFF	OFF	0(6)

1. The clock is gated in the core in Sleep mode.
2. WFI = wait for interrupt, ISR = interrupt service routine, WFE = wait for event.
3. The CPU subsystem peripherals that have a PERxLPEN bit, operate accordingly.
4. When the CPU is in Stop mode, the last EXTI wake-up source must be cleared by software.
5. When HSI or MSI is used, the state is controlled by HSISTOPEN and MSISTOPEN, otherwise the system oscillator is off.
6. A guaranteed minimum PWR_ON pulse low time can be defined by POPL bits in PWR_CR1.

Table 55. Functionalities depending on system operating mode

Peripheral[1]	Run mode	Stop mode SVOS high		Stop mode SVOS low		Standby mode		V_{BAT} mode
		-	Wake-up capability	-	Wake-up capability	-	Wake-up capability	
CPU	Y	R	-	R	-	-	-	-
NPU	O	O	-	R	-	-	-	-
Debug	O	O	O	R	-	-	-	-
ROM memory	Y	R	-	R	-	-	-	-
RAMCFG	O	R	-	R	-	-	-	-
I-TCM	O	R	-	R	-	R	-	-
I-TCM FLEXMEM	O	R	-	R	-	R	-	-
D-TCM	O	R	-	R	-	R	-	-
AXISRAM1	O	R	-	R	-	R[2]	-	-
AXISRAMx (x = 2, 3, 4)	O	O	-	R	-	-	-	-
I-TCM FLEXMEM extension	O	O	-	R	-	R[3]	-	-
D-TCM FLEXMEM extension	O	O	-	R	-	-	-	-
CACHEAXI	O	O	-	R	-	-	-	-
VENCRAM	O	O	-	R	-	-	-	-
GPU RAM	O	O	-	R	-	-	-	-
BKPSRAM	O	R	-	R	-	O	-	O
AHBSRAMx (x = 1, 2)	O	O	-	R	-	-	-	-
XSPIx (x = 1, 2, 3)	O	R	-	R	-	-	-	-
XSPIM	O	R	-	R	-	-	-	-
MCEx (x = 1, 2, 3, 4)	O	R	-	R	-	-	-	-
FMC	O	R	-	R	-	-	-	-
Backup registers	Y	R	-	R	-	R	-	R
Brownout reset (BOR)	Y	Y	Y	Y	Y	Y	Y	-
Programmable voltage detector (PVD)	O	O	O	O	O	-	-	-
Peripheral voltage monitor (PVM)	O	O	O	O	O	-	-	-
V08CAPH/V08CAPL monitoring	O	O	O	O	O	O	O	O
TEMPH/TEMPL monitoring	O	O	O	O	O	O	O	O
GPDMA1	O	R	-	R	-	-	-	-
HPDMA1	O	R	-	R	-	-	-	-

Table 55. Functionalities depending on system operating mode (continued)

Peripheral[1]	Run mode	Stop mode SVOS high		Stop mode SVOS low		Standby mode		V_{BAT} mode
		-	Wake-up capability	-	Wake-up capability	-	Wake-up capability	
High-speed internal (HSI)	O	O	-	-	-	-	-	-
High-speed external (HSE)	O	-	-	-	-	-	-	-
Low-speed internal (LSI)	O	O	-	O	-	O	-	-
Low-speed external (LSE)	O	O	-	O	-	O	-	O
Multi-speed internal (MSI)	O	O	-	-	-	-	-	-
HSE CSS (clock security system)	O	-	-	-	-	-	-	-
LSE CSS	O	O	O	O	O	O	O	O
RTC/auto wake-up	O	O	O	O	O	O	O	O
TAMP, number of tamper pins	7	7	O	4	O	4	O	4
USB1HS	O	R	O	R	-	-	-	-
USB2HS	O	R	O	R	-	-	-	-
UCPD1	O	R	O	R	-	-	-	-
SDMMCx (x = 1, 2)	O	R	-	R	-	-	-	-
FDCAN	O	R	-	R	-	-	-	-
MDIOS	O	R	O	R	-	-	-	-
ETH1	O	R	O	R	-	-	-	-
LPUART1	O	O	O	R	-	-	-	-
U(S)ARTx (x = 1 to 10)	O	O	O	R	-	-	-	-
I2Cx (x = 1 to 4)	O	O	O	R	-	-	-	-
I3Cx (x = 1, 2)	O	O	O	R	-	-	-	-
SPIx (x = 1 to 6)	O	O	O	R	-	-	-	-
SAIx (x = 1, 2)	O	R	-	R	-	-	-	-
ADF1	O	O	O	R	-	-	-	-
MDF1	O	O	O	R	-	-	-	-
DCMI	O	R	-	R	-	-	-	-
PSSI	O	R	-	R	-	-	-	-
DCMIPP	O	R	-	R	-	-	-	-
GPU	O	R	-	R	-	-	-	-
DMA2D	O	R	-	R	-	-	-	-

Table 55. Functionalities depending on system operating mode (continued)

Peripheral[1]	Run mode	Stop mode SVOS high		Stop mode SVOS low		Standby mode		V_{BAT} mode
		-	Wake-up capability	-	Wake-up capability	-	Wake-up capability	
GFXTIM	O	R	-	R	-	-	-	-
GFXMMU	O	R	-	R	-	-	-	-
JPEG	O	R	-	R	-	-	-	-
VENC	O	R	-	R	-	-	-	-
LTDC	O	R	-	R	-	-	-	-
ADCx (x = 1, 2)	O	R	-	R	-	-	-	-
VREFBUF	O	R	-	R	-	-	-	-
DTS	O	R	O	R	-	-	-	-
TIMx (x = 1 to 18)	O	R	-	R	-	-	-	-
LPTIMx (x = 1 to 5)	O	O	O	R	-	-	-	-
IWDG	O	O	O	O	O	O	O	-
WWDG	O	R	-	R	-	-	-	-
RNG	O	R	-	R	-	-	-	-
SAES	O	R	-	R	-	-	-	-
CRYP	O	R	-	R	-	-	-	-
HASH	O	R	-	R	-	-	-	-
CRC	O	R	-	R	-	-	-	-
GPIOs	O	O	O	O	O 4 pins	O	O 4 pins	-

1. Legend: Y = Yes (enable). O = Optional (disable by default. Can be enabled by software). R = data/state retained.
-= not available.
2. Only the first 80 Kbytes can optionally be retained (see *Section 10: SRAM configuration controller (RAMCFG)* for details).
3. Only the first 64 Kbytes can optionally be retained (see *Section 10: SRAM configuration controller (RAMCFG)* for details).

Debug mode

By default, the debug connection is lost if the application puts the MCU in Stop or Standby mode while debug features are used (the Cortex-M55 core is no longer clocked or powered).

However, by setting some configuration bits in DBGMCU control registers, the software can be debugged even when using the low-power modes extensively. For more details, refer to *Debug and low-power modes*.

13.6.2 Voltage scaling

The V_{CORE} domain is supplied from a single voltage regulator that supports voltage scaling with the following features:

- Run mode voltage scaling
  - VOS low
  - VOS high
- Stop mode voltage scaling
  - SVOS low
  - SVOS high

For more details on voltage scaling values, refer to the product datasheets.

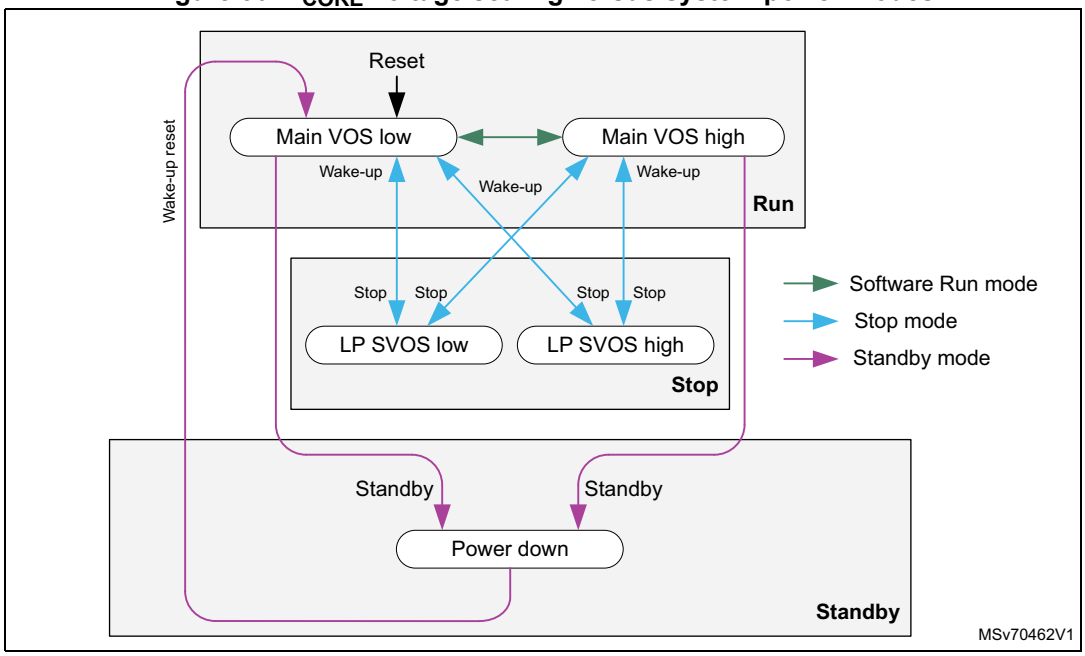
When using the internal SMPS step-down converter, after reset, the system starts on the lowest Run mode voltage scaling (VOS low). The voltage scaling can be changed on-the-fly by programming VOS in PWR_VOSCR, according to the required system performance.

Before entering Stop mode, the software can preselect the SVOS level in PWR_CPUCR. The Stop mode voltage scaling for SVOS low and SVOS high also sets the voltage regulator in low-power mode, to further reduce power consumption.

When exiting Stop mode, the MCU is in Run mode same range as before entering Stop mode. When exiting Standby mode, the Run mode voltage scaling is reset to the default VOS low value.

In Standby mode, the V_{CORE} supply is switched off.

Figure 30. V_{CORE} voltage scaling versus system power modes



13.6.3 Power management examples

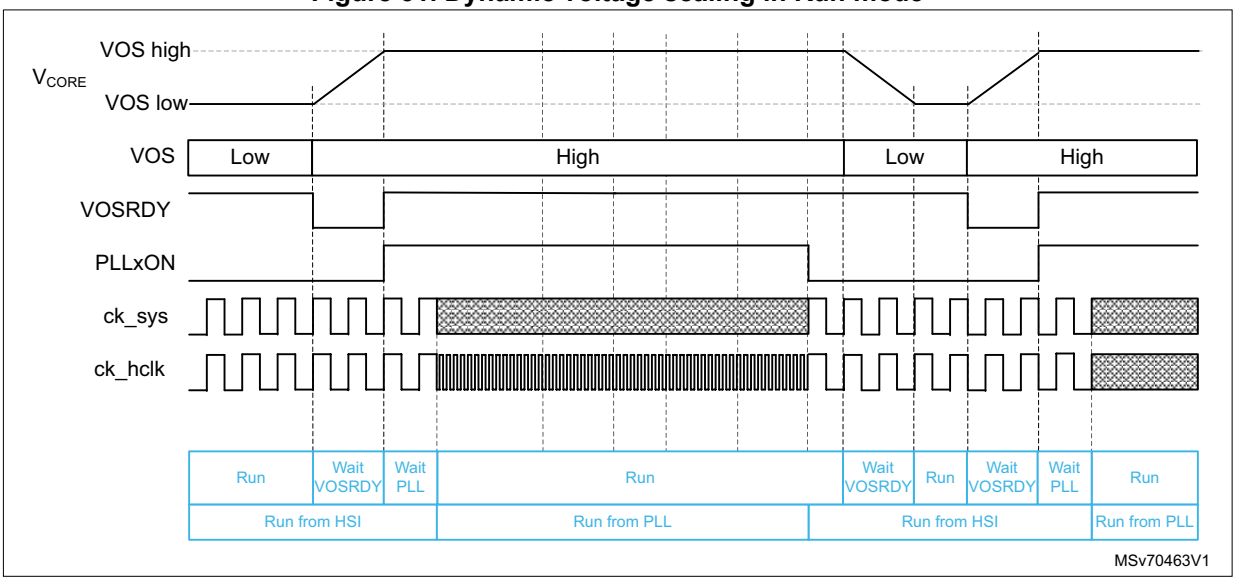
Example of V_{CORE} voltage scaling behavior in Run mode

Figure 31 illustrates the following system operation sequence example:

1. After reset, the system starts from HSI with VOS low.
2. The system performance is then increased to a high-speed clock from the PLL with voltage scaling VOS high. To do this:
   a) Program the voltage scaling to VOS high.
   b) Once the V_{CORE} supply has reached the required level indicated by VOSRDY, increase the clock frequency by enabling the PLL.
   c) Once the PLL is locked, switch the system clock.
3. Reduce the system performance to HSI clock with voltage scaling VOS low. To do this:
   a) Switch the clock to HSI.
   b) Disable the PLL.
   c) Decrease the voltage scaling to VOS low.
4. The system performance can then be increased to high-speed clock from the PLL. To do this:
   a) Program the voltage scaling to VOS high.
   b) Once the V_{CORE} supply has reached the required level indicated by VOSRDY, increase the clock frequency by enabling the PLL.
   c) Once the PLL is locked, switch the system clock.

When the system performance (clock frequency) is changed, VOS must be set accordingly.

Figure 31. Dynamic voltage scaling in Run mode



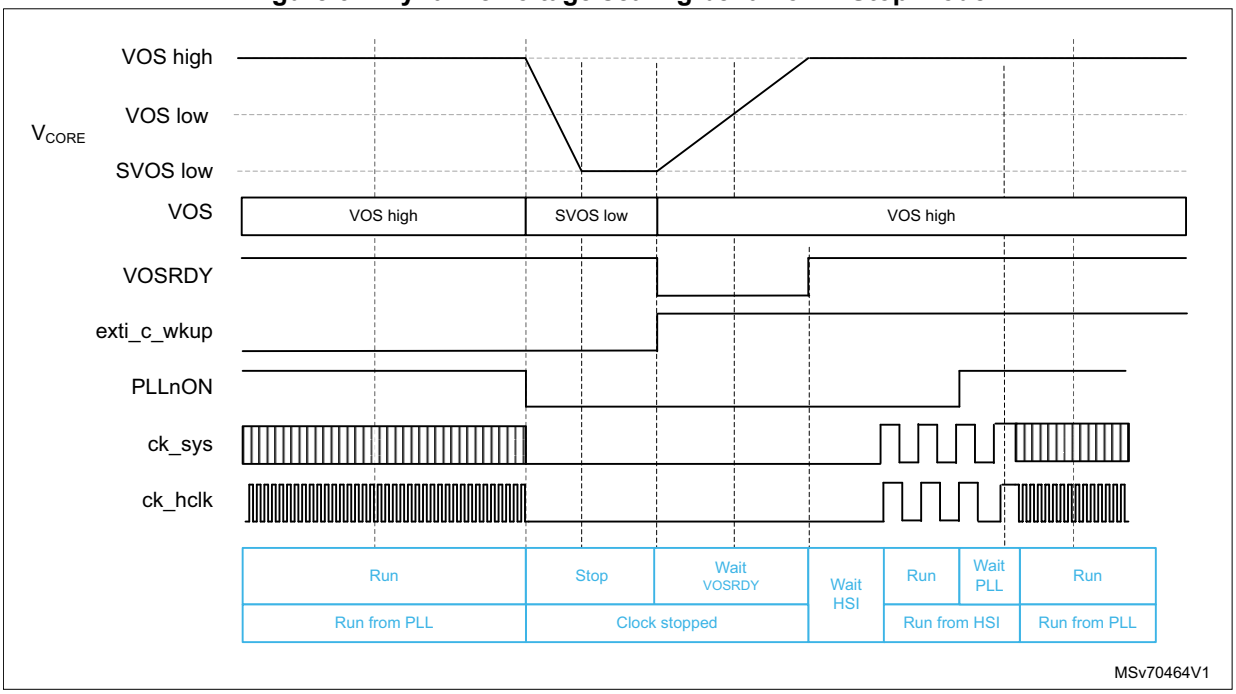
1. The status of the register bits at each step is shown in blue.

Example of V_{CORE} voltage scaling behavior in Stop mode

Figure 32 illustrates the following system operation sequence example:

1. The system runs from the PLL in high-performance mode (VOS high voltage scaling).
2. The CPU subsystem enters Stop mode, and the system enters Stop mode. The system clock is stopped, and the hardware lowers the voltage scaling to the software preselected SVOS low.
3. The CPU subsystem is then woken up. The system exits Stop mode, and the CPU subsystem exits Stop mode. The hardware always sets the same voltage range as before entering Stop mode, and waits for the requested supply level to be reached before enabling the HSI clock. Once the HSI clock is stable, the system clock is enabled and switch in Run mode under HSI clock.
4. The clock frequency can be increased by enabling the PLL. Once the PLL is locked, the system clock can be switched.

Figure 32. Dynamic voltage scaling behavior in Stop mode



1. The status of the register bits at each step is shown in blue.

Example of V_{CORE} voltage regulator and voltage scaling behavior in Standby mode

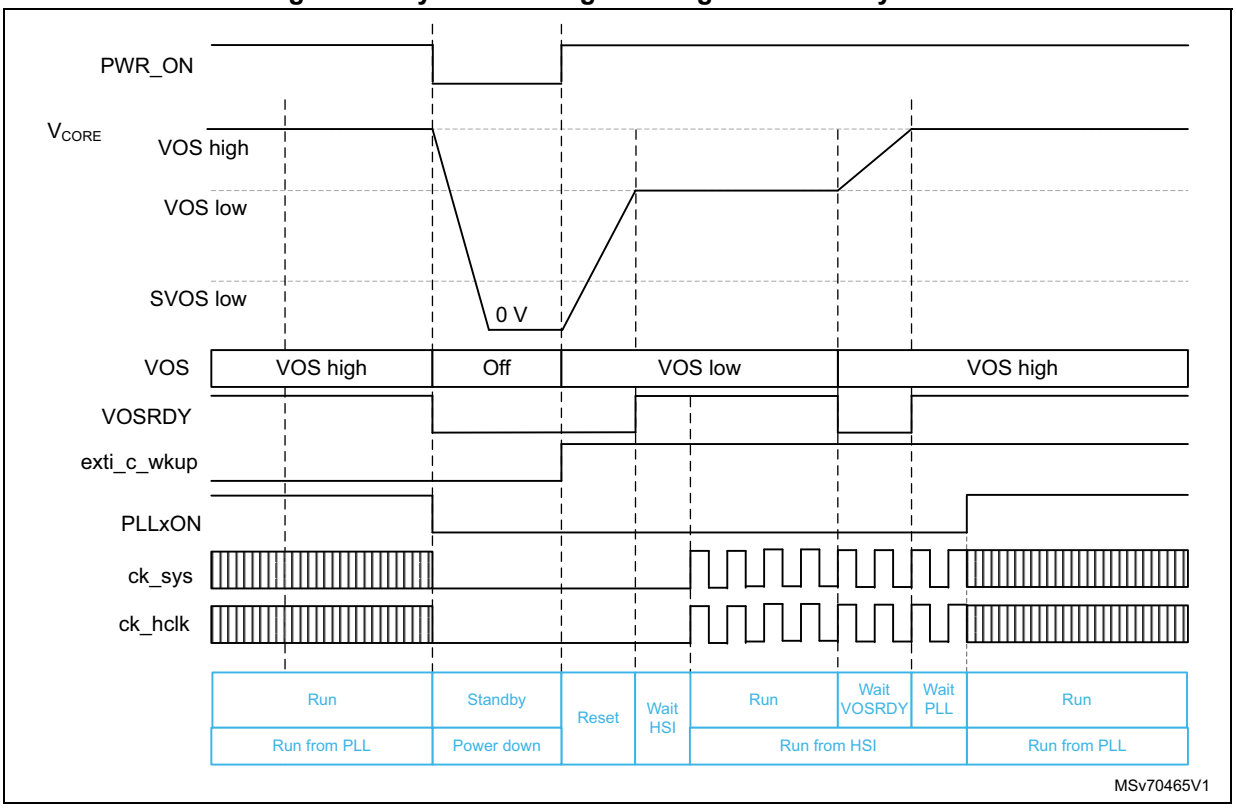
Figure 33 illustrates the following system operation sequence example:

1. The system runs from the PLL in high-performance mode (VOS high voltage scaling).
2. The CPU subsystem deallocates all the peripherals, and the V_{CORE} supply is switched off. The system enters Standby mode. The PWR_ON signal request $V_{DDA18PMU}$ and V_{DDSMPS} supplies to be powered off. If needed, a guarantee minimum PWR_ON pulse low time can be defined by POPL bits in PWR_CR1.
3. On a wake-up event, the external regulator suppling $V_{DDA18PMU}$ and V_{DDSMPS} is requested, via the PWR_ON signal. The system exits Standby mode. The hardware sets the voltage scaling to the default VOS low, and waits for the requested supply level

to be reached before enabling the default HSI oscillator. Once the HSI clock is stable, the system clock is enabled.

4. In a next step, increase the system performance with the following:
   a) The software first increases the voltage scaling to VOS high.
   b) Before enabling the PLL, the software waits for the requested supply level to be reached by monitoring VOSRDY.
   c) Once the PLL is locked, the system clock can be switched.

Figure 33. Dynamic Voltage Scaling from Standby mode



1. The status of the register bits at each step is shown in blue.

13.7 Low-power modes

Several low-power modes are available to save power when the CPU does not need to execute code (when waiting for an external event). The user must select the mode that gives the best compromise between low power consumption, short start-up time, and available wake-up sources:

- slowing down system clocks (see *Section 14.6.6: System clocks*)
- controlling individual peripheral clocks (see *Section 14.6.11: Peripheral clock-gating control*)
- low-power modes
  - Sleep (CPU clock stopped and still in Run mode)
  - Stop (system clock stopped)
  - Standby (system powered down)

13.7.1 Slowing down system clocks

In Run mode, the speed of the system clocks sys[a,b,c,d]_ck can be reduced. For more details, refer to *Section 14.6.6: System clocks*.

13.7.2 Controlling peripheral clocks

In Run mode, the peripherals bus clock can be stopped by configuring at any time PERxEN in RCC_xxxxENR to reduce power consumption.

To reduce power consumption in Sleep mode, the individual peripheral clocks can be disabled by configuring PERxLPEN bit in RCC_xxxxLPENR. For the peripherals still receiving a clock in Sleep mode, their clock can be slowed down before entering Sleep mode.

Example: For APB2 peripherals, the RCC register is RCC_APB2ENR, which contains bits like SPI4EN or SAI1EN.

13.7.3 Entering low-power modes

CPU subsystem Sleep and Stop modes are entered by the device when executing WFI or WFE instructions, or when SLEEPONEXIT is set on return-from-ISR in the Cortex-M55 System Control register.

The system can enter Stop or Standby mode when all EXTI wake-up sources are cleared.

13.7.4 Exiting low-power modes

The CPU subsystem exits Sleep mode through any interrupt or event; depending on how the low-power mode was entered:

- If the WFI instruction or return from ISR was used to enter the low-power mode, any peripheral interrupt acknowledged by the NVIC can wake up the system.
- If the WFE instruction is used to enter the low-power mode, the CPU exits the low-power mode as soon as an event occurs. The wake-up event can be generated by:
  - an NVIC IRQ interrupt

    When SEVONPEND = 0 in the Cortex-M55 System Control register, the interrupt must be enabled in the peripheral control register, and in the NVIC.

    When the MCU resumes from WFE, the peripheral interrupt pending bit and the NVIC peripheral IRQ channel pending bit in the NVIC interrupt clear pending register, have to be cleared. Only NVIC interrupts with sufficient priority wake up and interrupt the MCU.

    When SEVONPEND = 1 in the Cortex-M55 System Control register, the interrupt must be enabled in the peripheral control register and optionally in the NVIC. When the MCU resumes from WFE, the peripheral interrupt pending bit and, when enabled, the NVIC peripheral IRQ channel pending bit (in the NVIC interrupt clear pending register) have to be cleared.

    All NVIC interrupts wake up the MCU, even the disabled ones.

    Only enabled NVIC interrupts with sufficient priority wake-up and interrupt the MCU.
  - an event

    An EXTI line must be configured in event mode. When the CPU resumes from WFE, it is not necessary to clear the EXTI interrupt pending bit or the NVIC IRQ

channel pending bit, as pending bit corresponding to the event line is not set. The interrupt flag may have to be cleared in the peripheral.

The CPU subsystem exits Stop mode by enabling an EXTI interrupt or event depending on how the low-power mode was entered (see above).

The CPU subsystem exits Standby mode by enabling an external reset (NRST pin), an IWDG reset, an enabled WKUPx pin, or an RTC event.

Program execution restarts as after a system reset by fetching the vector tables in SYSCFG_INITSVTORCR and SYSCFG_INITNSVTORCR registers.

The default boot address can be changed to allow a fast restart on TCM when exiting a low power mode. The content of the vector table registers is maintained in Standby mode.

13.7.5 Sleep mode

The Sleep mode applies only to the CPU subsystem. In Sleep mode, the CPU clock is stopped. The CPU subsystem peripheral clocks operate according to the values of PERxLPEN bits in RCC_xxxxLPENR.

Entering Sleep mode

This mode is entered according to *Section 13.7.3*, when SLEEPDEEP bit in Cortex-M55 System Control register is cleared. Refer to *Table 56* for details on how to enter Sleep mode.

Exiting Sleep mode

The Sleep mode is exited according to *Section 13.7.4*. Refer to the table below for more details on how to exit Sleep mode.

Table 56. Sleep mode

Sleep mode	Description
Mode entry	WFI or WFE while: – SLEEPDEEP = 0 (refer to the Cortex-M55 System Control register) – CPU NVIC interrupts and events cleared
	On return from ISR while: – SLEEPDEEP = 0 and – SLEEPONEXIT = 1 (refer to the Cortex-M55 System Control register) – CPU NVIC interrupts and events cleared
Mode exit	If WFI or return from ISR was used for entry: – any interrupt enabled in NVIC (see *Table 134: STM32N6x5/x7xx vector table*) If WFE was used for entry and SEVONPEND = 0: – any event (see *Section 25: Extended interrupts and event controller (EXTI)*) If WFE was used for entry and SEVONPEND = 1: – any interrupt even when disabled in NVIC (see *Table 134: STM32N6x5/x7xx vector table*) or any event (see *Section 25: Extended interrupts and event controller (EXTI)*
Wake-up latency	None

13.7.6 Stop mode

The Stop mode applies to the MCU. In Stop mode, the CPU clock is stopped. All CPU subsystem peripheral clocks are stopped too, and only the CPU subsystem peripherals having a PERxLPEN bit operate accordingly.

In Stop mode, CPU subsystem peripherals having a kernel clock request can still request their kernel clock. For the peripheral having a PERxLPEN bit, this bit must be set to be able to request the kernel clock.

HSI or MSI can remain enabled in system Stop mode (HSISTOPEN and MSISTOPEN set in RCC_STOPCR). After exiting Stop mode, the clock is quickly available as kernel clock for peripherals. Other system oscillator sources are stopped in Stop mode, and require a starting time after exiting Stop mode.

In Stop mode and SVOS high, peripherals using the LSI or LSE clock, and peripherals having a kernel clock request, are still able to operate.

In system Stop mode, the following features can be selected to remain active by programming individual control bits:

- independent watchdog (IWDG)

  The IWDG is started by writing to its key register or by hardware option. Once started, it cannot be stopped except by a reset (see *Section 59: Independent watchdog (IWDG)*.
- real-time clock (RTC) that is configured via RTCEN in RCC_BDCR
- internal RC oscillator (LSI RC) that is configured via LSION in RCC_CSR
- external 32.768 kHz oscillator (LSE OSC) that is configured via LSEON in RCC_CSR.
- peripherals capable of running on the LSI or LSE clock
- peripherals having a kernel clock request
- internal RC oscillators (HSI and MSI) that are configured via HSISTOPEN and MSISTOPEN in RCC_STOPCR
- The ADCs can also consume power during Stop mode, unless they are disabled before entering this mode. To disable them, ADON bit in ADC_CR2 must be written to 0.

The selected SVOS low levels add an additional start-up delay when exiting from system Stop mode.

Entering Stop mode

The Stop mode is entered according to *Section 13.7.3*, when SLEEPDEEP bit in the Cortex-M55 System Control register is set.

The low-power mode security filter must be disabled by system option bytes (NRST_STOP) to enter Stop mode. Refer to *Section 14.5.4: Low-power mode security reset (lpwr_rst)* for details.

Refer to the tables below for details on how to enter to Stop mode.

Table 57. Stop mode SVOS high

Stop mode	Description
Mode entry	WFI or WFE while: SLEEPDEEP = 1 (refer to the Cortex-M55 System Control register) CPU NVIC interrupts and events cleared All CPU EXTI wake-up sources cleared
	On return from ISR while: SLEEPDEEP = 1 and SLEEPONEXIT = 1 (refer to the Cortex-M55 System Control register) CPU NVIC interrupts and events cleared All CPU EXTI wake-up sources cleared
Mode exit	If WFI or return from ISR was used for entry: EXTI interrupt enabled in NVIC (see *Table 134: STM32N6x5/x7xx vector table*, for peripheral that are not stopped or powered down) If WFE was used for entry and SEVONPEND = 0: EXTI event: Refer to *Section 25: Extended interrupts and event controller (EXTI)* for peripheral that are not stopped or powered down. If WFE was used for entry and SEVONPEND = 1: EXTI Interrupt even when disabled in NVIC: refer to *Table 134: STM32N6x5/x7xx vector table* or EXTI event: refer to *Section 25: Extended interrupts and event controller (EXTI)* for peripheral that are not stopped or powered down.
Wake-up latency	SMPS wake-up time from low-power mode (Run mode operating supply level to restore) + EXTI and RCC wake-up synchronization (see *Section 14: Reset and clock control (RCC)*)

Table 58. Stop mode SVOS low

Stop mode	Description
Mode entry	WFI or WFE while: SLEEPDEEP = 1 (refer to the Cortex-M55 System Control register) CPU NVIC interrupts and events cleared All CPU EXTI wake-up sources cleared
	On return from ISR while: SLEEPDEEP = 1 and SLEEPONEXIT = 1 (refer to the Cortex-M55 System Control register) CPU NVIC interrupts and events cleared All CPU EXTI wake-up sources cleared

Table 58. Stop mode SVOS low (continued)

Stop mode	Description
Mode exit	WKUP pins rising or falling edge, RTC alarm (alarm A and alarm B), RTC wake-up, tamper event, timestamp event, external reset in NRST pin, IWDG reset
Wake-up latency	SMPS wake-up time from low-power mode (Run mode operating supply level to restore) + EXTI and RCC wake-up synchronization (see *Section 14.5.12: Power-on and wake-up sequences*)

To allow peripherals having a kernel clock request to operate in Stop mode, the system must use SVOS high.

Note: Use a DSB instruction to ensure that outstanding memory transactions complete before entering Stop mode.

Exiting Stop mode

The Stop mode is exited according to *Section 13.7.4*.

Refer to *Table 57* and *Table 58* for more details on how to exit Stop mode.

When exiting Stop mode, the MCU is in Run mode same range as before entering Stop mode. The system starts on HSI or MSI.

STOPF status flag in PWR_CPUCR indicates that the system exited Stop mode.

I/O states in Stop mode

I/O pin configuration remains unchanged in Stop mode.

13.7.7 Standby mode

The Standby mode is used to achieve the lowest power consumption. Like Stop mode, it is based on CPU subsystem Stop mode. However, the V_{CORE} supply regulator is powered off.

When the system enters Standby mode, the voltage regulator is disabled. The complete V_{CORE} domain is consequently powered off. PLLs, HSI oscillator, MSI oscillator, and HSE oscillator are also switched off. SRAM and register contents are lost except for backup domain registers (RTC registers, RTC backup register, and backup RAM), and the retention domain (see *Section 13.4.4* and *Section 13.4.5*).

In system Standby mode, the following features can be selected by programming individual control bits:

- independent watchdog (IWDG)

  The IWDG is started by programming its key register or by hardware option. Once started, it cannot be stopped except by a reset (see *Section 59.3: IWDG implementation*).
- real-time clock (RTC) that is configured via RTCEN in RCC_BDCR
- internal RC oscillator (LSI RC) that is configured by LSION in RCC_CSR
- External 32.768 kHz oscillator (LSE OSC) that is configured by LSEON in RCC_CSR

Entering Standby mode

The Standby mode is entered according to *Section 13.7.3*, when PDDS is set to one in PWR_CPUCR.

The low-power mode security filter must be disabled by system option bytes (nRST_STDBY) to enter in Standby mode. Refer to *Section 14.5.4: Low-power mode security reset (lpwr_rst)* for details.

Refer to *Table 60* for more details on how to enter Standby mode.

Exiting Standby mode

The Standby mode is exited according to *Section 13.7.4*.

The system exits Standby mode when an external reset (NRST pin), an IWDG reset, a WKUP pin event, an RTC alarm, a tamper event, or a timestamp event is detected.
All registers are reset after waking up from Standby except for power control and status registers:

- PWR_CR1, PWR_CR4, PWR_BDCR1, PWR_BDCR2
- SBF and PDDS in PWR_CPUCR
- VDDIO4VRSTBY and VDDIO4VRSEL in PWR_SVMCR1
- VDDIO5VRSTBY and VDDIO5VRSEL in PWR_SVMCR2
- VDDIOxVRSEL in PWR_SVMCR3
- PWR_WKUPCR, PWR_WKUPSR, PWR_WKUPEPR

A pad or watchdog reset during Standby mode causes the program execution to start as after an application reset (rerun the boot ROM).

An enabled WKUP pin or an RTC event during Standby mode causes the program execution to start as after a system reset (fetch from SYSCFG_INITSVTORCR)

The default boot address can be changed to allow a fast restart on TCM when exiting a low power mode.

Refer to *Table 60* for more details on how to exit Standby mode.

SBF in PWR_CPUCR indicates that the system has exited Standby mode.

Table 59. Standby and Stop flags

SBF	STOPF	Description
0	1	System has been in Stop mode.
1	0	System has been in Standby mode.

Table 60. Standby mode

Standby mode	Description
Mode entry	The CPU subsystem is in Stop mode, and there is no active EXTI wake-up source. PDDS bit for select Standby All WKUPF bits in PWR_WKUPSR cleared
Mode exit	WKUP pins rising or falling edge, RTC alarm (alarm A and alarm B), RTC wake-up, tamper event, timestamp event, external reset in NRST pin, IWDG reset
Wake-up latency	System reset phase (see *Section 14.5.2: System and application resets (sys_rst, nreset_rstn)*)

I/O states in Standby mode

In Standby mode, all I/O pins are high impedance without pull, except for:

- RESET and PWR_ON pins (still available)
- RTC_AF1 pin if configured for tamper, time stamp, RTC alarm out, or RTC clock calibration out
- WKUP pins (if enabled): WKUP pin pull configuration can be defined through WKUPPUPD bits in PWR_WKUPEPR

13.7.8 Power mode output pins

To help the debug, the following signals are available as alternate functions:

- PWR_CSLEEP

  When set, PWR_CSLEEP indicates that the CPU is in Sleep mode:

  – WFI or WFE has been executed, and CPU stops execution.

  When cleared, PWR_CSLEEP indicates that the CPU is in Run mode.
- PWR_CSTOP

  When set, PWR_CSTOP indicates that the device is in Stop mode, meaning that the following conditions are true:

  – WFI or WFE has been executed with CPU SLEEPDEEP = 1.

  When cleared, PWR_CSTOP indicates that the device is in Run mode.

Table 61. Power mode output states versus MCU power modes

PWR_CSTOP	PWR_CSLEEP	MCU power modes(1)
X	0	Run mode (CPU executing)
0	1	Sleep mode (CPU sleep)
1	1	Stop mode (CPU SleepDeep)

1. PWR_CSLEEP and PWR_CSTOP are generated in the core domain, then they are not driven in Standby mode.

13.8 PWR security and privileged protection

After any application reset or system reset, the PWR does not filter any access (default configuration: nonsecure, any privileged) until the trusted agent has programmed the security and privileged protection.

The PWR is able to protect register bits from being modified by nonsecure and unprivileged accesses. The protection can be activated for the following features through PWR_SECCFGR and PWR_PRIVCFGR:

- system supply configuration
- voltage scaling
- low-power mode
- wake-up (WKUP) pins
- voltage detection and monitoring
- V_{BAT} mode

Table 62 summarizes the security and privilege configuration bits.

Table 62. PWR register security overview

PWR register	Secure/privilege configuration bit	Access type(1)	Description
PWR_CR1	PWR_SECCFGR.SEC[0] PWR_PRIVCFGR.PRIV[0]	RW/RWO	System supply configuration
PWR_CR2	PWR_SECCFGR.SEC[1] PWR_PRIVCFGR.PRIV[1]	RW	Programmable voltage detector
PWR_CR3	PWR_SECCFGR.SEC[2] PWR_PRIVCFGR.PRIV[2]	RW	V_{DDCORE} monitor
PWR_CR4	PWR_SECCFGR.SEC[3] PWR_PRIVCFGR.PRIV[3]	RW	I-TCM, D-TCM, and I-TCM FLEX MEM low-power control
PWR_VOSCR	PWR_SECCFGR.SEC[4] PWR_PRIVCFGR.PRIV[4]	RW	Voltage scaling selection
PWR_BDCR1	PWR_SECCFGR.SEC[5] PWR_PRIVCFGR.PRIV[5]	RW	V_{BAT} and temperature monitor
PWR_BDCR2		RW	BKPSRAM low-power control
PWR_DBPCR		RW	Disable backup domain write protection
PWR_CPUCR	PWR_SECCFGR.SEC[6] PWR_PRIVCFGR.PRIV[6]	RW	CPU power control
PWR_SVMCR1	PWR_SECCFGR.SEC[7] PWR_PRIVCFGR.PRIV[7]	RW	V_{DDIO4} peripheral voltage monitor
PWR_SVMCR2		RW	V_{DDIO5} peripheral voltage monitor
PWR_SVMCR3		RW	V_{DDIO2}, V_{DDIO3}, $V_{DD33USB}$, and $V_{DDA18ADC}$ peripheral voltage monitor
PWR_WKUPCR	PWR_SECCFGR.WKUPSEC[x] PWR_PRIVCFGR.WKUPPRIV[x] (X = 1 to 4)	W	Wake-up pin x (x = 1 to 4) clear register
PWR_WKUPSR		R	Wake-up pin x (x = 1 to 4) status register
PWR_WKUPEPR		RW	Wake-up pin x (x = 1 to 4) enable and polarity register

1. RW: read or write, R: read, W: write, RWO: read or write once.
   RC_W0: read or clear by writing 0. Writing 1 has no effect on the bit value.

13.8.1 Secure/nonsecure access filtering

To enable the filtering access based on this attribute, the authorized master agent must set SECx in PWR_SECCFGR related to the PWR feature.

When a register is configured as secure, read and write operations are allowed only by a secure access. Nonsecure read or write accesses are denied (RAZ/WI).

An illegal secure access event is generated to the IAC (illegal access controller). There is no bus error generated.

When a register is configured as nonsecure, read and write operations are allowed by both secure and nonsecure accesses.

13.8.2 Privileged/unprivileged access filtering

To enable the filtering access based on this attribute, the authorized master agent has to set PRIVx in PWR_PRIVCFGR related to the PWR feature.

When a register is configured as privileged, read and write operations are only allowed by a privileged access. Unprivileged read or write accesses are denied (RAZ/WI). An illegal privileged access event is generated to the IAC. There is no bus error generated.

When a register is configured as unprivileged, read and write operations are allowed by both privileged and unprivileged accesses.

13.9 PWR registers

The PWR registers can be accessed in word, half-word and byte format, unless otherwise specified.

13.9.1 PWR control register 1 (PWR_CR1)

Address offset: 0x000

Reset value: 0x0000 002X

This register is not reset by wake-up from Standby mode and application reset (such as NRST or IWDG), but only reset by V_{DD} POR.

Secure restriction is defined by SEC0 in PWR_SECCFGR. Privileged restriction is defined by PRIV0 in PWR_PRIVCFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	POPL[4:0]				
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPDS08V	MODE_PDN	Res.	SDEN	Res.	Res.
										rw	rw		rwo		

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **POPL[4:0]**: pwr_on pulse low configuration.

These bits are set and cleared by software. They define the minimum guaranteed duration of the pwr_on low pulse in Standby mode. The LSI oscillator is automatically enabled when needed by the POPL pulse low configuration.

00000: No guaranteed minimum low time

00001: ~ 1 ms guaranteed minimum low time (1 x 32 LSI cycles)

00010: ~ 2 ms guaranteed minimum low time (2 x 32 LSI cycles)

...

11111: ~ 31 ms guaranteed minimum low time (31 x 32 LSI cycles)

Bits 15:6 Reserved, must be kept at reset value.

Bit 5 **LPDS08V:** SMPS low-power mode enable (SVOS high only)

This bit is used to keep the SMPS in PWM mode (MR) in Stop SVOS high.

0: SMPS low-power mode disabled

1: SMPS low-power mode enabled (high-efficiency mode) (default)

Bit 4 **MODE_PDN**: Enables the pull down on output voltage during power-down mode
0: Pull-down disabled. The output is in high impedance during the shutdown (default).
1: Pull-down enabled

Bit 3 Reserved, must be kept at reset value.

Bit 2 **SDEN**: SMPS step-down converter enable
0: SMPS step-down converter disabled
1: SMPS step-down converter enabled (default)
This bit must be written as soon as possible after device start, and it can be written only once after POR.
When in bypass mode, SMPS is off, reading this bit returns 0, Write it to 0 as soon as possible after POR, or it will be impossible to enter low power modes. When SMPS is on, reading this bit returns 1 (default).

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

13.9.2 PWR control register 2 (PWR_CR2)

Address offset: 0x004

Reset value: 0x0000 0000

Reset on any system reset.

Secure restriction is defined by SEC1 in PWR_SECCFGR. Privilege restriction is defined by PRIV1 in PWR_PRIVCFGR.

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

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

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **PVDO**: Programmable voltage detect output
This bit is set and cleared by hardware. It is valid only if PVD is enabled by PVDEN.
0: Voltage level on PVD_IN is equal or higher than the internal VREFINT.
1: Voltage level on PVD_IN is lower than the internal VREFINT.
Note: The PVD is disabled in Standby mode, and after a system reset. For this reason, this bit is equal to 0 after Standby and system reset.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **PVDEN:** Programmable voltage detector enable
This bit is read only when PVDL is set in SYSCFG_CBR (when PVDL is set, there is no bus errors generated when writing this register).
0: PVD disabled
1: PVD enabled

13.9.3 PWR control register 3 (PWR_CR3)

Address offset: 0x008

Reset value: 0x0000 0000

Reset on any system reset.

Secure restriction is defined by SEC2 in PWR_SECCFGR. Privilege restriction is defined by PRIV2 in PWR_PRIVCFGR.

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

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

Bit 9 **VCOREH**: Monitored V_{DDCORE} level above high threshold

0: V_{DDCORE} level below high threshold level, or monitor disabled

1: V_{DDCORE} level equal or above high threshold level

Bit 8 **VCOREL**: Monitored V_{DDCORE} level above low threshold

0: V_{DDCORE} level above low threshold level, or monitor disabled

1: V_{DDCORE} level equal or below low threshold level

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **VCORELLS**: V_{DDCORE} voltage detector low-level selection

This bit selects the low-voltage threshold detected by the monitoring.

0: V_{DDCORE} low-voltage threshold 1 selected (VOS low)

1: V_{DDCORE} low-voltage threshold 2 selected (VOS high)

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **VCOREMONEN:** V_{DDCORE} monitoring enable

When this bit is set, the V_{DDCORE} supply monitoring is enabled.

0: V_{DDCORE} monitoring disabled

1: V_{DDCORE} monitoring enabled

13.9.4 PWR control register 4 (PWR_CR4)

Address offset: 0x00C

Reset value: 0x0000 0000

This register is not reset by wake-up from Standby mode, but by any application reset (such as NRST or IWDG). Access 6 wait states when writing this register.

When a system reset occurs during the register write cycle, the written data are not guaranteed.

Secure restriction is defined by SEC3 in PWR_SECCFGR. Privilege restriction is defined by PRIV3 in PWR_PRIVCFGR.

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

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

Bit 4 **TCMFLXRBSEN:** I-TCM FLEXMEM backup supply enable (used to maintain I-TCM FLEX MEM content in Standby mode)

When this bit is set, the I-TCM FLEXMEM is supplied from backup regulator in Standby mode. When this bit is reset, the I-TCM FLEXMEM can still be used in Run and Stop modes, but its content is lost in Standby mode.

0: I-TCM FLEXMEM backup supply disabled

1: I-TCM FLEXMEM backup supply enabled in Standby mode

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **TCMRBSEN:** I-TCM and D-TCM RAMs backup supply enable (used to maintain TCM RAMs content in Standby mode)

When this bit is set, I-TCM and D-TCM RAMs are supplied from backup regulator in Standby mode. When this bit is reset, I-TCM and D-TCM RAMs can still be used in Run and Stop modes, but their content is lost in Standby mode.

0: I-TCM and D-TCM RAMs backup supply disabled

1: I-TCM and D-TCM RAMs backup supply enabled in Standby mode

13.9.5 PWR voltage scaling control register (PWR_VOSCR)

Address offset: 0x020

Reset value: 0x0002 0002

Reset on any system reset.

Secure restriction is defined by SEC4 bit in PWR_SECCFGR. Privilege restriction is defined by PRIV4 bit in PWR_PRIVCFGR.

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

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

Bit 17 **ACTVOSRDY**: Voltage level ready bit for currently used ACTVOS
This bit is set to 1 by hardware when the SMPS step-down converter is disabled in PWR_CR1.
0: Voltage level invalid, above or below current ACTVOS
1: Voltage level valid, at current ACTVOS

Bit 16 **ACTVOS**: VOS currently applied for V_{CORE} voltage scaling selection
When the SMPS step-down converter is disabled, this bit is tied to 0, when the converter is enabled, this bit reflects the last VOS value applied to the PMU.
0: VOS low
1: VOS high

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

Bit 1 **VOSRDY**: VOS ready bit for V_{CORE} voltage scaling output selection
When the internal SMPS step-down converter is used, this bit indicates that all features allowed by the selected VOS can be used.
This bit is set to 1 by hardware when the SMPS step-down converter is disabled in PWR_CR1.
0: Not ready, voltage level below VOS selected level
1: Ready, voltage level at or above VOS selected level

Bit 0 **VOS**: Voltage scaling selection according to performance
When the SMPS step-down converter is disabled, this bit has no impact on the external supply voltage. It can be read/written to keep track of the external supply voltage setting.
When the SMPS step-down converter is enabled this bit controls the V_{CORE} voltage level, and is used to obtain the best trade-off between power consumption and performance:
– When increasing the performance, the voltage scaling must be changed before increasing the system frequency.
– When decreasing performance, the system frequency must be decreased before changing the voltage scaling.
0: VOS low level (default)
1: VOS high level

Note: Refer to electrical characteristics in the datasheet for more details.

13.9.6 PWR backup domain control register 1 (PWR_BDCR1)

Address offset: 0x024

Reset value: 0x0000 0000

This register is not reset by wake-up from Standby mode, application reset (such as NRST or IWDG), or V_{DD} POR, but it only by V_{SW} POR and VSWRST.

This register must not be accessed when VSWRST in RCC_BDCR resets the V_{SW} domain.

After reset, PWR_BDCR1 is write-protected. Prior to modifying its content, DBP must be set in PWR_DBPCR to disable the write protection.

Secure restriction is defined by SEC5 in PWR_SECCFGR. Privilege restriction is defined by PRIV5 in PWR_PRIVCFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TEMPH	TEMPL	V08CAPH	V08CAPL
												r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MONEN
															rw

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **TEMPH**: Temperature level monitoring versus high threshold
0: Temperature below high threshold level
1: Temperature equal or above high threshold level

Bit 18 **TEMPL**: Temperature level monitoring versus low threshold
0: Temperature above low threshold level
1: Temperature equal or below low threshold level

Bit 17 **V08CAPH**: V_{08CAP} level monitoring versus high threshold
0: V_{BAT} level below high threshold level.
1: V_{BAT} level equal or above high threshold level

Bit 16 **V08CAPL**: V_{08CAP} level monitoring versus low threshold
0: V_{BAT} level above low threshold level
1: V_{BAT} level equal or below low threshold level

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **MONEN:** V_{BAT} and temperature monitoring enable
When this bit is set, the V_{BAT} supply and temperature monitoring are enabled.
0: V_{BAT} and temperature monitoring disabled
1: V_{BAT} and temperature monitoring enabled

13.9.7 PWR backup domain control register 2 (PWR_BDCR2)

Address offset: 0x028

Reset value: 0x0000 0000

This register is not reset by wake-up from Standby mode, application reset (such as NRST or IWDG), or V_{DD} POR, but it is only reset by V_{SW} POR and VSWRST.

This register must not be accessed when VSWRST in RCC_BDCR resets the V_{SW} domain.

After reset, PWR_BDCR2 is write-protected. Prior to modifying its content, DBP must be set in PWR_DBPCR register to disable the write protection.

Secure restriction is defined by SEC5 in PWR_SECCFGR. Privilege restriction is defined by PRIV5 in PWR_PRIVCFGR.

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

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

Bit 0 **BKPRBSEN:** BKPSRAM backup supply enable (used to maintain BKPSRAM content in Standby and V_{BAT} modes).

When this bit is set, the backup ram is supplied from backup regulator in Standby and V_{BAT} modes. When this bit is reset, the backup ram can still be used in Run and Stop modes, but its content is lost in Standby and V_{BAT} modes.

0: BKPSRAM backup supply disabled

1: BKPSRAM backup supply enabled

13.9.8 PWR disable backup protection control register (PWR_DBPCR)

Address offset: 0x02C

Reset value: 0x0000 0000

Reset on any system reset

Secure restriction is defined by SEC5 in PWR_SECCFGR. Privilege restriction is defined by PRIV5 in PWR_PRIVCFGR.

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

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

Bit 0 **DBP*:*** Disable backup domain write protection

In reset state, all registers and SRAM in backup domain are protected against parasitic write access. This bit must be set to enable write access to these registers.

0: Write access to backup domain disabled

1: Write access to backup domain enabled

13.9.9 PWR CPU control register (PWR_CPUCR)

Address offset: 0x030

Reset value: 0x0001 0000

See individual bits for reset condition. Access six wait states when writing this register.

When a system reset occurs during the register write cycle, the written data are not guaranteed.

Secure restriction is defined by SEC6 in PWR_SECCFGR. Privilege restriction is defined by PRIV6 in PWR_PRIVCFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SVOS
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	SBF	STOPF	Res.	Res.	Res.	Res.	Res.	Res.	CSSF	PDDS
						r	r							rc_w1	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **SVOS**: System Stop mode voltage scaling selection
This bit is reset on any system reset.
0: SVOS low
1: SVOS high (default)

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **SBF**: Standby flag
This bit is set by hardware and cleared only by a POR or by setting the CSSF bit.
0: System has not been in Standby mode.
1: System has been in Standby mode.

Bit 8 **STOPF**: Stop flag
This bit is set by hardware and cleared by any reset, or by setting the CSSF bit.
0: System has not been in Stop mode.
1: System has been in Stop mode.

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

Bit 1 **CSSF**: Clear Standby and Stop flags (always read as 0)
This bit is reset on any system reset.
0: No effect
1: When written, clears the CPU flags (STOPF, SBF).
Note: This bit is cleared to 0 by hardware.

Bit 0 **PDDS**: Power-down deepsleep selection
This bit is reset only by a V_{DD} POR reset (not reset when exit standby mode). It allows the CPU to define the deepsleep mode for the system.
0: Stop mode when device enters deepsleep
1: Standby mode when device enters deepsleep

13.9.10 PWR supply voltage monitoring control register 1 (PWR_SVMCR1)

Address offset: 0x034

Reset value: 0x0000 0000

See individual bits for reset condition. Access six wait states when writing this register.

When a system reset occurs during the register write cycle, the written data are not guaranteed.

Secure restriction is defined by SEC7 in PWR_SECCFGR. Privilege restriction is defined by PRIV7 in PWR_PRIVCFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	VDDIO4 VRSTBY	VDDIO4 VRSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VDDIO4 RDY
						rw	rw								r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	VDDIO4 SV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VDDIO4 VMEN
							rw								rw

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

Bit 25 **VDDIO4VRSTBY:** V_{DDIO4} I/O voltage range Standby mode

When this bit is set, the VDDIO4VRSEL configuration is retained in Standby mode.

This bit must be set if the V_{DDIO4} is in 1v8 range in Standby mode, and when exiting Standby mode. It must not be set when V_{DDIO4} is in 3v3 range in Standby mode, or when exiting Standby mode. This bit is not reset by wake-up from Standby mode, but by any application reset (such as NRST or IWDG).

0: VDDIO4VRSEL not retained in Standby mode

1: VDDIO4VRSEL retained in Standby mode

Bit 24 **VDDIO4VRSEL:** V_{DDIO4} I/O voltage range selection

This bit is used to select the voltage range supported by I/Os supplied by V_{DDIO4}. It is not reset by wake-up from Standby mode, but by any application reset (such as NRST or IWDG), when VDDIO4VRSTBY is set, otherwise this bit is reset on any system reset.

0: 3v3 voltage range selected. If V_{DDIO4} is in 1v8 range with this setting, I/Os work in degraded mode.

1: 1v8 voltage range selected. HSLV_VDDIO4 option bit must be set to allow 1v8 voltage range operation. Setting this configuration while V_{DDIO4} is in 3v3 range damages the device.

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

Bit 16 **VDDIO4RDY**: V_{DDIO4} ready

0: V_{DDIO4} is below the threshold of the V_{DDIO4} voltage monitor.

1: V_{DDIO4} is equal or above the threshold of the V_{DDIO4} voltage monitor.

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **VDDIO4SV**: V_{DDIO4} independent I/O supply valid.

This bit is used to validate the V_{DDIO4} supply for electrical and logical isolation purpose. Setting this bit is mandatory to use PC[1], PC[12:6], and PH[9:2] I/Os. If V_{DDIO4} is not always present in the application, the V_{DDIO4} voltage monitor can be used to determine whether this supply is ready or not. This bit is reset on any system reset.

0: V_{DDIO4} is not present. Logical and electrical isolation is applied to ignore this supply.

1: V_{DDIO4} is valid.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **VDDIO4VMEN:** V_{DDIO4} independent I/O voltage monitor enable

This bit is reset on any system reset.

0: V_{DDIO4} voltage monitor disabled

1: V_{DDIO4} voltage monitor enabled

13.9.11 PWR supply voltage monitoring control register 2 (PWR_SVMCR2)

Address offset: 0x038

Reset value: 0x0000 0000

See individual bits for reset condition. Access six wait states when writing this register.

When a system reset occurs during the register write cycle, the written data are not guaranteed.

Secure restriction is defined by SEC7 in PWR_SECCFGR. Privilege restriction is defined by PRIV7 in PWR_PRIVCFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	VDDIO5 VRSTBY	VDDIO5 VRSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VDDIO5 RDY
						rw	rw								r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	VDDIO5 SV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VDDIO5 VMEN
							rw								rw

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

Bit 25 **VDDIO5VRSTBY:** V_{DDIO5} I/O voltage range Standby mode

When this bits is set, VDDIO5VRSEL configuration is retained in Standby mode.

This bit must be set if the V_{DDIO5} is in 1v8 range in Standby mode, and when exiting Standby mode. It must not be set when V_{DDIO5} is in 3v3 range in Standby mode, or when exiting Standby mode. This bit is not reset by wake-up from Standby mode, but by any application reset (such as NRST or IWDG).

0: VDDIO5VRSEL not retained in Standby mode

1: VDDIO5VRSEL retained in Standby mode.

Bit 24 **VDDIO5VRSEL:** V_{DDIO5} I/O voltage range selection

This bit is used to select the voltage range supported by I/Os supplied by V_{DDIO5}. It is not reset by wake-up from Standby mode, but by any application reset (such as NRST or IWDG), when VDDIO5VRSTBY bit is set, otherwise this bit is reset on any system reset.

0: 3v3 voltage range selected. If V_{DDIO5} is in 1v8 range with this setting, I/Os work in degraded mode.

1: 1v8 voltage range selected. HSLV_VDDIO5 option bit must be set to allow 1v8 voltage range operation. Setting this configuration while V_{DDIO5} is in 3v3 range damages the device.

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

Bit 16 **VDDIO5RDY**: V_{DDIO5} ready

0: V_{DDIO5} is below the threshold of the V_{DDIO5} voltage monitor.

1: V_{DDIO5} is equal or above the threshold of the V_{DDIO5} voltage monitor.

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **VDDIO5SV**: V_{DDIO5} independent supply valid

This bit is used to validate the V_{DDIO5} supply for electrical and logical isolation purpose. Setting this bit is mandatory to use PC[0], PC[5:2] and PE[4] I/Os. If V_{DDIO5} is not always present in the application, the V_{DDIO5} voltage monitor can be used to determine whether this supply is ready or not. This bit is reset on any system reset.

0: V_{DDIO5} is not present. Logical and electrical isolation is applied to ignore this supply.

1: V_{DDIO5} is valid.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **VDDIO5VMEN:** V_{DDIO5} independent voltage monitor enable
This bit is reset on any system reset.
0: V_{DDIO5} voltage monitor disabled
1: V_{DDIO5} voltage monitor enabled

13.9.12 PWR supply voltage monitoring control register 3 (PWR_SVMCR3)

Address offset: 0x03C

Reset value: 0x0000 0000

See individual bits for reset condition. Access six wait states when writing this register.

When a system reset occurs during the register write cycle, the written data are not guaranteed.

Secure restriction is defined by SEC7 in PWR_SECCFGR. Privilege restriction is defined by PRIV7 in PWR_PRIVCFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	VDDIO 3VRSEL	VDDIO 2VRSEL	VDDIO VRSEL	Res.	Res.	Res.	ARDY	Res.	USB33 RDY	VDDIO 3RDY	VDDIO 2RDY
					rw	rw	rw				r		r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	ASV	Res.	USB33 SV	VDDIO 3SV	VDDIO 2SV	Res.	Res.	Res.	AVMEN	Res.	USB33 VMEN	VDDIO 3VMEN	VDDIO 2VMEN
			rw		rw	rw	rw				rw		rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **VDDIO3VRSEL:** V_{DDIO3} I/O voltage range selection
This bit is used to select the voltage range supported by I/Os supplied by V_{DDIO3}. it is not reset by wake-up from Standby mode, but by any application reset (such as NRST or IWDG).
0: 3v3 voltage range selected. If V_{DDIO3} is in 1v8 range with this setting, I/Os work in degraded mode.
1: 1v8 voltage range selected. HSLV_VDDIO3 option bit must be set to allow 1v8 voltage range operation. Setting this configuration while V_{DDIO3} is in 3v3 range damages the device.

Bit 25 **VDDIO2VRSEL:** V_{DDIO2} I/O voltage range selection
This bit is used to select the voltage range supported by I/Os supplied by V_{DDIO2}. It is not reset by wake-up from Standby mode, but by any application reset (such as NRST or IWDG).
0: 3v3 voltage range selected. If V_{DDIO2} is in 1v8 range with this setting, I/Os work in degraded mode.
1: 1v8 voltage range selected. HSLV_VDDIO2 option bit must be set to allow 1v8 voltage range operation. Setting this configuration while V_{DDIO2} is in 3v3 range damages the device.

Bit 24 **VDDIOVRSEL:** V_{DD} I/O voltage range selection
This bit is used to select the voltage range supported by I/Os supplied by V_{DD}. It is not reset by wake-up from Standby mode, but by any application reset (such as NRST or IWDG).
0: 3v3 voltage range selected. If V_{DD} is in 1v8 range with this setting, I/Os work in degraded mode.
1: 1v8 voltage range selected. HSLV_VDD option bit must be set to allow 1v8 voltage range operation. Setting this configuration while V_{DD} is in 3v3 range damages the device.

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **ARDY**: $V_{DDA18ADC}$ ready
0: $V_{DDA18ADC}$ is below the threshold of the VDDA18ADC voltage monitor.
1: $V_{DDA18ADC}$ is equal or above the threshold of the VDDA18ADC voltage monitor.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **USB33RDY**: $V_{DD33USB}$ ready
0: $V_{DD33USB}$ is below the threshold of the USB33 voltage monitor.
1: $V_{DD33USB}$ is equal or above the threshold of the USB33 voltage monitor.

Bit 17 **VDDIO3RDY**: V_{DDIO3} ready
0: V_{DDIO3} is below the threshold of the VDDIO3 voltage monitor.
1: V_{DDIO3} is equal or above the threshold of the VDDIO3 voltage monitor.

Bit 16 **VDDIO2RDY**: V_{DDIO2} ready
0: V_{DDIO2} is below the threshold of the VDDIO2 voltage monitor.
1: V_{DDIO2} is equal or above the threshold of the VDDIO2 voltage monitor.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **ASV**: $V_{DDA18ADC}$ independent supply valid
This bit is used to validate the $V_{DDA18ADC}$ supply for electrical and logical isolation purpose. Setting this bit is mandatory to use the analog to digital converters. If $V_{DDA18ADC}$ is not always present in the application, the $V_{DDA18ADC}$ voltage monitor can be used to determine whether this supply is ready or not. This bit is reset on any system reset.
0: $V_{DDA18ADC}$ is not present. Logical and electrical isolation is applied to ignore this supply.
1: $V_{DDA18ADC}$ is valid.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **USB33SV**: $V_{DD33USB}$ independent supply valid
This bit is used to validate the $V_{DD33USB}$ supply for electrical and logical isolation purpose. Setting this bit is mandatory to use the USB2 HS PHYs. If $V_{DD33USB}$ is not always present in the application, the $V_{DD33USB}$ voltage monitor can be used to determine whether this supply is ready or not. This bit is reset on any system reset.
0: $V_{DD33USB}$ is not present. Logical and electrical isolation is applied to ignore this supply.
1: $V_{DD33USB}$ is valid.

Bit 9 **VDDIO3SV**: V_{DDIO3} independent supply valid
This bit is used to validate the V_{DDIO3} supply for electrical and logical isolation purpose. Setting this bit is mandatory to use PN[12:0] I/Os. If V_{DDIO3} is not always present in the application, the V_{DDIO3} voltage monitor can be used to determine whether this supply is ready or not. This bit is reset on any system reset.
0: V_{DDIO3} is not present. Logical and electrical isolation is applied to ignore this supply.
1: V_{DDIO3} is valid.

Bit 8 **VDDIO2SV**: V_{DDIO2} independent supply valid.
This bit is used to validate the V_{DDIO2} supply for electrical and logical isolation purpose. Setting this bit is mandatory to use PO[5:0] and PP[15:0] I/Os. If V_{DDIO2} is not always present in the application, the V_{DDIO2} voltage monitor can be used to determine whether this supply is ready or not. This bit is reset on any system reset.
0: V_{DDIO2} is not present. Logical and electrical isolation is applied to ignore this supply.
1: V_{DDIO2} is valid.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **AVMEN:** $V_{DDA18ADC}$ independent ADC voltage monitor enable
This bit is reset on any system reset.
0: $V_{DDA18ADC}$ voltage monitor disabled
1: $V_{DDA18ADC}$ voltage monitor enabled

Bit 3 Reserved, must be kept at reset value.

Bit 2 **USB33VMEN:** $V_{DD33USB}$ independent USB 33 voltage monitor enable.
This bit is reset on any system reset.
0: $V_{DD33USB}$ voltage monitor disabled
1: $V_{DD33USB}$ voltage monitor enabled

Bit 1 **VDDIO3VMEN:** V_{DDIO3} independent voltage monitor enable
This bit is reset on any system reset.
0: V_{DDIO3} voltage monitor disabled
1: V_{DDIO3} voltage monitor enabled

Bit 0 **VDDIO2VMEN:** V_{DDIO2} independent voltage monitor enable
This bit is reset on any system reset.
0: V_{DDIO2} voltage monitor disabled
1: V_{DDIO2} voltage monitor enabled

13.9.13 PWR wake-up clear register (PWR_WKUPCR)

Address offset: 0x050

Reset value: 0x0000 0000

This register is not reset on any system reset. Access six wait states when writing this register.

When a system reset occurs during the register write cycle, the written data are not guaranteed.

Secure restriction of each it WKUPCx (x = 1 to 4) is defined by WUPxSEC in PWR_SECCFGR. Privilege restriction of each WKUPCx (x = 1 to 4) is defined by WUPxPRIV in PWR_PRIVCFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPC4	WKUPC3	WKUPC2	WKUPC1
												w	w	w	w

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

Bit 3 **WKUPC4:** Clear wake-up flag for WKUP4 pin
These bits are always read as 0.
0: No effect
1: Writing 1 clears WKUPF4 in PWR_WKUPSR.
Note: This bit is cleared to 0 by hardware.

Bit 2 **WKUPC3:** Clear wake-up flag for WKUP3 pin
These bits are always read as 0.
0: No effect
1: Writing 1 clears WKUPF3 in PWR_WKUPSR.
Note: This bit is cleared to 0 by hardware.

Bit 1 **WKUPC2:** Clear wake-up flag for WKUP2 pin
These bits are always read as 0.
0: No effect
1: Writing 1 clears WKUPF2 in PWR_WKUPSR.
Note: This bit is cleared to 0 by hardware.

Bit 0 **WKUPC1:** Clear wake-up flag for WKUP1 pin
These bits are always read as 0.
0: No effect
1: Writing 1 clears WKUPF1 in PWR_WKUPSR.
Note: This bit is cleared to 0 by hardware.

13.9.14 PWR wake-up status register (PWR_WKUPSR)

Address offset: 0x054

Reset value: 0x0000 0000

This register is not reset by wake-up from Standby mode, but by any application reset (such as NRST or IWDG).

When clearing a WKUPFx, the AHB write access completes after the WKUPFx has been cleared.

When a system reset occurs during the register write cycle, the written data are not guaranteed.

Secure restriction of each WKUPFx (x = 1 to 4) is defined by WUPxSEC in PWR_SECCFGR. Privilege restriction of each WKUPFx (x = 1 to 4) is defined by WUPxPRIV in PWR_PRIVCFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPF4	WKUPF3	WKUPF2	WKUPF1
												r	r	r	r

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

Bit 3 **WKUPF4:** Wake-up flag for WKUP4 pin before enable
This bit is set by hardware and cleared only by a NRST reset, or by setting the WKUPC4 bit.
0: No wake-up event occurred
1: A wake-up event was received from WKUP4 pin.

Bit 2 **WKUPF3:** Wake-up flag for WKUP3 pin before enable
This bit is set by hardware and cleared only by a NRST reset, or by setting the WKUPC3 bit.
0: No wake-up event occurred
1: A wake-up event was received from WKUP3 pin.

Bit 1 **WKUPF2:** Wake-up flag for WKUP2 pin before enable
This bit is set by hardware and cleared only by a NRST reset, or by setting the WKUPC2 bit.
0: No wake-up event occurred
1: A wake-up event was received from WKUP2 pin.

Bit 0 **WKUPF1:** Wake-up flag for WKUP1 pin before enable
This bit is set by hardware and cleared only by a NRST reset, or by setting the WKUPC1 bit.
0: No wake-up event occurred
1: A wake-up event was received from WKUP1 pin.

13.9.15 PWR wake-up enable and polarity register (PWR_WKUPEPR)

Address offset: 0x058

Reset value: 0x0000 0000

This register is not reset by wake-up from Standby mode, but by any application reset (such as NRST or IWDG). Access six wait states when writing this register.

When a system reset occurs during the register write cycle, the written data are not guaranteed.

Secure restriction of each WKUPENx, WKUPPx, and WKUPPUPDx (x = 1 to 4) is defined by WUPxSEC in PWR_SECCFGR. Privilege restriction of each WKUPENx, WKUPPx, and WKUPPUPDx (x = 1 to 4) is defined by WUPxPRIV in PWR_PRIVCFGR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPPUPD4[1:0]		WKUPPUPD3[1:0]		WKUPPUPD2[1:0]		WKUPPUPD1[1:0]	
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	WKUP P4	WKUP P3	WKUP P2	WKUP P1	Res.	Res.	Res.	Res.	WKUP EN4	WKUP EN3	WKUP EN2	WKUP EN1
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:22 **WKUPPUPD4[1:0]**: Wake-up pull configuration for WKUP4 pin
These bits define the I/O pad pull configuration used when WKUPEN4 = 1 (the associated GPIO port pull configuration must be set to the same value or 00). The WKUP pin pull configuration is maintained in Standby mode.
00: No pulls
01: Pull-up
10: Pull-down
11: Reserved

Bits 21:20 **WKUPPUPD3[1:0]**: Wake-up pull configuration for WKUP3 pin

These bits define the I/O pad pull configuration used when WKUPEN3 = 1 (the associated GPIO port pull configuration must be set to the same value or 00). The WKUP pin pull configuration is maintained in Standby mode.

00: No pulls
01: Pull-up
10: Pull-down
11: Reserved

Bits 19:18 **WKUPPUPD2[1:0]**: Wake-up pull configuration for WKUP2 pin

These bits define the I/O pad pull configuration used when WKUPEN2 = 1 (the associated GPIO port pull configuration must be set to the same value or 00). The WKUP pin pull configuration is maintained in Standby mode.

00: No pulls
01: Pull-up
10: Pull-down
11: Reserved

Bits 17:16 **WKUPPUPD1[1:0]**: Wake-up pull configuration for WKUP1 pin

These bits define the I/O pad pull configuration used when WKUPEN1 = 1 (the associated GPIO port pull configuration must be set to the same value or 00). The WKUP pin pull configuration is maintained in Standby mode.

00: No pulls
01: Pull-up
10: Pull-down
11: Reserved

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

Bit 11 **WKUPP4**: Wake-up polarity bit for WKUP4 pin

This bit defines the polarity used for event detection on external WKUP4 pin.
0: Detection on high level (rising edge)
1: Detection on low level (falling edge)

Bit 10 **WKUPP3**: Wake-up polarity bit for WKUP3 pin

This bit defines the polarity used for event detection on external WKUP3 pin.
0: Detection on high level (rising edge)
1: Detection on low level (falling edge)

Bit 9 **WKUPP2**: Wake-up polarity bit for WKUP2 pin

This bit defines the polarity used for event detection on external WKUP2 pin.
0: Detection on high level (rising edge)
1: Detection on low level (falling edge)

Bit 8 **WKUPP1**: Wake-up polarity bit for WKUP1 pin

This bit defines the polarity used for event detection on external WKUP1 pin.
0: Detection on high level (rising edge)
1: Detection on low level (falling edge)

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **WKUPEN4:** Enable WKUP4 pin

0: An event on WKUP4 pin does not wake up the system from Standby and Stop modes.
1: A rising or falling edge on WKUP4 pin wakes up the system from Standby and Stop modes.

Bit 2 **WKUPEN3:** Enable WKUP3 pin

0: An event on WKUP3 pin does not wake up the system from Standby and Stop modes.
1: A rising or falling edge on WKUP3 pin wakes up the system from Standby and Stop modes.

Bit 1 **WKUPEN2:** Enable WKUP2 pin

0: An event on WKUP2 pin does not wake up the system from Standby and Stop modes.
1: A rising or falling edge on WKUP2 pin wakes up the system from Standby and Stop modes.

Bit 0 **WKUPEN1:** Enable WKUP1 pin

0: An event on WKUP1 pin does not wake up the system from Standby and Stop modes.
1: A rising or falling edge on WKUP1 pin wakes up the system from Standby and Stop modes.

13.9.16 PWR security configuration register (PWR_SECCFGR)

Address offset: 0x070

Reset value: 0x0000 0000

Reset on any system reset.

Nonsecure or unprivileged writes to this register are ignored, while any read is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPSEC4	WKUPSEC3	WKUPSEC2	WKUPSEC1
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **WKUPSEC4**: WKUP4 pin secure protection

0: Bits related to WKUP4 pin in PWR_WKUPCR, PWR_WKUPSR, and PWR_WKUPEPR can be read and written with secure or nonsecure access.
1: Bits related to WKUP4 pin in PWR_WKUPCR, PWR_WKUPSR, and PWR_WKUPEPR can be read and written only with secure access.

Bit 18 **WKUPSEC3**: WKUP3 pin secure protection

0: Bits related to WKUP3 pin in PWR_WKUPCR, PWR_WKUPSR, and PWR_WKUPEPR can be read and written with secure or nonsecure access.
1: Bits related to WKUP3 pin in PWR_WKUPCR, PWR_WKUPSR, and PWR_WKUPEPR can be read and written only with secure access.

Bit 17 **WKUPSEC2**: WKUP2 pin secure protection

0: Bits related to WKUP2 pin in PWR_WKUPCR, PWR_WKUPSR, and PWR_WKUPEPR can be read and written with secure or nonsecure access.
1: Bits related to WKUP2 pin in PWR_WKUPCR, PWR_WKUPSR, and PWR_WKUPEPR can be read and written only with secure access.

Bit 16 **WKUPSEC1**: WKUP1 pin secure protection
0: Bits related to WKUP1 pin in PWR_WKUPCR, PWR_WKUPSR, and PWR_WKUPEPR can be read and written with secure or nonsecure access.
1: Bits related to WKUP1 pin in PWR_WKUPCR, PWR_WKUPSR, and PWR_WKUPEPR can be read and written only with secure access.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **SEC7**: Peripheral voltage monitor secure protection
0: PWR_SVMCR1, PWR_SVMCR2, and PWR_SVMCR3 can be read and written with secure or nonsecure access.
1: PWR_SVMCR1, PWR_SVMCR2, and PWR_SVMCR3 can be read and written only with secure access.

Bit 6 **SEC6**: CPU power control secure protection
0: PWR_CPUCR can be read and written with secure or nonsecure access.
1: PWR_CPUCR can be read and written only with secure access.

Bit 5 **SEC5**: Backup domain secure protection
0: PWR_BDCR1, PWR_BDCR2, and PWR_DBPCR can be read and written with secure or nonsecure access.
1: PWR_BDCR1, PWR_BDCR2, and PWR_DBPCR can be read and written only with secure access.

Bit 4 **SEC4**: Voltage scaling selection secure protection
0: PWR_VOSCR can be read and written with secure or nonsecure access.
1: PWR_VOSCR can be read and written only with secure access.

Bit 3 **SEC3**: I-TCM, D-TCM, and I-TCM FLEXMEM low power control secure protection
0: PWR_CR4 can be read and written with secure or nonsecure access.
1: PWR_CR4 can be read and written only with secure access.

Bit 2 **SEC2**: V_{DDCORE} monitor secure protection
0: PWR_CR3 can be read and written with secure or nonsecure access.
1: PWR_CR3 can be read and written only with secure access.

Bit 1 **SEC1**: Programmable voltage detector secure protection
0: PWR_CR2 can be read and written with secure or nonsecure access.
1: PWR_CR2 can be read and written only with secure access.

Bit 0 **SEC0**: System supply configuration secure protection
0: PWR_CR1 can be read and written with secure or nonsecure access.
1: PWR_CR1 can be read and written only with secure access.

13.9.17 PWR privilege configuration register (PWR_PRIVCFGR)

Address offset: 0x074

Reset value: 0x0000 0000

Reset on any system reset.

Write access to this register is privileged only. Any read access is allowed on this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUP PRIV4	WKUP PRIV3	WKUP PRIV2	WKUP PRIV1
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **WKUPPRIV4**: WKUP4 pin privileged protection

0: Bits related to WKUP4 pin in PWR_WKUPCR, PWR_WKUPSR, and PWR_WKUPEPR can be read and written with privileged or unprivileged access.

1: Bits related to WKUP4 pin in PWR_WKUPCR, PWR_WKUPSR, and PWR_WKUPEPR can be read and written only with privileged access.

This bit can only be written by privileged application. If corresponding WKUPSEC4 is set in PWR_SECCFGR, this bit can only be written by secure privileged application.

Bit 18 **WKUPPRIV3**: WKUP3 pin privileged protection

0: Bits related to WKUP3 pin in PWR_WKUPCR, PWR_WKUPSR, and PWR_WKUPEPR can be read and written with privileged or unprivileged access.

1: Bits related to WKUP3 pin in PWR_WKUPCR, PWR_WKUPSR, and PWR_WKUPEPR can be read and written only with privileged access.

This bit can only be written by privileged application. If corresponding WKUPSEC3 is set in PWR_SECCFGR, this bit can only be written by secure privileged application.

Bit 17 **WKUPPRIV2**: WKUP2 pin privileged protection

0: Bits related to WKUP2 pin in PWR_WKUPCR, PWR_WKUPSR, and PWR_WKUPEPR can be read and written with privileged or unprivileged access.

1: Bits related to WKUP2 pin in PWR_WKUPCR, PWR_WKUPSR, and PWR_WKUPEPR can be read and written only with privileged access.

This bit can only be written by privileged application. If corresponding WKUPSEC2 is set in PWR_SECCFGR, this bit can only be written by secure privileged application.

Bit 16 **WKUPPRIV1**: WKUP1 pin privileged protection

0: Bits related to WKUP1 pin in PWR_WKUPCR, PWR_WKUPSR, and PWR_WKUPEPR can be read and written with privileged or unprivileged access.

1: Bits related to WKUP1 pin in PWR_WKUPCR, PWR_WKUPSR, and PWR_WKUPEPR can be read and written only with privileged access.

This bit can only be written by privileged application. If corresponding WKUPSEC1 is set in PWR_SECCFGR, this bit can only be written by secure privileged application.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **PRIV7**: Peripheral voltage monitor privileged protection

0: PWR_SVMCR1, PWR_SVMCR2, and PWR_SVMCR3 can be read and written with privileged or unprivileged access.

1: PWR_SVMCR1, PWR_SVMCR2, and PWR_SVMCR3 can be read and written only with privileged access.

This bit can only be written by privileged application. If corresponding SEC7 is set in PWR_SECCFGR, this bit can only be written by secure privileged application.

Bit 6 **PRIV6**: CPU power control privileged protection

0: PWR_CPUCR can be read and written with privileged or unprivileged access.

1: PWR_CPUCR can be read and written only with privileged access.

This bit can only be written by privileged application. If corresponding SEC6 is set in PWR_SECCFGR, this bit can only be written by secure privileged application.

Bit 5 **PRIV5**: Backup domain privileged protection

0: PWR_BDCR1, PWR_BDCR2, and PWR_DBPCR can be read and written with privileged or unprivileged access.

1: PWR_BDCR1, PWR_BDCR2, and PWR_DBPCR can be read and written only with privileged access.

This bit can only be written by privileged application. If corresponding SEC5 is set in PWR_SECCFGR, this bit can only be written by secure privileged application.

Bit 4 **PRIV4**: Voltage scaling selection privileged protection

0: PWR_VOSCR can be read and written with privileged or unprivileged access.

1: PWR_VOSCR can be read and written only with privileged access.

This bit can only be written by privileged application. If corresponding SEC4 is set in PWR_SECCFGR, this bit can only be written by secure privileged application.

Bit 3 **PRIV3**: I-TCM, D-TCM, and I-TCM FLEX MEM low power control privileged protection

0: PWR_CR4 can be read and written with privileged or unprivileged access.

1: PWR_CR4 can be read and written only with privileged access.

This bit can only be written by privileged application. If corresponding SEC3 is set in PWR_SECCFGR, this bit can only be written by secure privileged application.

Bit 2 **PRIV2**: V_{DDCORE} monitor privileged protection

0: PWR_CR3 can be read and written with privileged or unprivileged access.

1: PWR_CR3 can be read and written only with privileged access.

This bit can only be written by privileged application. If corresponding SEC2 is set in PWR_SECCFGR ,his bit can only be written by secure privileged application.

Bit 1 **PRIV1**: Programmable voltage detector privileged protection

0: PWR_CR2 can be read and written with privileged or unprivileged access.

1: PWR_CR2 can be read and written only with privileged access.

This bit can only be written by privileged application. If corresponding SEC1 is set in PWR_SECCFGR, this bit can only be written by secure privileged application.

Bit 0 **PRIV0**: System supply configuration privileged protection

0: PWR_CR1 can be read and written with privileged or unprivileged access.

1: PWR_CR1 can be read and written only with privileged access.

This bit can only be written by privileged application. If corresponding SEC0 is set in PWR_SECCFGR, this bit can only be written by secure privileged application.

13.9.18 PWR debug control register 1 (PWR_CRCFG1)

Address offset: 0x080

Reset value: 0xD000 0000

Bits [15:0] are not reset by wake-up from STANDBY mode, application reset (such as NRST, IWDG) and V_{DD} POR, but only by V_{SW} POR and VSWRST. Bits [31:16] are reset by system reset and when waking up from STANDBY mode.

Write access is protected and software must write a KEY (0XCAFECAFE) as a word access in this register to unlock a single write access. Once unlocked, the register accepts a single written access (byte, half word, or word), after it relocks.

Secure privileged write access only. Any read access is allowed on this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SD_OKSTART	SD_VS_READY	SD_LPM	VDDA18PMURDY	SD_STATUS_SPARE[3:0]				Res.	Res.	Res.	RLPSN	Res.	Res.	MODE_DVS	SYNC_ADC
r	r	r	r	r	r	r	r				rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	SEL_DLY_NOVL[2:0]			SEL_LC[1:0]		SLOPE_SFST[1:0]		SDFPWMEN	SDDISILM	MERGE_CLK_FSM	UNLOCKED
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **SD_OKSTART**: Step down converter start up phase status

Bit 30 **SD_VS_READY**: Step down converter voltage scaling status

Bit 29 **SD_LPM**: Step down converter low power mode status

Bit 28 **VDDA18PMURDY**: $V_{DDA18PMU}$ ready.
0: $V_{DDA18PMU}$ is below the threshold of the VDDA18PMU voltage monitor.
1: $V_{DDA18PMU}$ is equal or above the threshold of the VDDA18PMU voltage monitor.

Bits 27:24 **SD_STATUS_SPARE[3:0]:** Step down converter spare status bits

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **RLPSN**: RAM low power mode disable in STOP.
When set the RAMs does not enter low power mode when the system enters STOP
0: RAM enters low power mode when system enters STOP
1: RAM remains in normal mode when system enters STOP

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **MODE_DVS**: Mode selection during voltage scaling of the SD converter

Bit 16 **SYNC_ADC**:
0: SD_Converter clock free running
1: SD_Converter clock synchronized to ADC

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:8 **SEL_DLY_NOVL[2:0]**: NOVL delay selection

Bits 7:6 **SEL_LC[1:0]**: External inductor/capacitor selection for SD converter

Bits 5:4 **SLOPE_SFST[1:0]**: Set slope of the voltage ramp during the startup.

Bit 3 **SDFPWMEN**: Step down converter force PWM mode
0: SD_Converter Normal mode
1: SD_Converter forced PWM mode

Bit 2 **SDDISILM**: Step down converter current limitation disabling
0: Current limitation disabled
1: Current limitation enabled

Bit 1 **MERGE_CLK_FSM**: Synchronous clk in Ramp and Mos FSM

Bit 0 **UNLOCKED**: Debug register unlocked
0: accessed locked: key was not written and after each register write access
1: after key 0xCAFECAFE was written in this register

13.9.19 PWR debug control register 2 (PWR_CRCFG2)

Address offset: 0x084

Reset value: 0x0000 0000

Reset on any system reset.

Write access is protected and software must write a KEY (0XCAFECAFE) as a word access in this register to unlock a single write access. Once unlocked, the register accepts a single written access (byte, half word, or word), after it relocks.

Secure privileged write access only. Any read access is allowed on this register.

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

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **DBG_FSM[11:0]**:

Bits 15:4 **SEL_DLY_FSM[11:0]**: Various delay in the asynchronous FSM

Bit 3 Reserved, must be kept at reset value.

Bit 2 **DBG_OUT_EN**: Enable SD debug status outputs

Bit 1 **DBG_IN_EN**: Enable SD debug control inputs

Bit 0 **UNLOCKED**: Debug register unlocked
0: accessed locked: key was not written and after each register write access
1: after key 0xCAFECAFE was written in this register

13.9.20 PWR Debug control register 3 (PWR_CRCFG3)

Address offset: 0x088

Reset value: 0x0000 0000

Reset on any system reset.

Write access is protected and software must write a KEY (0XCAFECAFE) as a word access in this register to unlock a single write access. Once unlocked, the register accepts a single written access (byte, half word, or word), after it relocks.

Secure privileged write access only. Any read access is allowed on this register.

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

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:1 **DBG_CFG_BIT[11:0]**: Spare configuration bits

Bit 0 **UNLOCKED**: Debug register unlocked

0: accessed locked: key was not written and after each register write access

1: after key 0xCAFECAFE was written in this register

13.9.21 PWR register map

Table 63. PWR register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	PWR_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	POPL[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPDS08V	MODE_PDN		SDEN	Res.	Res.
	Reset value												0	0	0	0	0											1	0		1		
0x004	PWR_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PVDO	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PVDEN
	Reset value																								0								0
0x008	PWR_CR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VCOREH	VCOREL	Res.	Res.	Res.	VCORELLS	Res.	Res.	Res.	VCOREMONEN
	Reset value																							0	0				0				0
0x00C	PWR_CR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCMFLXRBSEN	Res.	Res.	Res.	TCMRBSEN
	Reset value																												0				0
0x010 - 0x01C	Reserved	Reserved																															
0x020	PWR_VOSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res	ACTVOSRDY	ACTVOS	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	VOSRDY	VOS
	Reset value															1	0															1	0
0x024	PWR_BDCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TEMPH	TEMPL	V08CAPH	V08CAPL	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	MONEN
	Reset value													0	0	0	0																0
0x028	PWR_BDCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPRBSEN
	Reset value																																0
0x02C	PWR_DBPCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBP
	Reset value																																0
0x030	PWR_CPUCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SVOS	Res.	Res.	Res.	Res.	Res.	Res.	SBF	STOPF	Res.	Res.	Res.	Res.	Res.	Res.	CSSF	PDDS
	Reset value																1							0	0							0	0
0x034	PWR_SVMCR1	Res.	Res.	Res.	Res.	Res.	Res.	VDDIO4VRSTBY	VDDIO4VRSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VDDIO4RDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VDDIO4SV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VDDIO4VMEN
	Reset value							0	0								0								0								0

Table 63. PWR register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x038	PWR_SVMCR2	Res.	Res.	Res.	Res.	Res.	Res.	VDDIO5VRSTBY	VDDIO5VRSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VDDIO5RDY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VDDIO5SV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VDDIO5VMEN
	Reset value							0	0								0								0								0
0x03C	PWR_SVMCR3	Res.	Res.	Res.	Res.	Res.	VDDIO3VRSEL	VDDIO2VRSEL	VDDIOVRSEL	Res.	Res.	Res.	ARDY	Res.	USB33RDY	VDDIO3RDY	VDDIO2RDY	Res.	Res.	Res.	ASV	Res.	USB33SV	VDDIO3SV	VDDIO2SV	Res.	Res.	Res.	AVMEN	Res.	USB33VMEN	VDDIO3VMEN	VDDIO2VMEN
	Reset value						0	0	0				0		0	0	0				0		0	0	0				0		0	0	0
0x040 - 0x04C	Reserved	Reserved																															
0x050	PWR_WKUPCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPC4	WKUPC3	WKUPC2	WKUPC1
	Reset value																													0	0	0	0
0x054	PWR_WKUPSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPF4	WKUPF3	WKUPF2	WKUPF1
	Reset value																													0	0	0	0
0x058	PWR_WKUPEPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPPUPD4[1:0]		WKUPPUPD3[1:0]		WKUPPUPD2[1:0]		WKUPPUPD1[1:0]		Res.	Res.	Res.	Res.	WKUPP4	WKUPP3	WKUPP2	WKUPP1	Res.	Res.	Res.	Res.	WKUPEN4	WKUPEN3	WKUPEN2	WKUPEN1
	Reset value									0	0	0	0	0	0	0	0					0	0	0	0					0	0	0	0
0x05C - 0x06C	Reserved	Reserved																															
0x070	PWR_SECCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPSEC4	WKUPSEC3	WKUPSEC2	WKUPSEC1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
	Reset value													0	0	0	0									0	0	0	0	0	0	0	0
0x074	PWR_PRIVCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPPRIV4	WKUPPRIV3	WKUPPRIV2	WKUPPRIV1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
	Reset value													0	0	0	0									0	0	0	0	0	0	0	0
0x080	PWR_CRCFG1	SD_OKSTART	SD_VS_READY	SD_PLM	VDDA18PMURDY	SD_STATUS_SPARE [3:0]				Res.	Res.	Res.	RLPSN	Res.	Res.	MODE_DVS	SYNC_ADC	Res.	Res.	Res.	Res.	Res.	Res.	SEL_DLY_NOVL [2:0]			SEL_LC[1:0]		SLOPE_SFST [1:0]		SDFPWMEN	SDDISILM	MERGE_CLK_FSM	UNLOCKED
	Reset value	1	1	0	1	0	0	0	0				0			0	0							0	0	0	0	0	0	0	0	0	0

Table 63. PWR register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x084	PWR_CRCFG2	Res.	Res.	Res.	Res.	DBG_FSM[11:0]												SEL_DLY_FSM[11:0]												Res.	DBG_OUT_EN	DBG_IN_EN	UNLOCKED
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
0x088	PWR_CRCFG3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_CFG_BIT[11:0]												UNLOCKED
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3* for the register boundary addresses.

14 Reset and clock control (RCC)

The RCC manages the clock and reset generation for the whole microcontroller.

The operating modes to whom this section refers are defined in *Section 13.6.1: Operating modes* of the PWR.

14.1 RCC main features

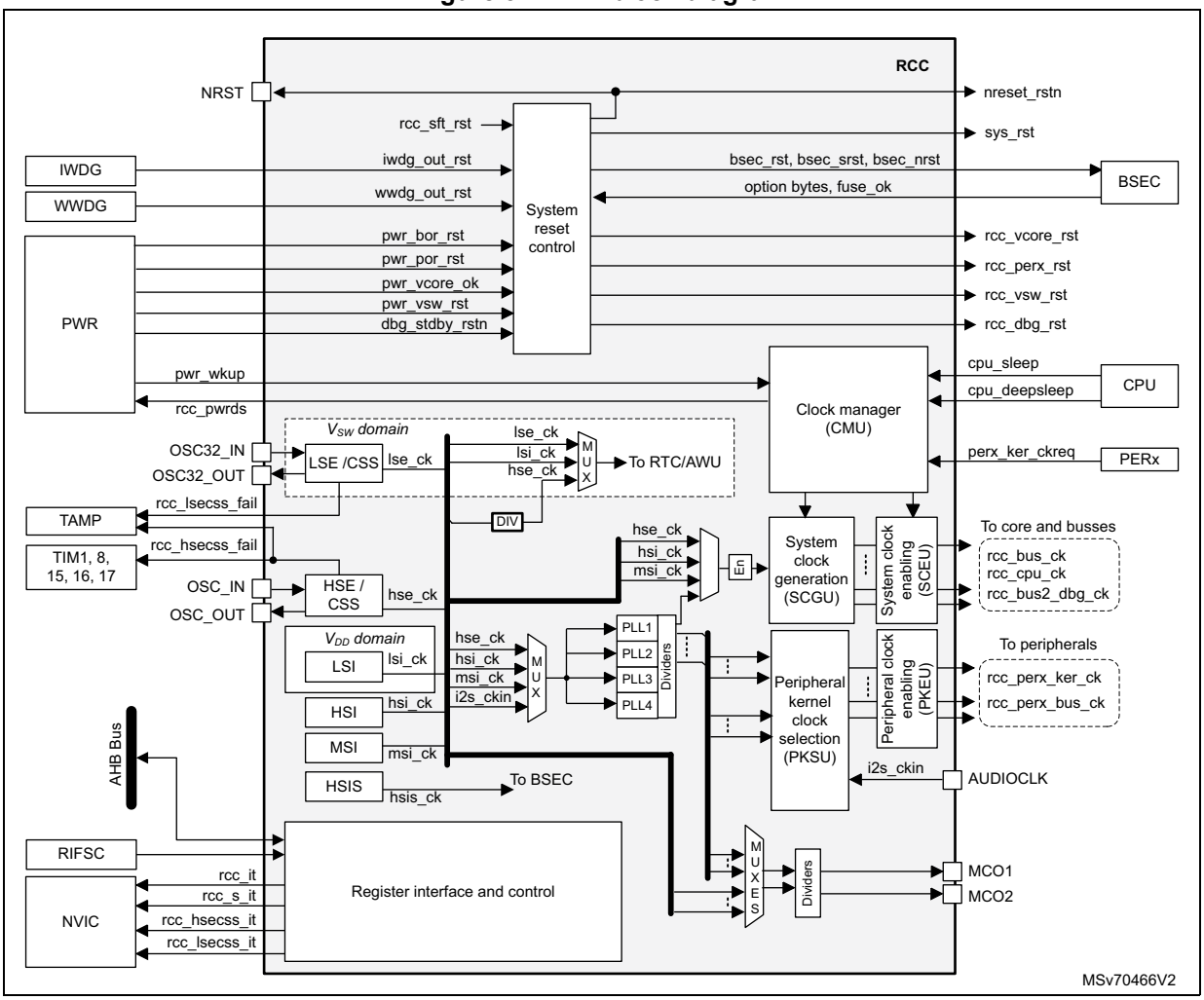
- AHB-Lite bus interface
- RIF (resource isolation framework) aware
- Reset part:
  - Generation of local and system reset
  - Bidirectional pin to reset the microcontroller and/or external devices
  - WWDG and IWDG reset supported
  - Power-on (POR) and brownout (BOR) resets initiated by the PWR
- Clock generation:
  - Generation and dispatching of clocks for the complete device
  - Four separate PLLs using integer or fractional ratios
  - Clock gating to reduce power dissipation
  - Two external oscillators:
    - High-speed external oscillator (HSE) supporting a wide range of crystals from 8 to 48 MHz (when the USBHSPHY is used, the HSE frequency must be 19.2, 20, or 24 MHz)
    - Low-speed external oscillator (LSE) for a 32.768 kHz crystal
  - Four internal oscillators:
    - High-speed internal oscillator (HSI)
    - Low-power internal oscillator (MSI)
    - Low-speed internal oscillator (LSI)
    - High-speed internal secure oscillator (HSIS)
  - Buffered clock outputs for external devices
  - Generation of two types of interrupt lines:
    - Dedicated interrupt lines for clock failure management
    - General interrupt line for other events (separated into secure and nonsecure)
  - Clock generation handling in Stop and Standby modes

14.2 RCC power domains

RCC interfaces with four power domains, namely core (V_{DDCORE}), retention (V_{DD}, V_{RET}), backup (V_{SW}, V_{BKP}), and analog ($V_{DDA18ADC}$).

14.3 RCC block diagram

Figure 34. RCC block diagram



14.4 RCC pins and internal signals

Table 64. RCC input/output signals connected to package pins or balls

Name	Type	Description
NRST(1)	Reset	System reset, can be used to provide reset to external devices
OSC32_IN	I	32 kHz oscillator input
OSC32_OUT	O	32 kHz oscillator output
OSC_IN	I	System oscillator input
OSC_OUT	O	System oscillator output
MCO1	O	Output clock 1 for external devices

Table 64. RCC input/output signals connected to package pins or balls (continued)

Name	Type	Description
MCO2	O	Output clock 2 for external devices
AUDIOCLK	I	External kernel clock input for digital audio interfaces: SPI/I2S, SAI, MDF, ADF

1. Bidirectional reset pin with embedded weak pull-up resistor.

The RCC exchanges signals with all components of the product. *Table 65* shows only the most significant internal signals.

Table 65. RCC internal input/output signals

Name	Type	Description
rcc_it	O	General interrupt request line (nonsecure)
rcc_s_it	O	General interrupt request line (secure)
rcc_hsecss_it	O	HSE clock security failure interrupt
rcc_lsecss_it	O	LSE clock security failure interrupt
rcc_hsecss_fail	O	Event indicating that an HSE clock security failure is detected
rcc_lsecss_fail	O	Event indicating that an LSE clock security failure is detected
nreset_rstn	I/O	Application reset
sys_rst	I/O	System reset
bsec_rstn	O	BSEC warm reset
bsec_srstn	O	BSEC scratch (cold) reset
bsec_hrstn	O	BSEC hot reset
fuse_ok	I	BSEC has finished loading the OTP (one-time programmable, which contains the option bytes)
option bytes	I	Configuration bits from BSEC, affecting RCC behavior (reset, clock, osc)
iwdg_out_rst	I	Reset line driven by the IWDG, indicating that a timeout occurred
wwdg_out_rst	I	Reset line driven by the WWDG, indicating that a timeout occurred
pwr_bor_rst	I	Brownout reset generated by the PWR
pwr_por_rst	I	Power-on reset generated by the PWR
pwr_vsw_rst	I	Power-on reset of the V_{SW} domain generated by the PWR
dbg_stdby_rstn	I	Standby emulation mode reset generated by the PWR
rcc_perx_rst	O	Reset generated by the RCC for the peripherals
pwr_wkup	I	Wake-up request generated by the PWR, and used to restore the clocks
rcc_pwrds	O	Informs the PWR that the RCC has stopped all clocks (PWR can then go to Stop or Standby mode)
cpu_sleep	I	Signals generated by the CPU, indicating if it is in Run, Sleep, or Stop mode
cpu_deepsleep	I	
cpu_SLEEPHOLDACKn	I	When this signal is asserted, the CPU does not advance in execution, and does not perform any memory operation

Table 65. RCC internal input/output signals (continued)

Name	Type	Description
rcc_SLEEPHOLDREQn	O	Request to extend the CPU sleep state regardless of wake-up events. If the CPU acknowledges this request (SLEEPHOLDACKn asserted), the CPU remains idle even when it receives a wake-up event.
perx_ker_ckreq	I	Generated by some peripherals to request the activation of their kernel clock
rcc_perx_ker_ck	O	Kernel clock signals generated by the RCC for some peripherals
rcc_perx_bus_ck	O	Bus interface clock signals generated by the RCC for peripherals
rcc_bus_ck	O	Clocks generated by the RCC for APB, AHB, and AXI bridges
rcc_cpu_ck	O	Clocks generated by the RCC for the CPU
ck_cpu_dbg	O	
rcc_bus2_dbg_ck	O	Debug components clock
ck_cpu_tsgen	O	TSGEN clock (timestamp generator)
ck_cpu_tpiu	O	TPIU clock (double data rate)

14.5 Functional description of RCC reset

The RCC handles the reset generation for the complete product, using events coming from different sources:

- assertion of the NRST pin from an external device
- a failure on the supply voltage applied to V_{DD} or V_{BAT}
- an exit from Standby mode
- a watchdog timeout
- a software command

The reset scope depends on the source that generates it.

14.5.1 Reset from the PWR

The PWR provides several reset signals to the RCC:

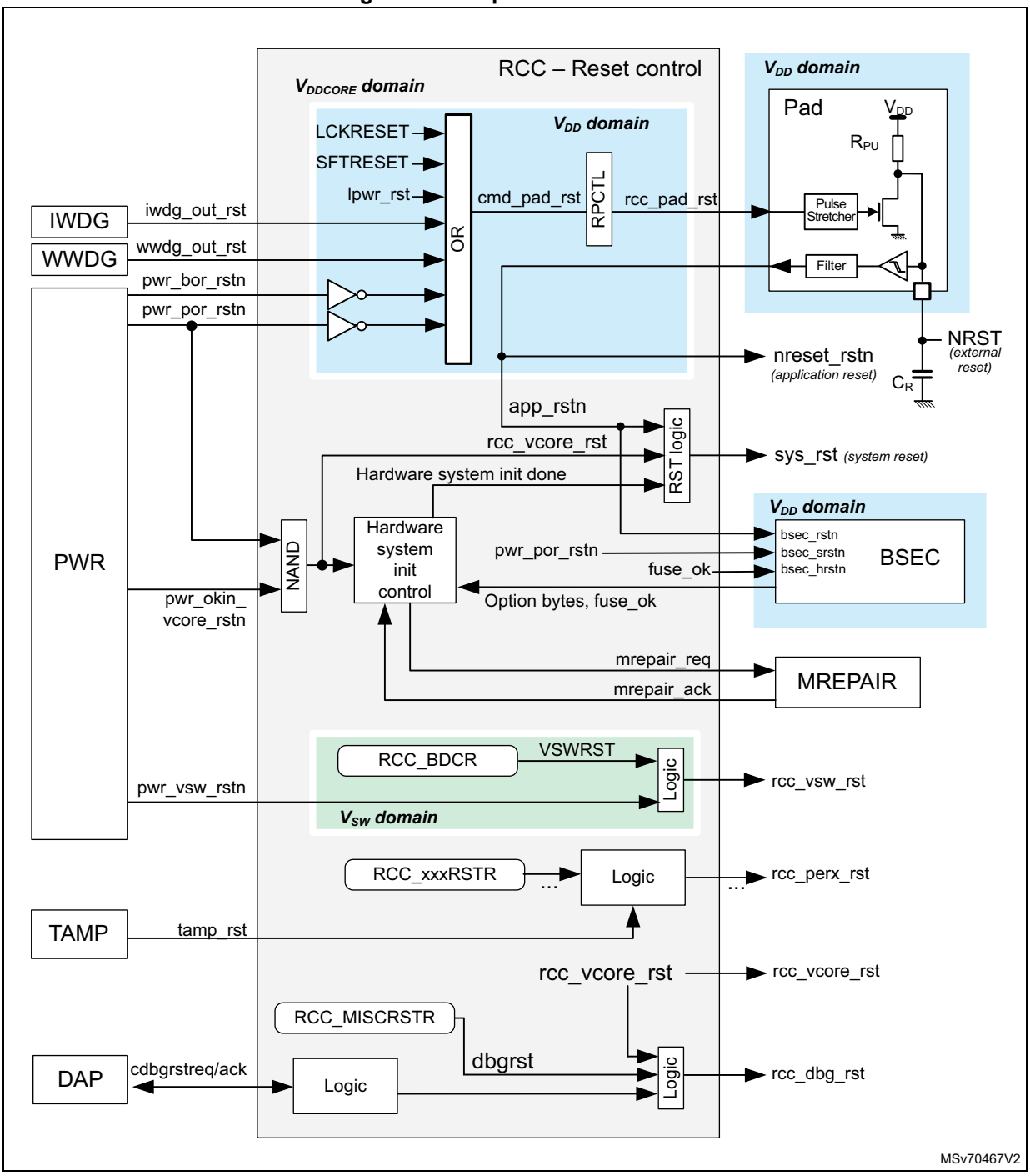
- power-on/off reset signal (pwr_por_rstn): asserted when the V_{DD} supply is lower than the V_{POR} threshold
- brownout reset signal (pwr_bor_rstn): asserted when the V_{DD} supply is lower than the V_{BOR} threshold
- core reset signal (pwr_okin_vcore_rstn): asserted when the V_{DDCORE} supply is not valid or available

Note: V_{DDCORE} is switched off when the product is in Standby mode. When the system exits Standby mode, pwr_okin_vcore_rstn is asserted while V_{DDCORE} from the regulator is not valid. pwr_okin_vcore_rstn is also asserted when the V_{DD} supply is not valid.

- V_{SW} domain reset signal (pwr_vsw_rstn): asserted when the V_{SW} supply is lower than the expected threshold

Refer to *Table 66* for more details.

Figure 35. Simplified reset circuit



14.5.2 System and application resets (sys_rst, nreset_rstn)

A system reset (sys_rst) resets most of the registers to their default values, unless otherwise specified in the register description (summary in *Table 66*).

A system reset can be generated from one of the following sources:

- an assertion of the NRST pin (external reset)

- a reset from the power-on/off reset block (pwr_por_rstn)
- a reset from the brownout reset block (pwr_bor_rstn), see *Section 13.5.2: Brownout reset (BOR)* for a detailed description of the BOR function
- a reset from the independent watchdogs (iwdg_out_rst)
- an exit from Standby mode (rcc_vcore_rst)
- a reset from the window watchdogs depending on WWDG configuration (wwdg_out_rst)
- a software reset (SFTRESET) signal, connected to SYSRESETREQ from the Cortex-M55 core
- a lockup reset (LCKRESET) signal, connected to LOCKUP from the Cortex-M55 core
- a reset from the low-power mode security reset, depending on the option byte configuration (lpwr_rst)

The application reset (nreset_rstn) is similar to the system reset, but it is not asserted when the system exits Standby mode.

Note: The sys_rst is actually a combination the native internal reset signal (int_sys_rstn) and the debug Standby reset signal (dbg_stdby_rstn). Some registers are reset by int_sys_rstn only. See Section 14.6.13 for more details about dbg_stdby_rstn.

The SYSRESETREQ bit in Cortex-M55 must be set to force a software reset on the device. Refer to the Cortex-M55 with FPU Technical Reference Manual for more details. There is also a SYSCFG register, which affects SYSRESETREQ.

14.5.3 NRST reset

The NRST is active low. A pulse stretcher guarantees a minimum reset pulse duration of 20 µs (see the datasheet for details). The NRST assertion can also be extended by adding the C_R capacitor.

It is not recommended to leave the NRST pin unconnected. When not used, connect this pin to ground via a 4.7 to 10 nF capacitor (C_R in *Figure 35*). As shown in *Figure 35*, a filter is present to suppress spurious coming from the NRST pin.

14.5.4 Low-power mode security reset (lpwr_rst)

To prevent critical applications from mistakenly entering a low-power mode, two low-power mode security resets are available. When enabled through RST_STOP and RST_STDBY option bytes, a system reset (sys_rst) is generated if the following conditions are met:

- The CPU mistakenly enters Stop mode.

  This type of reset is enabled by setting RST_STOP user option byte. If a Stop mode entry sequence is successfully executed, a system reset is generated.
- The system mistakenly enters Standby mode.

  This type of reset is enabled by setting RST_STDBY user option byte. If a Standby mode entry sequence is successfully executed, a system reset is generated.

LPWRRSTF bit in RCC_RSR indicates that a low-power mode security reset occurred (see row 8 in *Table 67*).

The lpwr_rst input is activated when a low-power mode security reset is required. This signal is generated by the PWR.

See *Section 5: OTP mapping (OTP)* for additional information, and *Table 54: Operating mode summary* for the overview of existing power modes.

14.5.5 Backup domain reset

A backup domain reset (rcc_vsw_rst) is generated when one of the following occurs:

- a software reset, triggered by setting VSWRST in RCC_BDCR: the BKPSRAM is not affected.
- V_{SW} voltage outside the operating range: the BKPSRAM content is no longer valid.

The RCC_BDCR register and all bitfields in the backup domain (including RTC) return to their reset values: these include RTCEN, RTCLPEN, RTCPRE, RTCSEL, LSECSSRA, LSECSSD, LSECSSON, LSERDY, LSERDYF, LSEON, LSEDRV, LSEEXT, LSEBYP, LSEGFON.

See *Section 13.4.4: Backup domain* and *Section 3: System security* for additional information.

14.5.6 CoreSight debug reset

CoreSight debug components can be reset in three different ways:

- using the DAP by setting CDBGRSTREQ in DP_CTRLSTAT

  This asserts the debug reset request signal (cdbgrstreq) connected to the RCC. The RCC then asserts the debug reset (rcc_dbg_rst), and a handshake signal cdbgrstack acknowledges the DAP request. The debug reset remains asserted while cdbgrstreq is asserted (see *Figure 35* for details).
- when the application sets DBGRST in RCC_MISCRSTR

  This asserts the rcc_dbg_rst reset, which is deasserted when DBGRST is cleared to 0.
- when a V_{DDCORE} power-on reset occurs (rcc_vcore_rst)

  This reset is asserted after a POR, or when the product exits Standby mode.

Refer to *Section 14.5.6* for details.

14.5.7 Option-byte loading

As shown in *Figure 37*, the option-byte loading (OBL via OTP_LD) sequence happens after a POR or a pin reset.

The system reset (sys_rst) is released only after the OBL has completed.

The BSEC manages an OTP array of fuse words, which hold the option-byte configuration for the device. This configuration must be set every time an app_rstn is asserted, and the system stays in reset until this configuration has been properly loaded (fuse_ok signal is received from the BSEC module).

The BSEC handles the following reset sources:

- a main reset (bsec_rstn) asserted when an app_rstn is asserted
- a scratch reset (bsec_srstn) asserted when a pwr_por_rstn is asserted
- a hot reset (bsec_hrstn) asserted when fuse_ok is low

14.5.8 Reset of peripherals

The application can reset individually any peripheral, whenever requested. This can be done via registers named RCC_xxxxRSTR (xxxx is the bus name on which the peripheral is connected).

To reset a peripheral, the corresponding reset bit must be set to 1 (peripheral clock not required), and then set back to 0 (peripheral clock must be enabled and running in advance). There is no need to enable a peripheral clock to reset a peripheral.

Caution: PKA, CRYP, SAES, and HASH may be reset directly in hardware upon a tamper event.

14.5.9 Reset pulse control (RPCTL)

The RPCTL allows the application to control the minimum activation time of the NRST pad. This feature is particularly helpful because some external devices may require a specific reset duration. In addition, the internal reset pulse, for example from IWDG, may be too short for external devices.

The RPCTL is located in the V_{DD} domain, and is reset only after a power-on reset.

The RPCTL is controlled by MRD[4:0] in RCC_RDCR.

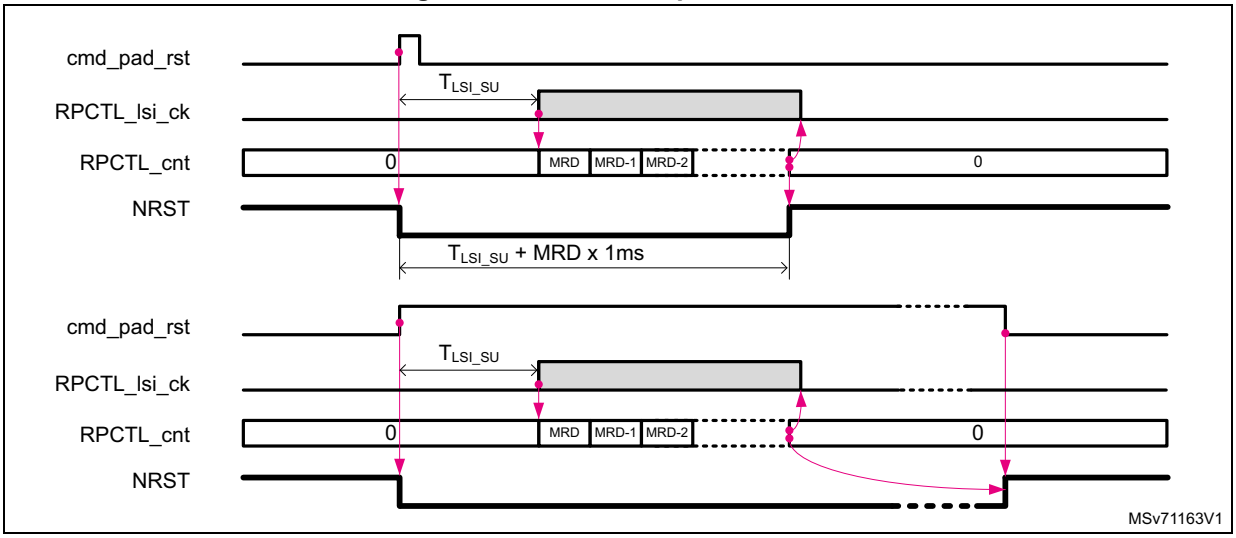
If MRD is 0, then the RPCTL is bypassed. The minimum activation time in this case is given by the pulse stretcher embedded in the reset pad (typically 20 µs).

If MRD is non-0, then the rising edge of cmd_pad_rst causes NRST to be immediately driven low, for at least the duration set by MRD[4:0]. The duration of the reset is unchanged if the cmd_pad_rst signal is active for longer than the duration set by MRD[4:0]. The minimum activation time is between 1 and 31 ms.

The RPCTL uses the LSI clock to measure time. When the cmd_pad_rst goes high, the RPCTL requests the LSI clock to control the reset duration. If the LSI was not enabled by another function, it may take some microseconds before obtaining LSI ready (TLSI_SU).

Figure 36 shows two scenarios.

Figure 36. NRST reset pulse control



14.5.10 Reset coverage summary

Table 66 gives a detailed view of the coverage of the most important reset sources.

Note: When V_{DD} is not valid, V_{DDCORE} is not valid as well.

Table 66. Reset coverage summary (1)

Reset functions		Main reset lines						
		pwr_por_rstn(2)	rcc_vcore_rst(3)	sys_rst	nreset_rstn(4)	rcc_dbg_rst	rcc_perx_rst	rcc_vsw_rstn
V_{DD} domain		X	-	-	-	-	-	-
MCU		X	X	X	-	-	-	-
WWDG		X	X	X	-	-	-	-
IWDG		X	-	-	X	-	-	-
AXI/AHB interconnections		X	X	X	-	-	-	-
Debug components (including DBGMCU): reset all the debug parts except the SWJ-DP function, which is reset by the NJTRST or rcc_vcore_rst resets.		X	X	-	-	X	-	-
Hardware system init: includes the memory repair.		X	X	-	-	-	-	-
RCC	RCC reset register (RCC_RSR)	X	-	-	-	-	-	-
	RCC control register (RCC_CR) and RCC APB5 Sleep enable register (RCC_RDCR)	X	-	-	-	-	-	-
	RCC bitfields in the backup domain	-	-	-	-	-	-	X
	Other RCC registers	X	X	X	-	-	-	-
PWR	PWR_CSR1	-	-	-	-	-	-	X
	PWR_CSR2	X	-	-	-	-	-	-
	PWR_CSR3: individual bits of this register do not have the same reset condition (see *Section 13: Power control (PWR)* for details).	X	X	X	-	-	-	-
	PWR_WKUPCR, PWR_WKUPFR, and PWR_WKUPEPR	X	-	-	X	-	-	-
	Other registers	X	X	X	-	-	-	-
RTC	Peripheral (except APB)	-	-	-	-	-	-	X
	Peripheral APB	X	X	X	-	-	X	-
BKPSRAM: after a reset of the V_{SW} domain, the BKPSRAM backup regulator is disabled. This function is controlled via BKPRBSEN (in PWR_BDCR2). If the rcc_vsw_rst reset is due to a too low V_{SW} voltage, the BKPSRAM content is lost.		-	-	-	-	-	-	X
Other peripherals		X	X	X	-	-	X	-

1. 'X' means that the function is reset by the corresponding reset line. '-' means that the function is not reset by the corresponding reset line.
2. pwr_por_rstn is asserted when the voltage applied to VDD is not valid. When pwr_por_rstn is asserted, the rcc_vcore_rst, NRST, sys_rst, and nreset_rstn are asserted as well.

3. rcc_vcore_rst is asserted when the voltage applied to VDD is not valid, or when the system exits Standby mode (because V_{DDCORE} is switched off). When rcc_vcore_rst is asserted, sys_rst and pwr_dbg_rst are asserted as well.
4. When nreset_rstn is asserted, sys_rst is asserted as well.

14.5.11 Reset source identification

The CPU can identify the reset source by checking reset flags in RCC_RSR or PWR_CSR3 registers.

The CPU can clear flags in RCC_RSR by setting RMVF bit in this register.

Table 67 shows how the status bits behave according to the situation that generated the reset. For example, when an IWDG timeout occurs (row 7), if the CPU reads RCC_RSR during the boot phase, both PINRSTF and IWDGRSTF bits are set, indicating that the IWDG also generated a pin reset.

Table 67. Reset source identification(1)

#	Situation generating a reset	SBF (2)	LPWRRSTF	IWDGRSTF	WWDGRST	LCKRSTF	SFTRSTF	BORRSTF	PINRSTF	PORRSTF
1	Power-on reset (pwr_por_rstn)	0	0	0	0	0	0	1	1	1
2	Pin reset (NRST)	0	0	0	0	0	0	0	1	0
3	Brownout reset (pwr_bor_rstn)	0	0	0	0	0	0	1	1	0
4	System reset generated by the CPU (SFTRESET)	0	0	0	0	0	1	0	1	0
5	System reset generated by the CPU (LCKRESET)	0	0	0	0	1	0	0	1	0
6	WWDG reset (wwdg_out_rst)	0	0	0	1	0	0	0	1	0
7	IWDG reset (iwdg_out_rst)	0	0	1	0	0	0	0	1	0
8	CPU erroneously enters Stop or Standby mode	0	1	0	0	0	0	0	1	0
9	The product exits Standby mode	1	0	0	0	0	0	0	1	0

1. Grayed cells highlight the register bits that are set.
2. The SBF bit is located in PWR_CSR3 register.

14.5.12 Power-on and wake-up sequences

For detailed diagrams, refer to *Section 13.4.1: System supply startup* in the PWR.

The time interval between the event that exits the device from a low-power, and the moment where the CPU is able to execute code, depends on the system state and on its configuration. *Figure 37* shows the most usual examples.

Power-on wake-up sequence

The sequence shown in *Figure 37* gives the most significant phases of the power-on wake-up. It is the longest sequence since the circuit was not powered.

Note: This sequence remains unchanged whatever V_{BAT} is present or not.

Boot from pin reset (NRST)

When a pin reset occurs, V_{DD} is still present. As a result:

- the regulator settling time is faster since the reference voltage is already stable
- the HSI restart delay may be needed if the HSI was not enabled when the NRST occurred, otherwise this restart delay phase is skipped.

Boot from system standby

When waking up from system standby, the reference voltage is stable since V_{DD} has not been removed. As a result, the regulator settling time is fast. Since V_{DDCORE} was not present, the restart delay for the HSI and HSIS cannot be skipped.

sys_rst remains asserted until HSIS runs and the memory repair completes.

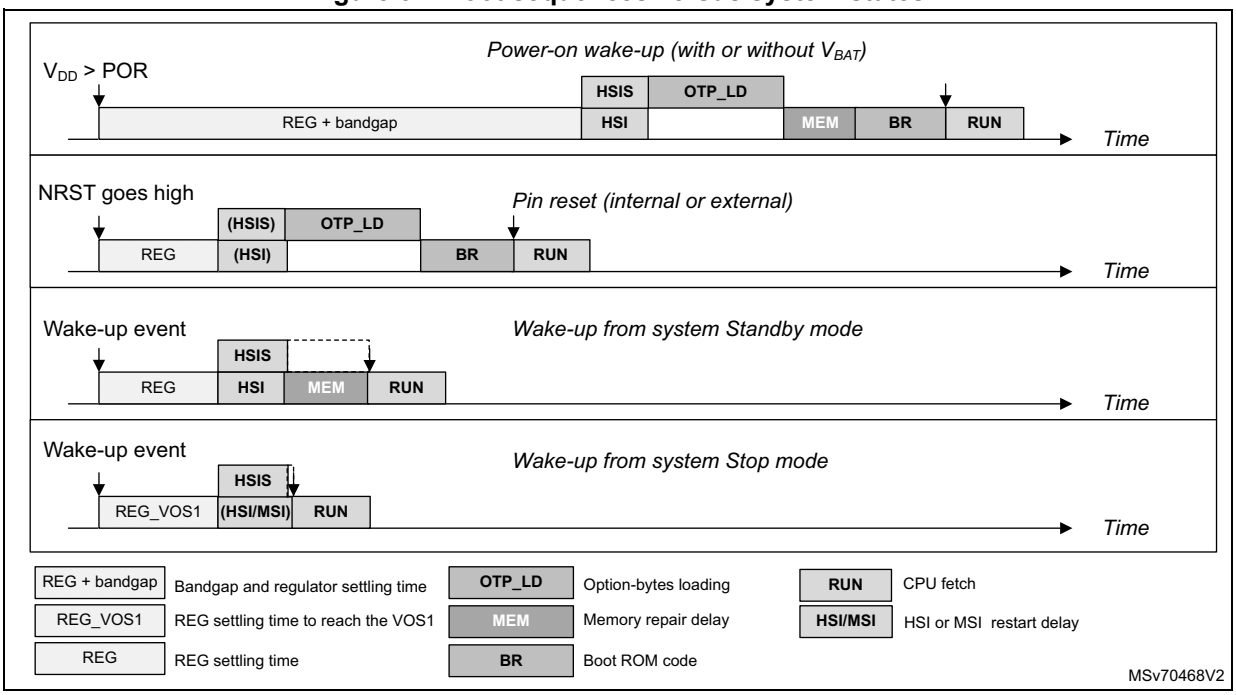
Restart from system stop

When restarting from system stop, V_{DD} is still present. As a result, the sequence is mainly composed of two steps:

1. The regulator settling time reaches V_{OS1} (default voltage).
2. HSI/MSI restart delay. This step can be skipped if HSISTOPEN = MSISTOPEN = 1 in RCC_STOPCR.

sys_rst remains asserted until HSIS runs.

Figure 37. Boot sequences versus system states



14.6 Functional description of RCC clocks

The RCC provides a wide choice of clock generators:

- HSI (high-speed internal oscillator) clock: ~ 8, 16, 32, or 64 MHz

- HSE (high-speed external oscillator) clock: 8 to 48 MHz
- LSE (low-speed external oscillator) clock: 32 kHz
- LSI (low-speed internal oscillator) clock: ~ 32 kHz
- MSI (low-power internal oscillator) clock: ~ 4 or 16 MHz

The RCC offers a high flexibility for the application to select the appropriate clock for the CPU and peripherals (in particular for peripherals that require a specific clock, such as SPI/I2S and SAI).

To optimize the power consumption, each clock source can be switched ON or OFF independently.

The RCC provides up to four PLLs; each of them can be configured in integer mode (with or without SSCG - spread spectrum clock generation), or fractional mode.

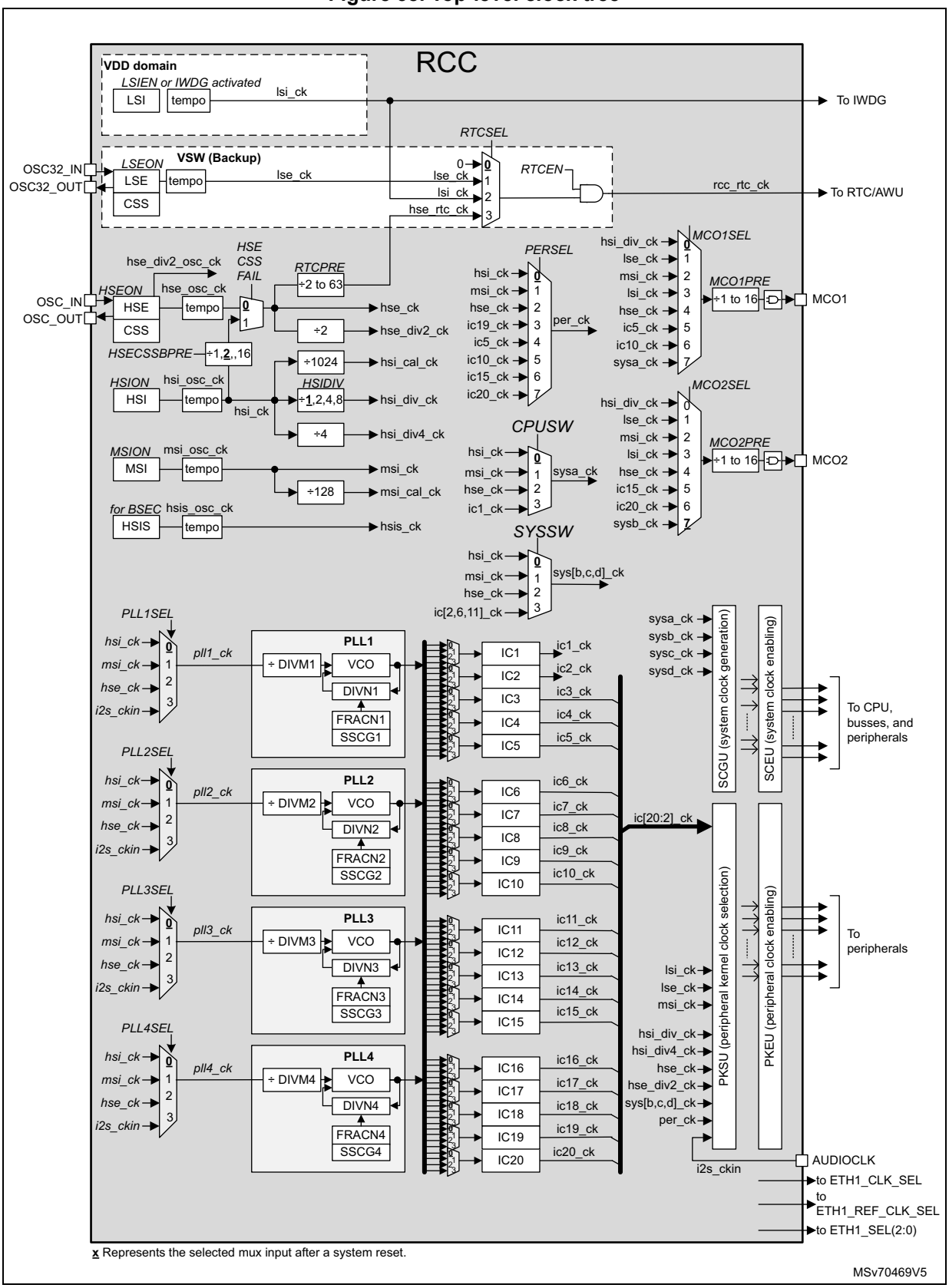
As shown in the *Figure 38*, the RCC offers two clock outputs (MCO1 and MCO2), with flexibility on the clock selection and frequency adjustment.

The SCGU (system clock generation unit) contains several prescalers to configure the CPU and bus matrix clock frequencies.

The PKSU (peripheral kernel clock selection unit) provides several dynamic switches, which give a large choice of kernel clock distribution to peripherals.

The PKEU (peripheral kernel clock enable unit) performs the peripheral kernel clock gating. The SCEU (system clock enable unit) performs the clock gating for the bus interface, cores, and the bus matrix.

Figure 38. Top-level clock tree



14.6.1 Clock naming convention

The RCC provides clocks to the complete circuit. To avoid misunderstandings, the following terms are used in this document:

- Peripheral clocks provided by the RCC to the peripherals

  Two kinds of clock are available, namely bus interface and kernel clocks

  A peripheral receives from the RCC a bus interface clock to access its registers, and thus control the peripheral operation. This clock is generally the AHB, APB, or AXI clock, depending on which bus the peripheral is connected to. Some peripherals need only a bus interface clock.

  Some peripherals require also a dedicated clock (named kernel clock) to handle the interface function. As an example, SAI must generate specific and accurate master clock frequencies, which require dedicated kernel clock frequencies.

  An advantage of decoupling the bus interface clock from the kernel clock is that the bus clock can be changed without reprogramming the peripheral.
- CPU clock: derived from a dedicated system clock (sysa_ck), is asynchronous to the bus clocks.
- Bus matrix clocks: provided to the different bridges (APB, AHB, AXI, or NoC), are derived from a system clock (sysb_ck).
- NPU and NPU AXI clocks: the NPU clock is derived from a dedicated system clock, sysc_ck, also used for the AXI matrix close to the NPU.
- AXISRAM1/2 clocks: derived from sysb_ck.
- AXISRAM3/4/5/6 clocks: derived from sysd_ck.

14.6.2 Oscillator description

Table 68 shows the oscillator states versus system modes, when the oscillators are enabled via registers. Available means that the resource can be used if activated via registers.

Table 68. Oscillator states versus system modes

System modes	V_{DDCORE} domain				V_{DD} domain	V_{SW} domain
	HSIS	HSE	HSI	MSI	LSI	LSE
Exit from system reset	On	Off	On	Off	Available	Available
Exit from system stop	On	Off	On(1)	On(2)	Available	Available
In Run/Sleep mode	On	Available	Available	Available	Available	Available
In Stop mode	Off	Off	Available(3)	Available(4)	Available	Available
In Standby mode	Off	Off	Off	Off	Available	Available
In V_{BAT} mode	Off	Off	Off	Off	Off	Available

1. If STOPWUCK = 0.
2. If STOPWUCK = 1.
3. HSI can remain activated in Stop mode if HSISTOPEN = 1, or if a peripheral selecting HSI generates a kernel clock request. **Caution**: HSI must be off if the PWR is programmed to use SVOS low.
4. MSI can remain activated in Stop mode if MSISTOPEN = 1, or if the peripheral selecting MSI generates a kernel clock request. **Caution**: MSI must be off if the PWR is programmed to use SVOS low.

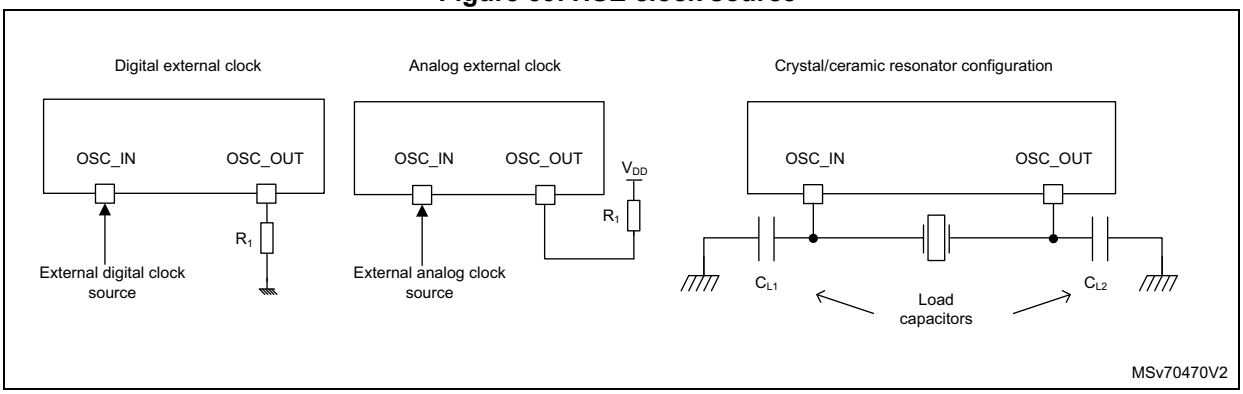
HSE oscillator

The HSE allows the application to provide a very accurate high-speed clock for the device. The HSE can generate an internal clock from two sources:

- external clock source (analog or digital)
- external crystal/ceramic resonator

Refer to the datasheet for the values of CL1, CL2, and R1.

Figure 39. HSE clock source



External clock source (HSE bypass)

In this mode, the oscillator is not used, and an external clock source must be provided to the OSC_IN pin. The external clock can be low swing (analog) or digital.

In order to allow the boot ROM to detect in which configuration the HSE is used, a resistor (R1) must be connected to GND or V_{DD} (see *Figure 39*).

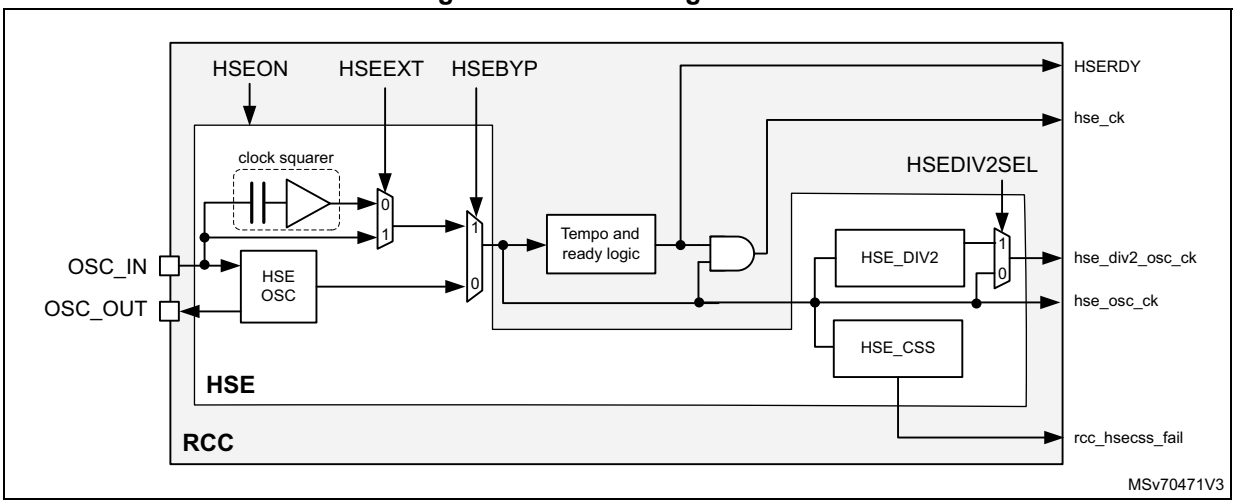
The resistor must be connected to GND when the HSE uses an external digital clock, and to V_{DD} when the HSE is using an external analog clock. The resistor must be removed if a crystal or ceramic resonator is used.

The external clock signal can be digital or analog (square, sinus, or triangle). An analog clock signal with a reduced amplitude is supported thanks to an internal clock squarer.

The input signal must have a duty cycle close to 50% (refer to the datasheet for additional information).

This mode is selected when HSEBYP = 1 in RCC_HSECFGR and HSEON = 1 in RCC_CR. In case of an analog clock input (low swing) HSEEXT must be set to 0 in RCC_HSECFGR. For a digital clock input, HSEEXT must be set to 1.

Figure 40. HSE clock generation



External crystal/ceramic resonator

A crystal/resonator can be connected as shown in *Figure 39*: the crystal/resonator and the load capacitors must be placed as close as possible to the oscillator pins in order to minimize the output distortion and startup stabilization time. The loading capacitance values must be adjusted according to the selected crystal or ceramic resonator. Refer to the electrical characteristics section of the datasheet for more details.

The oscillator mode is enabled by setting HSEBYP = 0 and HSEON = 1.

HSE ready logic

The HSERDY flag indicates when a valid clock is available at HSE output (hse_ck). When the HSE is enabled (HSEON = 1), the HSERDY flag goes to 1 when 1024 valid cycles of HSE have been detected. The hse_ck clock is not released until HSERDY goes to 1.

An interrupt can be generated if enabled in RCC_CIER.

HSE controls

The HSE can be switched on and off through HSEON.

The HSE is automatically disabled by hardware when the system enters Stop or Standby mode (see *Table 68*).

The HSE clock can also be driven to MCO1 and MCO2 outputs, and used as clock source for other application components.

HSE programming sequence

In order to initialize the HSE, the application must follow this sequence:

1. Make sure the HSE is not directly or indirectly used as system clock. If it is, switch to the HSI or MSI as clock source for system clock.
2. Disable the HSE by writing 0 to HSEON.
3. Check that the HSE is disabled by waiting HSERDY = 0.
4. If the oscillator mode is needed, select the oscillator mode with HSEBYP = 0.
5. If an external clock is connected to OSC_IN:
   – Select the bypass mode by setting HSEBYP = 1.

– If the input clock is a full-swing digital signal, set HSEEXT = 1.
– If the input clock is a low-swing signal, set HSEEXT = 0.

6. Enable again the HSE by writing 1 to HSEON.
7. Wait for HSERDY = 1, then the HSE is ready for use.

LSE oscillator

The LSE allows the application to provide a very accurate low-frequency clock for the device. The LSE can generate an internal clock from two possible sources:

- external user clock
- external crystal/ceramic resonator

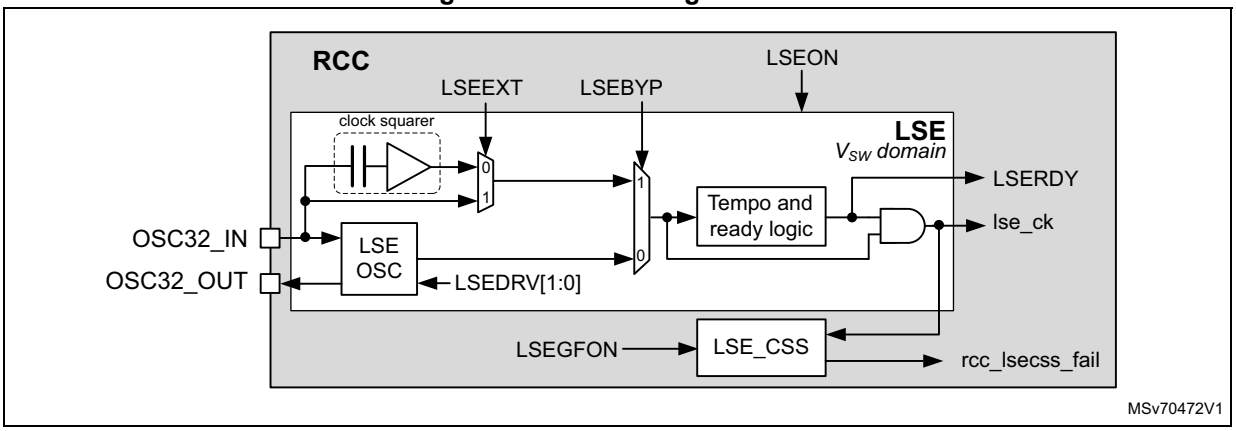
External clock source (LSE bypass)

In this mode, the oscillator is not used, and an external clock source must be provided to the OSC32_IN pin. The OSC32_OUT pin must be left high-Z.

The external clock signal can have a frequency up to 32.768 kHz, and can be digital or analog (square, sinus, or triangle). An analog clock signal with a reduced amplitude is supported thanks to an internal clock squarer. The input signal must have a duty cycle close to 50%. Refer to the datasheet for additional information.

This mode is selected by setting LSEBYP = 1 in RCC_LSECFGR, and LSEON = 1 in RCC_CR. In case of an analog clock input (low swing), LSEEXT must be set to 0 in RCC_LSECFGR. For a digital clock input, LSEEXT must be set to 1.

Figure 41. LSE clock generation



External crystal/ceramic resonator source (LSE crystal)

The LSE clock is generated from a 32.768 kHz crystal or ceramic resonator. It provides a low-power highly accurate clock source to the RTC for clock/calendar, or other timing functions. A crystal/resonator can be connected as shown in *Figure 39*. The crystal/resonator and the load capacitors must be placed as close as possible to the oscillator pins to minimize output distortion and startup stabilization time. The loading capacitance values must be adjusted according to the selected crystal or ceramic resonator. Refer to the electrical characteristics section of the datasheet for more details.

The oscillator mode is selected by setting LSEBYP bit to 0 and LSEON bit to 1.

The LSE offers a programmable driving capability (LSEDRV in RCC_LSECFGR) to modulate the amplifier driving capability. This driving capability is chosen according to the external crystal/ceramic component requirement to ensure a stable oscillation.

The driving capability must be set before enabling the LSE oscillator.

Warning: The driving capability must not be changed when the LSE is enabled. The LSE behavior is not guaranteed in that case.

LSE ready logic

The LSE offers an LSERDY flag, which indicates whether the LSE clock is available or not. When the LSE is enabled (LSEON = 1), LSERDY goes to 1 in RCC_SR when a certain number of valid LSE clock cycles has been detected. The lse_ck clock is not released until LSERDY goes to 1.

When LSEBYP = 0, the RCC waits 4096 clocks cycles before activating the LSERDY flag. When LSEBYP = 1, the RCC waits 16 clocks cycles.

An interrupt can be generated if enabled in RCC_CIER.

LSE controls

LSEBYP, LSEEXT, LSEDRV, and LSEON are write-protected by DBP in PWR_DBPCR. In order to modify the bits, DBP must be set 1.

The LSE oscillator is switched on and off using the LSEON bit.

The LSE remains enabled when the system enters Stop, Standby, or V_{BAT} mode (see *Table 68*).

The LSE clock can also be driven to MCOx outputs, and used as clock source for external components.

LSE programming sequence

To initialize the LSE, the application must follow the sequence hereafter:

1. Set DBP = 1 in PWR_DBPCR in order to allow write access.
2. Disable the LSE by writing to 0 to LSEON.
3. Check that the LSE is disabled by waiting LSERDY = 0.
4. If the oscillator mode is needed:
   - Select the oscillator mode by setting LSEBYP = 0.
   - Configure LSEDRV (if needed).
5. If an external clock is connected to OSC32_IN:
   - Select the bypass mode by setting LSEBYP = 1.
   - If the input clock is a full-swing digital signal, set LSEEXT = 1.
   - If the input clock is a low-swing signal, set LSEEXT = 0.
6. enable again the LSE by writing 1 to LSEON.
7. Wait for LSERDY = 1, then the LSE is ready for use.
8. If no further changes are needed, set DBP = 0 in PWR_DBPCR to write-protect the settings.

If the RTC is used, the LSE bypass must not be configured in digital mode, but in low-swing analog mode (default value after reset).

HSI oscillator

The HSI block provides the default clock to the device. It is a high-speed internal RC oscillator that can be used directly as system clock, peripheral clock, or as PLL input. A predivider allows the application to select an HSI output frequency of 8, 16, 32, or 64 MHz. This predivider is controlled by the HSIDIV in RCC_HSICFGR.

The HSI advantages are the following:

- low-cost clock source (no external crystals required)
- faster startup time than HSE (a few microseconds)
- reduced power consumption

The HSI frequency, even with frequency calibration, is less accurate than an external crystal oscillator or ceramic resonator.

HSI controls

The HSI can be switched on and off using HSION in RCC_CR. The HSIRDY flag in RCC_SR indicates if the HSI is stable or not. At startup, the HSI output clock is not released until HSIRDY is set to 1 by hardware.

The HSI clock can also be used as a backup source (auxiliary clock) if the HSE fails (see *CSS on HSE*).

The HSI can be disabled or not when the system enters Stop mode (see *Table 68*).

The HSI clock can also be driven to MCOx outputs, and used as clock source for other application components.

Care must be taken when the HSI is used as kernel clock for communication peripherals. The application must take into account the following parameters:

- the time interval between the moment where the peripheral generates a kernel clock request, and the moment where the clock is really available
- the frequency accuracy

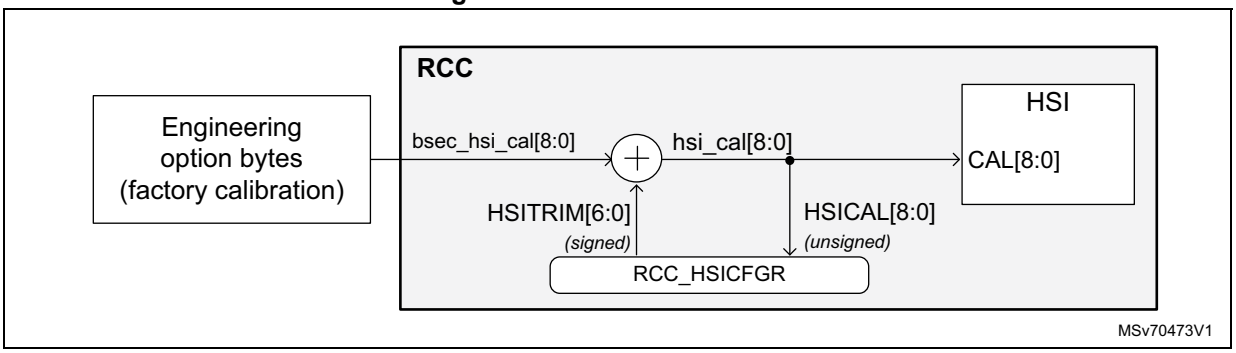
HSI calibration

RC oscillator frequencies can vary from one chip to another due to manufacturing process variations. That is why each device is factory calibrated by STMicroelectronics to achieve an ACC_{HSI} accuracy (refer to the product datasheet for more information).

After a power-on reset or pin reset, the factory calibration value is loaded in HSICAL[8:0] in RCC_HSICFGR.

If the application is subject to voltage or temperature variations, this may affect the RC oscillator frequency. The user application can trim the HSI frequency using HSITRIM[6:0] in RCC_HSICFGR.

Figure 42. HSI calibration flow



Note: The HSI clock divided by eight is also used for PAD compensation mechanism, and must be enabled if the PAD compensation mechanism is activated. Refer to Section 16: System configuration controller (SYSCFG) for additional details.

MSI oscillator

The MSI is a low-power RC oscillator that can be used directly as system clock, peripheral clock, or PLL input.

However, the following point must be considered: If the MSI clock is currently used as kernel clock for some peripherals, the application must ensure that the MSI frequency change does not disturb these peripherals.

The MSI advantages are the following:

- low-cost clock source (no external crystals required)
- faster startup time than HSE (a few microseconds)
- very low-power consumption

The MSI provides a clock frequency of 4 MHz (default MSIFREQSEL) or 16 MHz, while the HSI is able to provide a clock up to 64 MHz.

The MSI frequency, even with frequency calibration, is less accurate than an external crystal oscillator or ceramic resonator.

MSI controls

The MSI can be switched on and off through the MSION in RCC_CR. The MSIRDY flag in RCC_SR indicates whether the MSI is stable or not. At startup, the MSI output clock is not released until MSIRDY is set by hardware.

The MSI can be disabled or not when the system enters Stop mode (see *Table 68*).

The MSI clock can also be driven to MCOx outputs, and used as clock source for other application components.

Even if the MSI settling time is faster than the HSI, care must be taken when the MSI is used as kernel clock for communication peripherals: the application must take into account the following parameters:

- the interval between the moment when the peripheral generates a kernel clock request, and the moment when the clock is really available
- the frequency precision

MSI calibration

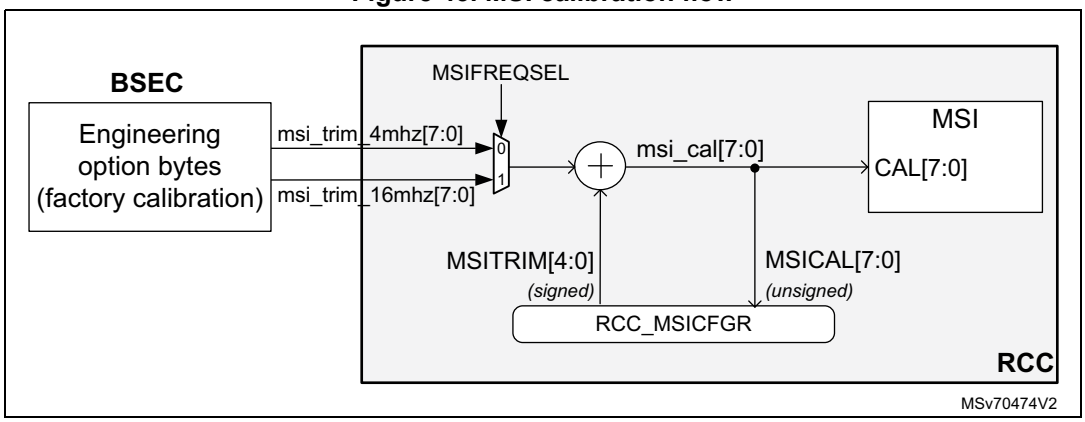
RC oscillator frequencies can vary because of manufacturing process variations. Each device is factory calibrated by ST to achieve the specified ACC_{MSI} accuracy (refer to the product datasheet for more information).

After a power-on or pin reset, the factory calibration value for 4 MHz is loaded in MSICAL[7:0] in RCC_MSICFGR.

If MSIFREQSEL is set to 16 MHz in RCC_MSICFGR, a different calibration value is provided by the BSEC.

Voltage or temperature variations can affect the RC oscillator frequency. The user application can trim the MSI frequency using MSITRIM[4:0] in RCC_MSICFGR.

Figure 43. MSI calibration flow



HSIS oscillator

The HSIS is a 64 MHz RC oscillator to clock only the BSEC. It is always activated after pwr_por_rstn or app_rstn reset.

When the system goes into Stop or Standby mode, the HSIS clock is disabled by hardware. Refer to *Section 14.6.7* for additional information.

HSIS calibration

RC oscillator frequencies can vary from one device to another, due to manufacturing process variations. To compensate for this, there is an HSISCAL[8:0] input on the oscillator.

The BSEC provides two calibration values (ambient and not ambient). The RCC selects between these two values using a select signal from the BSEC.

After a power-on reset, or pad reset, the factory calibration value is loaded in HSISCAL[8:0].

LSI oscillator

The LSI acts as a very low-power clock source that can be kept running when the system is in Stop or Standby mode for the IWDG and the auto-wake-up unit (AWU). The clock frequency is around 32 kHz. For more details, refer to the electrical characteristics section of the datasheet.

The LSI can be switched on and off using LSION. The LSIRDY flag indicates whether the LSI oscillator is stable or not. If an independent watchdog is started either by hardware or software, the LSI is forced on, and cannot be disabled.

The LSI remains enabled when the system enters Stop or Standby mode (see *Table 68*).

At LSI startup, the clock is not provided until the hardware sets LSIRDY. An interrupt can be generated if enabled in RCC_CIER.

Te LSI clock can also be driven to MCOx outputs, and used as a clock source for other application components.

14.6.3 Clock security system (CSS)

The CSS can detect a failure of either (or both) LSE and HSE oscillators. There are signals that can be connected to the TAMP (rcc_lsecss_fail and rcc_hsecss_fail), and signals for the interrupt controller (rcc_lsecss_it, rcc_hsecss_it, and rcc_it).

CSS on HSE

The CSS can be enabled by software via HSECSSON. This bit can be enabled even when HSEON = 0.

The CSS on HSE is activated when the HSE is enabled and ready, and when the software sets HSECSSON = 1. The CSS on HSE does no longer work when the HSE is disabled. For example, this function does not work when the system is in Stop mode.

HSECSSON cannot be cleared directly by software. It is cleared by hardware when a system reset occurs, or when the system enters Standby mode (see *Section 14.5.2*).

On an HSE failure, an HSI injection feature can automatically inject a divided HSI clock in replacement at the root of the HSE tree. Users of the failed HSE keep running, but potentially at a slightly lower frequency. The HSI injected clock is adapted to the HSE frequency by an integer division. The PLLs relocks, but at the same or lower speed.

To enable the automatic HSI injection, first configure HSECSSBPRE in RCC_HSECFGR, then set HSECSSBYP = 1.

The HSI division ratio is configured with HSECSSBPRE. For instance, with the HSI at 64 MHz and an HSE at 48 MHz, the division ratio must be configured to 2x (HSECSSBPRE = 1): a failed HSE is replaced by a clock at 64 / 2 = 32 MHz.

When the CSS on HSE is enabled, the following actions are done by the RCC if a failure is detected:

- If the HSI injection feature is enabled, the HSI oscillator is forced active, and the HSE clock is replaced by hsi_css_ck.
- rcc_hsecss_fail is asserted.
- The clock failure event (rcc_hsecss_fail) is also sent to the break inputs of advanced-control timers (TIM1/8/15/16/17).
- An NMI interrupt is generated to inform the software about the failure (rcc_hsecss_it). This allows the MCU to perform rescue operations. The NMI interrupt is asserted until HSECSSF = 0 in RCC_CICR. The HSECSSF flag can be cleared by setting HSECSSC = 1.
- A tamper event can also be triggered to clear content of backup registers and BKPSRAM.

CSS on LSE

A CSS on the LSE oscillator can be enabled by software by programming LSECSSON. This bit is disabled by hardware if one of the following conditions is met:

- after a V_{SW} hardware reset (pwr_vsw_rst)
- after a V_{SW} software reset via VSWRST bit

The software can also disable the CSS after an LSE failure detection.

The CSS on LSE works in all modes (Run, Stop, and Standby modes) including V_{BAT} mode.

The LSECSS provides a re-arm feature, offering the possibility to the software to re-arm the LSECSS, and to re-enable the LSE clock when a failure has been detected. This feature allows the application to decide if the LSE must be provided again to the RTC even if a failure occurred, or if another action must be performed. For example, the application can decide to reset the V_{SW} domain only if a certain number of consecutive LSE failures occurred, within a time window.

The LSECSS offers two flag signals:

- the LSECSSD able to retain an LSE failure even in V_{BAT} mode
- the LSECSSF used to generate an interrupt in case of LSE failure (flag not affected by a failure detected when the product is in V_{BAT} mode)

The sequence hereafter describes the LSE that enables sequence with the CSS enabled:

1. Follow the LSE enable procedure given in *LSE programming sequence*, except the last step.
2. Select the LSE clock via RTCSEL[1:0].
3. Set the LSECSSON bit to 1.
4. If no further changes are needed, clear DBP to 0 in PWR_DBPCR to write-protect accesses.

Note: The LSECSSON bit must be enabled after the LSE is enabled (LSEON set by software) and ready (LSERDY set by hardware), and after the RTC clock has been selected through RTCSEL.

If a failure is detected on the LSE, the hardware does the following:

- The LSE clock is no more delivered to the RTC.
- RTCSEL, LSECSSON, and LSEON are not changed by the hardware.
- A failure event is generated (rcc_lsecss_fail). This event allows the system to wake up from Standby mode, but also to protect the backup registers and BKPSRAM via TAMP. This event is also generated in V_{BAT} mode.
- The LSECSSF is activated (except in V_{BAT} mode) in order to generate an interrupt (rcc_lsecss_it, enabled by LSECSSIE).
- The LSECSSD is activated as well, retaining the first LSE failure even in V_{BAT} mode.

On the software side, different actions can be taken according to the application requirements. Three different cases are described hereafter in order to illustrate the hardware behavior, they can also be combined. The application can also decide to handle LSE failure differently.

Case A

The application no longer wants to use LSE when a failure is detected:

1. Unlock registers by setting DBP in PWR_DBPCR to 1.

2. Disable the CSS function (this step is mandatory):
   a) Clear LSECSSF if the interrupt was enabled for this event.
   b) Clear LSECSSON to 0.
   c) Clear LSEON to 0 in order to disable the LSE.
3. Change the clock source for the RTC if needed:
   a) Clear RTCEN to 0 to disable the RTC clock.
   b) Enable the new clock source for the RTC.
   c) Set RTCPRE if HSE is a new clock source.
   d) Select the proper clock source via RTCSEL.
   e) Set RTCEN to 1 to enable the RTC clock.
4. The application must perform specific actions for TAMP events if enabled (see *Section 4: Boot and security control (BSEC)* and *Section 61: Real-time clock (RTC)*).

- Lock registers by clearing DBP to 0 in PWR_DBPCR

Case B

The application wants to re-initialize the V_{SW} domain:

1. Unlock registers by setting the DBP bit of PWR_DBPCR to 1
2. Perform a VSW reset by setting VSWRST bit to 1, then back to 0.
3. The application must perform specific actions for TAMP events if enabled (see *Section 4: Boot and security control (BSEC)* and *Section 61: Real-time clock (RTC)*).
4. Re-initialize all components of the V_{SW} domain.
5. Lock registers by clearing DBP to 0 in PWR_DBPCR.

Case C

The application tries to reuse LSE when a failure is detected:

1. If the number of failures in a given time window is higher than a given threshold then go to case A or B. Otherwise, continue to next step.
2. Unlock registers by setting DBP to 1 in PWR_DBPCR.
3. Clear LSECSSF if interrupt was enabled for this event.
4. The application must perform specific actions for TAMP events if enabled (see *Section 4: Boot and security control (BSEC)* and *Section 61: Real-time clock (RTC)*).
5. Clear LSECSSON to 0.
6. Rearm the LSECSS function by writing 1 to LSECSSRA, then back to 0.
7. Wait for LSERDY = 1. The LSERDY flag must go to 1 after the oscillator settling time delay plus, 4096 periods of LSE clock. If it is not the case, it probably means that the LSE failure is permanent. LSECSSON cannot be set to 1. It is recommended to execute case A or B.
8. Set LSECSSON to 1.
9. When LSECSSON = 1, the LSE is enabled, and protected by LSECSS.
10. Lock registers by clearing DBP to 0 in PWR_DBPCR.

14.6.4 Clock output generation (MCO1/MCO2)

There are two MCO1 and MCO2 microcontroller clock output pins. A clock source can be selected for each output.The selected clock can be divided thanks to a configurable prescaler (refer to *Figure 38* for additional information on signal selection).

MCO1 and MCO2 are enabled using MCO1EN and MCO2EN in RCC_MISCENR.

The GPIO port corresponding to each MCO pin must be programmed in alternate function mode.

MCO1 and MCO2 are controlled via MCO1PRE[3:0], MCO1SEL[2:0], MCO2PRE[3:0], and MCO2SEL[2:0] located in RCC_CCIPR5.

MCO1PRE and MCO2PRE dividers provide a clock with a duty cycle of 50% for even divisions values, and around 53% for odd division values.

Note: MCO1 and MCO2 are available in Run, Stop, and Sleep modes.

Caution: The clock provided to the MCOx outputs must not exceed the maximum pin speed (refer to the product datasheet for information about the supported pin speed).

Table 69 shows the signals available on each MCO output.

Table 69. Clock output selection

MCO1SEL		MCO2SEL	
Position	**Clock source**	**Position**	**Clock source**
0	hsi_div_ck	0	hsi_div_ck
1	lse_ck	1	lse_ck
2	msi_ck	2	msi_ck
3	lsi_ck	3	lsi_ck
4	hse_ck	4	hse_ck
5	ic5_ck	5	ic15_ck
6	ic10_ck	6	ic20_ck
7	sysa_ck	7	sysb_ck

14.6.5 PLL description

The RCC features four PLLs with the same features.

A typical allocation is:

- PLL1, clocks to the CPU, buses, and storage (XSPI, SDMMC)
- PLL2, clocks to NPU and audio peripherals
- PLL3, clocks to CACHEAXI RAM and Ethernet
- PLL4, clocks to display, camera, FDCAN, and other peripherals

Each PLL has the following features:

- FREF frequency range:
  - 5 to 1200 MHz in integer mode
  - 10 to 1200 MHz in fractional mode
- VCO frequency range from 800 to 3200 MHz
- Three working modes:

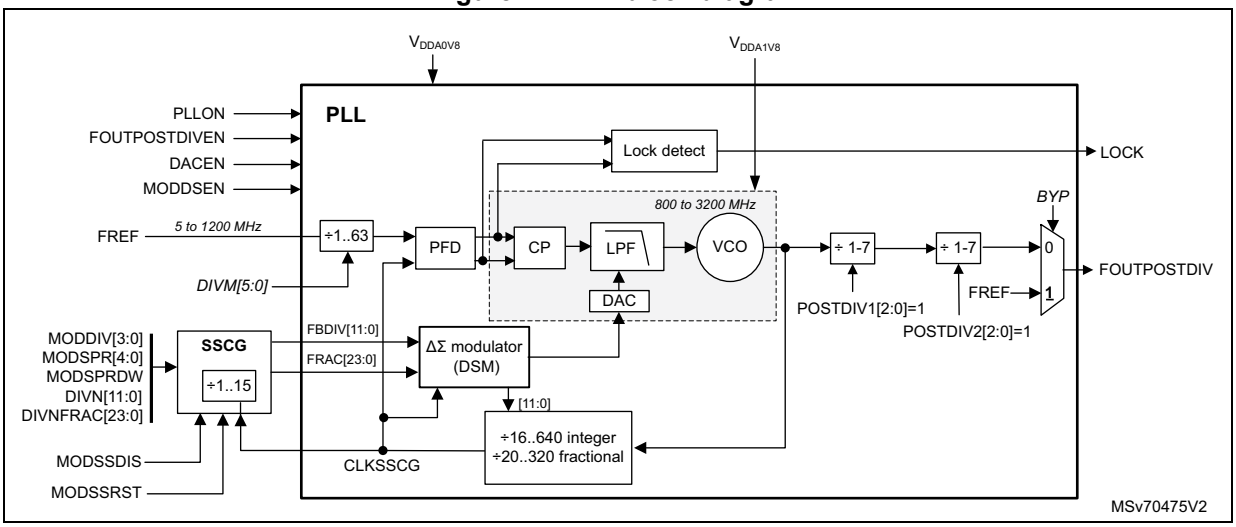
– fractional mode, using a 24-bit delta-sigma modulator (DSM)
– integer mode
– spread spectrum mode to reduce EMI. The fully digital spread spectrum clock generator (SSCG) is used in that case.

The internal post-dividers (POSTDIV1, POSTDIV2) are powered-off by default. They must be powered-on when the PLL is in use (PLLxPDIVEN).

The active post-dividers (ICx) are outside the PLL design. Each post-divider has a 4-way multiplexer before it, which can select any PLL output as input.

The DIVMx divider in RCC_PLLxCFGR1 must be properly programmed to keep the PFD input frequency below 50 MHz.

Figure 44. PLL block diagram



The PLL is enabled by setting PLLxON to 1 in RCC_CR. PLLxRDY in RCC_SR indicates that the PLL is ready (locked).

The DIVNx loop divider must be programmed to achieve the expected VCO output frequency before enabling the PLL. Changing the value on-the-fly can result in a spike on the VCO output proportional to the PFD frequency step. A frequency step of more than 0.01% per PFD clock period must be avoided. The SSCG typically steps the frequency by less than 0.005% per PFD clock period, so does not generate spikes.

The VCO output range must be respected.

The clock from FOUTPOSTDIV has a 50% duty-cycle (± 3%).

The ICx post-dividers provide clocks with 50% duty-cycle when dividing by an even value.

If an ICx post-divider enable is set to 0, its value can be changed without disabling any PLL.

The PLLs are disabled by hardware when the system enters Stop or Standby mode.

PLLs using HSE as reference clock are also disabled by hardware if an HSE failure is detected.

PLL programming recommendations

- Before enabling the PLLs, the user must ensure that the reference frequency (FREF) provided to the PLL is stable and in the correct range.
- POSTDIV1 and POSTDIV2 must be set to 1.
- PLLxPDIVEN must be set to 1 to output the clock FOUTPOSTDIV.
- When a PLL output is used, PLLxON and PLLxRDY must be set to 1. The application can then set any connected post-divider enable bits to 1 (in RCC_DIVEN).
- When a PLL output is not used, PLLxON must be cleared to 0. The application must also set any connected post-divider enable bits to 0 (in RCC_DIVEN).

Caution: The 4 MHz setting for the MSI oscillator cannot be used as FREF.

The PLLs can work in three different modes:

- integer
- fractional
- spread spectrum

Using PLLs in integer mode

The PLLx works in integer mode when the delta-sigma modulator (DSM) is loaded with a 0 value, and PLLxMODSSDIS = 1.

To load 0 into the DSM and to set DIVN, use the following sequence:

1. Clear PLLxON to 0.
2. Set DIVN value (valid range 16 to 640).
3. Clear DIVNFRAC (in RCC_PLLxCFGR2) and PLLxMODDSEN to 0.
4. Set PLLxMODSSRST to 1.
5. Set PLLxON to 1.

Caution: Do not update DIVN after the PLL has been enabled.

The VCO frequency (F_{VCO}) and output frequency expressions are the following:

$$F_{VCO} = F_{REF} \times \frac{DIVN}{DIVM}$$

$$F_{OUTPOSTDIV} = \frac{F_{VCO}}{POSTDIV1 \times POSTDIV2}$$

Using the PLLs in fractional mode

This mode is enabled when DSM ≠ 0, PLLxMODDSEN = 1, and PLLxMODSSDIS = 1.

To load the value into the DSM perform the following sequence:

1. Clear PLLxON to 0.
2. Set DIVN value (valid range 20 to 320).
3. Set DIVNFRAC (in RCC_PLLxCFGR2) to the required value, and set PLLxMODDSEN = DACEN 1.
4. Set PLLxMODSSRST to 1.
5. Set PLLxON to 1.

Caution: Do not update DIVN and DIVNFRAC after the PLL has been enabled.
The minimum FREF is 10 MHz in fractional mode.

The VCO frequency (F_{VCO}) and output frequency expressions are the following:

$$F_{VCO} = F_{REF} \times \frac{\left(DIVN + \frac{DIVNFRAC}{2^{24}}\right)}{DIVM}$$

$$F_{OUTPOSTDIV} = \frac{F_{VCO}}{POSTDIV1 \times POSTDIV2}$$

Using PLLs in spread spectrum mode

The spread spectrum mode is activated when the DSM is loaded with 0, and PLLxMODSSDIS is cleared to 0. This feature is available for all PLLs.

The spread spectrum method is to modulate the VCO frequency with a low-frequency triangular signal, in order to spread the clock energy into a wider frequency band. The amount of emitted EMI is then reduced.

The spread spectrum modulation is adjusted using the following fields:

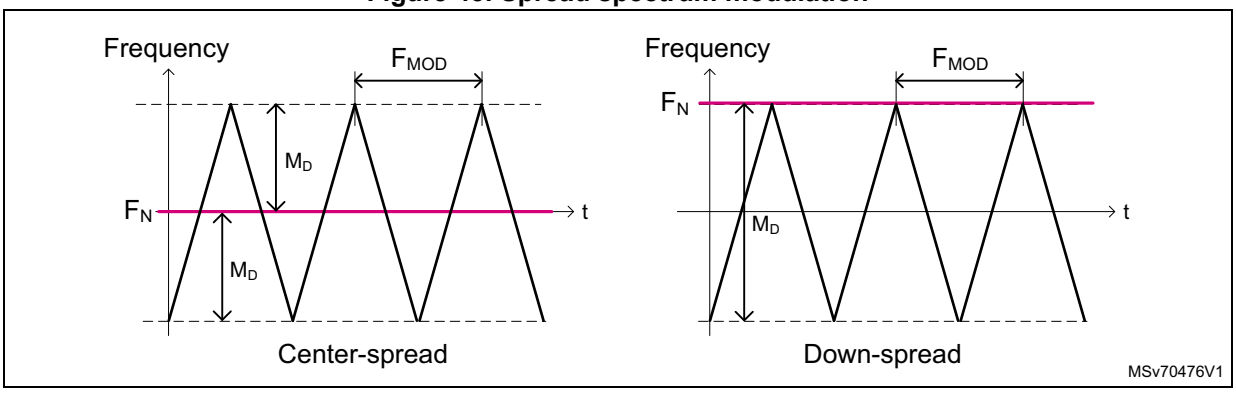
- MODDIV[3:0] to adjust the modulation frequency
- MODSPR[4:0] to adjust the modulation depth (or modulation index)
- MODSPRDW to define if the modulation is centered around the VCO frequency (center-spread), or lowered with respect to VCO frequency (down-spread)

MODDIV[3:0], MODSPR[4:0] and MODSPRDW are in RCC_PLLxCFGR3.

Figure 45 shows the SSCG modulating the nominal frequency (F_N), when MODSPRDW = 0 (center-spread), and MODSPRDW = 1 (down-spread). The nominal frequency is that output by the PLL in integer mode, when no clock spreading is applied.

Down-spread guarantees that the PLL output frequency does not exceed the programmed frequency value when SSCG is enabled.

Figure 45. Spread spectrum modulation



The peak modulation depth (in percentage) is given by the formula M_D (%) = MODSPR /10.

Note: Modulation is turned off when MODSPR = 0.

The modulation frequency (F_{MOD}) is given by:

$$F_{MOD} = \frac{F_{CLKSSCG}}{128 \times MODDIV}$$

where 128 is the number of points in the internal wave table.

Note: When the PLL is locked, $F_{CLKSSCG} = F_{PFD}$. The upper limit of the frequency of modulation (F_{MOD}) is set by the PLL bandwidth. The PLL bandwidth limits the maximum modulation to $F_{CLKSSCG}$ / 200, where $F_{CLKSSCG} = F_{REF}$ / DIVM or 50 MHz, whichever is lower.

To use the spread spectrum feature, to do the following:

1. Program the PLL to the nominal frequency (F_N) using the $F_{OUTPOSTDIV}$ formula from *Using PLLs in integer mode*
2. Compute the MODDIV value according to the desired modulation frequency (F_{MOD}):

$$MODDIV = ROUND\left(\frac{F_{CLKSSCG}}{128 \times F_{MOD}}\right)$$

3. Compute the MODSPR value according to the desired modulation depth (M_D).
4. Set the MODSPRDW value according to the desired modulation type (center-spread or down-spread).
5. Compute DIVN accordingly (DIVNFRAC=0):

$$DIVN = ROUND\left(\frac{F_N \times DIVM}{F_{REF}}\right)$$

6. Clear PLLxMODSSDIS, PLLxMODDSEN, and DACEN to 0.
7. Set PLLxMODSSRST to 1, and clear PLLxON to 0. PLLxON must be held at 0 for 1 µs to make sure the PLL is fully reset.
8. Set PLLxON, PLLxMODDSEN, and DACEN to 1 (see *Note:*).
9. Wait until the first edge of CLKSSCG, and then clear PLLxMODSSRST to 0 (this can be done before or after the PLL is locked).

Note: The spread spectrum accuracy relies on the PLL fractional-N capability, so MODDSEN and DACEN must be set to 1.

The user can check F_{MIN}, F_{MAX} as follows:

- Calculate F_N as above.
- If MODSPRDW = 0 (center-spread):

  $F_{MIN} = F_N \times (1 - M_D / 100)$ and $F_{MAX} = F_N \times (1 + M_D / 100)$
- If MODSPRDW = 1 (down-spread):

  $F_{MIN} = F_N \times (1 - M_D / 100)$ and $F_{MAX} = F_N$

Programming sequence for spread spectrum mode

The programming sequence to enable SSCG and the PLL is:

1. Deassert PLLxMODSSRST.
2. Set PLLxDIVN, PLLxDIVNFRAC, and PLLxMODSSDIS to 0.
3. Assert PLLxMODDSEN and PLLxDACEN.
4. Assert PLLxMODSSRST and deassert PLLxON.
5. PLLxON must be deasserted for 1µs to make sure PLL is fully reset. Then assert PLLxON.
6. Wait until PLLxRDY is asserted, then deassert PLLxMODSSRST.

14.6.6 System clocks

System clock selection

After a system reset, the HSI is selected as system clock (sys[a,b,c,d]_ck), and all PLLs are switched off.

The system clock can be stopped by hardware when the system enters Stop or Standby mode.

When the system runs, the user can select system clocks (sys[a,b,c,d]_ck) from the four following sources:

- HSE
- HSI
- MSI
- ic[1,2,6,11]_ck

This function is controlled by programming RCC_CFGR1. A switch from one clock source to another occurs only if the target clock source is ready (clock stable after startup delay or PLL locked). If a clock source that is not yet ready is selected, the switch occurs when the clock source is ready.

The SYSSW only selects the ic[2,6,11]_ck if all three IC dividers are enabled.

SWS bits in RCC_CFGR1 indicate which clock sources are currently selected. Other status bits in RCC_CR indicate which clock(s) is (are) ready.

System clock generation

Figure 46 shows a simplified view of the clock distribution for the CPU and buses. All the dividers shown in the block diagram can be changed on-the-fly, without generating timing violations. This feature is a very simple solution to adapt bus frequencies to application needs, thus optimizing the power consumption.

The AXI sys_bus_ck is divided by HPRE to generate the AHB clock. HPRE is controlled by RCC_CFGR2.

In addition to the divide values shown, PPRE1, PPRE2, PPRE4, and PPRE5 can divide by 32, 64, and 128.

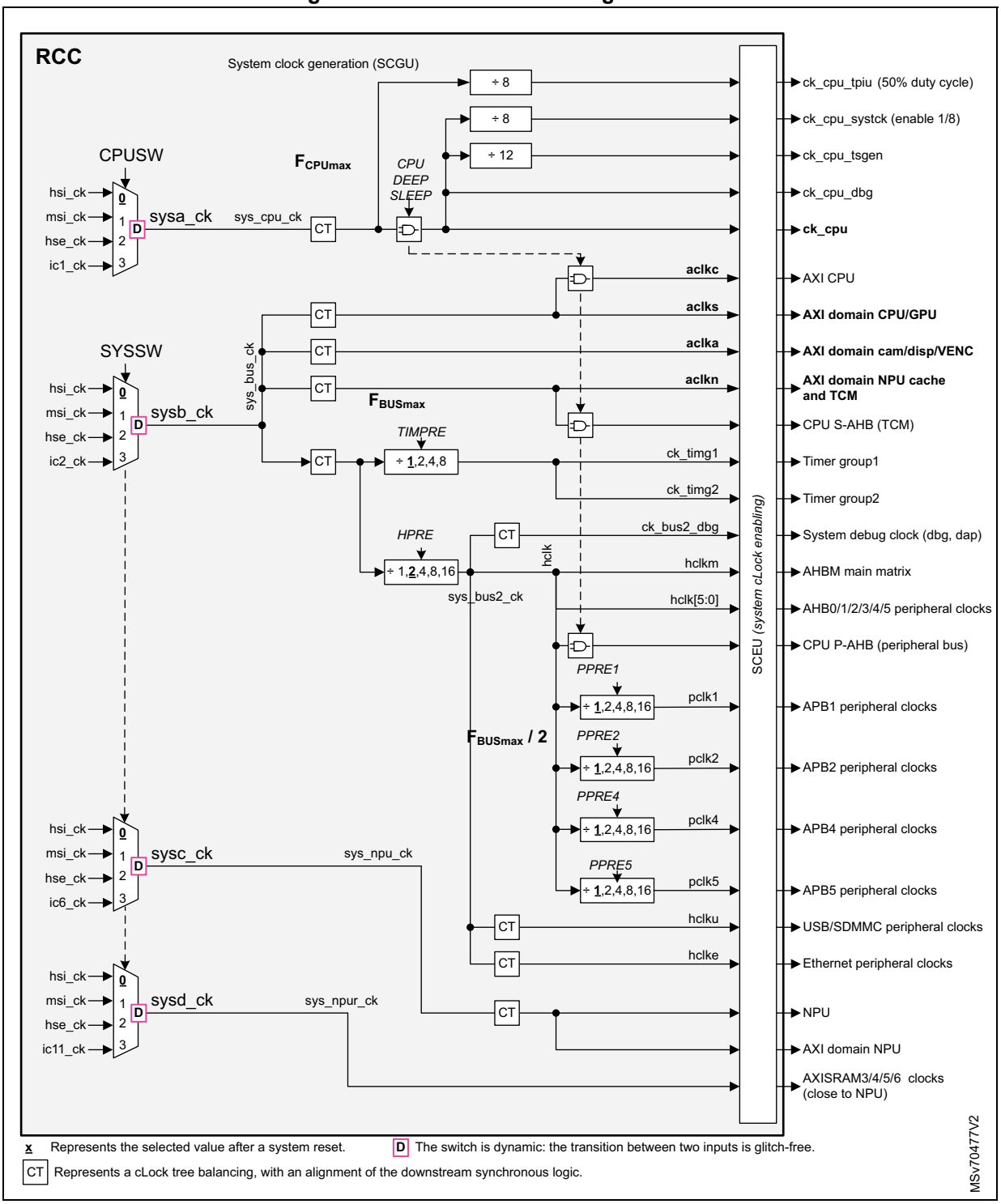
There is almost no clock protection, so the software must avoid configurations that can block the system.

Note: The application must respect the maximum allowed frequencies: F_{CPUmax} and F_{BUSmax}.

F_{BUS} represents the maximum allowed frequency for AHB and AXI buses (refer to the datasheet for the maximum values).

The trace clock (ck_cpu_tpiu) is generated from sys_cpu_ck clock, divided by eight. For additional information, refer to Clock distribution for trace and debug.

Figure 46. Core and bus clock generation



1. Dividers values can be changed on-the-fly. All dividers have 50% duty-cycles.

14.6.7 Clock generation in Stop and Standby modes

When the system enters Stop mode, all clocks (system and kernel) are stopped, and the following clock sources are disabled as well:

- MSI, HSI
- HSE
- PLL1, PLL2, PLL3, PLL4

Note: The MSI and HSI stay active based on xxxSTOPEN bits in RCC_STOPCR.

HSIS is also disabled.

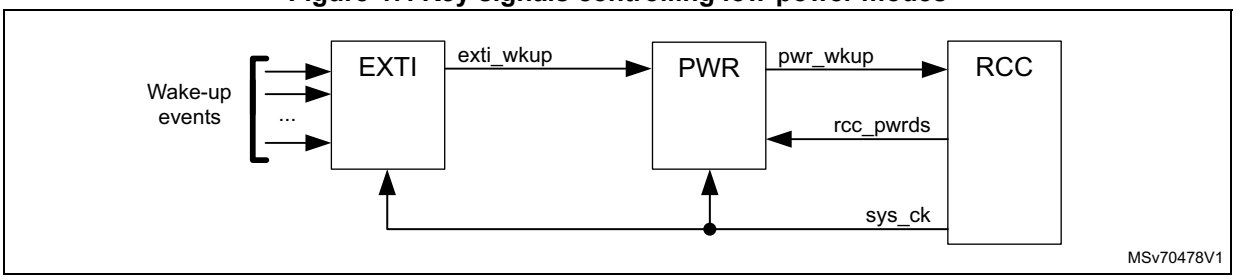
The content of the RCC registers is not altered, except CPUSW and SYSSW, forced to HSI or MSI (depending on STOPWUCK value), and PLLxON and HSEON, set to 0.

HSION and MSION are also modified, depending on STOPWUCK (see *Table 70*).

When the CPU requests to go in Stop mode, the RCC first stops all requested clocks, and informs the PWR that all clocks have been properly stopped. As shown in *Figure 47*, three main signals are used to control power transitions:

- rcc_pwrds: used to indicate to the PWR that the RCC stopped all clocks. The PWR can then go to Stop or Standby mode.
- pwr_wkup is used to indicate to the RCC to re-enable the clocks.
- The exti_wkup is used to indicate to the PWR that an event requests to exit the system from Stop mode.

Figure 47. Key signals controlling low-power modes



Exiting Stop mode

When the device exits system Stop mode via a wake-up event, HSIS is started automatically.

Note: sys_rst is only deasserted after HSIS has successfully started.

The application can select which other oscillator (HSI and/or MSI) is used to restart the system. STOPWUCK in RCC_CFGR1 selects the oscillator used as system clock. *Table 70* describes their behavior.

Table 70. STOPWUCK description

STOPWUCK	Activated oscillator when system exits Stop mode	Distributed clocks when system exits Stop mode	
		System clock	Kernel clock
0	HSI	HSI	HSI
1	MSI	MSI	MSI

During Stop mode

There are two specific cases where the HSI or MSI can be enabled during system Stop mode:

- When a dedicated peripheral requests the kernel clock, the peripheral receives the HSI or MSI according to the kernel clock source selected for this peripheral (via PERxSEL).
- When HSISTOPEN or MSISTOPEN are set in RCC_STOPCR, the HSI and MSI are kept running during Stop mode but the outputs are gated. The clock is then available immediately when the system exits Stop mode, or when a peripheral requests the kernel clock (see *Table 71* for details).

Caution: HSI and MSI are always off in Stop mode when the PWR is set to SVOS low.

Table 71. HSISTOPEN and MSISTOPEN behavior

HSISTOPEN (MSISTOPEN)	HSI (MSI) state during Stop mode	HSI (MSI) setting time
0	Off	$t_{su(HSI)}$ ($t_{su(MSI)}$) (1)
1	Running and gated	Immediate

1. $t_{su(HSI)}$ and $t_{su(MSI)}$ are the startup times of the HSI and MSI oscillators (see the datasheet for their values).

When the microcontroller exits Standby mode, the HSI is selected as system and kernel clock. RCC registers are reset to their initial values except for the backup domain configurations (LSE in RCC_CR/RCC_LSECFGR, RTC in RCC_CCIPR7, RCC_BDCR), and the reset cause (RCC_RSR, RCC_HWRSR).

Caution: When leaving Stop mode without reset (but not from Standby mode), the RCC returns in the same state as before, except for the software that has been forced to select the STOPWUCK source. When leaving Standby mode, the application can restore previous CPU clock settings, if needed.

Caution: If the system clock switch selection (SYSSW) is HSI or MSI oscillator, STOPWUCK (system clock selection after a wake-up from system Stop) must select the same oscillator.

14.6.8 Peripheral clock distribution

Some peripherals are designed to work with two different clock domains, operating asynchronously:

- a domain synchronous with the register and bus interface (ckg_bus_perx clock)
- a domain generally synchronous with the peripheral (kernel clock)

Other peripherals only need a bus interface clock, hence the user application has more freedom to choose an optimized clock frequency for the CPU, bus matrix, and for the kernel part of the peripheral. The user can change the bus frequency without reprogramming peripherals (example: an ongoing transfer with UART is not disturbed if its APB clock is changed on-the-fly).

Table 72 summarizes the clocks from RCC to the peripherals. The clock named per_ck is the output of a mux (see *Figure 38*).

Table 72. Peripheral clock distribution summary

Peripherals	Clock		Kernel clock MUX		Max kernel clock frequency (in MHz)	Type (1)(2)
	Type	Source	Position	Control field		
ADF1	Kernel	hclk2	**0**(3)	ADF1SEL	200	A
		per_ck	1			
		ic7_ck	2			
		ic8_ck	3			
		msi_ck	4			
		hsi_div_ck	5			
		I2S_CKIN	6			
		timg_ck	7			
	Bus	hclk2	-	-	200	-
ADC12	Kernel	hclk1	**0**(3)	ADC12SEL	125	A
		per_ck	1			
		ic7_ck	2			
		ic8_ck	3			
		msi_ck	4			
		hsi_div_ck	5			
		I2S_CKIN	6			
		timg_ck	7			
	Bus	hclk1	-	-	200	-
CACHEAXI	Bus	sys_bus_ck	-	-	400	-
		hclk5	-	-	200	-
CRC	Bus	hclk4	-	-	200	-
CRYP	Bus	hclk3	-	-	200	-
DBG	Kernel	sys_cpu_ck	-	-	800	A
	Bus	ck_bus2_dbg	-	-	200	-
DTS	Kernel	hsi_div8_ck	-	-	10	A
	Bus	pclk4	-	-	100	-
CSI	Kernel	As DCMIPP	-	DCMIPPSEL	-	A
	Bus	pclk5	-	-	200	-
CSIPHY	Kernel	ic18_ck	-	-	20	A
	Bus	pclk5	-	-	200	-

Table 72. Peripheral clock distribution summary (continued)

Peripherals	Clock		Kernel clock MUX		Max kernel clock frequency (in MHz)	Type (1)(2)
	Type	Source	Position	Control field		
DCMIPP	Kernel	pclk5	0	DCMIPPSEL	333	A
		per_ck	1			
		ic17_ck	2			
		hsi_div_ck	3			
	Bus	sys_busa_ck	-	-	400	-
		pclk5	-		200	
GPDMA1	Bus	hclk1	-	-	200	-
HPDMA1	Bus	hclk5	-	-	200	-
DMA2D	Bus	hclk5	-	-	200	-
		aclk				
DTS	Kernel	hsi_div8_ck	-	-	10	A
	Bus	pclk4	-	-	200	-
ETH1	Kernel	ETH1_TX_CLK	-	-	25	A
		ETH1_RX_CLK/ETH1_REF_CLK	**0**(3)	ETH1REFCLKSEL	125	A
		eth1_clk_fb	1			
		sys_bus2_ck	**0**(3)	ETH1CLKSEL	125	A
		per_ck	1			
		ic12_ck	2			
		hse_ck	3			
		sys_bus2_ck	0	ETH1PTPSEL	200	A
		per_ck	1			
		ic13_ck	2			
		hse_ck	3			
	Bus	hclk1	-	-	200	-
EXTI	Bus	pclk4	-	-	125	-
FDCAN	Kernel	pclk1	**0**(3)	FDCANSEL	150	A
		per_ck	1			
		ic19_ck	2			
		hse_ck	3			
	Bus	pclk1	-	-	200	-

Table 72. Peripheral clock distribution summary (continued)

Peripherals	Clock		Kernel clock MUX		Max kernel clock frequency (in MHz)	Type (1)(2)
	Type	Source	Position	Control field		
FMC	Kernel	hclk5	**0**(3)	FMCSEL	200	A
		per_ck	1			
		ic3_ck	2			
		ic4_ck	3			
	Bus	sys_buss_ck	-	-	400	-
		hclk5				
GPIOA-H, GPION-Q	Bus	hclk4	-	-	200	-
GPU2D	Bus	sys_buss_ck	-	-	400	-
GFXMMU	Bus	sys_buss_ck	-	-	400	-
		hclk5				
GFXTIM	Bus	pclk5	-	-	200	-
HASH	Bus	hclk3	-	-	200	-
I2C1, I2C2, I2C3	Kernel	pclk1	**0**(3)	I2C1SEL, I2C2SEL, I2C3SEL	100	A
		per_ck	1			
		ic10_ck	2			
		ic15_ck	3			
		msi_ck	4			
		hsi_div_ck	5			
	Bus	pclk1	-	-	100	-
I2C4	Kernel	pclk1	**0**(3)	I2C4SEL	100	A
		per_ck	1			
		ic10_ck	2			
		ic15_ck	3			
		msi_ck	4			
		hsi_div_ck	5			
	Bus	pclk4	-	-	100	-
I3C1, I3C2	Kernel	pclk1	**0**(3)	I3C1SEL, I3C2SEL	100	A
		per_ck	1			
		ic10_ck	2			
		ic15_ck	3			
		msi_ck				
		hsi_div_ck				
	Bus	pclk1	-	-	100	-

Table 72. Peripheral clock distribution summary (continued)

Peripherals	Clock		Kernel clock MUX		Max kernel clock frequency (in MHz)	Type (1)(2)
	Type	Source	Position	Control field		
IWDG	Kernel	lsi_ck	-	-	1	A
	Bus	pclk4	-	-	100	-
JPEG	Bus	hclk5	-	-	200	-
LPTIM1	Kernel	pclk1	**0**(3)	LPTIM1SEL	200	A
		per_ck	1			
		ic15_ck	2			
		lse_ck	3			
		lsi_ck	4			
		timg_ck	5			
	Bus	pclk1	-	-	200	-
LPTIM2, LPTIM3, LPTIM4, LPTIM5	Kernel	pclk4	**0**(3)	LPTIM2SEL. LPTIM3SEL. LPTIM4SEL, LPTIM5SEL	200	A
		per_ck	1			
		ic15_ck	2			
		lse_ck	3			
		lsi_ck	4			
		timg_ck	5			
	Bus	pclk4	-	-	200	-
LPUART1	Kernel	pclk4	**0**(3)	LPUART1SEL	100	A
		per_ck	1			
		ic9_ck	2			
		ic14_ck	3			
		lse_ck	4			
		msi_ck	5			
		hsi_div_ck	6			
	Bus	pclk4	-	-	100	-
LTDC	Kernel	pclk5	0	LTDCSEL	**86**	A
		per_ck	1			
		ic16_ck	2			
		hsi_div_ck	3			
	Bus	pclk5	-	-	200	-
		sys_busa_ck	-	-	400	
MCE1, MCE2, MCE3	Bus	aclk	-	As XSPI1, XSPI2, XSPI3	200	-
		hclk5	-			

Table 72. Peripheral clock distribution summary (continued)

Peripherals	Clock		Kernel clock MUX		Max kernel clock frequency (in MHz)	Type (1)(2)
	Type	Source	Position	Control field		
MCE4	Bus	aclk	-	As FMC	200	-
		hclk5	-			
MDF1	Kernel	hclk2	**0**(4)	MDF1SEL	200	A
		per_ck	1			
		ic7_ck	2			
		ic8_ck	3			
		msi_ck	4			
		hsi_div_ck	5			
		I2S_CKIN	6			
		timg_ck	7			
	Bus	hclk2	-	-	200	-
MDIOS	Bus	pclk1	-	-	200	-
NPU	Kernel	sys_npu_ck	-	-	1000	A
		sys_npur_ck	-	-	900	A
	Bus	sys_bus_ck	-	-	400	
		sys_bus2_ck	-	-	200	-
OTGPHY1, OTGPHY2	Kernel	hse_div2_ck	**0**(3)	OTGPHY1SEL, OTGPHY2SEL	48	A
		per_ck	1			
		ic15_ck	2			
		hse_div2_osc_ck	3			
	Kernel	otgphy1_ker_ck, otgphy2_ker_ck	0	OTGPHY1CK REFSEL, OTGPHY2CK REFSEL,	200	-
		hse_div2_osc_ck	1			
OTG1, OTG2	Kernel	phyclock	-	-	60	A
	Bus	hclku	-	-	200	-
PKA	Bus	hclk3	-	-	200	-
PWR	Bus	hclk4	-	-	200	-
PSSI	Kernel	hclk5	**0**(3)	PSSISEL	40	-
		per_ck	1			-
		ic20_ck	2			
		hsi_div_ck	3			
	Bus	hclk5	-	-	200	-
RCC	Bus	hclk	-	-	200	-

Table 72. Peripheral clock distribution summary (continued)

Peripherals	Clock		Kernel clock MUX		Max kernel clock frequency (in MHz)	Type (1)(2)
	Type	Source	Position	Control field		
RNG	Kernel	hsis_osc_ck	-	-	64	A
	Bus	hclk3	-	-	200	-
RTC(5)	Kernel	no clock	**0**(3)	RTCSEL	4	A
		lse_ck	1			
		lsi_ck	2			
		hse_ker_ck / (RTCDIV+1)	3			
	Bus	pclk4	-	-	100	-
SAES	Kernel	hclk3	-	-	200	A
	Bus	hclk3	-	-	200	-
SAI1, SAI2	Kernel	pclk2	**0**(3)	SAI1SEL, SAI2SEL	200	A
		per_ck	1			
		ic7_ck	2			
		ic8_ck	3			
		msi_ck	4			
		hsi_div_ck	5			
		I2S_CKIN	6			
		spdif_symb_ck	7			
	Bus	pclk2	-	-	200	-
SDMMC1	Kernel	sys_bus2_ck	**0**(3)	SDMMC1SEL	208	A
		per_ck	1			
		ic4_ck	2			
		ic5_ck	3			
	Bus	sys_bus2_ck	-	-	200	-
SDMMC2	Kernel	sys_bus2_ck	**0**(3)	SDMMC2SEL	208	A
		per_ck	1			
		ic4_ck	2			
		ic5_ck	3			
	Bus	sys_bus2_ck	-	-	200	-

Table 72. Peripheral clock distribution summary (continued)

Peripherals	Clock		Kernel clock MUX		Max kernel clock frequency (in MHz)	Type (1)(2)
	Type	Source	Position	Control field		
SPDIFRX1	Kernel	pclk1	**0**(3)	SPDIFRX1SEL	200	A
		per_ck	1			
		ic7_ck	2			
		ic8_ck	3			
		msi_ck	4			
		hsi_div_ck	5			
		I2S_CKIN	6			
	Bus	pclk1	-	-	200	-
SPI1	Kernel	pclk2	**0**(3)	SPI1SEL	200	A
		per_ck	1			
		ic8_ck	2			
		ic9_ck	3			
		msi_ck	4			
		hsi_div_ck	5			
		I2S_CKIN	6			
	Bus	pclk2	-	-	200	-
SPI2, SPI3	Kernel	pclk1	**0**(3)	SPI2SEL, SPI3SEL	200	A
		per_ck	1			
		ic8_ck	2			
		ic9_ck	3			
		msi_ck	4			
		hsi_div_ck	5			
		I2S_CKIN	6			
	Bus	pclk1	-	-	200	-
SPI4, SPI5	Kernel	pclk2	**0**(3)	SPI4SEL, SPI5SEL	133	A
		per_ck	1			
		ic9_ck	2			
		ic14_ck	3			
		msi_ck	4			
		hsi_div_ck	5			
		hse_ck	6			
	Bus	pclk2	-	-	200	-

Table 72. Peripheral clock distribution summary (continued)

Peripherals	Clock		Kernel clock MUX		Max kernel clock frequency (in MHz)	Type (1)(2)
	Type	Source	Position	Control field		
SPI6	Kernel	pclk4	**0**(3)	SPI1SEL	200	A
		per_ck	1			
		ic8_ck	2			
		ic9_ck	3			
		msi_ck	4			
		hsi_div_ck	5			
		I2S_CKIN	6			
	Bus	pclk4	-	-	200	-
TIM2, TIM3, TIM4, TIM5, TIM6, TIM7, TIM10, TIM11, TIM12, TIM13, TIM14	Kernel	timg1_ck	-	-	400	S
	Bus	pclk1	-	-	200	-
TIM1, TIM8, TIM9, TIM15, TIM16, TIM17, TIM18	Kernel	timg2_ck	-	-	400	S
	Bus	pclk2	-	-	200	-
USART1	Kernel	pclk2	**0**(3)	USART1SEL	100	A
		per_ck	1			
		ic9_ck	2			
		ic14_ck	3			
		lse_ck	4			
		msi_ck	5			
		hsi_div_cl	6			
	Bus	pclk2	-	-	100	-
USART2, USART3, UART4, UART5, UART7, UART8	Kernel	pclk1	**0**(3)	USART2SEL, USART3SEL, UART4SEL, UART5SEL, UART7SEL, UART8SEL	100	A
		per_ck	1			
		ic9_ck	2			
		ic14_ck	3			
		lse_ck	4			
		msi_ck	5			
		hsi_div_ck	6			
	Bus	pclk1	-	-	100	-

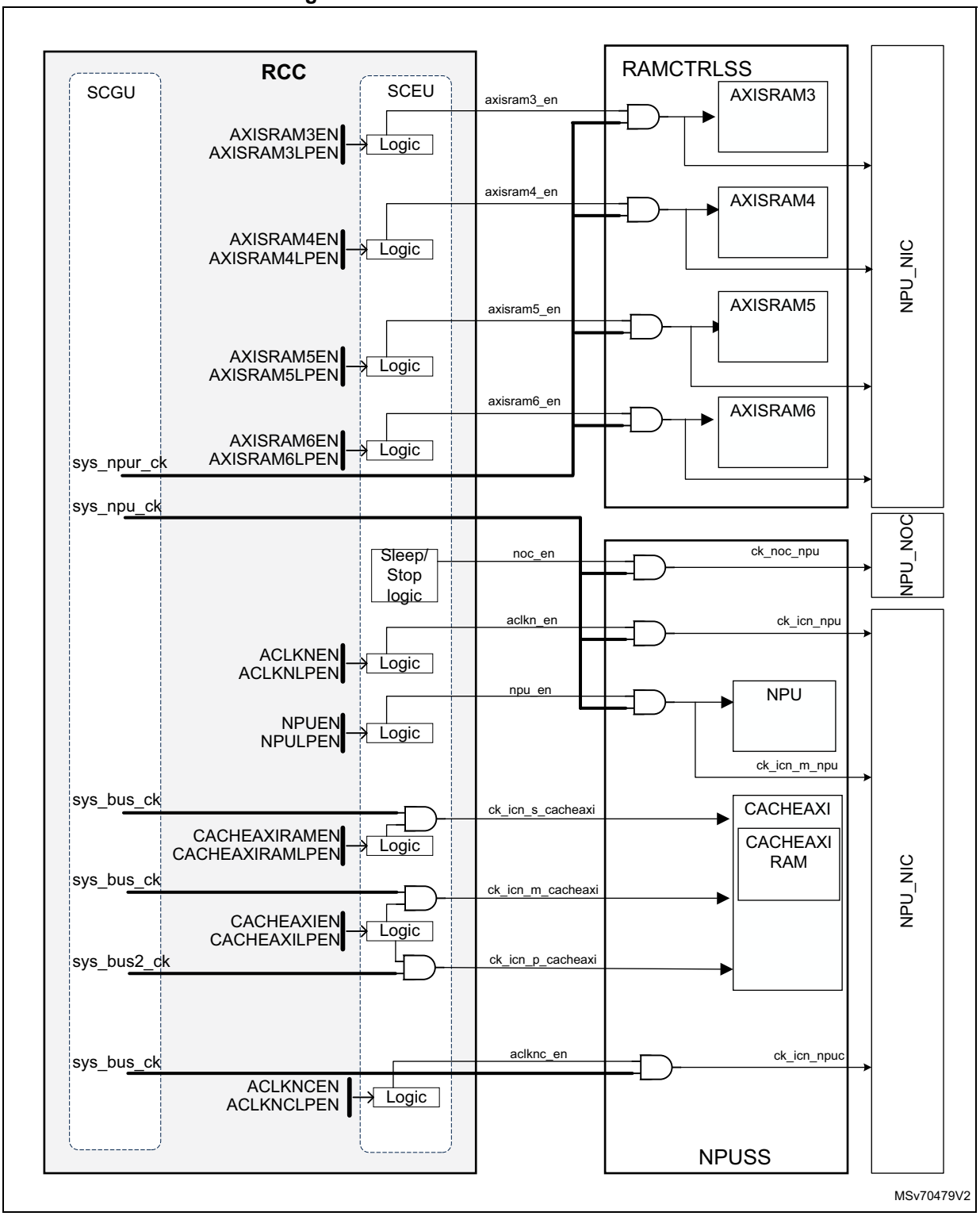
Table 72. Peripheral clock distribution summary (continued)

Peripherals	Clock		Kernel clock MUX		Max kernel clock frequency (in MHz)	Type (1)(2)
	Type	Source	Position	Control field		
USART6, UART9, USART10	Kernel	pclk2	**0**(3)	USART6SEL, UART9SEL, USART10SEL	100	A
		per_ck	1			
		ic9_ck	2			
		ic14_ck	3			
		lse_ck	4			
		msi_ck	5			
		hsi_div_ck	6			
	Bus	pclk2	-	-	100	-
UCPD1	Kernel	hsi_div4_ck	-	-	25	A
	Bus	pclk1	-	-	100	-
VENC	Bus	sys_busa_ck	-	-	400	-
		pclk5	-	-	200	-
VENCRAM	Bus	sys_buss_ck	-	-	400	-
VREFBUF	Bus	pclk4	-	-	200	-
WWDG1	Bus	pclk1	-	-	200	-
XSPI1, XSPI2, XSPI3	Kernel	hclk5	**0**(3)	XSPI1SEL, XSPI2SEL, XSPI3SEL	200	A
		per_ck	1			
		ic3_ck	2			
		ic4_ck	3			
	Bus	sys_buss_ck	-	-	200	-
XSPIM	Bus	hclk5	-	-	200	-

1. 'A' means that the kernel clock is asynchronous with respect to bus interface clock.
2. ‘S’ means that the kernel clock is synchronous with respect to bus interface clock.
3. Reset value.
4. Reset value.
5. The RTC switch is in the VSW voltage domain.

Clock distribution for the NPU

Figure 48. Clock distribution for the NPU



Clock distribution for graphic blocks (GPU, LTDC, DCMIPP, and PSSI)

Figure 49. Clock distribution for PSSI, CSI, and DCMIPP

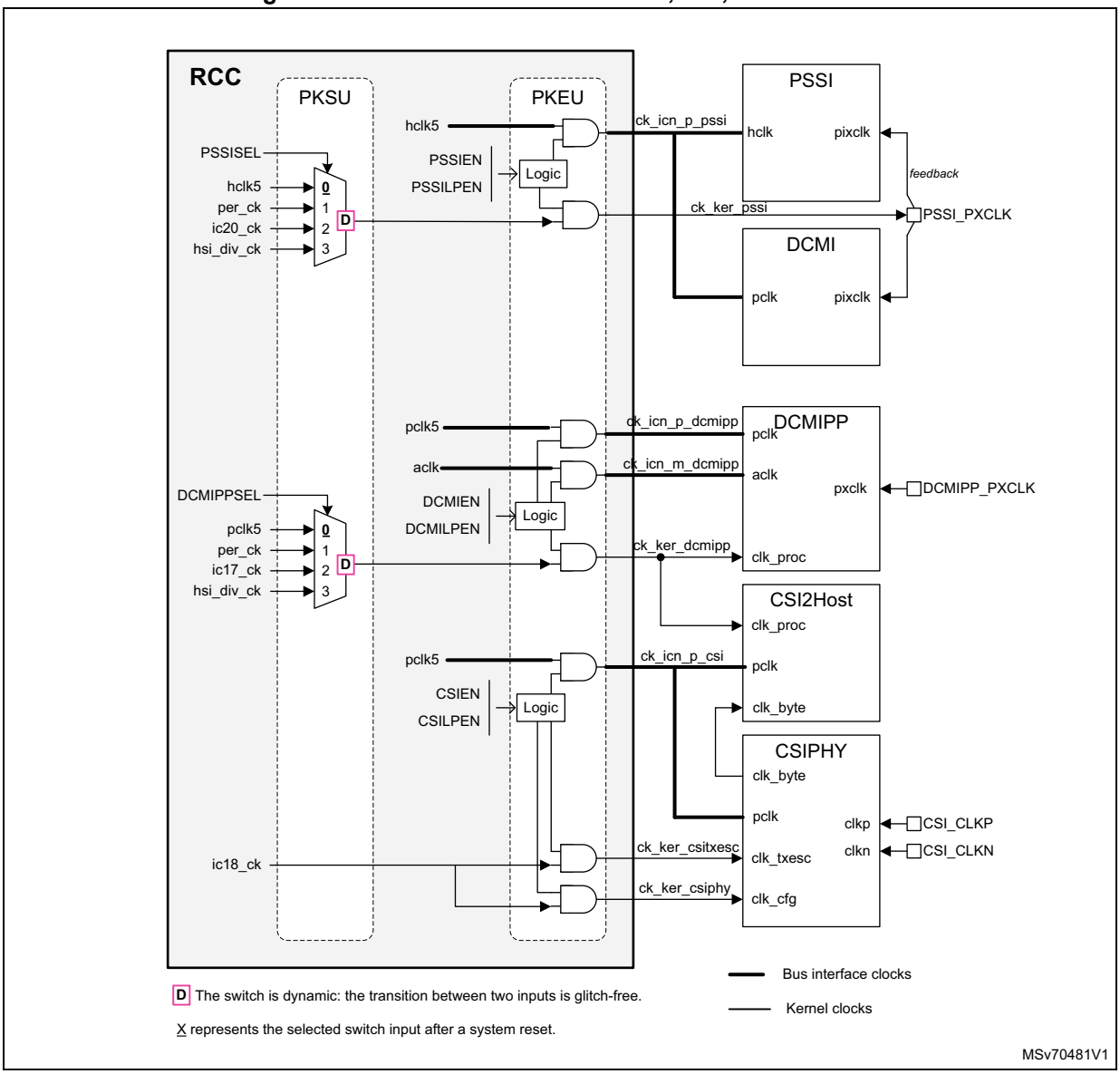
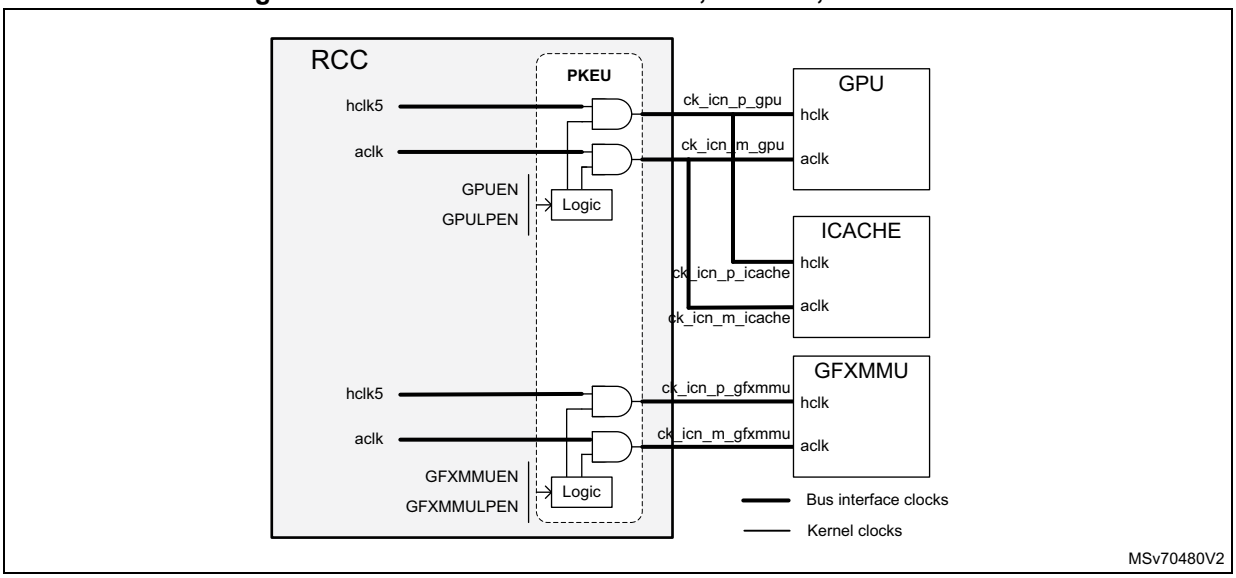


Figure 50. Clock distribution for GPU, ICACHE, and GFXMMU



The PSSI receives an AHB clock and a kernel clock (pxclk). The pxclk can be provided either by an external device via PSSI_PIXCK pin, or by the RCC.

Note: The clock generated by the RCC is provided to pxclk input by the feedback path of the PSSI_PIXCK pin. The drive of the PSSI_PIXCK is controlled by the PSSI.

Figure 51. Clock distribution for LTDC

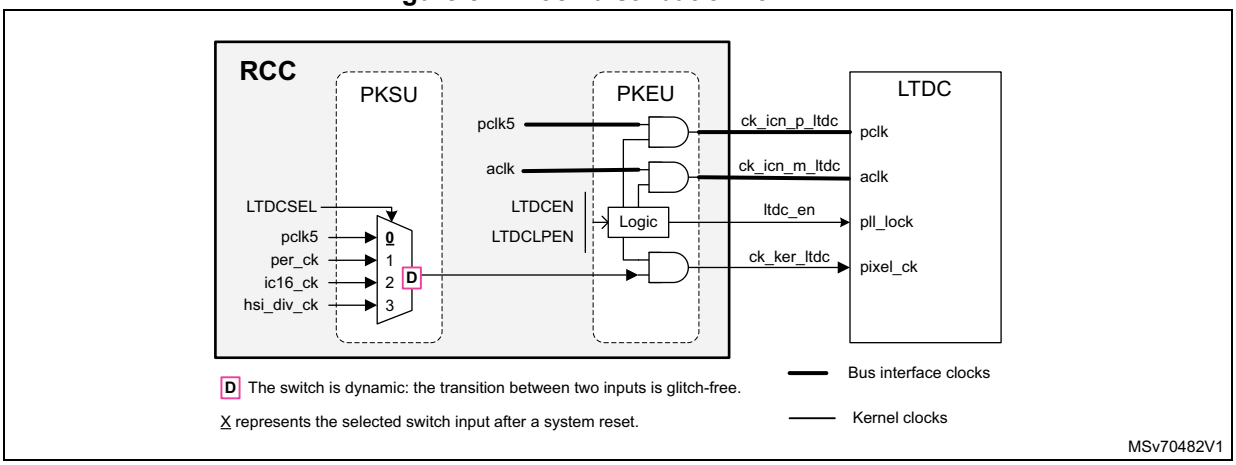
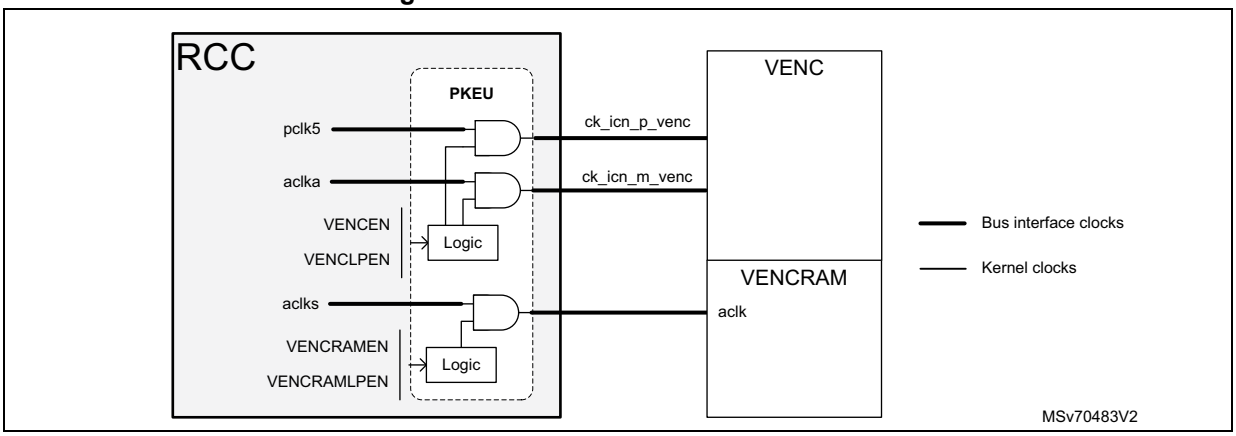


Figure 52. Clock distribution for VENC



Clock distribution for OTG1, OTG2, and UCPD1

Figure 53 shows the clock distribution for:

- USB Type-C Power Delivery block (UCPD1): uses ucpd1_ker_ck as kernel clock. ucpd1_ker_ck is directly generated from the HSI output divided by four.
- OTGPHYx: embeds a PLL that accepts a reference input frequency of 19.2, 20, or 24 MHz. The reference clock can be selected among one of the following:
  - hse_div2_osc_ck
  - hse_div2_ck
  - ic15_ck
  - per_ck

  hse_div2_osc_ck is direct from the HSE oscillator, without the tempo delay. It can be selected to be HSE or HSE/2.

  The OTGPHYx provides a 60 MHz clock (phyclock) to the OTGx.
- OTGx: uses the phyclock when working in HS or FS mode.

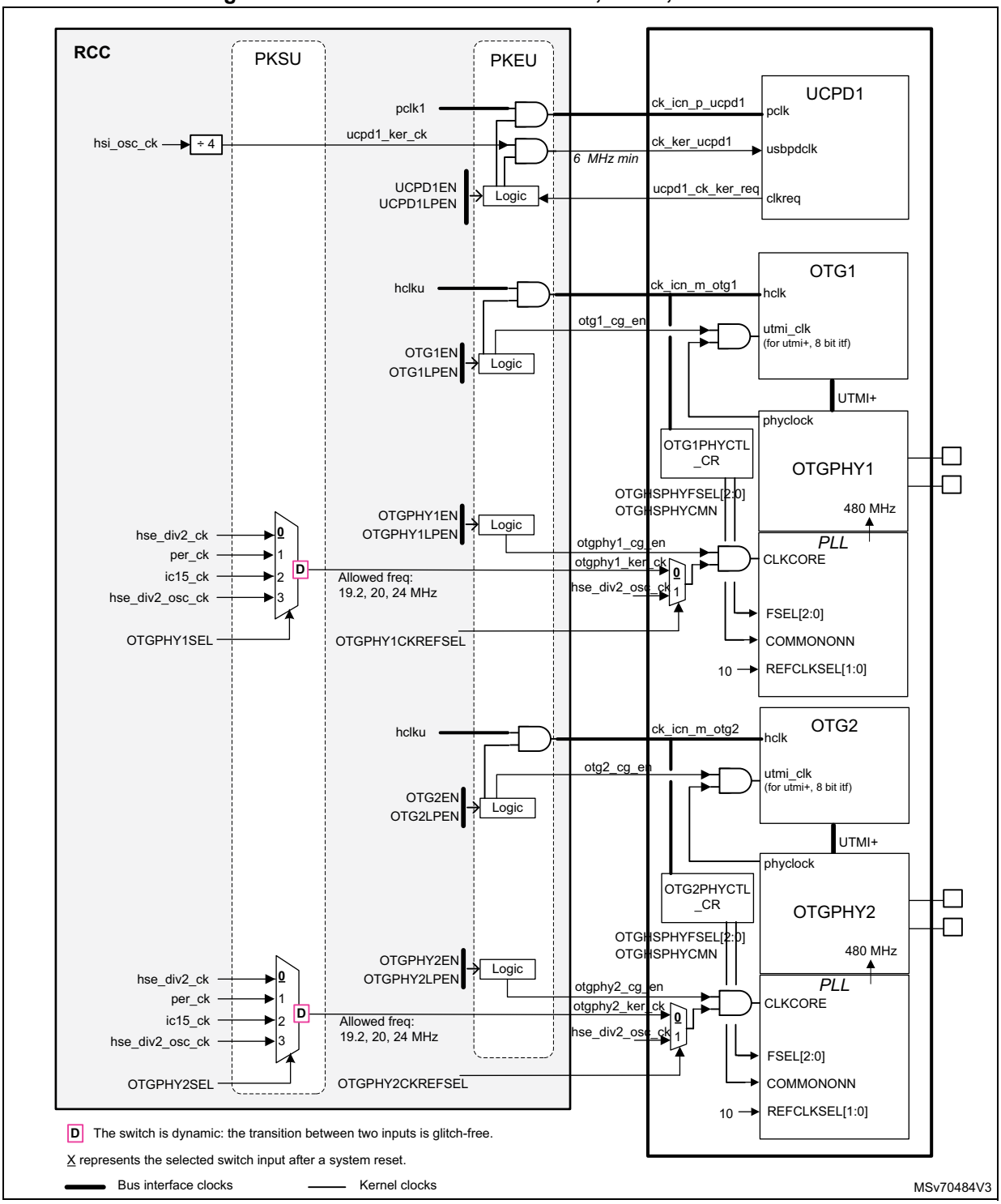
The reference clock selection for the OTGPHYx is performed by a simple multiplexer. To change the clock source, the application must use the following sequence:

1. Disable the OTGPHYx clock by clearing OTGPHYxEN to 0.
2. Change the clock source selector (OTGPHYxSEL) to the desired value.
3. Enable the OTGPHYx clock by setting OTGPHYxEN to 1.

Clocks provided by the RCC are controlled by enable bits in RCC_AHB5ENR.

Note: Before programming OTG1PHYCTL_CR, OTG1EN must be asserted (the OTG1PHYCTL_CR logic requires the clock enabled by OTG1EN).

Figure 53. Clock distribution for OTG1, OTG2, and UCPD1

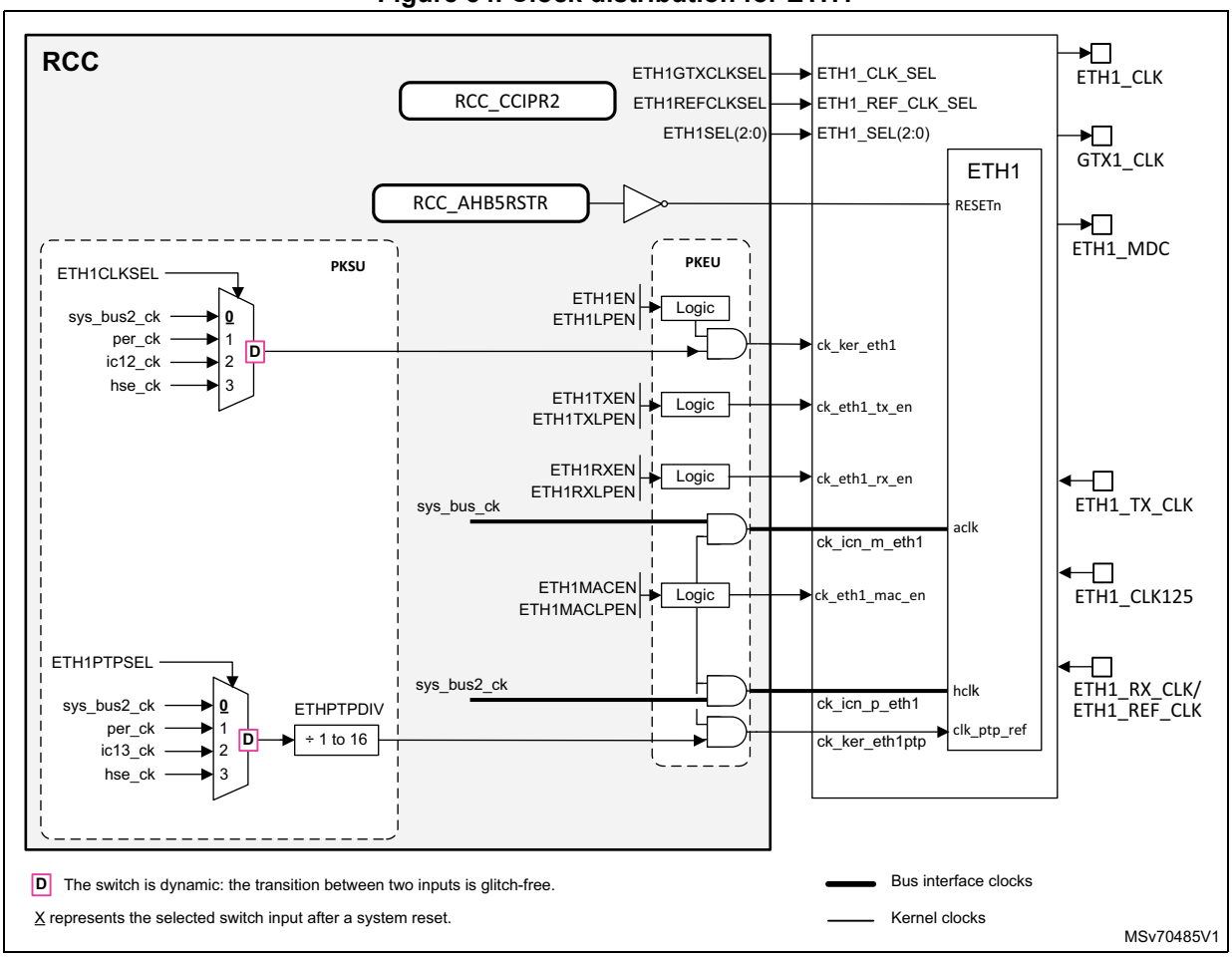


Clock distribution for ETH1

Kernel Ethernet clocks are provided by the RCC, who provides also clock selectors (CLK_SEL, REF_CLK_SEL, SEL), and clock enables for TX and RX used in the ETHSS.

Note: Bus and PTP clocks are gated via ETH1MACEN and ETH1MACLPEN bits.

Figure 54. Clock distribution for ETH1

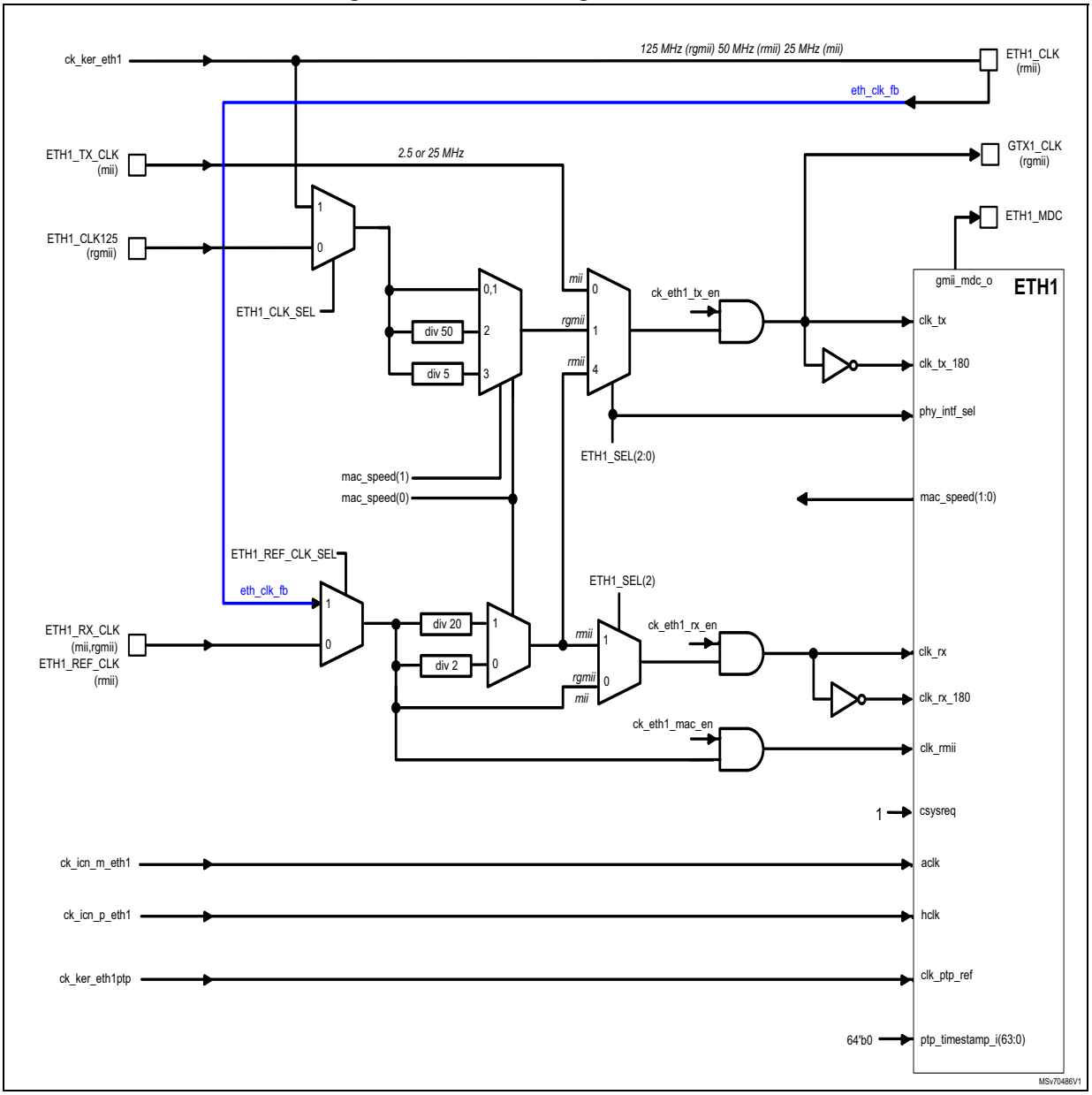


The ETH1 can generate a reference clock to the external PHY via the ETH1_CLK pin. The ETH1_CLK is generated only if all the following conditions are met:

- ETH1EN is enabled.
- The system is in Run or Sleep mode.
- The clock source for ck_ker_eth1 is available.

The clock management for ETH is very flexible and based on the PHY interface mode (MII, RMII, or RGMII). All clock signals (enables, selection, and pins) are shown in *Figure 55*.

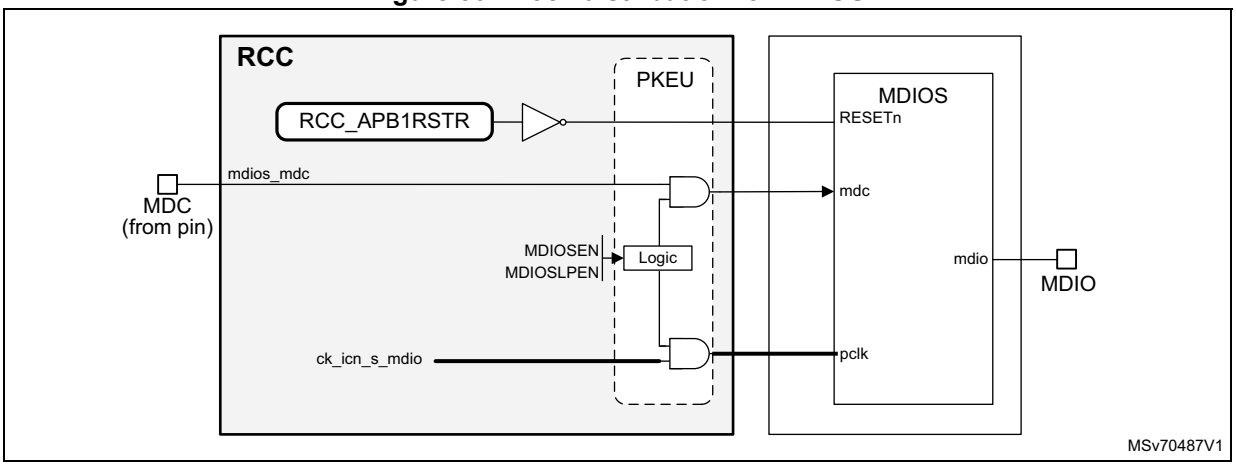
Figure 55. Clock management for ETH1



Clock distribution for MDIOS

MDIOS (MDIO slave) clocks are provided by the RCC.

Figure 56. Clock distribution for MDIOS



Clock distribution for FMC, XSPIs, and SDMMCs

The FMC kernel clock can be chosen between four different sources. For each XSPI, a clock switch can be used to select between four different sources. Each XSPI can be enabled independently.

The following steps are needed to correctly configure XSPI switches:

1. Switch on the desired clock source.
2. Ensure the clock source is ready, and conditions described above are met.
3. Set XSPIxSEL to the desired position.
4. Enable the XSPI clock (XSPIxEN = 1).

The XSPIs provide a clock to the external memory with a duty-cycle distortion lower than 5%. To this end, the kernel clock provided to the XSPIs has a typical duty cycle of 50%. In addition, the XSPIs embed a prescaler allowing clock division by even ratios.

Figure 57. Clock distribution for FMC and MCE4

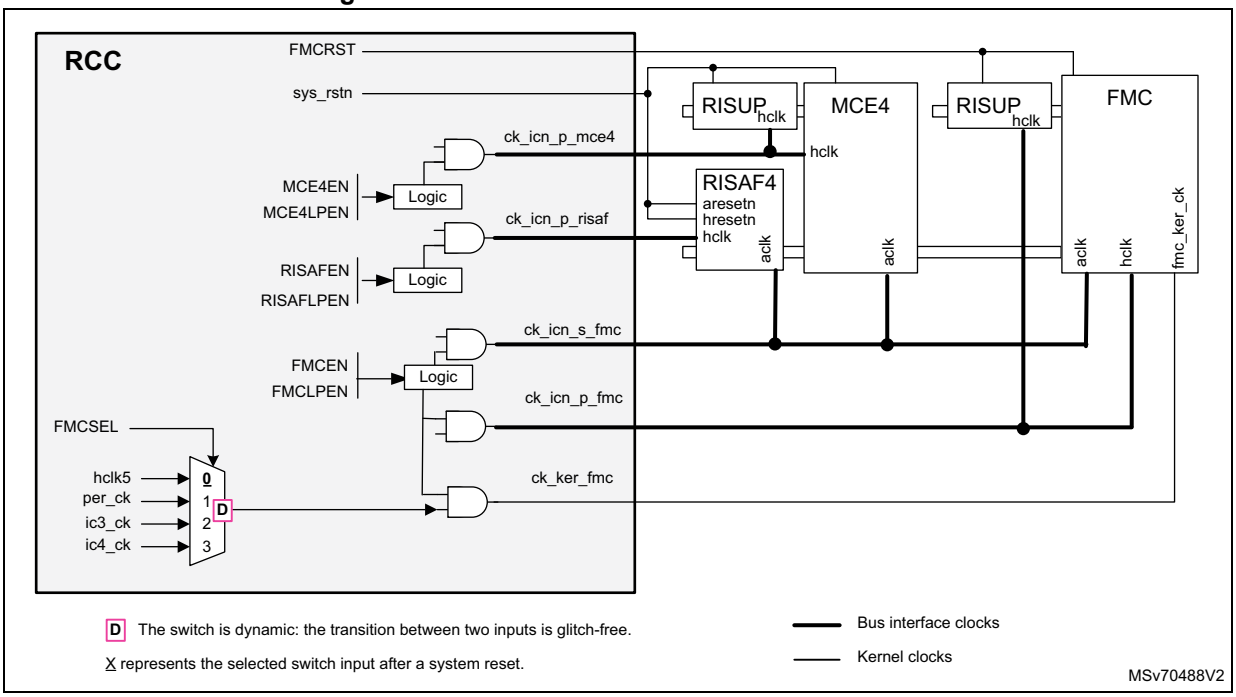
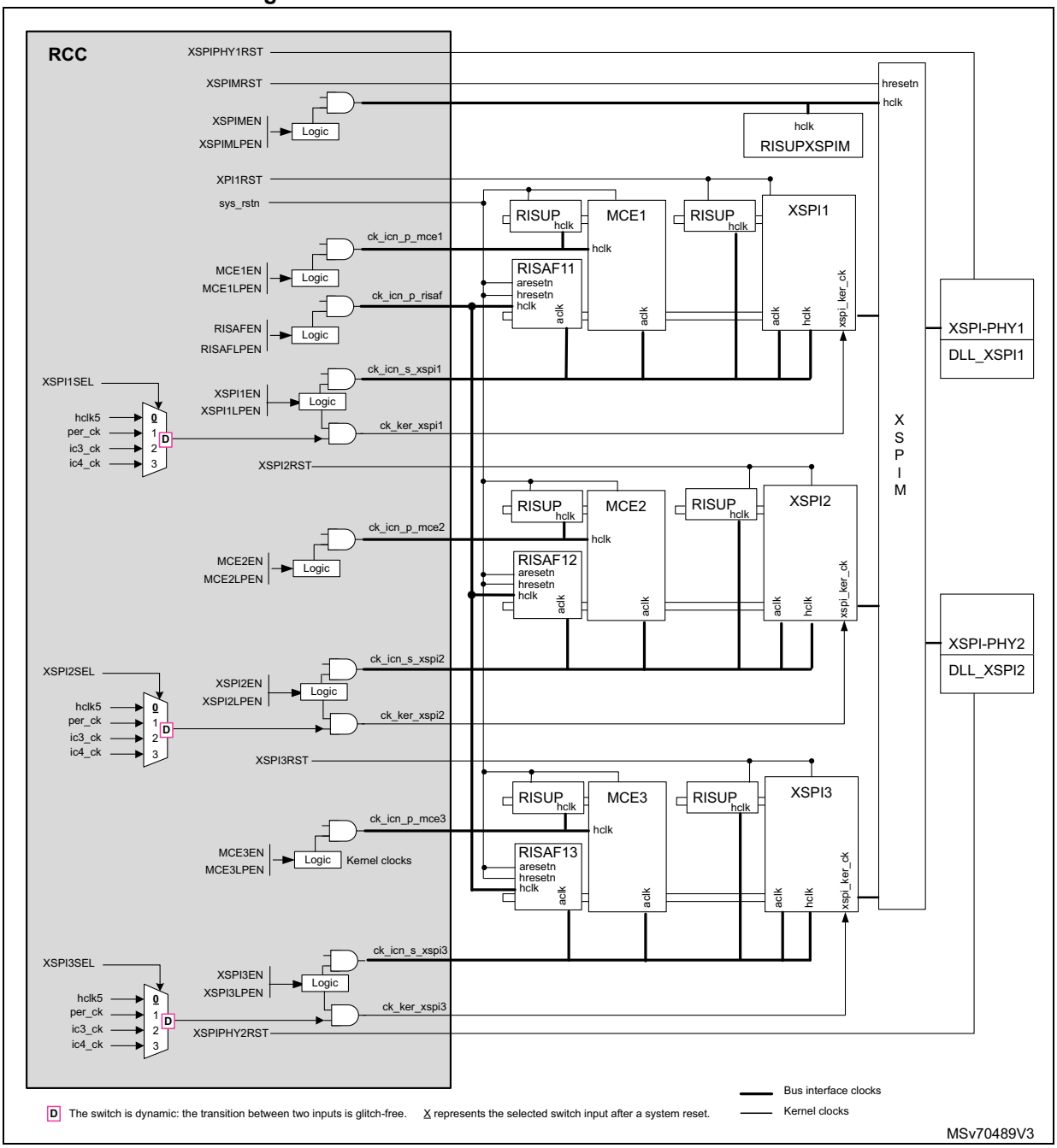


Figure 58. Clock distribution for XSPIs and MCE1/2/3



The SDMMC1 and SDMMC2 have separate kernel clocks. A clock switch allows the selection between four different sources. Each SDMMC can be enabled independently.

When an SDMMC is enabled via its SDMMCxEN bit, the associated SDMMC_SYSCONF is also enabled.

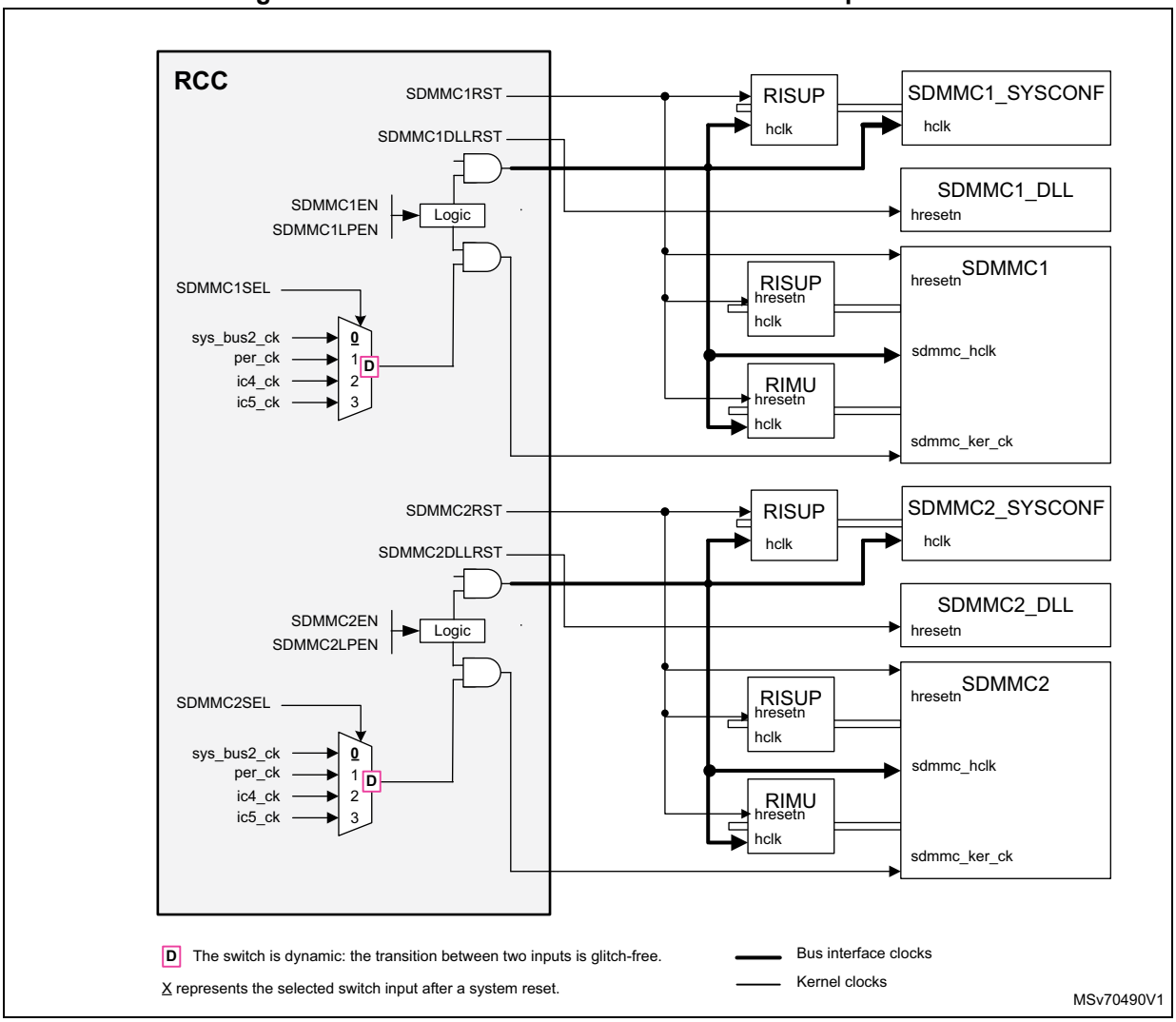
The application must configure the SDMMC to match the duty-cycle constraint of the interface clock.

Table 73. SDMMC interface clock constraints

SDMMC mode	Mode name	Interface clock frequency	Duty cycle constraint
SDIO	SDR12	25 MHz or less	30 - 70%
	SDR25	50 MHz or less	30 - 70%
	DDR50	50 MHz or less	45 - 55%
	SDR50	100 MHz or less	30 - 70%
e.MMC	Backward compatible	26 MHz or less	30 - 70%
	High-speed SDR	52 MHz or less	30 - 70%
	High-speed DDR	52 MHz or less	45 - 55%

For example, if the SDMMC works in SDR50, a kernel clock of 50 MHz, with a duty cycle better than 30-70% is enough. If the SDMMC works in DDR50, it is recommended to provide a kernel clock of 100 MHz, and to divide the frequency of the kernel clock by two, using the SDMMC divider, to ensure a duty-cycle very close to 50% for the SDMMC_CK.

Figure 59. Clock distribution for SDMMCx and companions



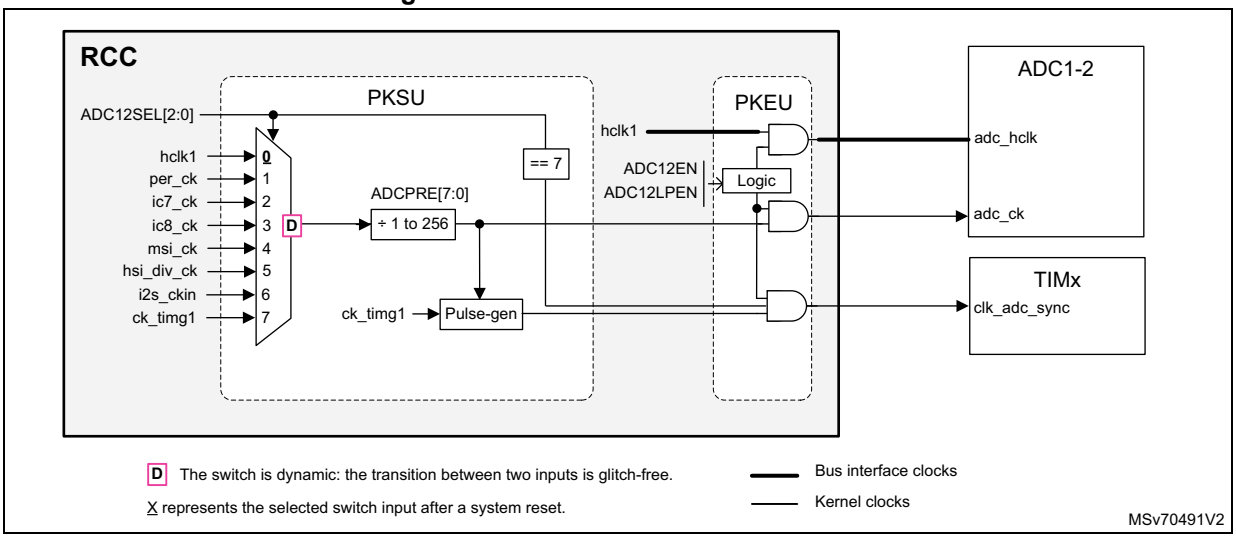
Clock distribution for ADC1/2

If the application requires that the ADC is precisely triggered by a TIMx timer without any uncertainty (fixed trigger latency), ck_timg1 must be selected as the kernel clock source. The other clock sources are asynchronous to TIMx, which results in an uncertain trigger instant due to the resynchronization between the two clock domains. The LPTIMx timers are also asynchronous.

The ADCPRE[7:0] divide value is set in RCC_CCIPR1.

clk_adc_sync is generated by Pulse-gen, one ck_timg1 cycle before the adc_ck rising edge.

Figure 60. Clock distribution for ADCs



Clock distribution for RTC/AWU clock

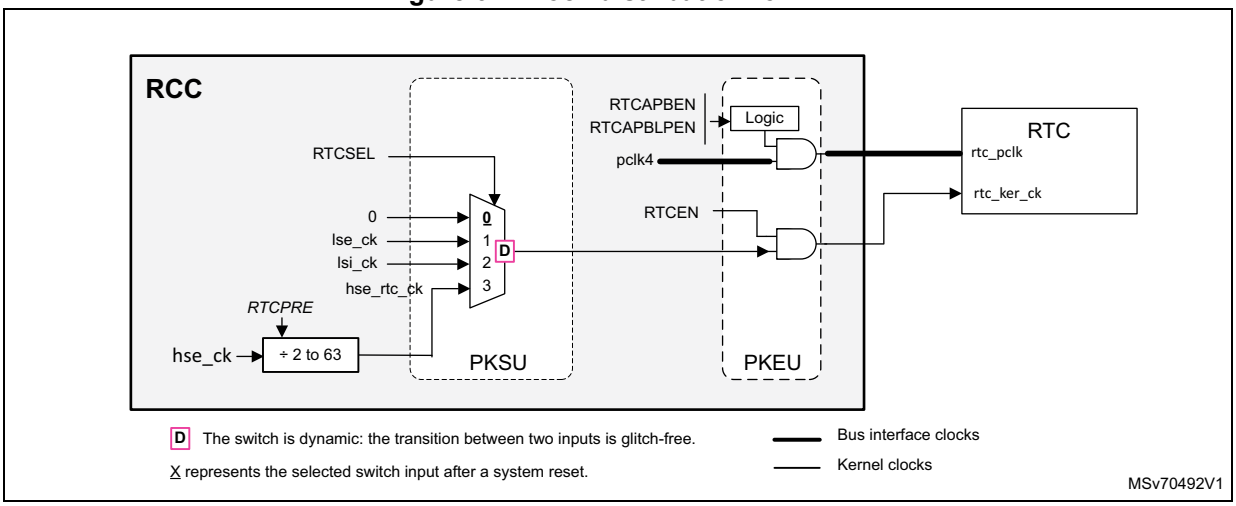
The rtc_ck clock source can be one of the following:

- hse_rtc_ck (hse_ker_ck divided by a programmable prescaler)
- lse_ck
- lsi_ck clock

The source clock is selected by programming RTCSEL and RTCPRE in RCC_CCIPR7. RTCSEL and RTCPRE are write-protected by DBP bit in PWR_DBPCR. In order to modify the bits, DBP must be set 1.

This selection cannot be modified without resetting the backup domain.

Figure 61. Clock distribution for RTC



If the LSE is selected as RTC clock, the RTC works normally even if the backup or the V_{DD} supply disappears.

The LSE clock is in the backup domain, whereas the other oscillators are not, with the following consequences:

- If LSE is selected as RTC clock, the RTC continues to work even if the V_{DD} supply is switched off, provided the V_{BAT} supply is maintained.
- If LSI is selected as the RTC clock, the AWU state is not guaranteed if the V_{DD} supply is powered off.
- If the HSE clock is used as RTC clock, the RTC state is not guaranteed if the V_{DD} supply is powered off, or if the $V_{VDDCORE}$ supply is powered off. In addition, the HSE is not available if the system goes to Stop mode.

rtc_ker_ck is enabled through RTCEN in RCC_APB4ENR.

The RTC bus interface clock (APB clock) is enabled through RTCAPBEN in RCC_APB4ENR, and RTCAPBLPEN in RCC_APB4LLPENR.

Note: To read the RTC calendar register when the APB clock frequency is less than seven times the RTC clock frequency ($F_{APB} < 7 \times F_{RTCLCK}$), the software must read the calendar time and date registers twice. Data are correct if the second read access to RTC_TR gives the same result of the first one. Otherwise, a third read access must be performed.

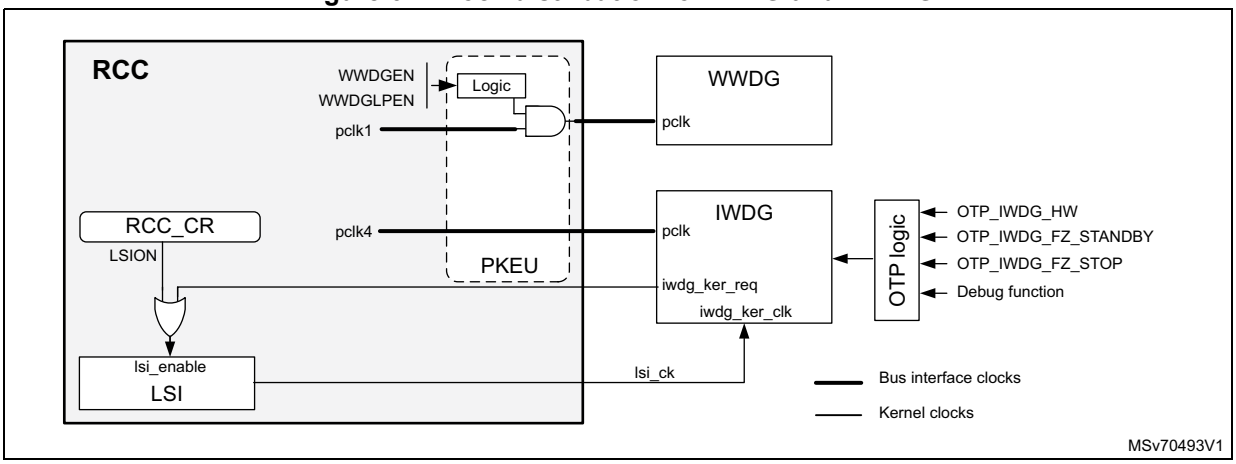
Clock distribution for watchdogs

The RCC provides the clock for the two watchdogs: the independent watchdog (IWDG), connected to the LSI, and the window watchdog (WWDG), connected to the APB clock.

If an IWDG is started by either hardware option or software access, the LSI is forced on, and cannot be disabled. After the LSI oscillator setup delay, the clock is provided to the IWDG.

The WWDG clock (pclk1) can be enabled by setting WWDGEN in RCC_APB1LENR. The software cannot stop WWDG down-counting by clearing WWDGEN to 0. The WWDG is frozen when the device goes to Stop mode.

Figure 62. Clock distribution for IWDG and WWDG



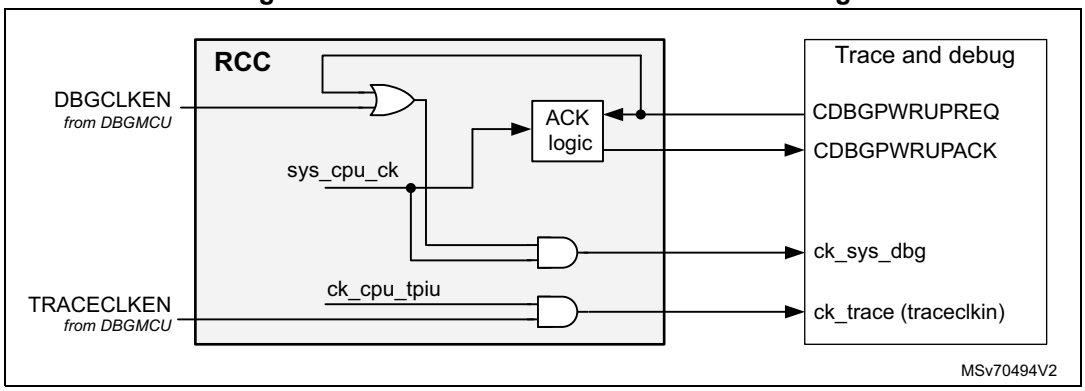
Clock distribution for trace and debug

The clock generation for the trace and debug is controlled by the DBGMCU.

DBGCLKEN in DBGMCU_CR allows the application to provide a clock to the debug components. This clock can also be enabled via the debug access port.

The trace clock generation is controlled via TRACECLKEN in DBGMCU_CR.

Figure 63. Clock distribution for trace and debug



14.6.9 General clock concept overview

The RCC handles the distribution of the CPU, bus interface, and peripheral clocks for the system, according to the CPU operating mode (refer to *Section 14.6.1* for details on clock definitions).

For each peripheral, the application can control the activation/deactivation of its kernel and bus interface clocks. Before using a peripheral, the CPU must enable it (by setting PERxEN to 1), and define if it remains active in Sleep mode (by setting PERxLPEN to 1). This is called allocation of a peripheral by the CPU (refer to *Section 14.6.10* for more details).

The peripheral allocation is used:

- by the RCC to automatically control the clock gating according to the CPU modes
- by the PWR to control V_{DDCORE} supply voltages

Memory handling

The CPU can access all memory areas available in the device:

- AXISRAM1 to AXISRAM6, CACHEAXI RAM, and FLEXRAM
- AHBSRAM1 and AHBSRAM2
- BKPSRAM

CACHEAXI RAM, and VENCRAM are disabled by default, see *RCC embedded memories enable register (RCC_MEMENR)*. The CPU must enable them before using these memories.

Read or write accesses to a peripheral register or memory, without the clocks enabled in the RCC registers, result in a system freeze. A system reset is required to unlock this.

If the access is performed by a debug interface (as an example, from a debug session inside an IDE), then a power-on reset is required to unlock the device, as the debug interface is not affected by a system reset.

Note: The memory interface clocks (flash memory and RAM interfaces) can be stopped by software during Sleep mode (via SRAMyLPEN bits).

Refer to *Section 14.6.11* and *Section 14.6.12* for details on clock enabling.

14.6.10 Peripheral allocation

The CPU can allocate a peripheral (and hence control its kernel and bus interface clock) by setting the dedicated PERxEN bit to 1. The CPU can control the peripheral clock gating when it is in Sleep mode via PERxLPEN bits.

The peripheral allocation bits (PERxEN bits) are used by the hardware to provide the kernel and bus interface clocks to the peripherals. Theses bits are also used to link peripherals to the CPU. The hardware can then safely gate peripheral clocks and bus matrix clocks, according to CPU states.

Clock switches and gating

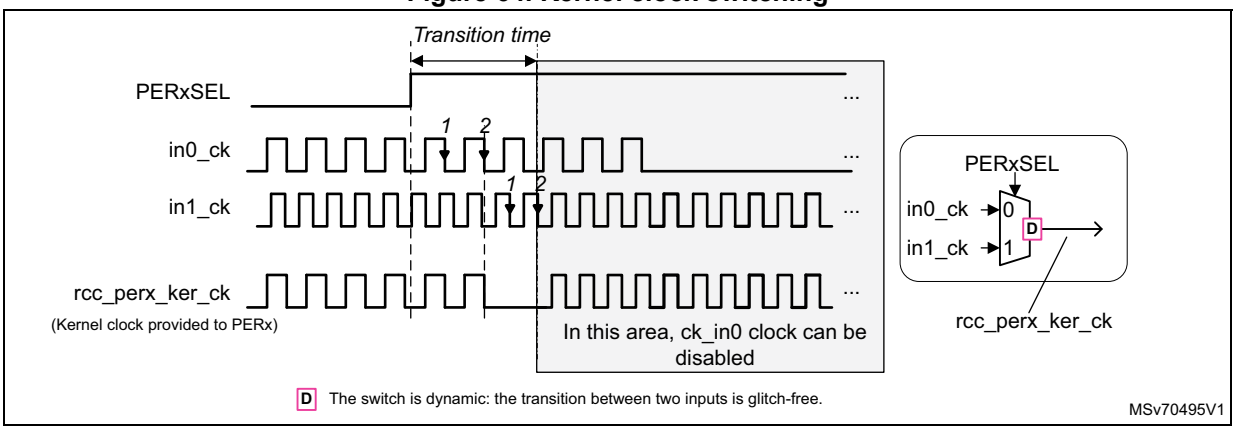
- Clock switching delays

  The input selected by the clock switches can be changed dynamically without generating spurs or timing violation. For example, if PERxSEL (in *Figure 64*) goes from 0 to 1, the switch first disables the clock output using the currently selected clock (in0_ck), and enables again the clock output using the new selected clock (in1_ck). Disable and enable commands are re-synchronized to their respective clocks. If one of the two clocks are not present, the sequence cannot be completed, and no clock is output. To recover from this situation, the user must either provide a valid clock to in1_ck input, or set back PERxSEL to 0.

  During the transition from one input to another, the kernel clock provided to the peripheral is gated, in the worst case, during two or three clock cycles of the new selected clock.

  As shown in *Figure 64*, both input clocks must be present during transition time.

Figure 64. Kernel clock switching



- Clock enabling delays

  In the same way, the clock gating logic synchronizes the enable command (coming generally from a kernel clock request, or PERxEN bits) with the selected clock, in order to avoid generation of spurious:

  – A maximum delay of two periods of the enabled clock can occur between the enable command and the first rising edge of the clock. The enable command can

be the rising edge of PERxEN in RCC_xxxxENR, or a kernel clock request asserted by a peripheral.

– A maximum delay of 1.5 periods of the disabled clock can occur between the disable command and the last falling edge of the clock. The disable command can be the falling edge of PERxEN in RCC_xxxxENR, or a kernel clock request released by a peripheral.

Note: Both kernel and bus interface clocks are affected by this re-synchronization delay.

14.6.11 Peripheral clock-gating control

As mentioned previously, each peripheral requires one or several bus interface clocks, named rcc_perx_bus_ck (for peripheral 'x'). These clocks can be an APB, AHB, or AXI clock, according to which bus(ses) the peripheral is connected.

The clocks used as bus interface for peripherals can be aclk[s,a,n], hclk[m,u,e], hclk[5:1], pclk[5:4], pclk[2:1], or ck_timg[1,2], depending on the bus connected to each peripheral.

Some peripherals also require dedicated clocks for their communication interface. These clocks are generally asynchronous with respect to the bus interface clock. They are named kernel clocks (perx_ker_ck). Both bus interface and kernel clocks can be gated according to several conditions detailed hereafter.

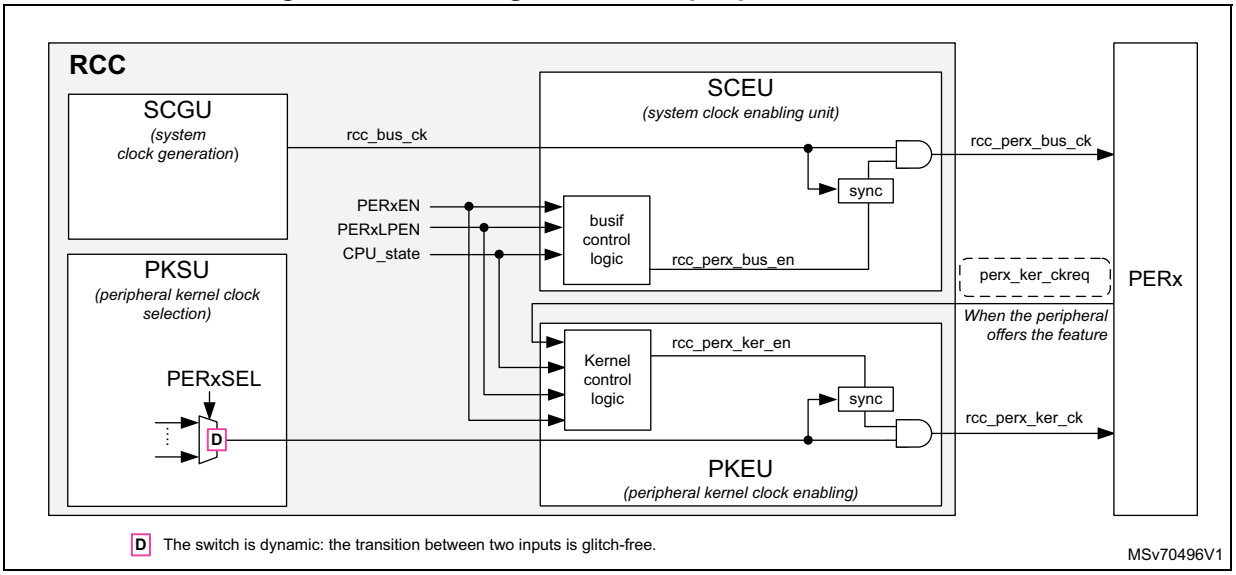
As shown in *Figure 65*, enabling kernel and bus interface clocks of each peripheral depends on several input signals:

- PERxEN and PERxLPEN bits

  PERxEN represents the peripheral enable (allocation) bit for the CPU. The CPU can write these bits to 1 via RCC_xxxxENR.
- CPU state (cpu_sleep and cpu_deepsleep signals)
- kernel clock request (perx_ker_ckreq) of the peripheral itself, when the feature is available

Refer to *Section 14.6.10* for more details.

Figure 65. Enable logic details for peripheral kernel clock



The clocks for all AHB and APB buses (AHBM, AHB1/2/3/4/5, APB1/2/4/5) are automatically enabled when a dependent peripheral is active. This may induce a chain: a peripheral activation activates the APB that activates the AHB, and activates the AHBM.

For instance, the manual UART4 activation induces the automatic APB1 activation (APB1 is used to configure the UART4). This induces the automatic AHB1 bus activation (AHB1 is needed to drive the APB1). This induces the activation of the central AHBM matrix (AHBM is needed to drive the AHB1).

High-bandwidth interconnect

The NoC and the two NPU AXI bus clocks (ck_icn_npu and ck_icn_npuc, see *Figure 48*) are permanently enabled in Run and Sleep modes, and permanently disabled in Stop and Standby modes.

An internal automatic clock-gating optimizes the NoC power consumption: when no transaction is ongoing in a bus section, an automatic clock-gating clocks and gates this bus.

The NPU interconnect clock (ck_icn_npu) can be disabled by setting ACLKNEN = 0 (disabled in Run and Sleep modes), or ACLKNLPEN = 0 (disabled in Sleep mode only). If the clock is disabled, the NPU cannot work (no interconnect downstream), and the CPU cannot access AXISRAM3/4/5/6, the CACHEAXI RAM, or the FLEXRAM.

The clock of the NPU interconnect (ck_icn_npuc) can be disabled by setting ACLKNCEN = 0 (disabled in Run and Sleep modes), or ACLKNCLPEN = 0 (disabled in Sleep mode only). If the clock is disabled, neither the CPU or NPU can access AXISRAM3/4/5/6, the CACHEAXI, the CACHEAXI RAM, or the FLEXRAM.

Table 74 gives a detailed description of the enabling logic of the peripheral clocks for peripherals located in the CPU domain and allocated by the CPU.

Table 74. Peripheral clock enabling

PERxEN	PERxLPEN	PERxSEL	perx_ker_ckreq	CPU state	rcc_perx_ker_ck	rcc_perx_bus_ck	Comments
0	X	X	X	X	-	-	No clock provided to the peripheral (PERxEN = 0)
1	X	X	X	Run	CK	CK	Kernel and bus interface clocks are provided to the peripheral (CPU in Run mode, and PERxEN = 1)
1	0	X	X	Sleep	-	-	No clock provided to the peripheral (CPU is in Sleep mode and PERxLPEN = 0)
1	1				CK	CK	Kernel and bus interface clocks are provided to the peripheral (CPU in Sleep mode and PERxLPEN = 1)

Table 74. Peripheral clock enabling (continued)

PERxEN	PERxLPEN	PERxSEL	perx_ker_ckreq	CPU state	rcc_perx_ker_ck	rcc_perx_bus_ck	Comments
1	0	X	X	Stop	-	-	No clock provided to the peripheral (PERxLPEN = 0)
1	1	no lsi_ck and no lse_ck and no hsi_ker_ck and no msi_ker_ck	X		-	-	No clock provided to the peripheral (CPU in Stop mode, and lse_ck or lsi_ck or hsi_ker_ck or msi_ker_ck not selected as kernel clock)
1	1	lsi_ck or lse_ck	X		CK	-	Kernel clock provided to the peripheral (PERxEN = PERxLPEN = 1, and lsi_ck or lse_ck selected and enabled) Bus interface clock not provided as the CPU is in Stop mode
1	1	hsi_ker_ck or msi_ker_ck	1		CK	-	Kernel clock provided to the peripheral (req_ker_perx = 1, PERxEN = PERxLPEN = 1, and hsi_ker_ck or msi_ker_ck selected and enabled) Bus interface clock not provided as the CPU is in Stop mode
1	1	hsi_ker_ck or msi_ker_ck	0		-	-	No clock provided to the peripheral (CPU in Stop mode, and no kernel clock request pending)

The kernel clock is provided to the peripherals when one of the following conditions is met:

1. The CPU is in Run mode and the peripheral is enabled.
2. The CPU is in Sleep mode and the peripheral is enabled with PERxLPEN = 1.
3. The CPU is in Stop mode, the peripheral is enabled with PERxLPEN = 1, the peripheral generates a kernel clock request, and the selected clock is hsi_ker_ck or msi_ker_ck.
4. The CPU is in Stop mode, the peripheral is enabled with PERxLPEN = 1, and the kernel source clock of the peripheral is lse_ck or lsi_ck.

The bus interface clock is provided to the peripherals only when conditions *1* or *2* are met.

14.6.12 CPU and bus matrix clock-gating control

The clocks of the CPU, AHB/AXI bridges, and APB buses are enabled as follows:

- The CPU clock (rcc_cpu_ck) is enabled when the CPU is in Run or Sleep mode.
- The AXI bridge clock is enabled when the CPU is in Run or Sleep mode.
- The AHB bus matrix clock is enabled if one of the following conditions is met:
  - The CPU is in Run mode.
  - The CPU is in Sleep mode, and at least one peripheral connected to this bridge has both its PERxEN and PERxLPEN set to 1.
  - The CPU is in Sleep mode, and AHB1, 2, 3, 4, or 5 has its clock enabled.
- The clocks of AHB1/2/3/4/5 bridges are enabled when one of the following conditions is met:

  - The CPU is in Run mode.
  - The CPU is in Sleep mode, and at least one peripheral connected to this bus has both its PERxEN and PERxLPEN set to 1.
  - The CPU is in Sleep mode, and the APB bus connected to the AHB bridge has its clock enabled.
- The APB1/2/4/5 clock buses are enabled when one of the following conditions is met:
  - The CPU is in Run mode.
  - The CPU is in Sleep mode, and at least one peripheral connected to this bus has both its PERxEN and PERxLPEN set to 1.

Refer to *Section 14.6.11* for details on the automatic clock-gating for AHBMEN, and all AHB/APB buses.

14.6.13 Low-power emulation modes

To ease the device debug, the RCC is able to handle an emulation mode for Stop and Standby modes.

Sleep emulation mode

The Sleep emulation mode is controlled by DBGSLEEP in DBGMCU_CR. When the processor goes to Sleep mode with DBGSLEEP = 1, the processor clock, the clocks of all enabled peripherals, debug parts, and interconnect are maintained activated.

Stop emulation mode

The Stop emulation mode is controlled by DBGSTOP in DBGMCU_CR. When the processor goes to Stop mode with DBGSTOP = 1:

- The processor, peripheral, and interconnect clocks remain active.
- The debug clock is active (ck_bus2_dbg, see *Figure 46*), and all debug parts remain clocked. All PLLs and OSCs remain active.

When a wake-up event occurs:

- The peripheral waking up the system sends the interrupt to the NVIC via the EXTI.
- If the PWR asserts the pwr_wkup signal, the RCC exits Stop mode. The RCC selects HSI or MSI as system clock, depending on STOPWUCK. The CPU exits Stop mode.

Standby emulation mode

The Standby emulation mode is controlled by DBGSTBY in DBGMCU_CR.

When the system goes to Standby mode with DBGSTBY = 1 and dbg_stdby_rstn = 0:

- the V_{DDCORE} voltage is not switched-off
- all CPU, interconnect, and peripherals are under reset except the debug part
- all PLLs and OSCs remain active
- the debug part is clocked
- peripherals on V_{SW} or V_{DD} domains are not reset

The system exits Standby mode when dbg_stdby_rstn is deasserted:

- the RCC deasserts the rcc_pwrds signal
- all peripherals are removed from reset
- the RCC enters Run state

14.7 RCC interrupts

The RCC provides the following interrupt lines:

- rcc_it: general interrupt line that provides events when PLLs or oscillators are ready (for nonsecure bits)
- rcc_s_it: general interrupt line that provides events when PLLs or oscillators are ready (for secure bits)
- rcc_hsecss_it: interrupt line dedicated to the failure detection of the HSE CSS
- rcc_lsecss_it: interrupt line dedicated to the failure detection of the LSE CSS

The interrupt enable is controlled via RCC_CIER, except for the HSE CSS failure. When the HSE CSS feature is enabled, the interrupt generation cannot be masked.

The interrupt flags can be checked via RCC_CIFR, and these flags can be cleared via RCC_CICR.

Note: The interrupt flags are not relevant if the corresponding interrupt enable bit is not set.

Table 75 gives a summary of the interrupt sources and the way to control them.

Table 75. Interrupt sources and control

Interrupt source	Description	Interrupt enable	Action to clear interrupt	Interrupt line
LSIRDYF	LSI ready	LSIRDYIE	Set LSIRDYC to 1	rcc_it, rcc_s_it
LSERDYF	LSE ready	LSERDYIE	Set LSERDYC to 1	
HSIDRYF	HSI ready	HSIDRYIE	Set HSIRDYC to 1	
HSERDYF	HSE ready	HSERDYIE	Set HSERDYC to 1	
MSIRDYF	MSI ready	MSIRDYIE	Set MSIRDYC to 1	
PLL1RDYF	PLL1 ready	PLL1RDYIE	Set PLL1RDYC to 1	
PLL2RDYF	PLL2 ready	PLL2RDYIE	Set PLL2RDYC to 1	
PLL3RDYF	PLL3 ready	PLL3RDYIE	Set PLL3RDYC to 1	
PLL4RDYF	PLL4 ready	PLL4RDYIE	Set PLL4RDYC to 1	
LSECSSF	LSE CSS failure	LSECSSIE (1)	Set LSECSSC to 1	rcc_lsecss_it
HSECSSF	HSE CSS failure	- (2)	Set HSECSSC to 1	rcc_hsecss_it

1. The security system feature must be enabled (LSECSSON = 1) to generate interrupts.
2. This interrupt cannot be masked when the security system feature is enabled (HSECSSON = 1).

14.8 RCC application information

14.8.1 HSE crystal auto-detection

The software can detect the frequency of the crystal connected to the HSE. The crystal choices are 19.2, 20.0, 24.0, 38.4, 40.0, 48.0 MHz. The closest crystal frequencies are 19.2 and 20 MHz (differing ~4%).

Measurement uses a timer clocked by a fast clock, triggered by the slower clock. Using the PWM input mode of input capture, it is possible to measure the full period of the slow clock.

LSE reference

If available, the LSE can trigger TIM16 to count hse_ck ticks. This is crystal accurate (better than using HSI).

Configure a PLL to generate (HSE * 8) MHz. Then set SYSSW to select ic2_ck for sysb_ck = ck_timg (TIMPRE = 1). This means sys_bus_ck relates to the HSE frequency.

LSE is connected to TIM16 - TI1_2.

Example: the multiplication by eight of hse_ck gives
(40 × 1000 × 8) / 32.768 = 9765 counts, and (38.4 × 1000 × 8) / 32.768 = 9375 counts.

19.2, 20, 24, 38.4, 40, 48 = 4687, 4882, 5859, 9375, 9765, 11718 counts

sys_bus_ck = 153.6, 160, 192, 307.2, 320, 384 MHz

HSI reference

HSI can be used when LSE is not available. A trimmed HSI can vary by ±4% (at 3 σ) across the full temperature range. This means that 19.2 and 20 MHz cannot be differentiated reliably.

If the DTS is available, the user can measure the temperature to compensate for the HSI drift.

The RCC generates an hse_cal_ck signal, which is HSE divided by 1024. This signal is connected to TIM17 - TI1_2.

hse_cal_ck has an expected frequency between 19.2 to 50 kHz. The HSI is about 64 MHz. To get 300 MHz, select HSI as input to a PLL, then ic2_ck for sysb_ck = ck_timg (TIMPRE = 1).

Example: The division by 1024 of hse_ck gives 300 × 1024 / 40 = 7680 counts, and 300 × 1024 / 38.4 = 8000 counts.

19.2, 20, 24, 38.4, 40, 48 = 16000, 15360, 12800, 8000, 7680, 6400 counts

A less accurate method is to use hse_ck output on MCO divided by 16. This gives, for example: 300 × 16 / 40 = 120 counts, and 300 × 16 / 38.4 = 125 counts.

19.2, 20, 24, 38.4, 40, 48 = 250, 240, 200, 125, 120, 100 counts

14.8.2 Calibration and clock frequency measurement using TIMx

Most of the clock source generator frequencies can be measured by means of the input capture of TIMx.

Calibrating HSI or MSI with the LSE

The main purpose of connecting the LSE to a TIMx input capture is to accurately measure the HSI or MSI. This requires to use the HSI or MSI as sys_bus_ck either directly, or via a PLL. The number of system clock counts between consecutive edges of the LSE signal gives a measurement of the internal clock period. Taking advantage of the high precision of LSE crystals (typically a few tens of ppm), the user can determine the internal clock frequency with the same resolution, and trim the source to compensate for variations due to manufacturing, temperature, or voltage.

The ratio between the two clock frequencies affects the measurement precision. The greater the ratio, the more accurate the calculation.

HSI and MSI oscillators have dedicated user-accessible calibration bits for this purpose (see RCC_HSICFGR and RCC_MSICFGR). When the HSI or MSI is used via a PLL, it is also possible to fine-tune the sys_bus_ck using the fractional divider of the PLL.

Calibrating LSI with HSI

The LSI frequency can also be measured. This is useful for applications that do not have a crystal. The ultra-low-power LSI oscillator has a large manufacturing process variation. The LSI clock frequency can be measured using the more precise HSI clock source. Using this measurement, the user can obtain a more accurate RTC timebase timeout (when LSI is used as RTC clock source), and/or a more accurate IWDG timeout.

14.8.3 Clock monitoring

Monitoring HSI with LSE

The purpose is to assist the software calibration procedure when the HSI frequency drifts out of a predefined range because of environmental variation (for example, temperature).

This monitoring can be enabled in all system modes where HSI is used, except Standby and V_{BAT} modes. When enabled, the number of HSI clock ticks between consecutive edges of the LSE clock is counted.

The HSI monitoring control is based on RCC_HSIMCR and RCC_HSIMSR.

The following steps can be used to enable the monitoring:

1. Enable LSE signal:
   a) Set DBP = 1 in PWR_DBPCR.
   b) Write 1 to LSEON.
   c) Wait for LSERDY = 1.
   d) Set DBP = 0 in PWR_DBPCR.
2. Set up HSI clock period monitoring:
   a) Write 1 to HSIMONEN in RCC_HSIMCR to start the monitoring.

14.8.4 Clock frequency limits

The maximum frequencies that can be set for each peripheral are detailed in *Table 76*.

Table 76. Maximum peripheral clock frequencies

Peripheral	Maximum kernel clock frequency (MHz)
PWR	200
RNG	64
RTC	4
SAES	200
SAI1	200
SAI2	200
SDMMC1	208
SDMMC2	208

Table 76. Maximum peripheral clock frequencies (continued)

Peripheral	Maximum kernel clock frequency (MHz)
SPDIFRX1	200
SPI1	200
SPI2	200
SPI3	200
SPI4	133
SPI5	133
SPI6	200
SYSCFG	200
TIMx	400
LPTIMx	200
DTS	10
USART1	100
USART2	100
USART3	100
UART4	100
UART5	100
USART6	100
UART7	100
UART8	100
UART9	100
USART10	100
LPUART1	100
UCPD1	25
VENC	400
VREFBUF	200
WWDG	200
MREPAIR	64

As an example, the PLL configuration to achieve these frequencies would use HSI as reference clock (64 MHz), and program the VCO of PLL1, 2, 3, and 4, to respectively, 800, 993.52, 875, and 514.5 MHz.

14.9 RCC security

The system RIF protects bus accesses to the RCC and peripheral registers.

The RIFSC indicates if an access to the RCC or other peripherals is secure and/or privileged. Signals are connected from the RIFSC to the RCC to communicate this information. The notation for these signals is S, P.

The RCC is able to protect register bits from being modified by nonsecure and unprivileged accesses.

If a peripheral RISUP is programmed as secure (or privileged), the peripheral clock and reset bits become secure (or privileged).

If the peripheral is TrustZone-aware, the peripheral clock and reset bits become secure (or privileged) as soon as at least one function is configured as secure (or privileged) by RIFSC.

Peripheral configuration registers inside the RCC can be also be made secure (or privileged) via a global override bit (PERSEC in RCC_SECCFGR3, PERPRIV in RCC_PRIVCFGR3).

After an application reset or system reset, the RCC does not filter any access until the trusted agent has configured the system.

14.9.1 Internal register protection

The following can be made secure and/or privileged (via RCC_SECCFGRx and RCC_PRIVCFGRx):

- internal and external oscillators (HSE, LSE, HSI, MSI, LSI)
- PLLs and AHB prescalers
- system clock-source selection
- MCO clock outputs
- reset flags
- automatic internal oscillator wake-up configuration

There are four access controls for RCC registers:

- SEC (secure)
- PRIV (privileged)
- LOCK (locked SEC and PRIV)
- PUB (public)

SEC (in RCC_SECCFGRx)

xxxSEC defines the secure status required for a write to the xxx configuration registers (for example, HSISEC for the HSI oscillator). When this bit is set, configuration register bits are writable by secure software only. This bit can be locked (cannot be changed) with the xxxLOCK bit, and is readable by all. Write access to RCC_SECCFGRx is controlled by S, P.

PRIV (in RCC_PRIVCFGRx)

xxxPRIV defines the privileged status required for a write to the xxx configuration registers (such as HSIPRIV for the HSI oscillator). When this bit is set, the configuration register bits are writable only by privileged software. This bit can be locked (cannot be changed) with the xxxLOCK bit, and is readable by all. Write access to RCC_PRIVCFGRx is controlled by xxxSEC, P. xxxPRIV can be set/cleared by write to 1 registers.

LOCK (in RCC_LOCKCFGRx)

xxxLOCK, when set, locks definitively xxxSEC and xxxPRIV settings: xxxLOCK is readable by all. Write access to RCC_LOCKCFGRx is controlled by S, P. xxxLOCK is a set-once bit.

PUB (in RCC_PUBCFGRx)

xxxPUB grants read access to configuration/status bits, regardless of security or privilege.

For example, if HSIPUB bit is set, any software can read the HSI oscillator configuration and status. If not set then normal access controls are enforced.

xxxPUB is readable by all. Write access to RCC_PUBCFGRx is controlled by xxxSEC and xxxPRIV. xxxPUB can be set/cleared by write-to-1 registers.

14.9.2 Internal register write-protection

There are different controls for write-protecting a register bit.

- xxxLOCK bits (described above)
- other internal protection logic (controlled from internal or external signals)

Only protection logic using the pwr_lock_backup signal is implemented.

14.10 RCC registers

14.10.1 RCC control register (RCC_CR)

Address offset: 0x0

Reset value: 0x0000 0008

This register is used to enable the RCC oscillators and PLLs in Run or Sleep mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	PLL4 ON	PLL3 ON	PLL2 ON	PLL1 ON	Res.	Res.	Res.	HSE ON	HSI ON	MSI ON	LSE ON	LSI ON
				rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **PLL4ON**: PLL4 enable

This bit is reset by int_sys_rstn. It is security-protected by PLL4SEC or PLL4PRIV, and is publicly readable if PLL4PUB = 1. It can be set with PLL4ONS, and cleared with PLL4ONC. This bit is set and reset by software. It cannot be cleared if PLL4 is currently used to generate the CPU or system clock.

0: PLL4 is OFF (default after reset)

1: PLL4 is ON

Bit 10 **PLL3ON**: PLL3 enable

This bit is reset by int_sys_rstn. It is security-protected by PLL3SEC or PLL3PRIV, and is publicly readable if PLL3PUB = 1. It can be set with PLL3ONS, and cleared with PLL3ONC. This bit is set and reset by software. It cannot be cleared if PLL3 is currently used to generate the CPU or system clock.

0: PLL3 is OFF (default after reset)

1: PLL3 is ON

Bit 9 **PLL2ON**: PLL2 enable

This bit is reset by int_sys_rstn. It is security-protected by PLL2SEC or PLL2PRIV, and is publicly readable if PLL2PUB = 1. It can be set with PLL2ONS, and cleared with PLL2ONC. This bit is set and reset by software. It cannot be cleared if PLL2 is currently used to generate the CPU or system clock.

0: PLL2 is OFF (default after reset)

1: PLL2 is ON

Bit 8 **PLL1ON**: PLL1 enable

This bit is reset by int_sys_rstn. It is security-protected by PLL1SEC or PLL1PRIV, and is publicly readable if PLL1PUB = 1. It can be set with PLL1ONS, and cleared with PLL1ONC. This bit is set and reset by software. It cannot be cleared if PLL1 is currently used to generate the CPU or system clock.

0: PLL1 is OFF (default after reset)

1: PLL1 is ON

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **HSEON**: HSE oscillator enable

This bit is reset by int_sys_rstn. It is security-protected by HSESEC or HSEPRIV, and is publicly readable if HSEPUB = 1. It can be set with HSEONS, and cleared with HSEONC. This bit is set and reset by software.

0: HSE is OFF (default after reset)

1: HSE is ON

Bit 3 **HSION**: HSI oscillator enable

This bit is reset by int_sys_rstn. It is security-protected by HSISEC or HSIPRIV, and is publicly readable if HSIPUB = 1. It can be set with HSIONS, and cleared with HSIONC. This bit is set and reset by software.

0: HSI is OFF

1: HSI is ON (default after reset)

Bit 2 **MSION**: MSI oscillator enable

This bit is reset by int_sys_rstn. It is security-protected by MSISEC or MSIPRIV, and is publicly readable if MSIPUB = 1. It can be set with MSIONS, and cleared with MSIONC. This bit is set and reset by software.

0: MSI is OFF (default after reset)

1: MSI is ON

Bit 1 **LSEON**: LSE oscillator enable

This bit is reset by rcc_vsw_rstn. It is in the V_{BKP} voltage domain. It is write-protected by the pwr_lock_backup_n signal. It is security-protected by LSESEC, LSEPRIV, and is publicly readable if LSEPUB = 1. It can be set with LSEONS, and cleared with LSEONC. This bit is set and reset by software.

0: LSE is OFF (default after reset)

1: LSE is ON

Bit 0 **LSION**: LSI oscillator enable

This bit is reset by nreset_rstn. It is in the V_{RET} voltage domain. This bit is security-protected by LSISEC or LSIPRIV, and is publicly readable if LSIPUB = 1. It can be set with LSIONS, and cleared with LSIONC. This bit is set and reset by software.

0: LSI is OFF (default after reset)

1: LSI is ON

14.10.2 RCC status register (RCC_SR)

Address offset: 0x4

Reset value: 0x0000 0000

This register is used to retrieve the status the RCC oscillators and PLLs.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	PLL4 RDY	PLL3 RDY	PLL2 RDY	PLL1 RDY	Res.	Res.	Res.	HSE RDY	HSI RDY	MSI RDY	LSE RDY	LSI RDY
				r	r	r	r				r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **PLL4RDY**: PLL4 clock ready flag

This bit is security-protected by PLL4SEC or PLL4PRIV, and is publicly readable if PLL4PUB = 1. It is set by hardware to indicate that the PLL4 is locked.

0: PLL4 unlocked (default after reset)

1: PLL4 locked

Bit 10 **PLL3RDY**: PLL3 clock ready flag

This bit is security-protected by PLL3SEC or PLL3PRIV, and is publicly readable if PLL3PUB = 1. It is set by hardware to indicate that the PLL3 is locked.

0: PLL3 unlocked (default after reset)

1: PLL3 locked

Bit 9 **PLL2RDY**: PLL2 clock ready flag

This bit is security-protected by PLL2SEC or PLL2PRIV, and is publicly readable if PLL2PUB = 1. it is set by hardware to indicate that the PLL2 is locked.

0: PLL2 unlocked (default after reset)

1: PLL2 locked

Bit 8 **PLL1RDY**: PLL1 clock ready flag

This bit is security-protected by PLL1SEC or PLL1PRIV, and is publicly readable if PLL1PUB = 1. It is set by hardware to indicate that the PLL1 is locked.

0: PLL1 unlocked (default after reset)

1: PLL1 locked

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **HSERDY**: HSE clock ready flag

This bit is security-protected by HSESEC or HSEPRIV, and is publicly readable if HSEPUB = 1.

It is set by hardware to indicate that the HSE oscillator is stable.

0: HSE not ready (default after reset)

1: HSE ready

Bit 3 **HSIRDY**: HSI clock ready flag

This bit is security-protected by HSISEC or HSIPRIV, and is publicly readable if HSIPUB = 1. It is set by hardware to indicate that the HSI oscillator is stable.

0: HSI not ready (default after reset)

1: HSI ready

Bit 2 **MSIRDY**: MSI clock ready flag

This bit is security-protected by MSISEC or MSIPRIV, and is publicly readable if MSIPUB = 1. It is set and reset by hardware to indicate that the MSI oscillator is stable.

0: MSI not ready (default after reset)

1: MSI ready

Bit 1 **LSERDY**: LSE clock ready flag

This bit is security-protected by LSESEC or LSEPRIV, and is publicly readable if LSEPUB = 1. It is set and reset by hardware to indicate that the LSE oscillator is stable. This bit requires 6 cycles of lse_ck before it is deasserted.

0: LSE not ready (default after reset)

1: LSE ready

Bit 0 **LSIRDY**: LSI clock ready flag

This bit is security-protected by LSISEC or LSIPRIV, and is publicly readable if LSIPUB = 1. It is set by hardware to indicate that the LSI oscillator is stable.

0: LSI not ready (default after reset)

1: LSI ready

14.10.3 RCC Stop mode control register (RCC_STOPCR)

Address offset: 0x8

Reset value: 0x0000 0002

This register is used to enable the RCC oscillators in Stop mode.

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

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

Bit 1 **HSISTOPEN**: HSI oscillator enable

This bit is reset by int_sys_rstn. It is security-protected by HSISEC or HSIPRIV, and is publicly readable if HSIPUB = 1. It can be set with HSISTOPENS, and cleared with HSISTOPENC. This bit is set and reset by software.

0: HSI is OFF

1: HSI is ON (default after reset)

Bit 0 **MSISTOPEN**: MSI oscillator enable

This bit is reset by int_sys_rstn. It is security-protected by MSISEC or MSIPRIV, and is publicly readable if MSIPUB = 1. It can be set with MSISTOPENS, and cleared with MSISTOPENC. This bit is set and reset by software.

0: MSI is OFF (default after reset)

1: MSI is ON

14.10.4 RCC configuration register 1 (RCC_CFGR1)

Address offset: 0x20

Reset value: 0x0000 0000

This register controls the selection of the CPU and system clocks, and their status (see *Figure 46* for various SYSSW inputs). The register is reset by sys_rstn, and is in the V_{CORE} voltage domain.

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

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **SYSSWS[1:0]**: System clock switch status

This bitfield is security-protected by SYSSEC, SYSPRIV, and is publicly readable if SYSPUB = 1. It is set and reset by hardware to show the source of the system bus clocks (sys_bus_ck).

00: hsi_ck selected as sysb_ck, sysc_ck, sysd_ck system clocks (default after reset)

01: msi_ck selected as sysb_ck, sysc_ck, sysd_ck system clocks

10: hse_ck selected as sysb_ck, sysc_ck, sysd_ck system clocks

11: ic2_ck selected as sysb_ck, ic6_ck selected as sysc_ck, ic11_ck as sysd_ck system clocks

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:24 **SYSSW[1:0]**: System clock switch selection

This bitfield is security-protected by SYSSEC, SYSPRIV, and is publicly readable if SYSPUB = 1. It is set by the software to select the source of the system bus clocks (sys_bus_ck).

00: hsi_ck selected as system clock (default after reset)

01: msi_ck selected as system clock

10: hse_ck selected as system clock

11: ic2_ck selected as system clock

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **CPUSWS[1:0]**: CPU clock switch status
This bitfield is security-protected by SYSSEC, SYSPRIV, and is publicly readable if SYSPUB = 1. It is set and reset by hardware to show the source of the CPU clock (sys_cpu_ck).
00: hsi_ck selected as system clock (default after reset)
01: msi_ck selected as system clock
10: hse_ck selected as system clock
11: ic1_ck selected as system clock

Bits 19:18 Reserved, must be kept at reset value.

Bits 17:16 **CPUSW[1:0]**: CPU clock switch selection
This bitfield is security-protected by SYSSEC, SYSPRIV, and is publicly readable if SYSPUB = 1. It is set by the software to select the source of the CPU clock (sys_cpu_ck).
00: hsi_ck selected as system clock (default after reset)
01: msi_ck selected as system clock
10: hse_ck selected as system clock
11: ic1_ck selected as system clock

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **STOPWUCK**: System clock selection after a wake up from system stop
This bit is security-protected by SYSSEC, SYSPRIV, and is publicly readable if SYSPUB = 1. It is set and reset by software to select the system wake-up clock from system stop.
0: HSI selected as wake-up clock from system stop (default after reset)
1: MSI selected as wake-up clock from system stop

14.10.5 RCC configuration register 2 (RCC_CFGR2)

Address offset: 0x24

Reset value: 0x0010 0000

This register controls the division factors of the central clocks: AHB, APB, and timer. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	TIMPRE[1:0]		Res.	HPRE[2:0]			Res.	PPRE5[2:0]		
						rw	rw		rw	rw	rw		rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PPRE4[2:0]			Res.	Res.	Res.	Res.	Res.	PPRE2[2:0]			Res.	PPRE1[2:0]		
	rw	rw	rw						rw	rw	rw		rw	rw	rw

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

Bits 25:24 **TIMPRE[1:0]**: Timer clock prescaler selection
This bitfield is security-protected by a SEC signal from RIFSC, the SYSSEC bit, a PRIV signal from RIFSC, or the SYSPRIV bit, and is publicly readable if SYSPUB = 1. It is set and reset by software to control the clock frequency of all the timers connected to APB1 and APB2 domains.
00: timg_ck = sys_bus_ck (default after reset)
01: timg_ck = sys_bus_ck / 2
10: timg_ck = sys_bus_ck / 4
11: timg_ck = sys_bus_ck / 8

Bit 23 Reserved, must be kept at reset value.

Bits 22:20 **HPRE[2:0]**: AHB clock prescaler

This bitfield is security-protected by SYSSEC or SYSPRIV, and is publicly readable if SYSPUB = 1. It is set and reset by software to control the division factor of the clocks between the system AXI bus and the configuration AHB bus. The AXI sys_bus_ck source clock is divided in function of HPRE, to generate the AHB clock.

The division ratio is as follows:

000: sys_bus2_ck= sys_bus_ck

001: sys_bus2_ck = sys_bus_ck / 2 (default after reset)

010: sys_bus2_ck= sys_bus_ck / 4

011: sys_bus2_ck = sys_bus_ck / 8

100: sys_bus2_ck = sys_bus_ck / 16

101: sys_bus2_ck = sys_bus_ck / 32

110: sys_bus2_ck = sys_bus_ck / 64

111: sys_bus2_ck = sys_bus_ck / 128

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **PPRE5[2:0]**: CPU domain APB5 prescaler

This bitfield is security-protected by SYSSEC or SYSPRIV, and is publicly readable if SYSPUB = 1. It is set and reset by software to control the division factor of rcc_pclk5.

000: rcc_pclk5 = sys_bus2_ck (default after reset)

001: rcc_pclk5 = sys_bus2_ck / 2

010: rcc_pclk5 = sys_bus2_ck / 4

011: rcc_pclk5 = sys_bus2_ck / 8

100: rcc_pclk5 = sys_bus2_ck / 16

101: rcc_pclk5 = sys_bus2_ck / 32

110: rcc_pclk5 = sys_bus2_ck / 64

111: rcc_pclk5 = sys_bus2_ck / 128

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **PPRE4[2:0]**: CPU domain APB4 prescaler

This bitfield is security-protected by SYSSEC or SYSPRIV, and is publicly readable if SYSPUB = 1. It is set and reset by software to control the division factor of rcc_pclk4.

000: rcc_pclk4 = sys_bus2_ck (default after reset)

001: rcc_pclk4 = sys_bus2_ck / 2

010: rcc_pclk4 = sys_bus2_ck / 4

011: rcc_pclk4 = sys_bus2_ck / 8

100: rcc_pclk4 = sys_bus2_ck / 16

101: rcc_pclk4 = sys_bus2_ck / 32

110: rcc_pclk4 = sys_bus2_ck / 64

111: rcc_pclk4 = sys_bus2_ck / 128

Bits 11:7 Reserved, must be kept at reset value.

Bits 6:4 **PPRE2[2:0]**: CPU domain APB2 prescaler
This bitfield is security-protected by SYSSEC or SYSPRIV, and is publicly readable if SYSPUB = 1. It is set and reset by software to control the division factor of rcc_pclk2.
000: rcc_pclk2 = sys_bus2_ck (default after reset)
001: rcc_pclk2 = sys_bus2_ck / 2
010: rcc_pclk2 = sys_bus2_ck / 4
011: rcc_pclk2 = sys_bus2_ck / 8
100: rcc_pclk2 = sys_bus2_ck / 16
101: rcc_pclk2 = sys_bus2_ck / 32
110: rcc_pclk2 = sys_bus2_ck / 64
111: rcc_pclk2 = sys_bus2_ck / 128

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **PPRE1[2:0]**: CPU domain APB1 prescaler
This bitfield is security-protected by SYSSEC or SYSPRIV, and is publicly readable if SYSPUB = 1. Is it set and reset by software to control the division factor of rcc_pclk1.
000: rcc_pclk1 = sys_bus2_ck (default after reset)
001: rcc_pclk1 = sys_bus2_ck / 2
010: rcc_pclk1 = sys_bus2_ck / 4
011: rcc_pclk1 = sys_bus2_ck / 8
100: rcc_pclk1 = sys_bus2_ck / 16
101: rcc_pclk1 = sys_bus2_ck / 32
110: rcc_pclk1 = sys_bus2_ck / 64
111: rcc_pclk1 = sys_bus2_ck / 128

14.10.6 RCC backup domain protection register (RCC_BDCR)

Address offset: 0x2C

Reset value: 0x0000 0000

This register controls the reset of the backup domain. It is reset by pwr_vsw_rstn, and is in the V_{BKP} voltage domain.

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

Bit 31 **VSWRST**: V_{switch} (V_{SW}) domain software reset.
This bit is write-protected by the pwr_lock_backup_n signal. It is security-protected by a SEC signal from RIFSC, the RSTSEC bit, a PRIV signal from RIFSC, or the RSTPRIV bit, and is publicly readable if RSTPUB = 1. Writing 1 to this VSWRST bit by software generates a pulse that resets the V_{SW} domain.
0: V_{SW} domain not reset (default after reset)
1: V_{SW} domain reset

Bits 30:0 Reserved, must be kept at reset value.

14.10.7 RCC reset status register for hardware (RCC_HWRSR)

Address offset: 0x30

Reset value: 0x00A0 0000

This register is used to monitor the resets that occurred. It is reset by pwr_por_rstn, and is in the V_{BKP} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	LPWRRSTF	Res.	WWDGRSTF	Res.	IWDGRSTF	Res.	SFTRSTF	PORRSTF	PINRSTF	BORRSTF	Res.	Res.	Res.	LCKRSTF	RMVF
	r		r		r		r	r	r	r				r	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bit 31 Reserved, must be kept at reset value.

Bit 30 **LPWRRSTF**: Illegal Stop or Standby flag

This bit is set by hardware when the CPU goes erroneously in Stop or Standby mode. It is reset by software by writing HWRMVF.

0: No illegal reset occurred (default after power-on reset)

1: Illegal Stop or Standby reset occurred

Bit 29 Reserved, must be kept at reset value.

Bit 28 **WWDGRSTF**: Window watchdog reset flag

This bit is set by hardware when a window watchdog reset occurs. it is reset by software by writing HWRMVF.

0: No window watchdog reset occurred from WWDG (default after power-on reset)

1: Window watchdog reset occurred from WWDG

Bit 27 Reserved, must be kept at reset value.

Bit 26 **IWDGRSTF**: Independent watchdog reset flag

This bit is set by hardware when an independent watchdog reset occurs. It is reset by software by writing HWRMVF.

0: No independent watchdog reset occurred (default after power-on reset)

1: Independent watchdog reset occurred

Bit 25 Reserved, must be kept at reset value.

Bit 24 **SFTRSTF**: Software system reset flag

This bit is set by hardware when the software system reset is due to the CPU.The CPU can generate a software system reset by writing SYSRESETREQ in AIRCR register of the CPU. This bit is reset by software by writing HWRMVF.

0: No software system reset occurred (default after power-on reset)

1: A software system reset has been generated by the CPU.

Bit 23 **PORRSTF**: POR/PDR reset flag

This bit is set by hardware when a POR/PDR occurs. it is reset by software by writing HWRMVF.

0: No POR/PDR occurred

1: POR/PDR occurred (default after power-on reset)

Bit 22 **PINRSTF**: Pin reset flag (NRST)

This bit is set by hardware when a reset from pin occurs. it is reset by software by writing HWRMVF.

0: No reset from pin occurred

1: Reset from pin occurred (default after power-on reset)

Bit 21 **BORRSTF**: BOR reset flag

This bit is set by hardware when a BOR occurs (pwr_bor_rst). it is reset by software by writing HWRMVF.

0: No BOR occurred

1: BOR occurred (default after power-on reset)

Bits 20:18 Reserved, must be kept at reset value.

Bit 17 **LCKRSTF**: CPU lockup reset flag

This bit is set by hardware when a reset from a CPU lockup occurs. Is it reset by software by writing RMVF.

0: No reset from CPU lockup occurred

1: Reset from CPU lockup occurred

Bit 16 **RMVF**: Remove reset flag

This bit is write-protected by the security bit. It is security-protected by a SEC signal from RIFSC, the RSTSEC bit, a PRIV signal from RIFSC, or the RSTPRIV bit, and is publicly readable if RSTPUB = 1. This bit is written by software to clear the value of the reset flags in this register.

0: Clear of the reset flags not activated (default after power-on reset)

1: Clear the value of the reset flags

Bits 15:0 Reserved, must be kept at reset value.

14.10.8 RCC reset register (RCC_RSR)

Address offset: 0x34

Reset value: 0x00A0 0000

This register is used to monitor the resets that occurred. It is reset by pwr_por_rstn, and is in the V_{BKP} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	LPWRRSTF	Res.	WWDGRSTF	Res.	IWDGRSTF	Res.	SFTRSTF	PORRSTF	PINRSTF	BORRSTF	Res.	Res.	Res.	LCKRSTF	RMVF
	r		r		r		r	r	r	r				r	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bit 31 Reserved, must be kept at reset value.

Bit 30 **LPWRRSTF**: Illegal Stop or Standby flag

This bit is set by hardware when the CPU goes erroneously in Stop or Standby mode. It is reset by software by writing RMVF.

0: No illegal reset occurred (default after power-on reset)

1: Illegal Stop or Standby reset occurred

Bit 29 Reserved, must be kept at reset value.

Bit 28 **WWDGRSTF**: Window watchdog reset flag

This bit is set by hardware when a window watchdog reset occurs. It is reset by software by writing RMVF.

0: No window watchdog reset occurred from WWDG (default after power-on reset)

1: Window watchdog reset occurred from WWDG

Bit 27 Reserved, must be kept at reset value.

Bit 26 **IWDGRSTF**: Independent watchdog reset flag

This bit is set by hardware when an independent watchdog reset occurs. It is reset by software by writing RMVF.

0: No independent watchdog reset occurred (default after power-on reset)

1: Independent watchdog reset occurred

Bit 25 Reserved, must be kept at reset value.

Bit 24 **SFTRSTF**: Software system reset flag

This bit is set by hardware when the software system reset is due to the CPU.The CPU can generate a software system reset by writing SYSRESETREQ in AIRCR register of the CPU. This bit is reset by software by writing RMVF.

0: No software system reset occurred (default after power-on reset)

1: A software system reset has been generated by the CPU.

Bit 23 **PORRSTF**: POR/PDR reset flag

This bit is set by hardware when a POR/PDR occurs. It is reset by software by writing RMVF.

0: No POR/PDR occurred

1: POR/PDR occurred (default after power-on reset)

Bit 22 **PINRSTF**: Pin reset flag (NRST)

This bit is set by hardware when a reset from pin occurs. It is reset by software by writing RMVF.

0: No reset from pin occurred

1: Reset from pin occurred (default after power-on reset)

Bit 21 **BORRSTF**: BOR reset flag

This bit is set by hardware when a BOR occurs (pwr_bor_rst). It is reset by software by writing RMVF.

0: No BOR occurred

1: BOR occurred (default after power-on reset)

Bits 20:18 Reserved, must be kept at reset value.

Bit 17 **LCKRSTF**: CPU lockup reset flag

This bit is set by hardware when a reset from a CPU lockup occurs. it is reset by software by writing RMVF.

0: No reset from CPU lockup occurred

1: Reset from CPU lockup occurred

Bit 16 **RMVF**: Remove reset flag

This bit is write-protected by the security bit. It is security-protected by a SEC signal from RIFSC, the SYSSEC bit, a PRIV signal from RIFSC, or the SYSPRIV bit, and is publicly readable if SYSPUB = 1. This bit is written by software to clear the value of the reset flags in this register.

0: Clear of the reset flags not activated (default after power-on reset)

1: Clear the value of the reset flags

Bits 15:0 Reserved, must be kept at reset value.

14.10.9 RCC LSE configuration register (RCC_LSECFGR)

Address offset: 0x40

Reset value: 0x0000 0000

This register is used to configure the LSE oscillator. It is reset by rcc_vsw_rstn, and is in the V_{BKP} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSEDRV[1:0]		LSE GFON	LSE EXT
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LSE BYP	Res.	Res.	Res.	Res.	Res.	LSE CSSD	LSE CSSRA	LSE CSSON	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw						r	rw	w							

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:18 **LSEDRV[1:0]**: LSE oscillator driving capability

This bitfield is write-protected by the pwr_lock_backup_n signal. It is security-protected by LSESEC or LSEPRIV, and is publicly readable if LSEPUB = 1. it is set by software to select the driving capability of the LSE oscillator.

00: Lowest drive (default after reset)

01: Medium-high drive

10: Medium-low drive

11: Highest drive

Bit 17 **LSEGFON**: LSE clock glitch filter enable

This bit is write-protected by the pwr_lock_backup_n signal. It is security-protected by LSESEC or LSEPRIV, and is publicly readable if LSEPUB = 1. This bit is set and cleared by software to enable the LSE clock glitch filter. it t can be written only when LSE is disabled (LSE ACT = 0).

0: LSE clock glitch filter disabled (default after reset)

1: LSE clock glitch filter enabled

Bit 16 **LSEEXT**: LSE clock type in bypass mode

This bit is write-protected by the pwr_lock_backup_n signal. It is security-protected by LSESEC or LSEPRIV, and is publicly readable if LSEPUB = 1. This bit is set and reset by software to select the external clock type (analog or digital).

The external clock must be enabled with the LSE enable bit to be used by the device.

This LSEEXT bit can be written only if the LSE oscillator is disabled.

0: LSE in analog mode (default after reset)

1: LSE in digital mode

Bit 15 **LSEBYP**: LSE clock bypass

This bit is write-protected by the pwr_lock_backup_n signal. It is security-protected by LSESEC or LSEPRIV, and is publicly readable if LSEPUB = 1. This bit is set and cleared by software to bypass the oscillator with an external clock.

The external clock must be enabled with the LSE enable bit to be used by the device.

This LSEBYP bit can be written even if the LSE oscillator is disabled.

0: LSE oscillator not bypassed (default after reset)

1: LSE oscillator bypassed with an external clock

Bits 14:10 Reserved, must be kept at reset value.

Bit 9 **LSECSSD**: LSE clock security system (CSS) failure detection

This bit is set by hardware to indicate when a failure has been detected by the CSS on the external LSE oscillator.

0: No failure detected on the oscillator (default after reset)

1: Failure detected on the oscillator

Bit 8 **LSECSSRA**: LSE clock security system (CSS) rearm function

This bit is write-protected by the pwr_lock_backup_n signal. It is security-protected by LSESEC or LSEPRIV, and is publicly readable if LSEPUB = 1. This bit is set by software. After an LSE failure detection, the software can re-enable the LSECSS by writing this bit to 1. Reading this bit returns the written value. Prior to set this bit to 1, LSECSSON must be set to 0. Refer to *CSS on LSE* for details.

0: Writing 0 has no effect (default after reset).

1: Writing 1 generates a rearm pulse for the LSECSS function.

Bit 7 **LSECSSON**: LSE clock security system (CSS) enable

This bit is write-protected by the pwr_lock_backup_n signal. It is security-protected by LSESEC or LSEPRIV, and is publicly readable if LSEPUB = 1. This bit is set by software to enable the CSS on the LSE oscillator. Refer to *LSE oscillator* for details on the activation and deactivation sequences. Once this LSECSSON bit is enabled, it cannot be disabled, except after an LSE failure detection (LSECSSD = 1).

0: CSS on the LSE oscillator OFF (default after reset)

1: CSS on the LSE oscillator ON

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

14.10.10 RCC MSI configuration register (RCC_MSICFGR)

Address offset: 0x44

Reset value: 0x0000 0000

This register is used to configure the MSI oscillator. It is reset by int_sys_rstn, and is in the V_{CORE} voltage domain.

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

Bit 31 Reserved, must be kept at reset value.

Bits 30:23 **MSICAL[7:0]**: MSI clock calibration

This bitfield is set by hardware by option-byte loading during a system reset. It is adjusted by software through MSITRIM bitfield. This MSICAL bitfield represents the sum of the engineering-option-byte calibration value and MSITRIM[4:0] value.

Bits 22:21 Reserved, must be kept at reset value.

Bits 20:16 **MSITRIM[4:0]**: MSI clock trimming
This bitfield is set by software to adjust calibration. It is added to the engineering option bytes (CAL_BSEC_Fuse[7:0]) loaded during the reset phase (bsec_msi_cal[7:0]), to form the calibration trimming value.
MSICAL[7:0] = MSITRIM[4:0] + CAL_BSEC_Fuse[7:0].

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **MSIFREQSEL**: MSI oscillator frequency selection
This bit is set and cleared by software.
0: MSI oscillator frequency is 4 MHz (default after backup domain reset).
1: MSI oscillator frequency is 16 MHz.

Bits 8:0 Reserved, must be kept at reset value.

14.10.11 RCC HSI configuration register (RCC_HSICFGR)

Address offset: 0x48

Reset value: 0x0000 0000

This register is used to configure the HSI oscillator. It is reset by pwr_okin_vcore_rstn, and is in the V_{CORE} voltage domain.

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

Bits 31:23 **HSICAL[8:0]**: HSI clock calibration
This bitfield is set by hardware by option-byte loading during a system reset. It is adjusted by software through HSITRIM bitfield. This HSICAL bitfield represents the sum of the engineering option byte calibration value and HSITRIM[6:0] value.

Bits 22:16 **HSITRIM[6:0]**: HSI clock trimming
This bitfield is set by software to adjust calibration. It represents a signed value, added to the engineering option bytes (bsec_hsi_cal[8:0]) loaded during the reset phase (bsec_hsi_cal), to form the calibration trimming value: HSICAL[8:0] = HSITRIM[6:0] + bsec_hsi_cal[8:0].
0x1-0x3F: bsec_hsi_cal[8:0] + {V}
0x40-0x7F: bsec_hsi_cal[8:0] - 128 + {v}
0x00: bsec_hsi_cal[8:0] (default after reset)

Bits 15:9 Reserved, must be kept at reset value.

Bits 8:7 **HSIDIV[1:0]**: HSI clock divider
This bitfield is set and reset by software to control the hsi_ck frequency (see *Figure 38*).
00: hsi_div_ck = hsi_ck (default after reset)
01: hsi_div_ck = hsi_ck / 2
10: hsi_div_ck = hsi_ck / 4
11: hsi_div_ck = hsi_ck / 8

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

14.10.12 RCC HSI monitor control register (RCC_HSIMCR)

Address offset: 0x4C

Reset value: 0x001F 07A1

This register is used to monitor and trim of the HSI oscillator. It is reset by int_sys_rstn, and is in the V_{CORE} voltage domain.

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

Bit 31 **HSIMONEN**: HSI clock period monitor enable
This bit is set and cleared by software.
0: Writing 0 disables the HSI clock period monitoring. Reading 0 means that the HSI clock period monitoring is disabled.
1: Writing 1 enables the HSI clock period monitoring. Reading 1 means that the HSI clock period monitoring is enabled.

Bits 30:22 Reserved, must be kept at reset value.

Bits 21:16 **HSIDEV[5:0]**: HSI clock count deviation value
This bitfield is set and cleared by software.

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **HSIREF[10:0]**: HSI clock-cycle counter reference value.
This bit contains the number of HSI clock cycles expected between two consecutive rising edges of the LSE signal. It is set by hardware.

14.10.13 RCC HSI monitor status register (RCC_HSIMSR)

Address offset: 0x50

Reset value: 0x0000 0000

This register is used to monitor and trim of the HSI oscillator. It is reset by int_sys_rstn, and is in the V_{CORE} voltage domain.

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

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:0 **HSIVAL[10:0]**: HSI clock-cycle counter measured value
This bitfield contains the number of HSI clock cycles measured between consecutive rising edges of the LSE signal. it is set by hardware.

14.10.14 RCC HSE configuration register (RCC_HSECFGR)

Address offset: 0x54

Reset value: 0x0000 0800

This register is used to configure the HSE oscillator. It is reset by int_sys_rstn, and is in the V_{CORE} voltage domain.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HSE BYP	HSECSSBPRE[3:0]				HSE CSSBYP	HSE CSSD	Res.	HSE CSSON	HSE DIV2SEL	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	r		rs	rw						

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **HSEEXT**: HSE clock type in bypass mode

This bit is set and reset by software to select the external clock type (analog or digital). The external clock must be enabled with the HSE enable bit to be used by the device. This HSEEXT bit can be written only if the HSE oscillator is disabled.

0: HSE in analog mode (default after reset)

1: HSE in digital mode

Bit 15 **HSEBYP**: HSE clock bypass

This bit is set and cleared by software to bypass the oscillator with an external clock. The external clock must be enabled with the HSE enable bit to be used by the device. This HSEBYP bit can be written even if the HSE oscillator is disabled.

0: HSE oscillator not bypassed (default after reset)

1: HSE oscillator bypassed with an external clock

Bits 14:11 **HSECSSBPRE[3:0]**: HSE CSS bypass divider

This bitfield is set and reset by software to divide the replacement internal HSI oscillator that bypasses the HSE oscillator when a failure is detected. Refer to *HSE oscillator* for details on the activation and deactivation sequences.

0000: HSI clock divided by 1
0001: HSI clock divided by 2
0010: HSI clock divided by 3
0011: HSI clock divided by 4
0100: HSI clock divided by 5
0101: HSI clock divided by 6
0110: HSI clock divided by 7
0111: HSI clock divided by 8
1000: HSI clock divided by 9
1001: HSI clock divided by 10
1010: HSI clock divided by 11
1011: HSI clock divided by 12
1100: HSI clock divided by 13
1101: HSI clock divided by 14
1110: HSI clock divided by 15
1111: HSI clock divided by 16

Bit 10 **HSECSSBYP**: HSE CSS bypass enable

This bit is set and reset by software to enable the CSS to bypass the HSE oscillator when a failure is detected, and to get a clock from the HSI oscillator. Refer to *HSE oscillator* for details on the activation and deactivation sequences.

0: CSS bypass of the HSE oscillator OFF (default after reset)

1: CSS bypass on the HSE oscillator ON

Bit 9 **HSECSSD**: HSE CSS failure detection

This bit is set by hardware to indicate when a failure has been detected by the CSS on the external HSE oscillator.

0: No failure detected on the oscillator (default after reset)

1: Failure detected on the oscillator

Bit 8 Reserved, must be kept at reset value.

Bit 7 **HSECSSON**: HSE CSS enable

This bit is set by software to enable the CSS on the HSE oscillator. Refer to *HSE oscillator* for details on the activation and deactivation sequences. Once this HSECSSON bit is enabled, it cannot be disabled, except after an HSE failure detection (HSECSSD = 1).

0: CSS on the HSE oscillator OFF (default after reset)

1: CSS on the HSE oscillator ON

Bit 6 **HSEDIV2SEL**: HSE div2 clock source select

This bit is set and reset by software to select the source of the div2 output clock.

0: HSE: hse_div2_osc_ck = hse_osc_ck (default after reset)

1: HSE: hse_div2_osc_ck = hse_osc_ck/2

Bits 5:0 Reserved, must be kept at reset value.

14.10.15 RCC PLL1 configuration register 1 (RCC_PLL1CFGR1)

Address offset: 0x80

Reset value: 0x0820 2500

This register is used to configure the main features of PLL1. It is reset by int_sys_rst,n and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	PLL1SEL[2:0]			PLL1BYP	Res.	PLL1DIVM[5:0]						PLL1DIVN[11:8]			
	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLL1DIVN[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw								

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **PLL1SEL[2:0]**: PLL1 source selection of the reference clock

This bitfield is set and reset by software to select system clock source (sys_ck).

000: hsi_ck selected as reference clock

001: msi_ck selected as reference clock

010: hse_ck selected as reference clock

011: I2S_CKIN selected as reference clock

Bit 27 **PLL1BYP**: PLL1 bypass
This bit is set and cleared by software to bypass the VCO, and to feed the output with the PLL reference clock.
0: PLL output is driven by the VCO, via the optional POSTDIV division.
1: PLL output is bypassed and driven by the PLL reference clock (default after reset).

Bit 26 Reserved, must be kept at reset value.

Bits 25:20 **PLL1DIVM[5:0]**: PLL1 reference input clock divide frequency ratio
This bitfield is set and cleared by software.
0x00: Not applicable when the PLL is enabled
0x01: Reference clock divided by 1 (min value)
0x2-0x3F: Reference clock divided by {v}. It is recommended to configure the maximum divided reference clock frequency close to FVCO / 16.

Bits 19:8 **PLL1DIVN[11:0]**: PLL1 integer part for the VCO multiplication factor
This bitfield is set and cleared by software to control the multiplication factor of the VCO.
VCO output frequency = reference clock * DIVN / DIVM when FRACV = 0.
In integer mode, this value must be set between 0x10 (16) and 0x280 (640).
In fractional mode, this value must be set between 0x14 (20) and 0x140 (320).

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

14.10.16 RCC PLL1 configuration register 2 (RCC_PLL1CFGR2)

Address offset: 0x84

Reset value: 0x0080 0000

This register is used to configure the optional fractional feedback division of PLL1. It is reset by int_sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL1DIVNFRAC[23:16]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLL1DIVNFRAC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:0 **PLL1DIVNFRAC[23:0]**: PLL1 fractional part of the VCO multiplication factor
This bitfield is set and cleared by software to control the VCO fractional multiplication factor.
VCO output frequency = reference clock * (DIVN + DIVNFRAC / 2^{24}) / DIVM.

14.10.17 RCC PLL1 configuration register 3 (RCC_PLL1CFGR3)

Address offset: 0x88

Reset value: 0x4900 000D

This register is used to configure the SSCG and optional inner features of PLL1. It is reset by int_sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	PLL1PDIVEN	PLL1PDIV1[2:0]			PLL1PDIV2[2:0]			Res.	Res.	Res.	PLL1MODSPR[4:0]				
	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	PLL1MODDIV[3:0]				Res.	Res.	Res.	PLL1MODSPRDW	PLL1MODDSEN	PLL1MODSSDIS	PLL1DACEN	PLL1MODSSRST
				rw	rw	rw	rw				rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **PLL1PDIVEN**: PLL1 post divider POSTDIV1, POSTDIV2, and PLL clock output enable
This bit is set and cleared by software to enable the post divider.
0: POSTDIV1 and POSTDIV2 powered down
1: POSTDIV1 and POSTDIV2 active (default after reset)

Bits 29:27 **PLL1PDIV1[2:0]**: PLL1 VCO frequency divider level 1
This bitfield is set and cleared by software to divide the VCO frequency output.
000: Not applicable
001: VCO output divided by 1 (minimum value) (default after reset)
010: VCO output divided by 2
011: VCO output divided by 3
100: VCO output divided by 4
101: VCO output divided by 5
110: VCO output divided by 6
111: VCO output divided by 7

Bits 26:24 **PLL1PDIV2[2:0]**: PLL1 VCO frequency divider level 2
This bitfield is set and cleared by software to divide the VCO frequency output.
000: Not applicable
001: VCO output divided by 1 (minimum value) (default after reset)
010: VCO output divided by 2
011: VCO output divided by 3
100: VCO output divided by 4
101: VCO output divided by 5
110: VCO output divided by 6
111: VCO output divided by 7

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:16 **PLL1MODSPR[4:0]**: PLL1 modulation spread depth adjustment
This bitfield is set and cleared by software to adjust the modulation depth of the clock spreading generator.

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

Bits 11:8 **PLL1MODDIV[3:0]**: PLL1 modulation division frequency adjustment
This bitfield is set and cleared by software to adjust the modulation frequency of the clock spreading generator. Corresponds to DIVVAL in *Figure 44*).

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **PLL1MODSPRDW**: PLL1 modulation spread spectrum down
This bit is set and cleared by software to select the clock spreading mode of PLL1.
0: Center-spread modulation selected (default after reset)
1: Down-spread modulation selected

Bit 3 **PLL1MODDSEN**: PLL1 modulation spread spectrum (and fractional divide) enable
This bit is set and cleared by software to enable the delta-sigma modulator.
0: Modulation spread spectrum and fractional divide not active
1: Modulation spread spectrum and fractional divide active (default after reset)

Bit 2 **PLL1MODSSDIS**: PLL1 modulation spread spectrum disable
This bit is set and cleared by software to enable the clock spreading generator of PLL1.
0: Modulation spread spectrum active (and fractional divide inactive)
1: Fractional divide active (and modulation spread spectrum inactive) (default after reset)

Bit 1 **PLL1DACEN**: PLL1 noise canceling DAC enable in fractional mode
This bit is set and cleared by software to enable the noise cancellation in fractional mode.
0: DAC not active (default after reset)
1: DAC active

Bit 0 **PLL1MODSSRST**: PLL1 modulation spread spectrum reset
This bit is set and cleared by software.
0: PLL1 modulation spread spectrum reset module released
1: PLL1 modulation spread spectrum reset module asserted (default after reset)

14.10.18 RCC PLL2 configuration register 1 (RCC_PLL2CFGR1)

Address offset: 0x90

Reset value: 0x0800 0000

This register is used to configure the main features of PLL2. It is reset by int_sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	PLL2SEL[2:0]			PLL2BYP	Res.	PLL2DIVM[5:0]						PLL2DIVN[11:8]			
	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLL2DIVN[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw								

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **PLL2SEL[2:0]**: PLL2 source selection of the reference clock
This bitfield is set and reset by software to select system clock source (sys_ck).
000: hsi_ck selected as reference clock
001: msi_ck selected as reference clock
010: hse_ck selected as reference clock
011: I2S_CKIN selected as reference clock

Bit 27 **PLL2BYP**: PLL2 bypass

This bit is set and cleared by software to bypass the VC,O and to feed the output with the PLL reference clock.

0: PLL output is driven by the VCO, via the optional POSTDIV division.

1: PLL output is bypassed and driven by the PLL reference clock (default after reset).

Bit 26 Reserved, must be kept at reset value.

Bits 25:20 **PLL2DIVM[5:0]**: PLL2 reference input clock divide frequency ratio

This bitfield is set and cleared by software.

0x2-0x3F: Reference clock is divided by {v}. The maximum divided reference clock frequency must be close to FVCO/16.

0x00: Not applicable when PLL is enabled

0x01: Reference clock is divided by 1 (min value).

Bits 19:8 **PLL2DIVN[11:0]**: PLL2 integer part for the VCO multiplication factor

This bitfield is set and cleared by software to control the multiplication factor of the VCO.

VCO output frequency = reference clock * DIVN / DIVM when FRACV = 0.

In integer mode, this value must be set between 0x10 (16) and 0x280 (640).

In fractional mode, this value must be set between 0x14 (20) and 0x140 (320).

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

14.10.19 RCC PLL2 configuration register 2 (RCC_PLL2CFGR2)

Address offset: 0x94

Reset value: 0x0000 0000

This register is used to configure the optional fractional feedback division of PLL2. It is reset by int_sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL2DIVNFRAC[23:16]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLL2DIVNFRAC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:0 **PLL2DIVNFRAC[23:0]**: PLL2 fractional part of the VCO multiplication factor

This bitfield is set and cleared by software to control the VCO fractional multiplication factor.

VCO output frequency = reference clock * (DIVN + DIVNFRAC / 2^{24}) / DIVM.

14.10.20 RCC PLL2 configuration register 3 (RCC_PLL2CFGR3)

Address offset: 0x98

Reset value: 0x4900 0005

This register is used to configure the SSCG and optional inner features of PLL2. It is reset by int_sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	PLL2PDIVEN	PLL2PDIV1[2:0]			PLL2PDIV2[2:0]			Res.	Res.	Res.	PLL2MODSPR[4:0]				
	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	PLL2MODDIV[3:0]				Res.	Res.	Res.	PLL2MODSPRDW	PLL2MODDSEN	PLL2MODSSDIS	PLL2DACEN	PLL2MODSSRST
				rw	rw	rw	rw				rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **PLL2PDIVEN**: PLL2 post divider POSTDIV1, POSTDIV2, and PLL clock output enable
This bit is set and cleared by software to enable the post divider.
0: POSTDIV1 and POSTDIV2 powered down
1: POSTDIV1 and POSTDIV2 active (default after reset)

Bits 29:27 **PLL2PDIV1[2:0]**: PLL2 VCO frequency divider level 1
This bitfield is set and cleared by software to divide the VCO frequency output.
000: Not applicable
001: VCO output divided by 1 (minimum value) (default after reset)
010: VCO output divided by 2
011: VCO output divided by 3
100: VCO output divided by 4
101: VCO output divided by 5
110: VCO output divided by 6
111: VCO output divided by 7

Bits 26:24 **PLL2PDIV2[2:0]**: PLL2 VCO frequency divider level 2
This bitfield is set and cleared by software to divide the VCO frequency output.
000: Not applicable
001: VCO output divided by 1 (minimum value) (default after reset)
010: VCO output divided by 2
011: VCO output divided by 3
100: VCO output divided by 4
101: VCO output divided by 5
110: VCO output divided by 6
111: VCO output divided by 7

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:16 **PLL2MODSPR[4:0]**: PLL2 modulation spread depth adjustment
This bitfield is set and cleared by software to adjust the modulation depth of the clock spreading generator.

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

Bits 11:8 **PLL2MODDIV[3:0]**: PLL2 modulation division frequency adjustment
This bitfield is set and cleared by software to adjust the modulation frequency of the clock spreading generator. It corresponds to DIVVAL in *Figure 44*.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **PLL2MODSPRDW**: PLL2 modulation down spread
This bit is set and cleared by software to select the clock spreading mode of PLL2.
0: Center-spread modulation selected (default after reset)
1: Down-spread modulation selected

Bit 3 **PLL2MODDSEN**: PLL2 modulation spread spectrum (and fractional divide) enable
This bit is set and cleared by software to enable the delta-sigma modulator.
0: Modulation spread spectrum and fractional divide not active (default after reset)
1: Modulation spread spectrum and fractional divide active

Bit 2 **PLL2MODSSDIS**: PLL2 modulation spread spectrum disable
This bit is set and cleared by software to enable the clock spreading generator of PLL2.
0: Modulation spread spectrum active (and fractional divide inactive)
1: Fractional divide active (and the Modulation spread spectrum inactive) (default after reset)

Bit 1 **PLL2DACEN**: PLL2 noise canceling DAC enable in fractional mode
This bit is set and cleared by software to enable the noise cancellation in fractional mode.
0: DAC not active (default after reset)
1: DAC active

Bit 0 **PLL2MODSSRST**: PLL2 modulation spread spectrum reset
This bit is set and cleared by software.
0: PLL2 modulation spread spectrum reset module released
1: PLL2 modulation spread spectrum reset module asserted (default after reset)

14.10.21 RCC PLL3 configuration register 1 (RCC_PLL3CFGR1)

Address offset: 0xA0

Reset value: 0x0800 0000

This register is used to configure the main features of PLL3. It is reset by int_sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	PLL3SEL[2:0]			PLL3BYP	Res.	PLL3DIVM[5:0]						PLL3DIVN[11:8]			
	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLL3DIVN[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw								

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **PLL3SEL[2:0]**: PLL3 source selection of the reference clock
This bitfield is set and reset by software to select system clock source (sys_ck).
000: hsi_ck selected as reference clock
001: msi_ck selected as reference clock
010: hse_ck selected as reference clock
011: I2S_CKIN selected as reference clock

Bit 27 **PLL3BYP**: PLL3 bypass
This bit is set and cleared by software to bypass the VCO and feed the output with the PLL reference clock.
0: PLL output is driven by the VCO, via the optional POSTDIV division.
1: PLL output is bypassed and driven by the PLL reference clock (default after reset).

Bit 26 Reserved, must be kept at reset value.

Bits 25:20 **PLL3DIVM[5:0]**: PLL3 reference input clock divide frequency ratio
This bitfield is set and cleared by software.
0x2-0x3F: Reference clock is divided by {v}. The maximum divided reference clock frequency must be close to FVCO/16.
0x00: Not applicable when PLL is enabled
0x01: Reference clock is divided by 1 (min value)

Bits 19:8 **PLL3DIVN[11:0]**: PLL3 Integer part for the VCO multiplication factor
This bitfield is set and cleared by software to control the multiplication factor of the VCO.
VCO output frequency = reference clock * DIVN / DIVM when FRACV = 0.
In integer mode, this value must be set between 0x10 (16) and 0x280 (640).
In fractional mode, this value must be set between 0x14 (20) and 0x140 (320).

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

14.10.22 RCC PLL3 configuration register 2 (RCC_PLL3CFGR2)

Address offset: 0xA4

Reset value: 0x0000 0000

This register is used to configure the optional fractional feedback division of PLL3. It is reset by int_sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL3DIVNFRAC[23:16]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLL3DIVNFRAC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:0 **PLL3DIVNFRAC[23:0]**: PLL3 fractional part of the VCO multiplication factor
This bitfield is set and cleared by software to control the VCO fractional multiplication factor.
VCO output frequency = reference clock * (DIVN + DIVNFRAC / 2^{24}) / DIVM.

14.10.23 RCC PLL3 configuration register 3 (RCC_PLL3CFGR3)

Address offset: 0xA8

Reset value: 0x4900 0005

This register is used to configure the SSCG and optional inner features of PLL3. It is reset by int_sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	PLL3PDIVEN	PLL3PDIV1[2:0]			PLL3PDIV2[2:0]			Res.	Res.	Res.	PLL3MODSPR[4:0]				
	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	PLL3MODDIV[3:0]				Res.	Res.	Res.	PLL3MODSPRDW	PLL3MODDSEN	PLL3MODSSDIS	PLL3DACEN	PLL3MODSSRST
				rw	rw	rw	rw				rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **PLL3PDIVEN**: PLL3 post divider POSTDIV1, POSTDIV2, and PLL clock output enable
This bit is set and cleared by software to enable the post divider.
0: POSTDIV1 and POSTDIV2 powered down
1: POSTDIV1 and POSTDIV2 active (default after reset)

Bits 29:27 **PLL3PDIV1[2:0]**: PLL3 VCO frequency divider level 1
This bitfield is set and cleared by software to divide the VCO frequency output.
000: Not applicable
001: VCO output divided by 1 (minimum value) (default after reset)
010: VCO output divided by 2
011: VCO output divided by 3
100: VCO output divided by 4
101: VCO output divided by 5
110: VCO output divided by 6
111: VCO output divided by 7

Bits 26:24 **PLL3PDIV2[2:0]**: PLL3 VCO frequency divider level 2
This bitfield is set and cleared by software to divide the VCO frequency output.
000: Not applicable
001: VCO output divided by 1 (minimum value) (default after reset)
010: VCO output divided by 2
011: VCO output divided by 3
100: VCO output divided by 4
101: VCO output divided by 5
110: VCO output divided by 6
111: VCO output divided by 7

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:16 **PLL3MODSPR[4:0]**: PLL3 modulation spread depth adjustment
This bitfield is set and cleared by software to adjust the modulation depth of the clock spreading generator.

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

Bits 11:8 **PLL3MODDIV[3:0]**: PLL3 modulation division frequency adjustment
This bitfield is set and cleared by software to adjust the modulation frequency of the clock spreading generator. It corresponds to DIVVAL in *Figure 44*.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **PLL3MODSPRDW**: PLL3 modulation down spread
This bit is set and cleared by software to select the clock spreading mode of PLL3.
0: Center-spread modulation selected (default after reset)
1: Down-spread modulation selected

Bit 3 **PLL3MODDSEN**: PLL3 modulation spread spectrum (and fractional divide) enable
This bit is set and cleared by software to enable the delta-sigma modulator.
0: Modulation spread spectrum and fractional divide not active (default after reset)
1: Modulation spread spectrum and fractional divide active

Bit 2 **PLL3MODSSDIS**: PLL3 modulation spread spectrum disable
This bit is set and cleared by software to enable the clock spreading generator of PLL3.
0: Modulation spread spectrum active (and fractional divide inactive)
1: Fractional divide active (and the modulation spread spectrum inactive) (default after reset)

Bit 1 **PLL3DACEN**: PLL3 noise canceling DAC enable in fractional mode
This bit is set and cleared by software to enable the noise cancellation in fractional mode.
0: DAC not active (default after reset)
1: DAC active

Bit 0 **PLL3MODSSRST**: PLL3 modulation spread spectrum reset
This bit is set and cleared by software.
0: PLL3 modulation spread spectrum reset module released
1: PLL3 modulation spread spectrum reset module asserted (default after reset)

14.10.24 RCC PLL4 configuration register 1 (RCC_PLL4CFGR1)

Address offset: 0xB0

Reset value: 0x0800 0000

This register is used to configure the main features of PLL4. It is reset by int_sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	PLL4SEL[2:0]			PLL4BYP	Res.	PLL4DIVM[5:0]						PLL4DIVN[11:8]			
	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLL4DIVN[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw								

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **PLL4SEL[2:0]**: PLL4 source selection of the reference clock
This bitfield is set and reset by software to select system clock source (sys_ck).
000: hsi_ck selected as reference clock
001: msi_ck selected as reference clock
010: hse_ck selected as reference clock
011: I2S_CKIN selected as reference clock

Bit 27 **PLL4BYP**: PLL4 bypass
This bit is set and cleared by software to bypass the VCO, and to feed the output with the PLL reference clock.
0: PLL output is driven by the VCO, via the optional POSTDIV division.
1: PLL output is bypassed and driven by the PLL reference clock (default after reset).

Bit 26 Reserved, must be kept at reset value.

Bits 25:20 **PLL4DIVM[5:0]**: PLL4 reference input clock divide frequency ratio
This bitfield is set and cleared by software.
0x2-0x3F: Reference clock is divided by {v}. The maximum divided reference clock frequency must be close to FVCO/16.
0x00: Not applicable when PLL is enabled
0x01: Reference clock is divided by 1 (min value).

Bits 19:8 **PLL4DIVN[11:0]**: PLL4 integer part for the VCO multiplication factor
This bitfield is set and cleared by software to control the multiplication factor of the VCO.
VCO output frequency = reference clock * DIVN / DIVM when FRACV = 0.
In integer mode, this value must be set between 0x10 (16) and 0x280 (640).
In fractional mode, this value must be set between 0x14 (20) and 0x140 (320).

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

14.10.25 RCC PLL4 configuration register 2 (RCC_PLL4CFGR2)

Address offset: 0xB4

Reset value: 0x0000 0000

This register is used to configure the optional fractional feedback division of PLL4. It is reset by int_sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL4DIVNFRAC[23:16]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLL4DIVNFRAC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:0 **PLL4DIVNFRAC[23:0]**: PLL4 fractional part of the VCO multiplication factor
This bitfield is set and cleared by software to control the VCO fractional multiplication factor.
VCO output frequency = reference clock * (DIVN + DIVNFRAC / 2^{24}) / DIVM.

14.10.26 RCC PLL4 configuration register 3 (RCC_PLL4CFGR3)

Address offset: 0xB8

Reset value: 0x4900 0005

This register is used to configure the SSCG and optional inner features of PLL4. It is reset by int_sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	PLL4PDIVEN	PLL4PDIV1[2:0]			PLL4PDIV2[2:0]			Res.	Res.	Res.	PLL4MODSPR[4:0]				
	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	PLL4MODDIV[3:0]				Res.	Res.	Res.	PLL4MODSPRDW	PLL4MODDSEN	PLL4MODSSDIS	PLL4DACEN	PLL4MODSSRST
				rw	rw	rw	rw				rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **PLL4PDIVEN**: PLL4 post divider POSTDIV1, POSTDIV2, and PLL clock output enable
This bit is set and cleared by software to enable the post divider.
0: POSTDIV1 and POSTDIV2 powered down
1: POSTDIV1 and POSTDIV2 active (default after reset)

Bits 29:27 **PLL4PDIV1[2:0]**: PLL4 VCO frequency divider level 1
This bitfield is set and cleared by software to divide the VCO frequency output.
000: Not applicable
001: VCO output divided by 1 (minimum value) (default after reset)
010: VCO output divided by 2
011: VCO output divided by 3
100: VCO output divided by 4
101: VCO output divided by 5
110: VCO output divided by 6
111: VCO output divided by 7

Bits 26:24 **PLL4PDIV2[2:0]**: PLL4 VCO frequency divider level 2
This bitfield is set and cleared by software to divide the VCO frequency output.
000: Not applicable
001: VCO output divided by 1 (minimum value) (default after reset)
010: VCO output divided by 2
011: VCO output divided by 3
100: VCO output divided by 4
101: VCO output divided by 5
110: VCO output divided by 6
111: VCO output divided by 7

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:16 **PLL4MODSPR[4:0]**: PLL4 modulation spread depth adjustment
This bitfield is set and cleared by software to adjust the modulation depth of the clock spreading generator.

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

Bits 11:8 **PLL4MODDIV[3:0]**: PLL4 modulation division frequency adjustment
This bitfield is set and cleared by software to adjust the modulation frequency of the clock spreading generator. It corresponds to DIVVAL in *Figure 44*.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **PLL4MODSPRDW**: PLL4 modulation down spread
This bit is set and cleared by software to select the clock spreading mode of PLL4.
0: Center-spread modulation selected (default after reset)
1: Down-spread modulation selected

Bit 3 **PLL4MODDSEN**: PLL4 modulation spread spectrum (and fractional divide) enable
This bitfield is set and cleared by software to enable the delta-sigma modulator.
0: Modulation spread spectrum and fractional divide not active (default after reset)
1: Modulation spread spectrum and fractional divide active

Bit 2 **PLL4MODSSDIS**: PLL4 modulation spread spectrum disable
This bit is set and cleared by software to enable the clock spreading generator of PLL4.
0: Modulation spread spectrum active (and fractional divide inactive)
1: Fractional divide active (and modulation spread spectrum inactive) (default after reset)

Bit 1 **PLL4DACEN**: PLL4 noise canceling DAC enable in fractional mode
This bit is set and cleared by software to enable the noise cancellation in fractional mode.
0: DAC not active (default after reset)
1: DAC active

Bit 0 **PLL4MODSSRST**: PLL4 modulation spread spectrum reset
This bit is set and cleared by software.
0: PLL4 modulation spread spectrum reset module released
1: PLL4 modulation spread spectrum reset module asserted (default after reset)

14.10.27 RCC IC1 configuration register (RCC_IC1CFGR)

Address offset: 0xC4

Reset value: 0x0002 0000

This register is used to configure the IC1 divider. It is reset by int_sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	IC1SEL[1:0]		Res.	Res.	Res.	Res.	IC1INT[7:0]							
		rw	rw					rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **IC1SEL[1:0]**: Divider IC1 source selection
This bitfield is set and reset by software to select the PLL output to feed for the channel IC1.
00: pll1_ck selected (default after reset)
01: pll2_ck selected
10: pll3_ck selected
11: pll4_ck selected

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:16 **IC1INT[7:0]**: Divider IC1 integer division factor

This bitfield is set and reset by software to control the frequency of the IC1 clock. The frequency is divided by the value IC1INT + 1: {v}: IC1 = pllx_ck / {v + 1}.

0x00: IC1 = pllx_ck

0x01: IC1 = pllx_ck / 2

0x02: IC1 = pllx_ck / 3 (default after reset)

0x03: IC1 = pllx_ck / 4

...

0xFF: IC1 = pllx_ck / 256

Bits 15:0 Reserved, must be kept at reset value.

14.10.28 RCC IC2 configuration register (RCC_IC2CFGR)

Address offset: 0xC8

Reset value: 0x0003 0000

This register is used to configure the IC2 divider. It is reset by int_sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	IC2SEL[1:0]		Res.	Res.	Res.	Res.	IC2INT[7:0]							
		rw	rw					rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **IC2SEL[1:0]**: Divider IC2 source selection

This bitfield is set and reset by software to select the PLL output to feed for the channel IC2.

00: pll1_ck selected (default after reset)

01: pll2_ck selected

10: pll3_ck selected

11: pll4_ck selected

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:16 **IC2INT[7:0]**: Divider IC2 integer division factor

This bitfield is set and reset by software to control the frequency of the IC2 clock. The frequency is divided by the value IC2INT + 1: {v}: IC2 = pllx_ck / {v + 1}.

0x00: IC2 = pllx_ck

0x01: IC2 = pllx_ck / 2

0x02: IC2 = pllx_ck / 3

0x03: IC2 = pllx_ck / 4 (default after reset)

...

0xFF: IC2 = pllx_ck / 256

Bits 15:0 Reserved, must be kept at reset value.

14.10.29 RCC IC3 configuration register (RCC_IC3CFGR)

Address offset: 0xCC

Reset value: 0x0000 0000

This register is used to configure the IC3 divider. It is reset by int_sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	IC3SEL[1:0]		Res.	Res.	Res.	Res.	IC3INT[7:0]							
		rw	rw					rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **IC3SEL[1:0]**: Divider IC3 source selection

This bitfield is set and reset by software to select the PLL output to feed for the channel IC3.
00: pll1_ck selected (default after reset)
01: pll2_ck selected
10: pll3_ck selected
11: pll4_ck selected

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:16 **IC3INT[7:0]**: Divider IC3 integer division factor

This bitfield is set and reset by software to control the frequency of the IC3 clock. The frequency is divided by the value IC3INT + 1: {v}: IC3 = pllx_ck / {v + 1}.
0x00: IC3 = pllx_ck (default after reset)
0x01: IC3 = pllx_ck / 2
0x02: IC3 = pllx_ck / 3
0x03: IC3 = pllx_ck / 4
...
0xFF: IC3 = pllx_ck / 256

Bits 15:0 Reserved, must be kept at reset value.

14.10.30 RCC IC4 configuration register (RCC_IC4CFGR)

Address offset: 0xD0

Reset value: 0x0000 0000

This register is used to configure the IC4 divider. It is reset by int_sys_rstn and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	IC4SEL[1:0]		Res.	Res.	Res.	Res.	IC4INT[7:0]							
		rw	rw					rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **IC4SEL[1:0]**: Divider IC4 source selection

This bitfield is set and reset by software to select the PLL output to feed for the channel IC4.
00: pll1_ck selected (default after reset)
01: pll2_ck selected
10: pll3_ck selected
11: pll4_ck selected

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:16 **IC4INT[7:0]**: Divider IC4 integer division factor

This bitfield is set and reset by software to control the frequency of the IC4 clock. The frequency is divided by the value IC4INT+1: {v}: IC4 = pllx_ck / {v + 1}.
0x00: IC4 = pllx_ck (default after reset)
0x01: IC4 = pllx_ck / 2
0x02: IC4 = pllx_ck / 3
0x03: IC4 = pllx_ck / 4
...
0xFF: IC4 = pllx_ck / 256

Bits 15:0 Reserved, must be kept at reset value.

14.10.31 RCC IC5 configuration register (RCC_IC5CFGR)

Address offset: 0xD4

Reset value: 0x0000 0000

This register is used to configure the IC5 divider. It is reset by int_sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	IC5SEL[1:0]		Res.	Res.	Res.	Res.	IC5INT[7:0]							
		rw	rw					rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **IC5SEL[1:0]**: Divider IC5 source selection

This bitfield is set and reset by software to select the PLL output to feed for the channel IC5.
00: pll1_ck selected (default after reset)
01: pll2_ck selected
10: pll3_ck selected
11: pll4_ck selected

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:16 **IC5INT[7:0]**: Divider IC5 integer division factor

This bitfield is set and reset by software to control the frequency of the IC5 clock. The frequency is divided by the value IC5INT + 1: {v}: IC5 = pllx_ck / {v + 1}.

0x00: IC5 = pllx_ck (default after reset)

0x01: IC5 = pllx_ck / 2

0x02: IC5 = pllx_ck / 3

0x03: IC5 = pllx_ck / 4

...

0xFF: IC5 = pllx_ck / 256

Bits 15:0 Reserved, must be kept at reset value.

14.10.32 RCC IC6 configuration register (RCC_IC6CFGR)

Address offset: 0xD8

Reset value: 0x0003 0000

This register is used to configure the IC6 divider. It is reset by int_sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	IC6SEL[1:0]		Res.	Res.	Res.	Res.	IC6INT[7:0]							
		rw	rw					rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **IC6SEL[1:0]**: Divider IC6 source selection

This bitfield is set and reset by software to select the PLL output to feed for the channel IC6.

00: pll1_ck selected (default after reset)

01: pll2_ck selected

10: pll3_ck selected

11: pll4_ck selected

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:16 **IC6INT[7:0]**: Divider IC6 integer division factor

This bitfield is set and reset by software to control the frequency of the IC6 clock. The frequency is divided by the value IC6INT+1: {v}: IC6 = pllx_ck / {v + 1}.

0x00: IC6 = pllx_ck

0x01: IC6 = pllx_ck / 2

0x02: IC6 = pllx_ck / 3

0x03: IC6 = pllx_ck / 4 (default after reset)

...

0xFF: IC6 = pllx_ck / 256

Bits 15:0 Reserved, must be kept at reset value.

14.10.33 RCC IC7 configuration register (RCC_IC7CFGR)

Address offset: 0xDC

Reset value: 0x1000 0000

This register is used to configure the IC7 divider. It is reset by int_sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	IC7SEL[1:0]		Res.	Res.	Res.	Res.	IC7INT[7:0]							
		rw	rw					rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **IC7SEL[1:0]**: Divider IC7 source selection

This bitfield is set and reset by software to select the PLL output to feed for the channel IC7.
00: pll1_ck selected
01: pll2_ck selected (default after reset)
10: pll3_ck selected
11: pll4_ck selected

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:16 **IC7INT[7:0]**: Divider IC7 integer division factor

This bitfield is set and reset by software to control the frequency of the IC7 clock. The frequency is divided by the value IC7INT + 1: {v}: IC7 = pllx_ck / {v + 1}.
0x00: IC7 = pllx_ck (default after reset)
0x01: IC7 = pllx_ck / 2
0x02: IC7 = pllx_ck / 3
0x03: IC7 = pllx_ck / 4
...
0xFF: IC7 = pllx_ck / 256

Bits 15:0 Reserved, must be kept at reset value.

14.10.34 RCC IC8 configuration register (RCC_IC8CFGR)

Address offset: 0xE0

Reset value: 0x1000 0000

This register is used to configure the IC8 divider. It is reset by int_sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	IC8SEL[1:0]		Res.	Res.	Res.	Res.	IC8INT[7:0]							
		rw	rw					rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **IC8SEL[1:0]**: Divider IC8 source selection

This bitfield is set and reset by software to select the PLL output to feed for the channel IC8.
00: pll1_ck selected
01: pll2_ck selected (default after reset)
10: pll3_ck selected
11: pll4_ck selected

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:16 **IC8INT[7:0]**: Divider IC8 integer division factor

This bitfield is set and reset by software to control the frequency of the IC8 clock. The frequency is divided by the value IC8INT + 1: {v}: IC8 = pllx_ck / {v + 1}.
0x00: IC8 = pllx_ck (default after reset)
0x01: IC8 = pllx_ck / 2
0x02: IC8 = pllx_ck / 3
0x03: IC8 = pllx_ck / 4
...
0xFF: IC8 = pllx_ck / 256

Bits 15:0 Reserved, must be kept at reset value.

14.10.35 RCC IC9 configuration register (RCC_IC9CFGR)

Address offset: 0xE4

Reset value: 0x1000 0000

This register is used to configure the IC9 divider. It is reset by int_sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	IC9SEL[1:0]		Res.	Res.	Res.	Res.	IC9INT[7:0]							
		rw	rw					rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **IC9SEL[1:0]**: Divider IC9 source selection

This bitfield is set and reset by software to select the PLL output to feed for the channel IC9.
00: pll1_ck selected
01: pll2_ck selected (default after reset)
10: pll3_ck selected
11: pll4_ck selected

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:16 **IC9INT[7:0]**: Divider IC9 integer division factor
This bitfield is set and reset by software to control the frequency of the IC9 clock. The frequency is divided by the value IC9INT + 1: {v}: IC9 = pllx_ck / {v + 1}.
0x00: IC9 = pllx_ck (default after reset)
0x01: IC9 = pllx_ck / 2
0x02: IC9 = pllx_ck / 3
0x03: IC9 = pllx_ck / 4
...
0xFF: IC9 = pllx_ck / 256

Bits 15:0 Reserved, must be kept at reset value.

14.10.36 RCC IC10 configuration register (RCC_IC10CFGR)

Address offset: 0xE8

Reset value: 0x1000 0000

This register is used to configure the IC10 divider. It is reset by int_sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	IC10SEL[1:0]		Res.	Res.	Res.	Res.	IC10INT[7:0]							
		rw	rw					rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **IC10SEL[1:0]**: Divider IC10 source selection
This bitfield is set and reset by software to select the PLL output to feed for the channel IC10.
00: pll1_ck selected
01: pll2_ck selected (default after reset)
10: pll3_ck selected
11: pll4_ck selected

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:16 **IC10INT[7:0]**: Divider IC10 integer division factor
This bitfield is set and reset by software to control the frequency of the IC10 clock. The frequency is divided by the value IC10INT + 1. {v}: IC10 = pllx_ck / {v + 1}.
0x00: IC10 = pllx_ck (default after reset)
0x01: IC10 = pllx_ck / 2
0x02: IC10 = pllx_ck / 3
0x03: IC10 = pllx_ck / 4
...
0xFF: IC10 = pllx_ck / 256

Bits 15:0 Reserved, must be kept at reset value.

14.10.37 RCC IC11 configuration register (RCC_IC11CFGR)

Address offset: 0xEC

Reset value: 0x0003 0000

This register is used to configure the IC11 divider. It is reset by int_sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	IC11SEL[1:0]		Res.	Res.	Res.	Res.	IC11INT[7:0]							
		rw	rw					rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **IC11SEL[1:0]**: Divider IC11 source selection

This bitfield is set and reset by software to select the PLL output to feed for the channel IC11.

00: pll1_ck selected (default after reset)

01: pll2_ck selected

10: pll3_ck selected

11: pll4_ck selected

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:16 **IC11INT[7:0]**: Divider IC11 integer division factor

This bitfield is set and reset by software to control the frequency of the IC11 clock. The frequency is divided by the value IC11INT + 1: {v}: IC11 = pllx_ck / {v + 1}.

0x00: IC11 = pllx_ck

0x01: IC11 = pllx_ck / 2

0x02: IC11 = pllx_ck / 3

0x03: IC11 = pllx_ck / 4 (default after reset)

...

0xFF: IC11 = pllx_ck / 256

Bits 15:0 Reserved, must be kept at reset value.

14.10.38 RCC IC12 configuration register (RCC_IC12CFGR)

Address offset: 0xF0

Reset value: 0x2000 0000

This register is used to configure the IC12 divider. It is reset by int_sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	IC12SEL[1:0]		Res.	Res.	Res.	Res.	IC12INT[7:0]							
		rw	rw					rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **IC12SEL[1:0]**: Divider IC12 source selection
This bitfield is set and reset by software to select the PLL output to feed for the channel IC12.
00: pll1_ck selected
01: pll2_ck selected
10: pll3_ck selected (default after reset)
11: pll4_ck selected

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:16 **IC12INT[7:0]**: Divider IC12 integer division factor
This bitfield is set and reset by software to control the frequency of the IC12 clock. The frequency is divided by the value IC12INT + 1: {v}: IC12 = pllx_ck / {v + 1}.
0x00: IC12 = pllx_ck (default after reset)
0x01: IC12 = pllx_ck / 2
0x02: IC12 = pllx_ck / 3
0x03: IC12 = pllx_ck / 4
...
0xFF: IC12 = pllx_ck / 256

Bits 15:0 Reserved, must be kept at reset value.

14.10.39 RCC IC13 configuration register (RCC_IC13CFGR)

Address offset: 0xF4

Reset value: 0x2000 0000

This register is used to configure the IC13 divider. It is reset by int_sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	IC13SEL[1:0]		Res.	Res.	Res.	Res.	IC13INT[7:0]							
		rw	rw					rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **IC13SEL[1:0]**: Divider IC13 source selection
This bitfield is set and reset by software to select the PLL output to feed for the channel IC13.
00: pll1_ck selected
01: pll2_ck selected
10: pll3_ck selected (default after reset)
11: pll4_ck selected

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:16 **IC13INT[7:0]**: Divider IC13 integer division factor

This bitfield is set and reset by software to control the frequency of the IC13 clock. The frequency is divided by the value IC13INT + 1: {v}: IC13 = pllx_ck / {v + 1}.

0x00: IC13 = pllx_ck (default after reset)

0x01: IC13 = pllx_ck / 2

0x02: IC13 = pllx_ck / 3

0x03: IC13 = pllx_ck / 4

...

0xFF: IC13 = pllx_ck / 256

Bits 15:0 Reserved, must be kept at reset value.

14.10.40 RCC IC14 configuration register (RCC_IC14CFGR)

Address offset: 0xF8

Reset value: 0x2000 0000

This register is used to configure the IC14 divider. It is reset by int_sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	IC14SEL[1:0]		Res.	Res.	Res.	Res.	IC14INT[7:0]							
		rw	rw					rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **IC14SEL[1:0]**: Divider IC14 source selection

This bitfield is set and reset by software to select the PLL output to feed for the channel IC14.

00: pll1_ck selected

01: pll2_ck selected

10: pll3_ck selected (default after reset)

11: pll4_ck selected

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:16 **IC14INT[7:0]**: Divider IC14 integer division factor

This bitfield is set and reset by software to control the frequency of the IC14 clock. The frequency is divided by the value IC14INT + 1. {v}: IC14 = pllx_ck / {v + 1}.

0x00: IC14 = pllx_ck (default after reset)

0x01: IC14 = pllx_ck / 2

0x02: IC14 = pllx_ck / 3

0x03: IC14 = pllx_ck / 4

...

0xFF: IC14 = pllx_ck / 256

Bits 15:0 Reserved, must be kept at reset value.

14.10.41 RCC IC15 configuration register (RCC_IC15CFGR)

Address offset: 0xFC

Reset value: 0x2000 0000

This register is used to configure the IC15 divider. It is reset by int_sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	IC15SEL[1:0]		Res.	Res.	Res.	Res.	IC15INT[7:0]							
		rw	rw					rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **IC15SEL[1:0]**: Divider IC15 source selection

This bitfield is set and reset by software to select the PLL output to feed for the channel IC15.
00: pll1_ck selected
01: pll2_ck selected
10: pll3_ck selected (default after reset)
11: pll4_ck selected

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:16 **IC15INT[7:0]**: Divider IC15 integer division factor

This bitfield is set and reset by software to control the frequency of the IC15 clock. The frequency is divided by the value IC15INT + 1: {v}: IC15 = pllx_ck / {v + 1}.
0x00: IC15 = pllx_ck (default after reset)
0x01: IC15 = pllx_ck / 2
0x02: IC15 = pllx_ck / 3
0x03: IC15 = pllx_ck / 4
...
0xFF: IC15 = pllx_ck / 256

Bits 15:0 Reserved, must be kept at reset value.

14.10.42 RCC IC16 configuration register (RCC_IC16CFGR)

Address offset: 0x100

Reset value: 0x3000 0000

This register is used to configure the IC16 divider. It is reset by int_sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	IC16SEL[1:0]		Res.	Res.	Res.	Res.	IC16INT[7:0]							
		rw	rw					rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **IC16SEL[1:0]**: Divider IC16 source selection

This bitfield is set and reset by software to select the PLL output to feed for the channel IC16.
00: pll1_ck i selected
01: pll2_ck selected
10: pll3_ck selected
11: pll4_ck selected (default after reset)

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:16 **IC16INT[7:0]**: Divider IC16 integer division factor

This bitfield is set and reset by software to control the frequency of the IC16 clock. The frequency is divided by the value IC16INT + 1: {v}: IC16 = pllx_ck / {v + 1}.
0x00: IC16 = pllx_ck (default after reset)
0x01: IC16 = pllx_ck / 2
0x02: IC16 = pllx_ck / 3
0x03: IC16 = pllx_ck / 4

Bits 15:0 Reserved, must be kept at reset value.

14.10.43 RCC IC17 configuration register (RCC_IC17CFGR)

Address offset: 0x104

Reset value: 0x3000 0000

This register is used to configure the IC17 divider. It is reset by int_sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	IC17SEL[1:0]		Res.	Res.	Res.	Res.	IC17INT[7:0]							
		rw	rw					rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **IC17SEL[1:0]**: Divider IC17 source selection

This bitfield is set and reset by software to select the PLL output to feed for the channel IC17.
00: pll1_ck selected
01: pll2_ck selected
10: pll3_ck selected
11: pll4_ck selected (default after reset)

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:16 **IC17INT[7:0]**: Divider IC17 integer division factor

This bitfield is set and reset by software to control the frequency of the IC17 clock. The frequency is divided by the value IC17INT + 1: {v}: IC17 = pllx_ck / {v + 1}.
0x00: IC17 = pllx_ck (default after reset)
0x01: IC17 = pllx_ck / 2
0x02: IC17 = pllx_ck / 3
0x03: IC17 = pllx_ck / 4

Bits 15:0 Reserved, must be kept at reset value.

14.10.44 RCC IC18 configuration register (RCC_IC18CFGR)

Address offset: 0x108

Reset value: 0x3000 0000

This register is used to configure the IC18 divider. It is reset by int_sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	IC18SEL[1:0]		Res.	Res.	Res.	Res.	IC18INT[7:0]							
		rw	rw					rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **IC18SEL[1:0]**: Divider IC18 source selection

This bitfield is set and reset by software to select the PLL output to feed for the channel IC18.
00: pll1_ck selected
01: pll2_ck selected
10: pll3_ck selected
11: pll4_ck selected (default after reset)

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:16 **IC18INT[7:0]**: Divider IC18 integer division factor

This bitfield is set and reset by software to control the frequency of the IC18 clock. The frequency is divided by the value IC18INT + 1: {v}: IC18 = pllx_ck / {v + 1}.
0x00: IC18 = pllx_ck (default after reset)
0x01: IC18 = pllx_ck / 2
0x02: IC18 = pllx_ck / 3
0x03: IC18 = pllx_ck / 4

Bits 15:0 Reserved, must be kept at reset value.

14.10.45 RCC IC19 configuration register (RCC_IC19CFGR)

Address offset: 0x10C

Reset value: 0x3000 0000

This register is used to configure the IC19 divider. It is reset by int_sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	IC19SEL[1:0]		Res.	Res.	Res.	Res.	IC19INT[7:0]							
		rw	rw					rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **IC19SEL[1:0]**: Divider IC19 source selection

This bitfield is set and reset by software to select the PLL output to feed for the channel IC19.
00: pll1_ck selected
01: pll2_ck selected
10: pll3_ck selected
11: pll4_ck selected (default after reset)

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:16 **IC19INT[7:0]**: Divider IC19 integer division factor

This bitfield is set and reset by software to control the frequency of the IC19 clock. The frequency is divided by the value IC19INT + 1: {v}: IC19 = pllx_ck / {v + 1}.
0x00: IC19 = pllx_ck (default after reset)
0x01: IC19 = pllx_ck / 2
0x02: IC19 = pllx_ck / 3
0x03: IC19 = pllx_ck / 4

Bits 15:0 Reserved, must be kept at reset value.

14.10.46 RCC IC20 configuration register (RCC_IC20CFGR)

Address offset: 0x110

Reset value: 0x3000 0000

This register is used to configure the IC20 divider. It is reset by int_sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	IC20SEL[1:0]		Res.	Res.	Res.	Res.	IC20INT[7:0]							
		rw	rw					rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **IC20SEL[1:0]**: Divider IC20 source selection

This bitfield is set and reset by software to select the PLL output to feed for the channel IC20.
00: pll1_ck selected
01: pll2_ck selected
10: pll3_ck selected
11: pll4_ck selected (default after reset)

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:16 **IC20INT[7:0]**: Divider IC20 integer division factor

This bitfield is set and reset by software to control the frequency of the IC20 clock. The frequency is divided by the value IC20INT + 1. {v}: IC20 = pllx_ck / {v + 1}.
0x00: IC20 = pllx_ck (default after reset)
0x01: IC20 = pllx_ck / 2
0x02: IC20 = pllx_ck / 3
0x03: IC20 = pllx_ck / 4

Bits 15:0 Reserved, must be kept at reset value.

14.10.47 RCC clock-source interrupt enable register (RCC_CIER)

Address offset: 0x124

Reset value: 0x0002 0000

This register controls the enabling (unmasking) of the interrupts.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPIE	Res.	Res.	Res.	Res.	Res.	Res.	HSECSSIE	LSECSSIE
							rw							rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	PLL4RDYIE	PLL3RDYIE	PLL2RDYIE	PLL1RDYIE	Res.	Res.	Res.	HSERDYIE	HSIRDYIE	MSIRDYIE	LSERDYIE	LSIRDYIE
				rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **WKUPIE**: CPU wake-up from Stop interrupt enable

This bit is set and reset by software to enable/disable interrupt caused by CPU wake-up from Stop mode.

0: Wake-up interrupt disabled (default after reset)

1: Wake-up interrupt enabled

Bits 23:18 Reserved, must be kept at reset value.

Bit 17 **HSECSSIE**: HSE CSS interrupt enable

This bit is set and reset by software to enable/disable interrupt caused by the CSS on external 32 kHz oscillator.

0: HSE CSS interrupt disabled

1: HSE CSS interrupt enabled (default after reset)

Bit 16 **LSECSSIE**: LSE CSS interrupt enable

This bit is set and reset by software to enable/disable interrupt caused by the CSS on external 32 kHz oscillator.

0: LSE CSS interrupt disabled (default after reset)

1: LSE CSS interrupt enabled

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

Bit 11 **PLL4RDYIE**: PLL4 ready interrupt enable

This bit is set and reset by software to enable/disable interrupt caused by PLL4 lock.

0: PLL4 lock interrupt disabled (default after reset)

1: PLL4 lock interrupt enabled

Bit 10 **PLL3RDYIE**: PLL3 ready interrupt enable

This bit is set and reset by software to enable/disable interrupt caused by PLL3 lock.

0: PLL3 lock interrupt disabled (default after reset)

1: PLL3 lock interrupt enabled

Bit 9 **PLL2RDYIE**: PLL2 ready interrupt enable

This bit is set and reset by software to enable/disable interrupt caused by PLL2 lock.

0: PLL2 lock interrupt disabled (default after reset)

1: PLL2 lock interrupt enabled

Bit 8 **PLL1RDYIE**: PLL1 ready interrupt enable
This bit is set and reset by software to enable/disable interrupt caused by PLL1 lock.
0: PLL1 lock interrupt disabled (default after reset)
1: PLL1 lock interrupt enabled

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **HSERDYIE**: HSE ready interrupt enable
This bit is set and reset by software to enable/disable interrupt caused by the HSE oscillator stabilization.
0: HSE ready interrupt disabled (default after reset)
1: HSE ready interrupt enabled

Bit 3 **HSIRDYIE**: HSI ready interrupt enable
This bit is set and reset by software to enable/disable interrupt caused by the HSI oscillator stabilization.
0: HSI ready interrupt disabled (default after reset)
1: HSI ready interrupt enabled

Bit 2 **MSIRDYIE**: MSI ready interrupt enable
This bit is set and reset by software to enable/disable interrupt caused by the MSI oscillator stabilization.
0: MSI ready interrupt disabled (default after reset)
1: MSI ready interrupt enabled

Bit 1 **LSERDYIE**: LSE ready interrupt enable
This bit is set and reset by software to enable/disable interrupt caused by the LSE oscillator stabilization.
0: LSE ready interrupt disabled (default after reset)
1: LSE ready interrupt enabled

Bit 0 **LSIRDYIE**: LSI ready interrupt enable
This bit is set and reset by software to enable/disable interrupt caused by the LSI oscillator stabilization.
0: LSI ready interrupt disabled (default after reset)
1: LSI ready interrupt enabled

14.10.48 RCC clock-source interrupt flag register (RCC_CIFR)

Address offset: 0x128

Reset value: 0x0000 0000

This register returns the triggered interrupts. It is reset by sys_rstn.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPF	Res.	Res.	Res.	Res.	Res.	Res.	HSECSSF	LSECSSF
							r							r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	PLL4RDYF	PLL3RDYF	PLL2RDYF	PLL1RDYF	Res.	Res.	Res.	HSERDYF	HSIRDYF	MSIRDYF	LSERDYF	LSIRDYF
				r	r	r	r				r	r	r	r	r

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **WKUPF**: CPU wake-up from Stop interrupt flag
This bit is reset by software by writing the WKUPFC bit. It is set by hardware when the CPU needs to exit Stop mode.
0: No wake-up interrupt caused by the PWR (default after reset)
1: Wake-up interrupt caused by the PWR

Bits 23:18 Reserved, must be kept at reset value.

Bit 17 **HSECSSF**: HSE ready interrupt flag
This bit is reset by software by writing the HSECSSC bit. It is set by hardware when the HSE clock becomes stable and HSECSSIE is set.
0: No clock ready interrupt caused by the HSE (default after reset)
1: Clock ready interrupt caused by the HSE

Bit 16 **LSECSSF**: LSE ready interrupt flag
This bit is reset by software by writing the LSECSSC bit. It is set by hardware when the LSE clock becomes stable and LSECSSIE is set.
0: No clock ready interrupt caused by the LSE (default after reset)
1: Clock ready interrupt caused by the LSE

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

Bit 11 **PLL4RDYF**: PLL4 ready interrupt flag
This bit is reset by software by writing the PLL4RDYC bit. It is set by hardware when the PLL4 clock becomes stable and PLL4RDYIE is set.
0: No clock ready interrupt caused by the PLL4 (default after reset)
1: Clock ready interrupt caused by the PLL4

Bit 10 **PLL3RDYF**: PLL3 ready interrupt flag
This bit is reset by software by writing the PLL3RDYC bit. It is set by hardware when the PLL3 clock becomes stable and PLL3RDYIE is set.
0: No clock ready interrupt caused by the PLL3 (default after reset)
1: Clock ready interrupt caused by the PLL3

Bit 9 **PLL2RDYF**: PLL2 ready interrupt flag
This bit is reset by software by writing the PLL2RDYC bit. it is set by hardware when the PLL2 clock becomes stable and PLL2RDYIE is set.
0: No clock ready interrupt caused by the PLL2 (default after reset)
1: Clock ready interrupt caused by the PLL2

Bit 8 **PLL1RDYF**: PLL1 ready interrupt flag
This bit is reset by software by writing the PLL1RDYC bit. It is set by hardware when the PLL1 clock becomes stable and PLL1RDYIE is set.
0: No clock ready interrupt caused by the PLL1 (default after reset)
1: Clock ready interrupt caused by the PLL1

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **HSERDYF**: HSE ready interrupt flag
This bit is reset by software by writing the HSERDYC bit. It is set by hardware when the HSE clock becomes stable and HSERDYIE is set.
0: No clock ready interrupt caused by the HSE (default after reset)
1: Clock ready interrupt caused by the HSE

Bit 3 **HSIRDYF**: HSI ready interrupt flag

This bit is reset by software by writing the HSIRDYC bit. It is set by hardware when the HSI clock becomes stable and HSIRDYIE is set.

0: No clock ready interrupt caused by the HSI (default after reset)

1: Clock ready interrupt caused by the HSI

Bit 2 **MSIRDYF**: MSI ready interrupt flag

This bit is reset by software by writing the MSIRDYC bit. It is set by hardware when the MSI clock becomes stable and MSIRDYIE is set.

0: No clock ready interrupt caused by the MSI (default after reset)

1: Clock ready interrupt caused by the MSI

Bit 1 **LSERDYF**: LSE ready interrupt flag

This bit is reset by software by writing the LSERDYC bit. It is set by hardware when the LSE clock becomes stable and LSERDYIE is set.

0: No clock ready interrupt caused by the LSE (default after reset)

1: Clock ready interrupt caused by the LSE

Bit 0 **LSIRDYF**: LSI ready interrupt flag

This bit is reset by software by writing the LSIRDYC bit. It is set by hardware when the LSI clock becomes stable and LSIRDYIE is set.

0: No clock ready interrupt caused by the LSI (default after reset)

1: Clock ready interrupt caused by the LSI

14.10.49 RCC clock-source interrupt clear register (RCC_CICR)

Address offset: 0x12C

Reset value: 0x0000 0000

This register clears the interrupts.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPFC	Res.	Res.	Res.	Res.	Res.	Res.	HSECSSC	LSECSSC
							w							w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	PLL4RDYC	PLL3RDYC	PLL2RDYC	PLL1RDYC	Res.	Res.	Res.	HSERDYC	HSIRDYC	MSIRDYC	LSERDYC	LSIRDYC
				w	w	w	w				w	w	w	w	w

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **WKUPFC**: CPU wake-up ready interrupt clear

This bit is set by software to clear WKUPF. It is reset by hardware when clear done.

0: WKUPF not modified (default after reset)

1: WKUPF cleared

Bits 23:18 Reserved, must be kept at reset value.

Bit 17 **HSECSSC**: HSE ready interrupt clear

This bit is set by software to clear HSECSSF. it is reset by hardware when clear done.

0: HSECSSF not modified (default after reset)

1: HSECSSF cleared

Bit 16 **LSECSSC**: LSE ready interrupt clear
This bit is set by software to clear LSECSSF. It is reset by hardware when clear done.
0: LSECSSF not modified (default after reset)
1: LSECSSF cleared

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

Bit 11 **PLL4RDYC**: PLL4 ready interrupt clear
This bit is set by software to clear PLL4RDYF. It is reset by hardware when clear done.
0: PLL4RDYF not modified (default after reset)
1: PLL4RDYF cleared

Bit 10 **PLL3RDYC**: PLL3 ready interrupt clear
This bit is set by software to clear PLL3RDYF. it is reset by hardware when clear done.
0: PLL3RDYF not modified (default after reset)
1: PLL3RDYF cleared

Bit 9 **PLL2RDYC**: PLL2 ready interrupt clear
This bit is set by software to clear PLL2RDYF. It is reset by hardware when clear done.
0: PLL2RDYF not modified (default after reset)
1: PLL2RDYF cleared

Bit 8 **PLL1RDYC**: PLL1 ready interrupt clear
This bit is set by software to clear PLL1RDYF. It is reset by hardware when clear done.
0: PLL1RDYF not modified (default after reset)
1: PLL1RDYF cleared

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **HSERDYC**: HSE ready interrupt clear
This bit is set by software to clear HSERDYF. It is reset by hardware when clear done.
0: HSERDYF not modified (default after reset)
1: HSERDYF cleared

Bit 3 **HSIRDYC**: HSI ready interrupt clear
This bit is set by software to clear HSIRDYF. It is reset by hardware when clear done.
0: HSIRDYF not modified (default after reset)
1: HSIRDYF cleared

Bit 2 **MSIRDYC**: MSI ready interrupt clear
This bit is set by software to clear MSIRDYF. It is reset by hardware when clear done.
0: MSIRDYF not modified (default after reset)
1: MSIRDYF cleared

Bit 1 **LSERDYC**: LSE ready interrupt clear
This bit is set by software to clear LSERDYF. it is reset by hardware when clear done.
0: LSERDYF not modified (default after reset)
1: LSERDYF cleared

Bit 0 **LSIRDYC**: LSI ready interrupt clear
This bit is set by software to clear LSIRDYF. It is reset by hardware when clear done.
0: LSIRDYF not modified (default after reset)
1: LSIRDYF cleared

14.10.50 RCC clock configuration for independent peripheral register 1 (RCC_CCIPR1)

Address offset: 0x144

Reset value: 0x0000 0000

This register is used to configure various clocks. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCMIPPSEL[1:0]		Res.	Res.	Res.	Res.
										rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADCPRE[7:0]								Res.	ADC12SEL[2:0]			Res.	ADF1SEL[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw		rw	rw	rw

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

Bits 21:20 **DCMIPPSEL[1:0]**: Source selection for the DCMIPP kernel clock
This bitfield is set and reset by software.
00: pclk5 selected as reference clock
01: per_ck selected as reference clock
10: ic17_ck selected as reference clock
11: hsi_div_ck selected as reference clock

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:8 **ADCPRE[7:0]**: ADC12 kernel clock divider selection (for clock ck_ker_adc12)
This bitfield is set and reset by software. The division ratio is linear: {v}: ck_ker_adc12 / {v+1}.
0x00: ck_ker_adc12 divided by 1
0x01: ck_ker_adc12 divided by 2

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ADC12SEL[2:0]**: Source selection for the ADC12 kernel clock
This bitfield is set and reset by software.
000: hclk1 selected as reference clock
001: per_ck selected as reference clock
010: ic7_ck selected as reference clock
011: ic8_ck selected as reference clock
100: msi_ck selected as reference clock
101: hsi_div_ck selected as reference clock
110: I2S_CKIN selected as reference clock
111: timg_ck selected as reference clock

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **ADF1SEL[2:0]**: Source selection for the ADF1 kernel clock
This bitfield is set and reset by software.
000: hclk2 selected as reference clock
001: per_ck selected as reference clock
010: ic7_ck selected as reference clock
011: ic8_ck selected as reference clock
100: msi_ck selected as reference clock
101: hsi_div_ck selected as reference clock
110: I2S_CKIN selected as reference clock
111: timg_ck selected as reference clock

14.10.51 RCC clock configuration for independent peripheral register 2 (RCC_CCIPR2)

Address offset: 0x148

Reset value: 0x0000 0000

This register is used to configure various clocks. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETH1GTXCLKSEL	Res.	Res.	Res.	ETH1REFCLKSEL	Res.	ETH1SEL[2:0]		
							rw				rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ETH1CLKSEL[1:0]		Res.	Res.	Res.	ETH1PWRDOWNACK	ETH1PTPDIV[3:0]				Res.	Res.	ETH1PTPSEL[1:0]	
		rw	rw				r	rw	rw	rw	rw			rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **ETH1GTXCLKSEL**: Ethernet 1 reference Tx RGMII 125 MHz clock selection
This bit is set and reset by software.
0: External clock (ETH1_CLK125) is used. Need to program AFmux.
1: Internal clock from the RCC is used.

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **ETH1REFCLKSEL**: Ethernet 1 reference Rx clock selection
This bit is set and reset by software.
0: External clock (ETH_RX/REF_CLK) is used. This is RX clock for an RGMII or MII PHY, or REF clock for an RMII PHY. Need to program AFmux.
1: Internal clock (ck_ker_eth1 from RCC) is used. To be used when the RMII 50 MHz (pad ETH1_CLK) is generated to the RMII PHY.

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **ETH1SEL[2:0]**: Ethernet 1 PHY interface selection
This bitfield is set and reset by software.
Note: Apply this configuration while the ETH1 is under reset, before enabling the ETH1 clocks.
000: MII
001: RGMII
100: RMII
Others: Reserved

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

Bits 13:12 **ETH1CLKSEL[1:0]**: Source selection for the ETH1 kernel clock
This bitfield is set and reset by software.
00: hclke selected as reference clock
01: per_ck selected as reference clock
10: ic12_ck selected as reference clock
11: hse_ck selected as reference clock

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **ETH1PWRDOWNACK**: Ethernet 1 Power-down status
This bit is set and reset by software. It is asserted when the power-down sequence start has been acknowledged.
0: Power-down sequence start not yet acknowledged
1: Power-down sequence start acknowledged

Bits 7:4 **ETH1PTPDIV[3:0]**: ETH1 kernel clock divider selection (for clock ck_ker_eth1ptp)
This bitfield is set and reset by software. The division ratio is linear.
0000: ck_ker_eth1ptp divided by 1
0001: ck_ker_eth1ptp divided by 2
0010: ck_ker_eth1ptp divided by 3
0011: ck_ker_eth1ptp divided by 4
0100: ck_ker_eth1ptp divided by 5
0101: ck_ker_eth1ptp divided by 6
0110: ck_ker_eth1ptp divided by 7
0111: ck_ker_eth1ptp divided by 8
1000: ck_ker_eth1ptp divided by 9
1001: ck_ker_eth1ptp divided by 10
1010: ck_ker_eth1ptp divided by 11
1011: ck_ker_eth1ptp divided by 12
1100: ck_ker_eth1ptp divided by 13
1101: ck_ker_eth1ptp divided by 14
1110: ck_ker_eth1ptp divided by 15
1111: ck_ker_eth1ptp divided by 16

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **ETH1PTPSEL[1:0]**: Source selection for the ETH1 kernel clock
This bitfield is set and reset by software.
00: hclke selected as reference clock
01: per_ck selected as reference clock
10: ic13_ck selected as reference clock
11: hse_ck selected as reference clock

14.10.52 RCC clock configuration for independent peripheral register 3 (RCC_CCIPR3)

Address offset: 0x14C

Reset value: 0x0000 0003

This register is used to configure various clocks. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

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

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

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

Bits 5:4 **FMCSEL[1:0]**: Source selection for the FMC kernel clock
This bitfield is set and reset by software.
00: hclk5 selected as reference clock
01: per_ck selected as reference clock
10: ic3_ck selected as reference clock
11: ic4_ck selected as reference clock
Others: Reserved, no clock provided as reference clock

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **FDCANSEL[1:0]**: Source selection for the FDCAN kernel clock
This bitfield is set and reset by software.
00: pclk1 selected as reference clock
01: per_ck selected as reference clock
10: ic19_ck selected as reference clock
11: hse_ck selected as reference clock
Others: Reserved, no clock provided as reference clock

14.10.53 RCC clock configuration for independent peripheral register 4 (RCC_CCIPR4)

Address offset: 0x150

Reset value: 0x0000 0000

This register is used to configure various clocks. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	LTDCSEL[1:0]		Res.	I3C2SEL[2:0]			Res.	I3C1SEL[2:0]		
						rw	rw		rw	rw	rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	I2C4SEL[2:0]			Res.	I2C3SEL[2:0]			Res.	I2C2SEL[2:0]			Res.	I2C1SEL[2:0]		
	rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw	rw

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

Bits 25:24 **LTDCSEL[1:0]**: Source selection for the LTDC kernel clock
This bitfield is set and reset by software.
00: pclk5 selected as reference clock
01: per_ck selected as reference clock
10: ic16_ck selected as reference clock
11: hsi_div_ck selected as reference clock
Others: Reserved, no clock provided as reference clock

Bit 23 Reserved, must be kept at reset value.

Bits 22:20 **I3C2SEL[2:0]**: Source selection for the I3C2 kernel clock
This bitfield is set and reset by software.
000: pclk1 selected as reference clock
001: per_ck selected as reference clock
010: ic10_ck selected as reference clock
011: ic15_ck selected as reference clock
100: msi_ck selected as reference clock
101: hsi_div_ck selected as reference clock
Others: Reserved, no clock provided as reference clock

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **I3C1SEL[2:0]**: Source selection for the I3C1 kernel clock
This bitfield is set and reset by software.
000: pclk1 selected as reference clock
001: per_ck selected as reference clock
010: ic10_ck selected as reference clock
011: ic15_ck selected as reference clock
100: msi_ck selected as reference clock
101: hsi_div_ck selected as reference clock
Others: Reserved, no clock provided as reference clock

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **I2C4SEL[2:0]**: Source selection for the I2C4 kernel clock
This bitfield is set and reset by software.
000: pclk1 selected as reference clock
001: per_ck selected as reference clock
010: ic10_ck selected as reference clock
011: ic15_ck selected as reference clock
100: msi_ck selected as reference clock
101: hsi_div_ck selected as reference clock
Others: Reserved, no clock provided as reference clock

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **I2C3SEL[2:0]**: Source selection for the I2C3 kernel clock
This bitfield is set and reset by software.
000: pclk1 selected as reference clock
001: per_ck selected as reference clock
010: ic10_ck selected as reference clock
011: ic15_ck selected as reference clock
100: msi_ck selected as reference clock
101: hsi_div_ck selected as reference clock
Others: Reserved, no clock provided as reference clock

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **I2C2SEL[2:0]**: Source selection for the I2C2 kernel clock
This bitfield is set and reset by software.
000: pclk1 selected as reference clock
001: per_ck selected as reference clock
010: ic10_ck selected as reference clock
011: ic15_ck selected as reference clock
100: msi_ck selected as reference clock
101: hsi_div_ck selected as reference clock
Others: Reserved, no clock provided as reference clock

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **I2C1SEL[2:0]**: Source selection for the I2C1 kernel clock
This bitfield is set and reset by software.
000: pclk1 selected as reference clock
001: per_ck selected as reference clock
010: ic10_ck selected as reference clock
011: ic15_ck selected as reference clock
100: msi_ck selected as reference clock
101: hsi_div_ck selected as reference clock
Others: Reserved, no clock provided as reference clock

14.10.54 RCC clock configuration for independent peripheral register 5 (RCC_CCIPR5)

Address offset: 0x154

Reset value: 0x0000 F0F0

This register is used to configure various clocks. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MDF1SEL[2:0]		
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCO2PRE[3:0]				Res.	MCO2SEL[2:0]			MCO1PRE[3:0]				Res.	MCO1SEL[2:0]		
rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **MDF1SEL[2:0]**: Source selection for the MDF1 kernel clock
This bitfield is set and reset by software.
000: hclk2 selected as reference clock
001: per_ck selected as reference clock
010: ic7_ck selected as reference clock
011: ic8_ck selected as reference clock
100: msi_ck selected as reference clock
101: hsi_div_ck selected as reference clock
110: I2S_CKIN selected as reference clock
111: timg_ck selected as reference clock

Bits 15:12 **MCO2PRE[3:0]**: MCO2 Kernel clock divider selection (for clock MCO2)
This bitfield is set and reset by software. The division ratio is linear.
0000: MCO2 divided by 1
0001: MCO2 divided by 2
0010: MCO2 divided by 3
0011: MCO2 divided by 4
0100: MCO2 divided by 5
0101: MCO2 divided by 6
0110: MCO2 divided by 7
0111: MCO2 divided by 8
1000: MCO2 divided by 9
1001: MCO2 divided by 10
1010: MCO2 divided by 11
1011: MCO2 divided by 12
1100: MCO2 divided by 13
1101: MCO2 divided by 14
1110: MCO2 divided by 15
1111: MCO2 divided by 16

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **MCO2SEL[2:0]**: Source selection for the MCO2 kernel clock
This bitfield is set and reset by software.
000: hsi_div_ck selected as reference clock (default after reset)
001: lse_ck selected as reference clock
010: msi_ck selected as reference clock
011: lsi_ck selected as reference clock
100: hse_ck selected as reference clock
101: ic15_ck selected as reference clock
110: ic20_ck selected as reference clock
111: sysb_ck selected as reference clock

Bits 7:4 **MCO1PRE[3:0]**: MCO1 Kernel clock divider selection (for clock MCO1)
This bitfield is set and reset by software. The division ratio is linear.
0000: MCO1 divided by 1
0001: MCO1 divided by 2
0010: MCO1 divided by 3
0011: MCO1 divided by 4
0100: MCO1 divided by 5
0101: MCO1 divided by 6
0110: MCO1 divided by 7
0111: MCO1 divided by 8
1000: MCO1 divided by 9
1001: MCO1 divided by 10
1010: MCO1 divided by 11
1011: MCO1 divided by 12
1100: MCO1 divided by 13
1101: MCO1 divided by 14
1110: MCO1 divided by 15
1111: MCO1 divided by 16

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **MCO1SEL[2:0]**: Source selection for the MCO1 kernel clock
This bitfield is set and reset by software.
000: hsi_div_ck selected as reference clock (default after reset)
001: lse_ck selected as reference clock
010: msi_ck selected as reference clock
011: lsi_ck selected as reference clock
100: hse_ck selected as reference clock
101: ic5_ck selected as reference clock
110: ic10_ck selected as reference clock
111: sysa_ck selected as reference clock

14.10.55 RCC clock configuration for independent peripheral register 6 (RCC_CCIPR6)

Address offset: 0x158

Reset value: 0x0000 0000

This register is used to configure various clocks. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTG PHY2CK REF SEL	Res.	Res.	OTGPHY2 SEL[1:0]		Res.	Res.	Res.	OTG PHY1CK REF SEL
							rw			rw	rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	OTGPHY1SEL[1:0]		Res.	Res.	XSPI3SEL[1:0]		Res.	Res.	XSPI2SEL[1:0]		Res.	Res.	XSPI1SEL[1:0]	
		rw	rw			rw	rw			rw	rw			rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **OTGPHY2CKREFSEL**:
This bitfield is set and reset by software.
0: otgphy2_ker_ck selected
1: hse_div2_osc_ck selected

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **OTGPHY2SEL[1:0]**: Source selection for the OTGPHY2 kernel clock
This bitfield is set and reset by software.
00: hse_div2_ck selected as reference clock
01: per_ck selected as reference clock
10: ic15_ck selected as reference clock
11: hse_div2_osc_ck selected as reference clock
Others: Reserved, no clock provided as reference clock

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **OTGPHY1CKREFSEL**:
This bitfield is set and reset by software.
0: otgphy1_ker_ck selected
1: hse_div2_osc_ck selected

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

Bits 13:12 **OTGPHY1SEL[1:0]**: Source selection for the OTGPHY1 kernel clock
This bitfield is set and reset by software.
00: hse_div2_ck selected as reference clock
01: per_ck selected as reference clock
10: ic15_ck selected as reference clock
11: hse_div2_osc_ck selected as reference clock
Others: Reserved, no clock provided as reference clock

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:8 **XSPI3SEL[1:0]**: Source selection for the XSPI3 kernel clock
This bitfield is set and reset by software.
00: hclk5 selected as reference clock
01: per_ck selected as reference clock
10: ic3_ck selected as reference clock
11: ic4_ck selected as reference clock
Others: Reserved, no clock provided as reference clock

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **XSPI2SEL[1:0]**: Source selection for the XSPI2 kernel clock
This bitfield is set and reset by software.
00: hclk5 selected as reference clock
01: per_ck selected as reference clock
10: ic3_ck selected as reference clock
11: ic4_ck selected as reference clock
Others: Reserved, no clock provided as reference clock

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **XSPI1SEL[1:0]**: Source selection for the XSPI1 kernel clock
This bitfield is set and reset by software.
00: hclk5 selected as reference clock
01: per_ck selected as reference clock
10: ic3_ck selected as reference clock
11: ic4_ck selected as reference clock
Others: Reserved, no clock provided as reference clock

14.10.56 RCC clock configuration for independent peripheral register 7 (RCC_CCIPR7)

Address offset: 0x15C

Reset value: 0x0000 0200

This register is used to configure various clocks. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	SAI2SEL[2:0]			Res.	SAI1SEL[2:0]			Res.	Res.	RTCPRE[5:4]	
					rw	rw	rw		rw	rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTCPRE[3:0]				Res.	Res.	RTCSEL[1:0]		Res.	Res.	PSSISEL[1:0]		Res.	PERSEL[2:0]		
rw	rw	rw	rw			rw	rw			rw	rw		rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:24 **SAI2SEL[2:0]**: Source selection for the SAI2 kernel clock
This bitfield is set and reset by software.
000: pclk2 selected as reference clock
001: per_ck selected as reference clock
010: ic7_ck selected as reference clock
011: ic8_ck selected as reference clock
100: msi_ck selected as reference clock
101: hsi_div_ck selected as reference clock
110: I2S_CKIN selected as reference clock
111: spdif_symb_ck selected as reference clock

Bit 23 Reserved, must be kept at reset value.

Bits 22:20 **SAI1SEL[2:0]**: Source selection for the SAI1 kernel clock
This bitfield is set and reset by software.
000: pclk2 selected as reference clock
001: per_ck selected as reference clock
010: ic7_ck selected as reference clock
011: ic8_ck selected as reference clock
100: msi_ck selected as reference clock
101: hsi_div_ck selected as reference clock
110: I2S_CKIN selected as reference clock
111: spdif_symb_ck selected as reference clock

Bits 19:18 Reserved, must be kept at reset value.

Bits 17:12 **RTCPRE[5:0]**: RTC OSC clock divider selection (for clock hse_ck)
This bitfield is set and reset by software. The division ratio is linear: {v}: hse_ck / {v+1}.
0x00: hse_ck divided by 1
0x01: hse_ck divided by 2

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:8 **RTCSEL[1:0]**: Source selection for the RTC kernel clock
This bitfield is write-protected by the pwr_lock_backup_n signal. It is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit.
This bitfield is set and reset by software.
01: lse_ck selected as reference clock
10: lsi_ck selected as reference clock
11: hse_rtc_ck selected as reference clock
Others: Reserved, no clock provided as reference clock

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **PSSISEL[1:0]**: Source selection for the PSSI kernel clock
This bitfield is set and reset by software.
00: hclk5 selected as reference clock
01: per_ck selected as reference clock
10: ic20_ck selected as reference clock
11: hsi_div_ck selected as reference clock
Others: Reserved, no clock provided as reference clock

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **PERSEL[2:0]**: Source selection for the PER kernel clock
This bitfield is set and reset by software.
000: hsi_ck selected as reference clock
001: msi_ck selected as reference clock
010: hse_ck selected as reference clock
011: ic19_ck selected as reference clock
100: ic5_ck selected as reference clock
101: ic10_ck selected as reference clock
110: ic15_ck selected as reference clock
111: ic20_ck selected as reference clock

14.10.57 RCC clock configuration for independent peripheral register 8 (RCC_CCIPR8)

Address offset: 0x160

Reset value: 0x0000 0000

This register is used to configure various clocks. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC2SEL [1:0]		Res.	Res.	SDMMC1SEL [1:0]	
										rw	rw			rw	rw

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

Bits 5:4 **SDMMC2SEL[1:0]**: Source selection for the SDMMC2 kernel clock
This bitfield is set and reset by software.
00: hclku selected as reference clock
01: per_ck selected as reference clock
10: ic4_ck selected as reference clock
11: ic5_ck selected as reference clock
Others: Reserved, no clock provided as reference clock

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **SDMMC1SEL[1:0]**: Source selection for the SDMMC1 kernel clock
This bitfield is set and reset by software.
00: hclku selected as reference clock
01: per_ck selected as reference clock
10: ic4_ck selected as reference clock
11: ic5_ck selected as reference clock
Others: Reserved, no clock provided as reference clock

14.10.58 RCC clock configuration for independent peripheral register 9 (RCC_CCIPR9)

Address offset: 0x164

Reset value: 0x0000 0000

This register is used to configure various clocks. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	SPI6SEL[2:0]			Res.	SPI5SEL[2:0]			Res.	SPI4SEL[2:0]		
					rw	rw	rw		rw	rw	rw		rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SPI3SEL[2:0]			Res.	SPI2SEL[2:0]			Res.	SPI1SEL[2:0]			Res.	SPDIFRX1SEL[2:0]		
	rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:24 **SPI6SEL[2:0]**: Source selection for the SPI6 kernel clock
This bitfield is set and reset by software.
000: pclk4 selected as reference clock
001: per_ck selected as reference clock
010: ic8_ck selected as reference clock
011: ic9_ck selected as reference clock
100: msi_ck selected as reference clock
101: hsi_div_ck selected as reference clock
110: I2S_CKIN selected as reference clock
Others: Reserved, no clock provided as reference clock

Bit 23 Reserved, must be kept at reset value.

Bits 22:20 **SPI5SEL[2:0]**: Source selection for the SPI5 kernel clock
This bitfield is set and reset by software.
000: pclk2 selected as reference clock
001: per_ck selected as reference clock
010: ic9_ck selected as reference clock
011: ic14_ck selected as reference clock
100: msi_ck selected as reference clock
101: hsi_div_ck selected as reference clock
110: hse_ck selected as reference clock
Others: Reserved, no clock provided as reference clock

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **SPI4SEL[2:0]**: Source selection for the SPI4 kernel clock
This bitfield is set and reset by software.
000: pclk2 selected as reference clock
001: per_ck selected as reference clock
010: ic9_ck selected as reference clock
011: ic14_ck selected as reference clock
100: msi_ck selected as reference clock
101: hsi_div_ck selected as reference clock
110: hse_ck selected as reference clock
Others: Reserved, no clock provided as reference clock

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **SPI3SEL[2:0]**: Source selection for the SPI3 kernel clock
This bitfield is set and reset by software.
000: pclk1 selected as reference clock
001: per_ck selected as reference clock
010: ic8_ck selected as reference clock
011: ic9_ck selected as reference clock
100: msi_ck selected as reference clock
101: hsi_div_ck selected as reference clock
110: I2S_CKIN selected as reference clock
Others: Reserved, no clock provided as reference clock

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **SPI2SEL[2:0]**: Source selection for the SPI2 kernel clock
This bitfield is set and reset by software.
000: pclk1 selected as reference clock
001: per_ck selected as reference clock
010: ic8_ck selected as reference clock
011: ic9_ck selected as reference clock
100: msi_ck selected as reference clock
101: hsi_div_ck selected as reference clock
110: I2S_CKIN selected as reference clock
Others: Reserved, no clock provided as reference clock

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **SPI1SEL[2:0]**: Source selection for the SPI1 kernel clock
This bitfield is set and reset by software.
000: pclk2 selected as reference clock
001: per_ck selected as reference clock
010: ic8_ck selected as reference clock
011: ic9_ck selected as reference clock
100: msi_ck selected as reference clock
101: hsi_div_ck selected as reference clock
110: I2S_CKIN selected as reference clock
Others: Reserved, no clock provided as reference clock

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **SPDIFRX1SEL[2:0]**: Source selection for the SPDIFRX1 kernel clock
This bitfield is set and reset by software.
000: pclk1 selected as reference clock
001: per_ck selected as reference clock
010: ic7_ck selected as reference clock
011: ic8_ck selected as reference clock
100: msi_ck selected as reference clock
101: hsi_div_ck selected as reference clock
110: I2S_CKIN selected as reference clock
Others: Reserved, no clock provided as reference clock

14.10.59 RCC clock configuration for independent peripheral register 12 (RCC_CCIPR12)

Address offset: 0x170

Reset value: 0x0000 0000

This register is used to configure various clocks. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	LPTIM5SEL[2:0]			Res.	LPTIM4SEL[2:0]			Res.	LPTIM3SEL[2:0]		
					rw	rw	rw		rw	rw	rw		rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LPTIM2SEL[2:0]			Res.	LPTIM1SEL[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw	rw	rw		rw	rw	rw								

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:24 **LPTIM5SEL[2:0]**: Source selection for the LPTIM5 kernel clock
This bitfield is set and reset by software.
000: pclk4 selected as reference clock
001: per_ck selected as reference clock
010: ic15_ck selected as reference clock
011: lse_ck selected as reference clock
100: lsi_ck selected as reference clock
101: timg_ck selected as reference clock
Others: Reserved, no clock provided as reference clock

Bit 23 Reserved, must be kept at reset value.

Bits 22:20 **LPTIM4SEL[2:0]**: Source selection for the LPTIM4 kernel clock
This bitfield is set and reset by software.
000: pclk4 selected as reference clock
001: per_ck selected as reference clock
010: ic15_ck selected as reference clock
011: lse_ck selected as reference clock
100: lsi_ck selected as reference clock
101: timg_ck selected as reference clock
Others: Reserved, no clock provided as reference clock

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **LPTIM3SEL[2:0]**: Source selection for the LPTIM3 kernel clock
This bitfield is set and reset by software.
000: pclk4 selected as reference clock
001: per_ck selected as reference clock
010: ic15_ck selected as reference clock
011: lse_ck selected as reference clock
100: lsi_ck selected as reference clock
101: timg_ck selected as reference clock
Others: Reserved, no clock provided as reference clock

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **LPTIM2SEL[2:0]**: Source selection for the LPTIM2 kernel clock
This bitfield is set and reset by software.
000: pclk4 selected as reference clock
001: per_ck selected as reference clock
010: ic15_ck selected as reference clock
011: lse_ck selected as reference clock
100: lsi_ck selected as reference clock
101: timg_ck selected as reference clock
Others: Reserved, no clock provided as reference clock

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **LPTIM1SEL[2:0]**: Source selection for the LPTIM1 kernel clock
This bitfield is set and reset by software.
000: pclk1 selected as reference clock
001: per_ck selected as reference clock
010: ic15_ck selected as reference clock
011: lse_ck selected as reference clock
100: lsi_ck selected as reference clock
101: timg_ck selected as reference clock
Others: Reserved, no clock provided as reference clock

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

14.10.60 RCC clock configuration for independent peripheral register 13 (RCC_CCIPR13)

Address offset: 0x174

Reset value: 0x0000 0000

This register is used to configure various clocks. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	UART8SEL[2:0]			Res.	UART7SEL[2:0]			Res.	USART6SEL[2:0]			Res.	UART5SEL[2:0]		
	rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	UART4SEL[2:0]			Res.	USART3SEL[2:0]			Res.	USART2SEL[2:0]			Res.	USART1SEL[2:0]		
	rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **UART8SEL[2:0]**: Source selection for the UART8 kernel clock
This bitfield is set and reset by software.
000: pclk1 selected as reference clock
001: per_ck selected as reference clock
010: ic9_ck selected as reference clock
011: ic14_ck selected as reference clock
100: lse_ck selected as reference clock
101: msi_ck selected as reference clock
110: hsi_div_ck selected as reference clock
Others: Reserved, no clock provided as reference clock

Bit 27 Reserved, must be kept at reset value.

Bits 26:24 **UART7SEL[2:0]**: Source selection for the UART7 kernel clock
This bitfield is set and reset by software.
000: pclk1 selected as reference clock
001: per_ck selected as reference clock
010: ic9_ck selected as reference clock
011: ic14_ck selected as reference clock
100: lse_ck selected as reference clock
101: msi_ck selected as reference clock
110: hsi_div_ck selected as reference clock
Others: Reserved, no clock provided as reference clock

Bit 23 Reserved, must be kept at reset value.

Bits 22:20 **USART6SEL[2:0]**: Source selection for the USART6 kernel clock
This bitfield is set and reset by software.
000: pclk2 selected as reference clock
001: per_ck selected as reference clock
010: ic9_ck selected as reference clock
011: ic14_ck selected as reference clock
100: lse_ck selected as reference clock
101: msi_ck selected as reference clock
110: hsi_div_ck selected as reference clock
Others: Reserved, no clock provided as reference clock

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **UART5SEL[2:0]**: Source selection for the UART5 kernel clock
This bitfield is set and reset by software.
000: pclk1 selected as reference clock
001: per_ck selected as reference clock
010: ic9_ck selected as reference clock
011: ic14_ck selected as reference clock
100: lse_ck selected as reference clock
101: msi_ck selected as reference clock
110: hsi_div_ck selected as reference clock
Others: Reserved, no clock provided as reference clock

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **UART4SEL[2:0]**: Source selection for the UART4 kernel clock
This bitfield is set and reset by software.
000: pclk1 selected as reference clock
001: per_ck selected as reference clock
010: ic9_ck selected as reference clock
011: ic14_ck selected as reference clock
100: lse_ck selected as reference clock
101: msi_ck selected as reference clock
110: hsi_div_ck selected as reference clock
Others: Reserved, no clock provided as reference clock

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **USART3SEL[2:0]**: Source selection for the USART3 kernel clock
This bitfield is set and reset by software.
000: pclk1 selected as reference clock
001: per_ck selected as reference clock
010: ic9_ck selected as reference clock
011: ic14_ck selected as reference clock
100: lse_ck selected as reference clock
101: msi_ck selected as reference clock
110: hsi_div_ck selected as reference clock
Others: Reserved, no clock provided as reference clock

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **USART2SEL[2:0]**: Source selection for the USART2 kernel clock
This bitfield is set and reset by software.
000: pclk1 selected as reference clock
001: per_ck selected as reference clock
010: ic9_ck selected as reference clock
011: ic14_ck selected as reference clock
100: lse_ck selected as reference clock
101: msi_ck selected as reference clock
110: hsi_div_ck selected as reference clock
Others: Reserved, no clock provided as reference clock

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **USART1SEL[2:0]**: Source selection for the USART1 kernel clock
This bitfield is set and reset by software.
000: pclk2 selected as reference clock
001: per_ck selected as reference clock
010: ic9_ck selected as reference clock
011: ic14_ck selected as reference clock
100: lse_ck selected as reference clock
101: msi_ck selected as reference clock
110: hsi_div_ck selected as reference clock
Others: Reserved, no clock provided as reference clock

14.10.61 RCC clock configuration for independent peripheral register 14 (RCC_CCIPR14)

Address offset: 0x178

Reset value: 0x0000 0000

This register is used to configure various clocks. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	LPUART1SEL[2:0]			Res.	USART10SEL[2:0]			Res.	UART9SEL[2:0]		
					rw	rw	rw		rw	rw	rw		rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:8 **LPUART1SEL[2:0]**: Source selection for the LPUART1 kernel clock
This bitfield is set and reset by software.
000: pclk4 selected as reference clock
001: per_ck selected as reference clock
010: ic9_ck selected as reference clock
011: ic14_ck selected as reference clock
100: lse_ck selected as reference clock
101: msi_ck selected as reference clock
110: hsi_div_ck selected as reference clock
Others: Reserved, no clock provided as reference clock

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **USART10SEL[2:0]**: Source selection for the USART10 kernel clock
This bitfield is set and reset by software.
000: pclk2 selected as reference clock
001: per_ck selected as reference clock
010: ic9_ck selected as reference clock
011: ic14_ck selected as reference clock
100: lse_ck selected as reference clock
101: msi_ck selected as reference clock
110: hsi_div_ck selected as reference clock
Others: Reserved, no clock provided as reference clock

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **UART9SEL[2:0]**: Source selection for the UART9 kernel clock

This bitfield is set and reset by software.
000: pclk2 selected as reference clock
001: per_ck selected as reference clock
010: ic9_ck selected as reference clock
011: ic14_ck selected as reference clock
100: lse_ck selected as reference clock
101: msi_ck selected as reference clock
110: hsi_div_ck selected as reference clock
Others: Reserved, no clock provided as reference clock

14.10.62 RCC miscellaneous configurations reset register (RCC_MISCRSTR)

Address offset: 0x208

Reset value: 0x0000 0000

This register is used to reset the miscellaneous configurations. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC2DLLRST	SDMMC1DLLRST	Res.	XSPIPHY2RST	XSPIPHY1RST	Res.	Res.	Res.	DBGRST
							rw	rw		rw	rw				rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **SDMMC2DLLRST**: SDMMC2DLL reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with SDMMC2DLLRSTS, and cleared with SDMMC2DLLRSTC. This bit is set and reset by software.
0: SDMMC2DLL not under reset (default after reset)
1: SDMMC2DLL under reset

Bit 7 **SDMMC1DLLRST**: SDMMC1DLL reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with SDMMC1DLLRSTS, and cleared with SDMMC1DLLRSTC. This bit is set and reset by software.
0: SDMMC1DLL not under reset (default after reset)
1: SDMMC1DLL under reset

Bit 6 Reserved, must be kept at reset value.

Bit 5 **XSPIPHY2RST**: XSPIPHY2 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with XSPIPHY2RST,S and cleared with XSPIPHY2RSTC. This bit is set and reset by software.
0: XSPIPHY2 not under reset (default after reset)
1: XSPIPHY2 under reset

Bit 4 **XSPIPHY1RST**: XSPIPHY1 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with XSPIPHY1RSTS, and cleared with XSPIPHY1RSTC. This bit is set and reset by software.

0: XSPIPHY1 not under reset (default after reset)

1: XSPIPHY1 under reset

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **DBGRST**: DBG reset

This bit is always security-protected. It can be set with DBGRSTS, and cleared with DBGRSTC. This bit is set and reset by software.

0: DBG not under reset (default after reset)

1: DBG under reset

14.10.63 RCC embedded memories reset register (RCC_MEMRSTR)

Address offset: 0x20C

Reset value: 0x0000 0000

This register is used to reset the embedded memories. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	BOOTROMRST	VENCRAMRST	CACHEAXIRAMRST	FLEXRAMRST	AXISRAM2RST	AXISRAM1RST	Res.	AHBSRAM2RST	AHBSRAM1RST	AXISRAM6RST	AXISRAM5RST	AXISRAM4RST	AXISRAM3RST
			rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **BOOTROMRST**: BootROM reset

This bit is always security-protected. It can be set with BOOTROMRSTS, and cleared with BOOTROMRSTC. This bit is set and reset by software.

0: BootROM not under reset (default after reset)

1: BootROM under reset

Bit 11 **VENCRAMRST**: VENCRAM reset

This bit is security-protected by a SEC or a PRIV signal from RIFSC, and is publicly readable if VENCRAMPUB = 1. It can be set with VENCRAMRSTS, and cleared with VENCRAMRSTC. This bit is set and reset by software.

0: VENCRAM not under reset (default after reset)

1: VENCRAM under reset

Bit 10 **CACHEAXIRAMRST**: CACHEAXIRAM reset

This bit is security-protected by a SEC or a PRIV signal from RIFSC, and is publicly readable if CACHEAXIRAMPUB = 1. It can be set with CACHEAXIRAMRSTS, and cleared with CACHEAXIRAMRSTC. This bit is set and reset by software.

0: CACHEAXIRAM not under reset (default after reset)

1: CACHEAXIRAM under reset

Bit 9 **FLEXRAMRST**: FLEXRAM reset

This bit is security-protected by a SEC or a PRIV signal from RIFSC, and is publicly readable if FLEXRAMPUB = 1. It can be set with FLEXRAMRSTS, and cleared with FLEXRAMRSTC. This bit is set and reset by software.

0: FLEXRAM not under reset (default after reset)

1: FLEXRAM under reset

Bit 8 **AXISRAM2RST**: AXISRAM2 reset

This bit is security-protected by a SEC or a PRIV signal from RIFSC, and is publicly readable if AXISRAM2PUB = 1. it can be set with AXISRAM2RSTS, and cleared with AXISRAM2RSTC. This bit is set and reset by software.

0: AXISRAM2 s not under reset (default after reset)

1: AXISRAM2 under reset

Bit 7 **AXISRAM1RST**: AXISRAM1 reset

This bit is security-protected by a SEC or a PRIV signal from RIFSC, and is publicly readable if AXISRAM1PUB = 1. It can be set with AXISRAM1RSTS, and cleared with AXISRAM1RSTC. This bit is set and reset by software.

0: AXISRAM1 not under reset (default after reset)

1: AXISRAM1 under reset

Bit 6 Reserved, must be kept at reset value.

Bit 5 **AHBSRAM2RST**: AHBSRAM2 reset

This bit is security-protected by a SEC or a PRIV signal from RIFSC, and is publicly readable if AHBSRAM2PUB = 1. It can be set with AHBSRAM2RSTS, and cleared with AHBSRAM2RSTC. This bit is set and reset by software.

0: AHBSRAM2 not under reset (default after reset)

1: AHBSRAM2 under reset

Bit 4 **AHBSRAM1RST**: AHBSRAM1 reset

This bit is security-protected by a SEC or a PRIV signal from RIFSC, and is publicly readable if AHBSRAM1PUB = 1. It can be set with AHBSRAM1RSTS, and cleared with AHBSRAM1RSTC. This bit is set and reset by software.

0: AHBSRAM1 not under reset (default after reset)

1: AHBSRAM1 under reset

Bit 3 **AXISRAM6RST**: AXISRAM6 reset

This bit is security-protected by a SEC or a PRIV signal from RIFSC, and is publicly readable if AXISRAM6PUB = 1. It can be set with AXISRAM6RSTS, and cleared with AXISRAM6RSTC. This bit is set and reset by software.

0: AXISRAM6 not under reset (default after reset)

1: AXISRAM6 under reset

Bit 2 **AXISRAM5RST**: AXISRAM5 reset

This bit is security-protected by a SEC or a PRIV signal from RIFSC, and is publicly readable if AXISRAM5PUB = 1. It can be set with AXISRAM5RSTS, and cleared with AXISRAM5RSTC. This bit is set and reset by software.

0: AXISRAM5 not under reset (default after reset)

1: AXISRAM5 under reset

Bit 1 **AXISRAM4RST**: AXISRAM4 reset

This bit is security-protected by a SEC or a PRIV signal from RIFSC, and is publicly readable if AXISRAM4PUB = 1. It can be set with AXISRAM4RSTS, and cleared with AXISRAM4RSTC. This bit is set and reset by software.

0: AXISRAM4 not under reset (default after reset)

1: AXISRAM4 under reset

Bit 0 **AXISRAM3RST**: AXISRAM3 reset

This bit is security-protected by a SEC or a PRIV signal from RIFSC, and is publicly readable if AXISRAM3PUB = 1. It can be set with AXISRAM3RSTS, and cleared with AXISRAM3RSTC. This bit is set and reset by software.

0: AXISRAM3 not under reset (default after reset)

1: AXISRAM3 under reset

14.10.64 RCC AHB1 reset register (RCC_AHB1RSTR)

Address offset: 0x210

Reset value: 0x0000 0000

This register is used to reset the AHB1. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC12RST	GPDMA1RST	Res.	Res.	Res.	Res.
										rw	rw				

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

Bit 5 **ADC12RST**: ADC12 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. it can be set with ADC12RSTS and cleared with ADC12RSTC. This bit is set and reset by software.

0: ADC12 not under reset (default after reset)

1: ADC12 under reset

Bit 4 **GPDMA1RST**: GPDMA1 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with GPDMA1RSTS and cleared with GPDMA1RSTC. This bit is set and reset by software.

0: GPDMA1 not under reset (default after reset)

1: GPDMA1 under reset

Bits 3:0 Reserved, must be kept at reset value.

14.10.65 RCC AHB2 reset register (RCC_AHB2RSTR)

Address offset: 0x214

Reset value: 0x0000 0000

This register is used to reset the AHB2. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADF1RST	MDF1RST
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	RAMCFGRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			rw												

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

Bit 17 **ADF1RST**: ADF1 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with ADF1RSTS and cleared with ADF1RSTC. This bit is set and reset by software.

0: ADF1 not under reset (default after reset)

1: ADF1 under reset

Bit 16 **MDF1RST**: MDF1 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with MDF1RSTS and cleared with MDF1RSTC. This bit is set and reset by software.

0: MDF1 not under reset (default after reset)

1: MDF1 under reset

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **RAMCFGRST**: RAMCFG reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with RAMCFGRSTS and cleared with RAMCFGRSTC. This bit is set and reset by software.

0: RAMCFG not under reset (default after reset)

1: RAMCFG under reset

Bits 11:0 Reserved, must be kept at reset value.

14.10.66 RCC AHB3 reset register (RCC_AHB3RSTR)

Address offset: 0x218

Reset value: 0x0000 0000

This register is used to reset the AHB3. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

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

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **IACRST**: IAC reset

This bit is always security-protected. It can be set with IACRSTS, and cleared with IACRSTC. This bit is set and reset by software.

0: IAC not under reset (default after reset)

1: IAC under reset

Bit 9 Reserved, must be kept at reset value.

Bit 8 **PKARST**: PKA reset

This bit is security-protected by the SYSSEC bit, the SYSPRIV bit. It can be set with PKARSTS, and cleared with PKARSTC. This bit is set and reset by software.

0: PKA not under reset (default after reset)

1: PKA under reset

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **SAESRST**: SAES reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with SAESRSTS, and cleared with SAESRSTC. This bit is set and reset by software.

0: SAES not under reset (default after reset)

1: SAES under reset

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CRYPRST**: CRYP reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with CRYPRSTS, and cleared with CRYPRSTC. This bit is set and reset by software.

0: CRYP not under reset (default after reset)

1: CRYP under reset

Bit 1 **HASHRST**: HASH reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with HASHRSTS, and cleared with HASHRSTC. This bit is set and reset by software.

0: HASH not under reset (default after reset)

1: HASH under reset

Bit 0 **RNGRST**: RNG reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with RNGRSTS, and cleared with RNGRSTC. This bit is set and reset by software.

0: RNG not under reset (default after reset)

1: RNG under reset

14.10.67 RCC AHB4 reset register (RCC_AHB4RSTR)

Address offset: 0x21C

Reset value: 0x0000 0000

This register is used to reset the AHB4. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCRST	PWRRST	Res.	GPIOQRST
												rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIOPRST	GPIOORST	GPIONRST	Res.	Res.	Res.	Res.	Res.	GPIOHRST	GPIOGRST	GPIOFRST	GPIOERST	GPIODRST	GPIOCRST	GPIOBRST	GPIOARST
rw	rw	rw						rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **CRCRST**: CRC reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with CRCRST,S and cleared with CRCRSTC.This bit is set and reset by software.

0: CRC not under reset (default after reset)

1: CRC under reset

Bit 18 **PWRRST**: PWR reset

This bit is always security-protected. It can be set with PWRRSTS, and cleared with PWRRSTC. This bit is set and reset by software.

0: PWR not under reset (default after reset)

1: PWR under reset

Bit 17 Reserved, must be kept at reset value.

Bit 16 **GPIOQRST**: GPIO Q reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with GPIOQRSTS, and cleared with GPIOQRSTC. This bit is set and reset by software.

0: GPIO Q not under reset (default after reset)

1: GPIO Q under reset

Bit 15 **GPIOPRST**: GPIO P reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with GPIOPRSTS, and cleared with GPIOPRSTC. This bit is set and reset by software.

0: GPIO P not under reset (default after reset)

1: GPIO P under reset

Bit 14 **GPIOORST**: GPIO O reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with GPIOORSTS, and cleared with GPIOORSTC. This bit is set and reset by software.

0: GPIO O not under reset (default after reset)

1: GPIO O under reset

Bit 13 **GPIONRST**: GPIO N reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with GPIONRSTS, and cleared with GPIONRSTC. This bit is set and reset by software.

0: GPIO N not under reset (default after reset)

1: GPIO N under reset

Bits 12:8 Reserved, must be kept at reset value.

Bit 7 **GPIOHRST**: GPIO H reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with GPIOHRSTS, and cleared with GPIOHRSTC. This bit is set and reset by software.

0: GPIO H not under reset (default after reset)

1: GPIO H under reset

Bit 6 **GPIOGRST**: GPIO G reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with GPIOGRSTS, and cleared with GPIOGRSTC. This bit is set and reset by software.

0: GPIO G not under reset (default after reset)

1: GPIO G under reset

Bit 5 **GPIOFRST**: GPIO F reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with GPIOFRSTS, and cleared with GPIOFRSTC. This bit is set and reset by software.

0: GPIO F not under reset (default after reset)

1: GPIO F under reset

Bit 4 **GPIOERST**: GPIO E reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with GPIOERSTS, and cleared with GPIOERSTC. This bit is set and reset by software.

0: GPIO E not under reset (default after reset)

1: GPIO E under reset

Bit 3 **GPIODRST**: GPIO D reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with GPIODRSTS, and cleared with GPIODRSTC. This bit is set and reset by software.

0: GPIO D not under reset (default after reset)

1: GPIO D under reset

Bit 2 **GPIOCRST**: GPIO C reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with GPIOCRSTS, and cleared with GPIOCRSTC. This bit is set and reset by software.

0: GPIO C not under reset (default after reset)

1: GPIO C under reset

Bit 1 **GPIOBRST**: GPIO B reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with GPIOBRSTS, and cleared with GPIOBRSTC. This bit is set and reset by software.

0: GPIO B not under reset (default after reset)

1: GPIO B under reset

Bit 0 **GPIOARST**: GPIO A reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with GPIOARSTS, and cleared with GPIOARSTC. This bit is set and reset by software.

0: GPIO A not under reset (default after reset)

1: GPIO A under reset

14.10.68 RCC AHB5 reset register (RCC_AHB5RSTR)

Address offset: 0x220

Reset value: 0x0000 0000

This register is used to reset the AHB5. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NPU RST	CACHE AXI RST	OTG2R ST	OTGP HY2 RST	OTGP HY1 RST	OTG1 RST	ETH1 RST	OTG2P HYCTL RST	OTG1P HYCTL RST	Res.	Res.	GPU2D RST	GFXM MU RST	Res.	XSPI3 RST	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw			rw	rw		rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	XSPIM RST	XSPI2 RST	Res.	Res.	Res.	SDMM C1 RST	SDMM C2 RST	PSSI RST	XSPI1 RST	FMC RST	JPEG RST	Res.	DMA2D RST	HP DMA1 RST
		rw	rw				rw	rw	rw	rw	rw	rw		rw	rw

Bit 31 **NPURST**: NPU reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with NPURSTS, and cleared with NPURSTC. This bit is set and reset by software.

0: NPU not under reset (default after reset)

1: NPU under reset

Bit 30 **CACHEAXIRST**: CACHEAXI reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with CACHEAXIRSTS, and cleared with CACHEAXIRSTC. This bit is set and reset by software.

0: CACHEAXI not under reset (default after reset)

1: CACHEAXI under reset

Bit 29 **OTG2RST**: OTG2 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with OTG2RSTS, and cleared with OTG2RSTC. This bit is set and reset by software.

0: OTG2 not under reset (default after reset)

1: OTG2 under reset

Bit 28 **OTGPHY2RST**: OTGPHY2 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with OTGPHY2RSTS, and cleared with OTGPHY2RSTC. This bit is set and reset by software.

0: OTGPHY2 not under reset (default after reset)

1: OTGPHY2 under reset

Bit 27 **OTGPHY1RST**: OTGPHY1 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with OTGPHY1RSTS, and cleared with OTGPHY1RSTC. This bit is set and reset by software.

0: OTGPHY1 not under reset (default after reset)

1: OTGPHY1 under reset

Bit 26 **OTG1RST**: OTG1 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with OTG1RSTS, and cleared with OTG1RSTC. This bit is set and reset by software.

0: OTG1 not under reset (default after reset)

1: OTG1 under reset

Bit 25 **ETH1RST**: ETH1 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with ETH1RSTS, and cleared with ETH1RSTC. This bit is set and reset by software.

0: ETH1 not under reset (default after reset)

1: ETH1 under reset

Bit 24 **OTG2PHYCTLRST**: OTG2PHYCTL reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with OTG2PHYCTLRSTS, and cleared with OTG2PHYCTLRSTC. This bit is set and reset by software.

0: OTG2PHYCTL not under reset (default after reset)

1: OTG2PHYCTL under reset

Bit 23 **OTG1PHYCTLRST**: OTG1PHYCTL reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with OTG1PHYCTLRSTS, and cleared with OTG1PHYCTLRSTC. This bit is set and reset by software.

0: OTG1PHYCTL not under reset (default after reset)

1: OTG1PHYCTL under reset

Bits 22:21 Reserved, must be kept at reset value.

Bit 20 **GPU2DRST**: GPU2D reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with GPURSTS, and cleared with GPURSTC. This bit is set and reset by software.

0: GPU2D not under reset (default after reset)

1: GPU2D under reset

Bit 19 **GFXMMURST**: GFXMMU reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with GFXMMURSTS, and cleared with GFXMMURSTC. This bit is set and reset by software.

0: GFXMMU not under reset (default after reset)

1: GFXMMU under reset

Bit 18 Reserved, must be kept at reset value.

Bit 17 **XSPI3RST**: XSPI3 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with XSPI3RSTS, and cleared with XSPI3RSTC. This bit is set and reset by software.

0: XSPI3 not under reset (default after reset)

1: XSPI3 under reset

Bits 16:14 Reserved, must be kept at reset value.

Bit 13 **XSPIMRST**: XSPIM reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with XSPIMRSTS, and cleared with XSPIMRSTC. This bit is set and reset by software.

0: XSPIM not under reset (default after reset)

1: XSPIM under reset

Bit 12 **XSPI2RST**: XSPI2 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with XSPI2RSTS, and cleared with XSPI2RSTC. This bit is set and reset by software.

0: XSPI2 not under reset (default after reset)

1: XSPI2 under reset

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **SDMMC1RST**: SDMMC1 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with SDMMC1RSTS, and cleared with SDMMC1RSTC. This bit is set and reset by software.

0: SDMMC1 not under reset (default after reset)

1: SDMMC1 under reset

Bit 7 **SDMMC2RST**: SDMMC2 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with SDMMC2RSTS, and cleared with SDMMC2RSTC. This bit is set and reset by software.

0: SDMMC2 not under reset (default after reset)

1: SDMMC2 under reset

Bit 6 **PSSIRST**: PSSI reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with PSSIRSTS, and cleared with PSSIRSTC. This bit is set and reset by software.

0: PSSI not under reset (default after reset)

1: PSSI under reset

Bit 5 **XSPI1RST**: XSPI1 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with XSPI1RSTS, and cleared with XSPI1RSTC. This bit is set and reset by software.

0: XSPI1 not under reset (default after reset)

1: XSPI1 under reset

Bit 4 **FMCRST**: FMC reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with FMCRSTS and cleared with FMCRSTC. This bit is set and reset by software.

0: FMC not under reset (default after reset)

1: FMC under reset

Bit 3 **JPEGRST**: JPEG reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with JPEGRSTS, and cleared with JPEGRSTC. This bit is set and reset by software.

0: JPEG not under reset (default after reset)

1: JPEG under reset

Bit 2 Reserved, must be kept at reset value.

Bit 1 **DMA2DRST**: DMA2D reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. it can be set with DMA2DRSTS, and cleared with DMA2DRSTC. This bit is set and reset by software.

0: DMA2D not under reset (default after reset)

1: DMA2D under reset

Bit 0 **HPDMA1RST**: HPDMA1 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with HPDMA1RSTS, and cleared with HPDMA1RSTC. This bit is set and reset by software.

0: HPDMA1 not under reset (default after reset)

1: HPDMA1 under reset

14.10.69 RCC APB1L reset register (RCC_APB1LRSTR)

Address offset: 0x224

Reset value: 0x0000 0000

This register is used to reset the APB1L. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UART8 RST	UART7 RST	Res.	Res.	Res.	Res.	I3C2 RST	I3C1 RST	I2C3 RST	I2C2 RST	I2C1 RST	UART5 RST	UART4 RST	USART3 RST	USART2 RST	SPDIF RX1 RST
rw	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3 RST	SPI2 RST	TIM11 RST	TIM10 RST	WWDG RST	Res.	LPTIM1 RST	TIM14 RST	TIM13 RST	TIM12 RST	TIM7 RST	TIM6 RST	TIM5 RST	TIM4 RST	TIM3 RST	TIM2 RST
rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UART8RST**: UART8 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with UART8RSTS, and cleared with UART8RSTC. This bit is set and reset by software.

0: UART8 not under reset (default after reset)

1: UART8 under reset

Bit 30 **UART7RST**: UART7 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with UART7RSTS, and cleared with UART7RSTC. This bit is set and reset by software.

0: UART7 not under reset (default after reset)

1: UART7 under reset

Bits 29:26 Reserved, must be kept at reset value.

Bit 25 **I3C2RST**: I3C2 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with I3C2RSTS, and cleared with I3C2RSTC. This bit is set and reset by software.

0: I3C2 not under reset (default after reset)

1: I3C2 i under reset

Bit 24 **I3C1RST**: I3C1 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with I3C1RSTS, and cleared with I3C1RSTC. This bit is set and reset by software.

0: I3C1 not under reset (default after reset)

1: I3C1 under reset

Bit 23 **I2C3RST**: I2C3 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with I2C3RSTS, and cleared with I2C3RSTC. This bit is set and reset by software.

0: I2C3 not under reset (default after reset)

1: I2C3 under reset

Bit 22 **I2C2RST**: I2C2 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with I2C2RSTS, and cleared with I2C2RSTC. This bit is set and reset by software.

0: I2C2 not under reset (default after reset)

1: I2C2 under reset

Bit 21 **I2C1RST**: I2C1 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with I2C1RST, S and cleared with I2C1RSTC. This bit is set and reset by software.

0: I2C1 not under reset (default after reset)

1: I2C1 under reset

Bit 20 **UART5RST**: UART5 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with UART5RSTS, and cleared with UART5RSTC. This bit is set and reset by software.

0: UART5 not under reset (default after reset)

1: UART5 under reset

Bit 19 **UART4RST**: UART4 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with UART4RSTS, and cleared with UART4RSTC. This bit is set and reset by software.

0: UART4 not under reset (default after reset)

1: UART4 under reset

Bit 18 **USART3RST**: USART3 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with USART3RSTS, and cleared with USART3RSTC.
This bit is set and reset by software.
0: USART3 not under reset (default after reset)
1: USART3 under reset

Bit 17 **USART2RST**: USART2 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with USART2RSTS, and cleared with USART2RSTC.
This bit is set and reset by software.
0: USART2 not under reset (default after reset)
1: USART2 under reset

Bit 16 **SPDIFRX1RST**: SPDIFRX1 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. it can be set with SPDIFRX1RSTS, and cleared with SPDIFRX1RSTC. This bit is set and reset by software.
0: SPDIFRX1 not under reset (default after reset)
1: SPDIFRX1 under reset

Bit 15 **SPI3RST**: SPI3 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. it can be set with SPI3RSTS, and cleared with SPI3RSTC. This bit is set and reset by software.
0: SPI3 not under reset (default after reset)
1: SPI3 under reset

Bit 14 **SPI2RST**: SPI2 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with SPI2RSTS, and cleared with SPI2RSTC. This bit is set and reset by software.
0: SPI2 not under reset (default after reset)
1: SPI2 under reset

Bit 13 **TIM11RST**: TIM11 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM11RSTS, and cleared with TIM11RSTC. This bit is set and reset by software.
0: TIM11 not under reset (default after reset)
1: TIM11 under reset

Bit 12 **TIM10RST**: TIM10 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. IT can be set with TIM10RSTS, and cleared with TIM10RSTC. This bit is set and reset by software.
0: TIM10 not under reset (default after reset)
1: TIM10 under reset

Bit 11 **WWDGRST**: WWDG reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with WWDGRSTS, and cleared with WWDGRSTC. This bit is set and reset by software.
0: WWDG not under reset (default after reset)
1: WWDG under reset

Bit 10 Reserved, must be kept at reset value.

Bit 9 **LPTIM1RST**: LPTIM1 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with LPTIM1RSTS, and cleared with LPTIM1RSTC. This bit is set and reset by software.

0: LPTIM1 not under reset (default after reset)

1: LPTIM1 under reset

Bit 8 **TIM14RST**: TIM14 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM14RSTS, and cleared with TIM14RSTC. This bit is set and reset by software.

0: TIM14 not under reset (default after reset)

1: TIM14 under reset

Bit 7 **TIM13RST**: TIM13 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM13RSTS, and cleared with TIM13RSTC. This bit is set and reset by software.

0: TIM13 not under reset (default after reset)

1: TIM13 under reset

Bit 6 **TIM12RST**: TIM12 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM12RSTS, and cleared with TIM12RSTC. This bit is set and reset by software.

0: TIM12 not under reset (default after reset)

1: TIM12 under reset

Bit 5 **TIM7RST**: TIM7 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM7RSTS, and cleared with TIM7RSTC. This bit is set and reset by software.

0: TIM7 not under reset (default after reset)

1: TIM7 under reset

Bit 4 **TIM6RST**: TIM6 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM6RSTS, and cleared with TIM6RSTC. This bit is set and reset by software.

0: TIM6 not under reset (default after reset)

1: TIM6 under reset

Bit 3 **TIM5RST**: TIM5 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM5RSTS, and cleared with TIM5RSTC. This bit is set and reset by software.

0: TIM5 not under reset (default after reset)

1: TIM5 under reset

Bit 2 **TIM4RST**: TIM4 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM4RSTS, and cleared with TIM4RSTC. This bit is set and reset by software.

0: TIM4 not under reset (default after reset)

1: TIM4 under reset

Bit 1 **TIM3RST**: TIM3 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM3RSTS, and cleared with TIM3RSTC. This bit is set and reset by software.

0: TIM3 not under reset (default after reset)

1: TIM3 under reset

Bit 0 **TIM2RST**: TIM2 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM2RSTS, and cleared with TIM2RSTC. This bit is set and reset by software.

0: TIM2 not under reset (default after reset)

1: TIM2 under reset

14.10.70 RCC APB1H reset register (RCC_APB1HRSTR)

Address offset: 0x228

Reset value: 0x0000 0000

This register is used to reset the APB1H. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD1 RST	Res.	Res.
													rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCAN RST	Res.	Res.	MDIOS RST	Res.	Res.	Res.	Res.	Res.
							rw			rw					

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **UCPD1RST**: UCPD1 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. it can be set with UCPD1RSTS, and cleared with UCPD1RSTC. This bit is set and reset by software.

0: UCPD1 not under reset (default after reset)

1: UCPD1 under reset

Bits 17:9 Reserved, must be kept at reset value.

Bit 8 **FDCANRST**: FDCAN reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with FDCANRSTS, and cleared with FDCANRSTC. This bit is set and reset by software.

0: FDCAN not under reset (default after reset)

1: FDCAN under reset

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **MDIOSRST**: MDIOS reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with MDIOSRSTS, and cleared with MDIOSRSTC. This bit is set and reset by software.

0: MDIOS not under reset (default after reset)

1: MDIOS under reset

Bits 4:0 Reserved, must be kept at reset value.

14.10.71 RCC APB2 reset register (RCC_APB2RSTR)

Address offset: 0x22C

Reset value: 0x0000 0000

This register is used to reset the APB2. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI2RST	SAI1RST	SPI5RST	TIM9RST	TIM17RST	TIM16RST	TIM15RST
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIM18RST	Res.	SPI4RST	SPI1RST	Res.	Res.	Res.	Res.	USART10RST	UART9RST	USART6RST	USART1RST	Res.	Res.	TIM8RST	TIM1RST
rw		rw	rw					rw	rw	rw	rw			rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **SAI2RST**: SAI2 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with SAI2RSTS, and cleared with SAI2RSTC. This bit is set and reset by software.

0: SAI2 not under reset (default after reset)

1: SAI2 under reset

Bit 21 **SAI1RST**: SAI1 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with SAI1RSTS, and cleared with SAI1RSTC. This bit is set and reset by software.

0: SAI1 not under reset (default after reset)

1: SAI1 under reset

Bit 20 **SPI5RST**: SPI5 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. it can be set with SPI5RSTS, and cleared with SPI5RSTC. This bit is set and reset by software.

0: SPI5 not under reset (default after reset)

1: SPI5 under reset

Bit 19 **TIM9RST**: TIM9 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM9RSTS, and cleared with TIM9RSTC. This bit is set and reset by software.

0: TIM9 not under reset (default after reset)

1: TIM9 under reset

Bit 18 **TIM17RST**: TIM17 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM17RSTS, and cleared with TIM17RSTC. This bit is set and reset by software.

0: TIM17 not under reset (default after reset)

1: TIM17 under reset

Bit 17 **TIM16RST**: TIM16 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM16RSTS, and cleared with TIM16RSTC. This bit is set and reset by software.

0: TIM16 not under reset (default after reset)

1: TIM16 under reset

Bit 16 **TIM15RST**: TIM15 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM15RSTS, and cleared with TIM15RSTC. This bit is set and reset by software.

0: TIM15 not under reset (default after reset)

1: TIM15 under reset

Bit 15 **TIM18RST**: TIM18 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM18RSTS, and cleared with TIM18RSTC. This bit is set and reset by software.

0: TIM18 not under reset (default after reset)

1: TIM18 under reset

Bit 14 Reserved, must be kept at reset value.

Bit 13 **SPI4RST**: SPI4 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with SPI4RSTS, and cleared with SPI4RSTC. This bit is set and reset by software.

0: SPI4 not under reset (default after reset)

1: SPI4 under reset

Bit 12 **SPI1RST**: SPI1 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with SPI1RSTS, and cleared with SPI1RSTC. This bit is set and reset by software.

0: SPI1 not under reset (default after reset)

1: SPI1 under reset

Bits 11:8 Reserved, must be kept at reset value.

Bit 7 **USART10RST**: USART10 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with USART10RSTS, and cleared with USART10RSTC. This bit is set and reset by software.

0: USART10 not under reset (default after reset)

1: USART10 under reset

Bit 6 **UART9RST**: UART9 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with UART9RSTS, and cleared with UART9RSTC. This bit is set and reset by software.

0: UART9 not under reset (default after reset)

1: UART9 under reset

Bit 5 **USART6RST**: USART6 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with USART6RSTS, and cleared with USART6RSTC. This bit is set and reset by software.

0: USART6 not under reset (default after reset)

1: USART6 under reset

Bit 4 **USART1RST**: USART1 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with USART1RSTS, and cleared with USART1RSTC. This bit is set and reset by software.

0: USART1 not under reset (default after reset)

1: USART1 under reset

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **TIM8RST**: TIM8 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM8RSTS, and cleared with TIM8RSTC. This bit is set and reset by software.

0: TIM8 not under reset (default after reset)

1: TIM8 under reset

Bit 0 **TIM1RST**: TIM1 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. it can be set with TIM1RSTS, and cleared with TIM1RSTC. This bit is set and reset by software.

0: TIM1 not under reset (default after reset)

1: TIM1 under reset

14.10.72 RCC APB4L reset register (RCC_APB4LRSTR)

Address offset: 0x234

Reset value: 0x0000 0000

This register is used to reset the APB4L. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VREFBUFRST	Res.	Res.	LPTIM5RST	LPTIM4RST	LPTIM3RST	LPTIM2RST	Res.	I2C4RST	Res.	SPI6RST	Res.	LPUART1RST	HDPRST	Res.	Res.
rw			rw	rw	rw	rw		rw		rw		rw	rw		

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **RTCRST**: RTC reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. it can be set with RTCRSTS, and cleared with RTCRSTC. This bit is set and reset by software.

0: RTC not under reset (default after reset)

1: RTC under reset

Bit 15 **VREFBUFRST**: VREFBUF reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. it can be set with VREFBUFRSTS, and cleared with VREFBUFRSTC. This bit is set and reset by software.

0: VREFBUF not under reset (default after reset)

1: VREFBUF under reset

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **LPTIM5RST**: LPTIM5 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. it can be set with LPTIM5RSTS, and cleared with LPTIM5RSTC. This bit is set and reset by software.

0: LPTIM5 not under reset (default after reset)

1: LPTIM5 under reset

Bit 11 **LPTIM4RST**: LPTIM4 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. it can be set with LPTIM4RSTS, and cleared with LPTIM4RSTC. This bit is set and reset by software.

0: LPTIM4 not under reset (default after reset)

1: LPTIM4 under reset

Bit 10 **LPTIM3RST**: LPTIM3 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. it can be set with LPTIM3RSTS, and cleared with LPTIM3RSTC. This bit is set and reset by software.

0: LPTIM3 not under reset (default after reset)

1: LPTIM3 under reset

Bit 9 **LPTIM2RST**: LPTIM2 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. it can be set with LPTIM2RSTS, and cleared with LPTIM2RSTC. This bit is set and reset by software.

0: LPTIM2 not under reset (default after reset)

1: LPTIM2 under reset

Bit 8 Reserved, must be kept at reset value.

Bit 7 **I2C4RST**: I2C4 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. it can be set with I2C4RSTS, and cleared with I2C4RSTC. This bit is set and reset by software.

0: I2C4 not under reset (default after reset)

1: I2C4 under reset

Bit 6 Reserved, must be kept at reset value.

Bit 5 **SPI6RST**: SPI6 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. it can be set with SPI6RSTS, and cleared with SPI6RSTC. This bit is set and reset by software.

0: SPI6 not under reset (default after reset)

1: SPI6 under reset

Bit 4 Reserved, must be kept at reset value.

Bit 3 **LPUART1RST**: LPUART1 reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. it can be set with LPUART1RST, S and cleared with LPUART1RSTC. This bit is set and reset by software.

0: LPUART1 not under reset (default after reset)

1: LPUART1 under reset

Bit 2 **HDPRST**: HDP reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. it can be set with HDPRSTS, and cleared with HDPRSTC. This bit is set and reset by software.

0: HDP not under reset (default after reset)

1: HDP under reset

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

14.10.73 RCC APB4H reset register (RCC_APB4HRSTR)

Address offset: 0x238

Reset value: 0x0000 0000

This register is used to reset the APB4H. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

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

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

Bit 2 **DTSRST**: DTS reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. it can be set with DTSRSTS, and cleared with DTSRSTC. This bit is set and reset by software.

0: DTS not under reset (default after reset)

1: DTS under reset

Bit 1 Reserved, must be kept at reset value.

Bit 0 **SYSCFGRST**: SYSCFG reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. it can be set with SYSCFGRSTS, and cleared with SYSCFGRSTC. This bit is set and reset by software.

0: SYSCFG not under reset (default after reset)

1: SYSCFG under reset

14.10.74 RCC APB5 reset register (RCC_APB5RSTR)

Address offset: 0x23C

Reset value: 0x0000 0000

This register is used to reset the APB5. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

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

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CSIRST**: CSI reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. it can be set with CSIRSTS, and cleared with CSIRSTC. This bit is set and reset by software.

0: CSI not under reset (default after reset)

1: CSI under reset

Bit 5 **VENCRST**: VENC reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. it can be set with VENCRSTS, and cleared with VENCRSTC. This bit is set and reset by software.

0: VENC not under reset (default after reset)

1: VENC under reset

Bit 4 **GFXTIMRST**: GFXTIM reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. it can be set with GFXTIMRSTS, and cleared with GFXTIMRSTC.

This bit is set and reset by software.

0: GFXTIM not under reset (default after reset)

1: GFXTIM under reset

Bit 3 Reserved, must be kept at reset value.

Bit 2 **DCMIPPRST**: DCMIPP reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with DCMIPPRSTS, and cleared with DCMIPPRSTC.
This bit is set and reset by software.
0: DCMIPP not under reset (default after reset)
1: DCMIPP under reset

Bit 1 **LTDCRST**: LTDC reset

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with LTDCRST, S and cleared with LTDCRSTC. This bit is set and reset by software.
0: LTDC not under reset (default after reset)
1: LTDC under reset

Bit 0 Reserved, must be kept at reset value.

14.10.75 RCC IC dividers enable register (RCC_DIVENR)

Address offset: 0x240

Reset value: 0x0000 0000

This register is used to enable the IC dividers in Run, Sleep, and Stop modes. It is reset by int_sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC20EN	IC19EN	IC18EN	IC17EN
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC16EN	IC15EN	IC14EN	IC13EN	IC12EN	IC11EN	IC10EN	IC9EN	IC8EN	IC7EN	IC6EN	IC5EN	IC4EN	IC3EN	IC2EN	IC1EN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **IC20EN**: IC20 enable

This bit is security-protected by IC20SEC or IC20PRIV, and is publicly readable if IC20PUB = 1. It can be set with IC20ENS, and cleared with IC20ENC. This bit is set and reset by software.
0: IC20 disabled (default after reset)
1: IC20 enabled

Bit 18 **IC19EN**: IC19 enable

This bit is security-protected by IC19SEC or IC19PRIV, and is publicly readable if IC19PUB = 1. It can be set with IC19ENS, and cleared with IC19ENC. This bit is set and reset by software.
0: IC19 disabled (default after reset)
1: IC19 enabled

Bit 17 **IC18EN**: IC18 enable

This bit is security-protected by IC18SEC or IC18PRIV, and is publicly readable if IC18PUB = 1. It can be set with IC18ENS, and cleared with IC18ENC. This bit is set and reset by software.
0: IC18 disabled (default after reset)
1: IC18 enabled

Bit 16 **IC17EN**: IC17 enable

This bit is security-protected by IC17SEC or IC17PRIV, and is publicly readable if IC17PUB = 1. It can be set with IC17ENS, and cleared with IC17ENC. This bit is set and reset by software.

0: IC17 disabled (default after reset)

1: IC17 enabled

Bit 15 **IC16EN**: IC16 enable

This bit is security-protected by IC16SEC or IC16PRIV, and is publicly readable if IC16PUB = 1. It can be set with IC16ENS, and cleared with IC16ENC. This bit is set and reset by software.

0: IC16 disabled (default after reset)

1: IC16 enabled

Bit 14 **IC15EN**: IC15 enable

This bit is security-protected by IC15SEC or IC15PRIV, and is publicly readable if IC15PUB = 1. It can be set with IC15ENS, and cleared with IC15ENC. This bit is set and reset by software.

0: IC15 disabled (default after reset)

1: IC15 enabled

Bit 13 **IC14EN**: IC14 enable

This bit is security-protected by IC14SEC or IC14PRIV, and is publicly readable if IC14PUB = 1. It can be set with IC14ENS, and cleared with IC14ENC. This bit is set and reset by software.

0: IC14 disabled (default after reset)

1: IC14 enabled

Bit 12 **IC13EN**: IC13 enable

This bit is security-protected by IC13SEC or IC13PRIV, and is publicly readable if IC13PUB = 1. It can be set with IC13ENS, and cleared with IC13ENC. This bit is set and reset by software.

0: IC13 disabled (default after reset)

1: IC13 enabled

Bit 11 **IC12EN**: IC12 enable

This bit is security-protected by IC12SEC or IC12PRIV, and is publicly readable if IC12PUB = 1. It can be set with IC12ENS, and cleared with IC12ENC. This bit is set and reset by software.

0: IC12 disabled (default after reset)

1: IC12 enabled

Bit 10 **IC11EN**: IC11 enable

This bit is security-protected by IC11SEC or IC11PRIV, and is publicly readable if IC11PUB = 1. It can be set with IC11ENS, and cleared with IC11ENC. This bit is set and reset by software.

0: IC11 disabled (default after reset)

1: IC11 enabled

Bit 9 **IC10EN**: IC10 enable

This bit is security-protected by IC10SEC or IC10PRIV, and is publicly readable if IC10PUB = 1. It can be set with IC10ENS, and cleared with IC10ENC. This bit is set and reset by software.

0: IC10 disabled (default after reset)

1: IC10 enabled

Bit 8 **IC9EN**: IC9 enable
This bit is security-protected by IC9SEC or IC9PRIV, and is publicly readable if IC9PUB = 1. It can be set with IC9ENS, and cleared with IC9ENC. This bit is set and reset by software.
0: IC9 disabled (default after reset)
1: IC9 enabled

Bit 7 **IC8EN**: IC8 enable
This bit is security-protected by IC8SEC or IC8PRIV, and is publicly readable if IC8PUB = 1. It can be set with IC8ENS, and cleared with IC8ENC. This bit is set and reset by software.
0: IC8 disabled (default after reset)
1: IC8 enabled

Bit 6 **IC7EN**: IC7 enable
This bit is security-protected by IC7SEC or IC7PRIV, and is publicly readable if IC7PUB = 1. It can be set with IC7ENS, and cleared with IC7ENC. This bit is set and reset by software.
0: IC7 disabled (default after reset)
1: IC7 enabled

Bit 5 **IC6EN**: IC6 enable
This bit is security-protected by IC6SEC or IC6PRIV, and is publicly readable if IC6PUB = 1. It can be set with IC6ENS, and cleared with IC6ENC. This bit is set and reset by software.
0: IC6 disabled (default after reset)
1: IC6 enabled

Bit 4 **IC5EN**: IC5 enable
This bit is security-protected by IC5SEC or IC5PRIV, and is publicly readable if IC5PUB = 1. It can be set with IC5ENS, and cleared with IC5ENC. This bit is Set and reset by software.
0: IC5 disabled (default after reset)
1: IC5 enabled

Bit 3 **IC4EN**: IC4 enable
This bit is security-protected by IC4SEC or IC4PRIV, and is publicly readable if IC4PUB = 1. It can be set with IC4ENS, and cleared with IC4ENC. This bit is set and reset by software.
0: IC4 disabled (default after reset)
1: IC4 enabled

Bit 2 **IC3EN**: IC3 enable
This bit is security-protected by IC3SEC or IC3PRIV, and is publicly readable if IC3PUB = 1. It can be set with IC3ENS, and cleared with IC3ENC. This bit is set and reset by software.
0: IC3 disabled (default after reset)
1: IC3 enabled

Bit 1 **IC2EN**: IC2 enable
This bit is security-protected by IC2SEC or IC2PRIV, and is publicly readable if IC2PUB = 1. It can be set with IC2ENS, and cleared with IC2ENC. This bit is set and reset by software.
0: IC2 disabled (default after reset)
1: IC2 enabled

Bit 0 **IC1EN**: IC1 enable
This bit is security-protected by IC1SEC or IC1PRIV, and is publicly readable if IC1PUB = 1. It can be set with IC1ENS, and cleared with IC1ENC. This bit is set and reset by software.
0: IC1 disabled (default after reset)
1: IC1 enabled

14.10.76 RCC embedded buses enable register (RCC_BUSENR)

Address offset: 0x244

Reset value: 0x0000 0003

This register is used to enable the embedded buses in Run and Sleep modes (in Sleep mode, each bit of this register is AND-ed with the LPEN bit). It is reset by sys_rst, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	APB5EN	APB4EN	APB3EN	APB2EN	APB1EN	AHB5EN	AHB4EN	AHB3EN	AHB2EN	AHB1EN	AHBMEN	ACLKNCEN	ACLKNEN
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **APB5EN**: APB5 enable

This bit is security-protected by the APB5SEC bit, the APB5PV bit, and is publicly readable if APB5PUB = 1. It can be set with APB5ENS, and cleared with APB5ENC. The bit field is non-user. Set and reset by software. Debug field, must be kept at reset value.

0: APB5 disabled (default after reset)

1: APB5 enabled

Bit 11 **APB4EN**: APB4 enable

The bit field is security-protected by the APB4SEC bit, the APB4PV bit, and is publicly readable if APB4PUB=1. The bit can be set with APB4ENS and cleared with APB4ENC. The bit field is non-user. Set and reset by software. Debug field, must be kept at reset value.

0: APB4 disabled (default after reset)

1: APB4 enabled

Bit 10 **APB3EN**: APB3 enable

The bit field is security-protected by the APB3SEC bit, the APB3PV bit, and is publicly readable if APB3PUB=1. The bit can be set with APB3ENS and cleared with APB3ENC. The bit field is non-user. Set and reset by software. Debug field, must be kept at reset value.

0: APB3 disabled (default after reset)

1: APB3 enabled

Bit 9 **APB2EN**: APB2 enable

The bit field is security-protected by the APB2SEC bit, the APB2PV bit, and is publicly readable if APB2PUB=1. The bit can be set with APB2ENS and cleared with APB2ENC. The bit field is non-user. Set and reset by software. Debug field, must be kept at reset value.

0: APB2 is disabled (default after reset)

1: APB2 is enabled

Bit 8 **APB1EN**: APB1 enable

The bit field is security-protected by the APB1SEC bit, the APB1PV bit, and is publicly readable if APB1PUB=1. The bit can be set with APB1ENS and cleared with APB1ENC. The bit field is non-user. Set and reset by software. Debug field, must be kept at reset value.

0: APB1 is disabled (default after reset)

1: APB1 is enabled

Bit 7 **AHB5EN**: AHB5 enable

The bit field is security-protected by the AHB5SEC bit, the AHB5PV bit, and is publicly readable if AHB5PUB=1. The bit can be set with AHB5ENS and cleared with AHB5ENC. The bit field is non-user. Set and reset by software. Debug field, must be kept at reset value.

0: AHB5 is disabled (default after reset)

1: AHB5 is enabled

Bit 6 **AHB4EN**: AHB4 enable

The bit field is security-protected by the AHB4SEC bit, the AHB4PV bit, and is publicly readable if AHB4PUB=1. The bit can be set with AHB4ENS and cleared with AHB4ENC. The bit field is non-user. Set and reset by software. Debug field, must be kept at reset value.

0: AHB4 is disabled (default after reset)

1: AHB4 is enabled

Bit 5 **AHB3EN**: AHB3 enable

The bit field is security-protected by the AHB3SEC bit, the AHB3PV bit, and is publicly readable if AHB3PUB=1. The bit can be set with AHB3ENS and cleared with AHB3ENC. The bit field is non-user. Set and reset by software. Debug field, must be kept at reset value.

0: AHB3 is disabled (default after reset)

1: AHB3 is enabled

Bit 4 **AHB2EN**: AHB2 enable

The bit field is security-protected by the AHB2SEC bit, the AHB2PV bit, and is publicly readable if AHB2PUB=1. The bit can be set with AHB2ENS and cleared with AHB2ENC. The bit field is non-user. Set and reset by software. Debug field, must be kept at reset value.

0: AHB2 is disabled (default after reset)

1: AHB2 is enabled

Bit 3 **AHB1EN**: AHB1 enable

The bit field is security-protected by the AHB1SEC bit, the AHB1PV bit, and is publicly readable if AHB1PUB=1. The bit can be set with AHB1ENS and cleared with AHB1ENC. The bit field is non-user. Set and reset by software. Debug field, must be kept at reset value.

0: AHB1 is disabled (default after reset)

1: AHB1 is enabled

Bit 2 **AHBMEN**: AHBM enable

The bit field is security-protected by the AHBMSEC bit, the AHBMPV bit, and is publicly readable if AHBMPUB = 1. The bit can be set with AHBMENS and cleared with AHBMENC. The bit field is non-user. Set and reset by software. Debug field, must be kept at reset value.

0: AHBM is disabled (default after reset)

1: AHBM is enabled

Bit 1 **ACLKNCEN**: ACLKNC enable

This bit is security-protected by ACLKNCSEC or ACLKNCPRIV, and is publicly readable if ACLKNCPUB = 1. It can be set with ACLKNCENS, and cleared with ACLKNCENC. This bit is set and reset by software.

0: ACLKNC disabled

1: ACLKNC enabled (default after reset)

Bit 0 **ACLKNEN**: ACLKN enable

This bit is security-protected by ACLKNSEC or ACLKNPRIV, and is publicly readable if ACLKNPUB = 1. It can be set with ACLKNENS, and cleared with ACLKNENC. This bit is set and reset by software.

0: ACLKN disabled

1: ACLKN enabled (default after reset)

14.10.77 RCC miscellaneous configurations enable register (RCC_MISCENR)

Address offset: 0x248

Reset value: 0x0000 0000

This register is used to enable the miscellaneous configurations in Run and Sleep modes (in Sleep mode, each bit of this register is AND-ed with the LPEN bit). It is reset by int_sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PEREN	Res.	Res.	XSPIPHYCOMPEN	MCO2EN	MCO1EN	DBGEN
									rw			rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **PEREN**: PER enable

This bit is security-protected by PERSEC or PERPRIV. it can be set with PERENS, and cleared with PERENC. This bit is set and reset by software.

0: PER disabled (default after reset)

1: PER enabled

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **XSPIPHYCOMPEN**: XSPIPHYCOMP enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. it can be set with XSPIPHYCOMPENS, and cleared with XSPIPHYCOMPENC. This bit is set and reset by software.

0: XSPIPHYCOMP disabled (default after reset)

1: XSPIPHYCOMP enabled

Bit 2 **MCO2EN**: MCO2 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. it can be set with MCO2ENS, and cleared with MCO2ENC. This bit is set and reset by software.

0: MCO2 disabled (default after reset)

1: MCO2 enabled

Bit 1 **MCO1EN**: MCO1 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. it can be set with MCO1ENS, and cleared with MCO1ENC. This bit is set and reset by software.

0: MCO1 disabled (default after reset)

1: MCO1 enabled

Bit 0 **DBGEN**: DBG enable

This bit is always security-protected. It can be set with DBGENS, and cleared with DBGENC.This bit is set and reset by software.

0: DBG disabled (default after reset)

1: DBG enabled

14.10.78 RCC embedded memories enable register (RCC_MEMENR)

Address offset: 0x24C

Reset value: 0x0000 13F0

This register is used to enable the embedded memories in Run and Sleep modes (in Sleep mode, each bit of this register is AND-ed with the LPEN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	BOOTROMEN	VENCRAMEN	CACHEAXIRAMEN	FLEXRAMEN	AXISRAM2EN	AXISRAM1EN	BKPSRAMEN	AHBSRAM2EN	AHBSRAM1EN	AXISRAM6EN	AXISRAM5EN	AXISRAM4EN	AXISRAM3EN
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **BOOTROMEN**: BootROM enable

This bit is always security-protected. It can be set with BOOTROMENS, and cleared with BOOTROMENC. This bit is set and reset by software.

0: BootROM disabled

1: BootROM enabled (default after reset)

Bit 11 **VENCRAMEN**: VENCRAM enable

This bit is security-protected by a SEC or a PRIV signal from RIFSC, and is publicly readable if VENCRAMPUB = 1. It can be set with VENCRAMENS, and cleared with VENCRAMENC. This bit is set and reset by software.

0: VENCRAM disabled (default after reset)

1: VENCRAM enabled

Bit 10 **CACHEAXIRAMEN**: CACHEAXIRAM enable

This bit is security-protected by a SEC or a PRIV signal from RIFSC, and is publicly readable if CACHEAXIRAMPUB = 1. It can be set with CACHEAXIRAMENS, and cleared with CACHEAXIRAMENC. This bit is set and reset by software.

0: CACHEAXIRAM disabled (default after reset)

1: CACHEAXIRAM enabled

Bit 9 **FLEXRAMEN**: FLEXRAM enable

This bit is security-protected by a SEC or a PRIV signal from RIFSC, and is publicly readable if FLEXRAMPUB = 1. It can be set with FLEXRAMENS, and cleared with FLEXRAMENC. This bit is set and reset by software.

0: FLEXRAM disabled

1: FLEXRAM enabled (default after reset)

Bit 8 **AXISRAM2EN**: AXISRAM2 enable

This bit is security-protected by a SEC or a PRIV signal from RIFSC, and is publicly readable if AXISRAM2PUB = 1. It can be set with AXISRAM2ENS, and cleared with AXISRAM2ENC. This bit is set and reset by software.

0: AXISRAM2 disabled

1: AXISRAM2 enabled (default after reset)

Bit 7 **AXISRAM1EN**: AXISRAM1 enable

This bit is security-protected by a SEC or a PRIV signal from RIFSC, and is publicly readable if AXISRAM1PUB = 1. It can be set with AXISRAM1ENS, and cleared with AXISRAM1ENC. This bit is set and reset by software.

0: AXISRAM1 disabled

1: AXISRAM1 enabled (default after reset)

Bit 6 **BKPSRAMEN**: BKPSRAM enable

This bit is security-protected by a SEC or a PRIV signal from RIFSC, and is publicly readable if BKPSRAMPUB = 1. It can be set with BKPSRAMENS, and cleared with BKPSRAMENC. This bit is set and reset by software.

0: BKPSRAM disabled

1: BKPSRAM enabled (default after reset)

Bit 5 **AHBSRAM2EN**: AHBSRAM2 enable

This bit is security-protected by a SEC or a PRIV signal from RIFSC, and is publicly readable if AHBSRAM2PUB = 1. It can be set with AHBSRAM2ENS, and cleared with AHBSRAM2ENC. This bit is set and reset by software.

0: AHBSRAM2 disabled

1: AHBSRAM2 enabled (default after reset)

Bit 4 **AHBSRAM1EN**: AHBSRAM1 enable

This bit is security-protected by a SEC or a PRIV signal from RIFSC, and is publicly readable if AHBSRAM1PUB = 1. It can be set with AHBSRAM1ENS, and cleared with AHBSRAM1ENC. This bit is set and reset by software.

0: AHBSRAM1 disabled

1: AHBSRAM1 enabled (default after reset)

Bit 3 **AXISRAM6EN**: AXISRAM6 enable

This bit is security-protected by a SEC or a PRIV signal from RIFSC, and is publicly readable if AXISRAM6PUB = 1. It can be set with AXISRAM6ENS, and cleared with AXISRAM6ENC. This bit is set and reset by software.

0: AXISRAM6 disabled (default after reset)

1: AXISRAM6 enabled

Bit 2 **AXISRAM5EN**: AXISRAM5 enable

This bit is security-protected by a SEC or a PRIV signal from RIFSC, and is publicly readable if AXISRAM5PUB = 1. It can be set with AXISRAM5ENS, and cleared with AXISRAM5ENC. This bit is set and reset by software.

0: AXISRAM5 disabled (default after reset)

1: AXISRAM5 enabled

Bit 1 **AXISRAM4EN**: AXISRAM4 enable

This bit is security-protected by a SEC or a PRIV signal from RIFSC, and is publicly readable if AXISRAM4PUB = 1. It can be set with AXISRAM4ENS, and cleared with AXISRAM4ENC. This bit is set and reset by software.

0: AXISRAM4 disabled (default after reset)

1: AXISRAM4 enabled

Bit 0 **AXISRAM3EN**: AXISRAM3 enable

This bit is security-protected by a SEC or a PRIV signal from RIFSC, and is publicly readable if AXISRAM3PUB = 1. It can be set with AXISRAM3ENS, and cleared with AXISRAM3ENC. This bit is set and reset by software.

0: AXISRAM3 disabled (default after reset)

1: AXISRAM3 enabled

14.10.79 RCC AHB1 enable register (RCC_AHB1ENR)

Address offset: 0x250

Reset value: 0x0000 0000

This register is used to enable the AHB1 in Run and Sleep modes (in Sleep mode, each bit of this register is AND-ed with the LPEN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC12EN	GPDMA1EN	Res.	Res.	Res.	Res.
										rw	rw				

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

Bit 5 **ADC12EN**: ADC12 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with ADC12ENS, and cleared with ADC12ENC. This bit is set and reset by software.

0: ADC12 disabled (default after reset)

1: ADC12 enabled

Bit 4 **GPDMA1EN**: GPDMA1 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with GPDMA1ENS, and cleared with GPDMA1ENC. This bit is set and reset by software.

0: GPDMA1 disabled (default after reset)

1: GPDMA1 enabled

Bits 3:0 Reserved, must be kept at reset value.

14.10.80 RCC AHB2 enable register (RCC_AHB2ENR)

Address offset: 0x254

Reset value: 0x0000 1000

This register is used to enable the AHB2 in Run and Sleep modes (in Sleep mode, each bit of this register is AND-ed with the LPEN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADF1EN	MDF1EN
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	RAMCFGEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			rw												

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

Bit 17 **ADF1EN**: ADF1 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with ADF1ENS, and cleared with ADF1ENC. This bit is set and reset by software.

0: ADF1 disabled (default after reset)

1: ADF1 enabled

Bit 16 **MDF1EN**: MDF1 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with MDF1ENS, and cleared with MDF1ENC. This bit is set and reset by software.

0: MDF1 disabled (default after reset)

1: MDF1 enabled

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **RAMCFGEN**: RAMCFG enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with RAMCFGENS, and cleared with RAMCFGENC. This bit is set and reset by software.

0: RAMCFG disabled

1: RAMCFG enabled (default after reset)

Bits 11:0 Reserved, must be kept at reset value.

14.10.81 RCC AHB3 enable register (RCC_AHB3ENR)

Address offset: 0x258

Reset value: 0x0000 4600

This register is used to enable the AHB3 in Run and Sleep modes (in Sleep mode, each bit of this register is AND-ed with the LPEN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RISAFEN	Res.	Res.	Res.	IACEN	RIFSCEN	PKAEN	Res.	Res.	Res.	SAESEN	Res.	CRYPEN	HASHEN	RNGEN
	rw				rw	rw	rw				rw		rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **RISAFEN**: RISAF enable

This bit is always security-protected. It can be set with RISAFENS, and cleared with RISAFENC. This bit is set and reset by software.

0: RISAF disabled

1: RISAF enabled (default after reset)

Bits 13:11 Reserved, must be kept at reset value.

Bit 10 **IACEN**: IAC enable

This bit is always security-protected. It can be set with IACENS, and cleared with IACENC. This bit is set and reset by software.

0: IAC disabled

1: IAC enabled (default after reset)

Bit 9 **RIFSCEN**: RIFSC enable

This bit is always security-protected. It can be set with RIFSCENS, and cleared with RIFSCENC. This bit is set and reset by software.

0: RIFSC disabled

1: RIFSC enabled (default after reset)

Bit 8 **PKAEN**: PKA enable

This bit is security-protected by SYSSEC or SYSPRIV. It can be set with PKAENS, and cleared with PKAENC. This bit is set and reset by software.

0: PKA disabled (default after reset)

1: PKA enabled

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **SAESEN**: SAES enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. it can be set with SAESENS, and cleared with SAESENC. This bit is set and reset by software.

0: SAES disabled (default after reset)

1: SAES enabled

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CRYPEN**: CRYP enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with CRYPENS, and cleared with CRYPENC. This bit is set and reset by software.

0: CRYP disabled (default after reset)

1: CRYP enabled

Bit 1 **HASHEN**: HASH enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with HASHENS, and cleared with HASHENC. This bit is set and reset by software.

0: HASH disabled (default after reset)

1: HASH enabled

Bit 0 **RNGEN**: RNG enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with RNGENS, and cleared with RNGENC. This bit is set and reset by software.

0: RNG disabled (default after reset)

1: RNG enabled

14.10.82 RCC AHB4 enable register (RCC_AHB4ENR)

Address offset: 0x25C

Reset value: 0x0004 0000

This register is used to enable the AHB4 in Run and Sleep modes (in Sleep mode, each bit of this register is AND-ed with the LPEN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCEN	PWREN	Res.	GPIOQEN
												rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIOPEN	GPIOOEN	GPIONEN	Res.	Res.	Res.	Res.	Res.	GPIOHEN	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
rw	rw	rw						rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **CRCEN**: CRC enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with CRCENS, and cleared with CRCENC. This bit is set and reset by software.

0: CRC disabled (default after reset)

1: CRC enabled

Bit 18 **PWREN**: PWR enable

This bit is always security-protected. It can be set with PWRENS, and cleared with PWRENC. This bit is set and reset by software.

0: PWR disabled

1: PWR enabled (default after reset)

Bit 17 Reserved, must be kept at reset value.

Bit 16 **GPIOQEN**: GPIO Q enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with GPIOQENS, and cleared with GPIOQENC. This bit is set and reset by software.

0: GPIO Q disabled (default after reset)

1: GPIO Q enabled

Bit 15 **GPIOPEN**: GPIO P enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with GPIOPENS, and cleared with GPIOPENC. This bit is set and reset by software.

0: GPIO P disabled (default after reset)

1: GPIO P enabled

Bit 14 **GPIOOEN**: GPIO O enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with GPIOOENS, and cleared with GPIOOENC. This bit is set and reset by software.

0: GPIO O disabled (default after reset)

1: GPIO O enabled

Bit 13 **GPIONEN**: GPIO N enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with GPIONENS, and cleared with GPIONENC. This bit is set and reset by software.

0: GPIO N disabled (default after reset)

1: GPIO N enabled

Bits 12:8 Reserved, must be kept at reset value.

Bit 7 **GPIOHEN**: GPIO H enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with GPIOHENS, and cleared with GPIOHENC. This bit is set and reset by software.

0: GPIO H disabled (default after reset)

1: GPIO H enabled

Bit 6 **GPIOGEN**: GPIO G enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with GPIOGENS, and cleared with GPIOGENC. This bit is set and reset by software.

0: GPIO G disabled (default after reset)

1: GPIO G enabled

Bit 5 **GPIOFEN**: GPIO F enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. t can be set with GPIOFENS, and cleared with GPIOFENC. This bit is set and reset by software.

0: GPIO F disabled (default after reset)

1: GPIO F enabled

Bit 4 **GPIOEEN**: GPIO E enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with GPIOEENS, and cleared with GPIOEENC. This bit is set and reset by software.

0: GPIO E disabled (default after reset)

1: GPIO E enabled

Bit 3 **GPIODEN**: GPIO D enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with GPIODENS, and cleared with GPIODENC. This bit is set and reset by software.

0: GPIO D disabled (default after reset)

1: GPIO D enabled

Bit 2 **GPIOCEN**: GPIO C enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with GPIOCENS, and cleared with GPIOCENC. This bit is set and reset by software.

0: GPIO C disabled (default after reset)

1: GPIO C enabled

Bit 1 **GPIOBEN**: GPIO B enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with GPIOBENS, and cleared with GPIOBENC. This bit is set and reset by software.

0: GPIO B disabled (default after reset)

1: GPIO B enabled

Bit 0 **GPIOAEN**: GPIO A enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with GPIOAENS, and cleared with GPIOAENC. This bit is set and reset by software.

0: GPIO A disabled (default after reset)

1: GPIO A enabled

14.10.83 RCC AHB5 enable register (RCC_AHB5ENR)

Address offset: 0x260

Reset value: 0x0000 0000

This register is used to enable the AHB5 in Run and Sleep modes (in Sleep mode, each bit of this register is AND-ed with the LPEN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NPU EN	CACHE AXI EN	OTG2 EN	OTG PHY2 EN	OTG PHY1 EN	OTG1 EN	ETH1 EN	ETH1 RX EN	ETH1 TX EN	ETH1 MAC EN	Res.	GPU2D EN	GFX MMU EN	MCE4 EN	XSPI3 EN	MCE3 EN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCE2 EN	MCE1 EN	XSPIM EN	XSPI2 EN	Res.	Res.	Res.	SDMM C1 EN	SDMM C2 EN	PSSI EN	XSPI1 EN	FMC EN	JPEG EN	Res.	DMA2D EN	HP DMA1 EN
rw	rw	rw	rw				rw	rw	rw	rw	rw	rw		rw	rw

Bit 31 **NPUEN**: NPU enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with NPUENS, and cleared with NPUENC. This bit is set and reset by software.

0: NPU disabled (default after reset)

1: NPU enabled

Bit 30 **CACHEAXIEN**: CACHEAXI enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with CACHEAXIENS and cleared with CACHEAXIENC. This bit is set and reset by software.

0: CACHEAXI disabled (default after reset)

1: CACHEAXI enabled

Bit 29 **OTG2EN**: OTG2 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with OTG2ENS, and cleared with OTG2ENC. This bit is set and reset by software.

0: OTG2 i disabled (default after reset)

1: OTG2 enabled

Bit 28 **OTGPHY2EN**: OTGPHY2 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with OTGPHY2ENS, and cleared with OTGPHY2ENC. This bit is set and reset by software.

0: OTGPHY2 disabled (default after reset)

1: OTGPHY2 enabled

Bit 27 **OTGPHY1EN**: OTGPHY1 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, tor he PERPRIV bit. It can be set with OTGPHY1ENS, and cleared with OTGPHY1ENC. This bit is set and reset by software.

0: OTGPHY1 disabled (default after reset)

1: OTGPHY1 enabled

Bit 26 **OTG1EN**: OTG1 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with OTG1ENS, and cleared with OTG1ENC. This bit is set and reset by software.

0: OTG1 disabled (default after reset)

1: OTG1 enabled

Bit 25 **ETH1EN**: ETH1 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with ETH1ENS, and cleared with ETH1ENC. This bit is set and reset by software.

0: ETH1 disabled (default after reset)

1: ETH1 enabled

Bit 24 **ETH1RXEN**: ETH1RX enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with ETH1RXENS, and cleared with ETH1RXENC. This bit is set and reset by software.

0: ETH1RX disabled (default after reset)

1: ETH1RX enabled

Bit 23 **ETH1TXEN**: ETH1TX enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with ETH1TXENS, and cleared with ETH1TXENC. This bit is set and reset by software.

0: ETH1TX disabled (default after reset)

1: ETH1TX enabled

Bit 22 **ETH1MACEN**: ETH1MAC enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with ETH1MACENS, and cleared with ETH1MACENC. This bit is set and reset by software.

0: ETH1MAC disabled (default after reset)

1: ETH1MAC enabled

Bit 21 Reserved, must be kept at reset value.

Bit 20 **GPU2DEN**: GPU2D enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with GPU2DENS, and cleared with GPU2DENC. This bit is set and reset by software.

0: GP2DU2D disabled (default after reset)

1: GPU enabled

Bit 19 **GFXMMUEN**: GFXMMU enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with GFXMMUENS, and cleared with GFXMMUENC. This bit is set and reset by software.

0: GFXMMU disabled (default after reset)

1: GFXMMU enabled

Bit 18 **MCE4EN**: MCE4 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with MCE4ENS, and cleared with MCE4ENC. This bit is set and reset by software.

0: MCE4 disabled (default after reset)

1: MCE4 enabled

Bit 17 **XSPI3EN**: XSPI3 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with XSPI3ENS, and cleared with XSPI3ENC. This bit is set and reset by software.

0: XSPI3 disabled (default after reset)

1: XSPI3 enabled

Bit 16 **MCE3EN**: MCE3 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with MCE3ENS, and cleared with MCE3ENC. This bit is set and reset by software.

0: MCE3 disabled (default after reset)

1: MCE3 enabled

Bit 15 **MCE2EN**: MCE2 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with MCE2ENS, and cleared with MCE2ENC. This bit is set and reset by software.

0: MCE2 disabled (default after reset)

1: MCE2 enabled

Bit 14 **MCE1EN**: MCE1 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with MCE1ENS, and cleared with MCE1ENC. This bit is set and reset by software.

0: MCE1 disabled (default after reset)

1: MCE1 enabled

Bit 13 **XSPIMEN**: XSPIM enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with XSPIMENS, and cleared with XSPIMENC. This bit is set and reset by software.

0: XSPIM disabled (default after reset)

1: XSPIM enabled

Bit 12 **XSPI2EN**: XSPI2 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with XSPI2ENS, and cleared with XSPI2ENC. This bit is set and reset by software.

0: XSPI2 disabled (default after reset)

1: XSPI2 enabled

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **SDMMC1EN**: SDMMC1 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with SDMMC1ENS, and cleared with SDMMC1ENC. This bit is set and reset by software.

0: SDMMC1 disabled (default after reset)

1: SDMMC1 enabled

Bit 7 **SDMMC2EN**: SDMMC2 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with SDMMC2ENS, and cleared with SDMMC2ENC. This bit is set and reset by software.

0: SDMMC2 disabled (default after reset)

1: SDMMC2 enabled

Bit 6 **PSSIEN**: PSSI enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with PSSIENS, and cleared with PSSIENC. This bit is set and reset by software.

0: PSSI disabled (default after reset)

1: PSSI enabled

Bit 5 **XSPI1EN**: XSPI1 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with XSPI1ENS, and cleared with XSPI1ENC. This bit is set and reset by software.

0: XSPI1 disabled (default after reset)

1: XSPI1 enabled

Bit 4 **FMCEN**: FMC enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with FMCENS, and cleared with FMCENC. This bit is set and reset by software.

0: FMC disabled (default after reset)

1: FMC enabled

Bit 3 **JPEGEN**: JPEG enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with JPEGENS, and cleared with JPEGENC. This bit is set and reset by software.

0: JPEG disabled (default after reset)

1: JPEG enabled

Bit 2 Reserved, must be kept at reset value.

Bit 1 **DMA2DEN**: DMA2D enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with DMA2DENS, and cleared with DMA2DENC. This bit is set and reset by software.

0: DMA2D disabled (default after reset)

1: DMA2D enabled

Bit 0 **HPDMA1EN**: HPDMA1 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. it can be set with HPDMA1ENS, and cleared with HPDMA1ENC. This bit is set and reset by software.

0: HPDMA1 disabled (default after reset)

1: HPDMA1 enabled

14.10.84 RCC APB1L enable register (RCC_APB1LENR)

Address offset: 0x264

Reset value: 0x0000 0000

This register is used to enable the APB1L in Run and Sleep modes (in Sleep mode, each bit of this register is AND-ed with the LPEN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UART8EN	UART7EN	Res.	Res.	Res.	Res.	I3C2EN	I3C1EN	I2C3EN	I2C2EN	I2C1EN	UART5EN	UART4EN	USART3EN	USART2EN	SPDIFRX1EN
rw	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3EN	SPI2EN	TIM11EN	TIM10EN	WWDGEN	Res.	LPTIM1EN	TIM14EN	TIM13EN	TIM12EN	TIM7EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2EN
rw	rw	rw	rw	rs		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UART8EN**: UART8 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with UART8ENS, and cleared with UART8ENC. This bit is set and reset by software.

0: UART8 disabled (default after reset)

1: UART8 enabled

Bit 30 **UART7EN**: UART7 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with UART7ENS, and cleared with UART7ENC. This bit is set and reset by software.

0: UART7 disabled (default after reset)

1: UART7 enabled

Bits 29:26 Reserved, must be kept at reset value.

Bit 25 **I3C2EN**: I3C2 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with I3C2ENS, and cleared with I3C2ENC. This bit is set and reset by software.

0: I3C2 disabled (default after reset)

1: I3C2 enabled

Bit 24 **I3C1EN**: I3C1 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with I3C1ENS, and cleared with I3C1ENC. This bit is set and reset by software.

0: I3C1 disabled (default after reset)

1: I3C1 enabled

Bit 23 **I2C3EN**: I2C3 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with I2C3ENS, and cleared with I2C3ENC. This bit is set and reset by software.

0: I2C3 disabled (default after reset)

1: I2C3 enabled

Bit 22 **I2C2EN**: I2C2 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with I2C2ENS, and cleared with I2C2ENC. This bit is set and reset by software.

0: I2C2 disabled (default after reset)

1: I2C2 enabled

Bit 21 **I2C1EN**: I2C1 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with I2C1ENS, and cleared with I2C1ENC. This bit is set and reset by software.

0: I2C1 disabled (default after reset)

1: I2C1 enabled

Bit 20 **UART5EN**: UART5 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit,. It can be set with UART5ENS, and cleared with UART5ENC. This bit is set and reset by software.

0: UART5 disabled (default after reset)

1: UART5 enabled

Bit 19 **UART4EN**: UART4 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with UART4ENS, and cleared with UART4ENC. This bit is set and reset by software.

0: UART4 disabled (default after reset)

1: UART4 enabled

Bit 18 **USART3EN**: USART3 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with USART3ENS, and cleared with USART3ENC. This bit is set and reset by software.

0: USART3 disabled (default after reset)

1: USART3 enabled

Bit 17 **USART2EN**: USART2 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with USART2ENS, and cleared with USART2ENC. This bit is set and reset by software.

0: USART2 disabled (default after reset)

1: USART2 enabled

Bit 16 **SPDIFRX1EN**: SPDIFRX1 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with SPDIFRX1ENS, and cleared with SPDIFRX1ENC. This bit is set and reset by software.

0: SPDIFRX1 disabled (default after reset)

1: SPDIFRX1 enabled

Bit 15 **SPI3EN**: SPI3 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with SPI3ENS, and cleared with SPI3ENC. This bit is set and reset by software.

0: SPI3 disabled (default after reset)

1: SPI3 enabled

Bit 14 **SPI2EN**: SPI2 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with SPI2ENS, and cleared with SPI2ENC. This bit is set and reset by software.

0: SPI2 disabled (default after reset)

1: SPI2 enabled

Bit 13 **TIM11EN**: TIM11 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM11ENS, and cleared with TIM11ENC. This bit is set and reset by software.

0: TIM11 disabled (default after reset)

1: TIM11 enabled

Bit 12 **TIM10EN**: TIM10 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM10ENS, and cleared with TIM10ENC. This bit is set and reset by software.

0: TIM10 disabled (default after reset)

1: TIM10 enabled

Bit 11 **WWDGEN**: WWDG enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. it can be set with WWDGENS. This bit is set and reset by software.

0: WWDG is disabled (default after reset)

1: WWDG is enabled

Bit 10 Reserved, must be kept at reset value.

Bit 9 **LPTIM1EN**: LPTIM1 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with LPTIM1ENS, and cleared with LPTIM1ENC. This bit is set and reset by software.

0: LPTIM1 is disabled (default after reset)

1: LPTIM1 is enabled

Bit 8 **TIM14EN**: TIM14 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM14ENS, and cleared with TIM14ENC. This bit is set and reset by software.

0: TIM14 is disabled (default after reset)

1: TIM14 is enabled

Bit 7 **TIM13EN**: TIM13 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. it can be set with TIM13ENS, and cleared with TIM13ENC. This bit is set and reset by software.

0: TIM13 disabled (default after reset)

1: TIM13 enabled

Bit 6 **TIM12EN**: TIM12 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM12ENS, and cleared with TIM12ENC. This bit is set and reset by software.

0: TIM12 disabled (default after reset)

1: TIM12 enabled

Bit 5 **TIM7EN**: TIM7 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM7ENS, and cleared with TIM7ENC. This bit is set and reset by software.

0: TIM7 disabled (default after reset)

1: TIM7 enabled

Bit 4 **TIM6EN**: TIM6 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM6ENS, and cleared with TIM6ENC. This bit is set and reset by software.

0: TIM6 disabled (default after reset)

1: TIM6 enabled

Bit 3 **TIM5EN**: TIM5 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM5ENS, and cleared with TIM5ENC. This bit is set and reset by software.

0: TIM5 disabled (default after reset)

1: TIM5 enabled

Bit 2 **TIM4EN**: TIM4 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM4ENS, and cleared with TIM4ENC. This bit is set and reset by software.

0: TIM4 disabled (default after reset)

1: TIM4 enabled

Bit 1 **TIM3EN**: TIM3 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM3ENS, and cleared with TIM3ENC. This bit is set and reset by software.

0: TIM3 disabled (default after reset)

1: TIM3 enabled

Bit 0 **TIM2EN**: TIM2 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM2ENS, and cleared with TIM2ENC. This bit is set and reset by software.

0: TIM2 disabled (default after reset)

1: TIM2 enabled

14.10.85 RCC APB1H enable register (RCC_APB1HENR)

Address offset: 0x268

Reset value: 0x0000 0000

This register is used to enable the APB1H in Run and Sleep modes (in Sleep mode, each bit of this register is AND-ed with the LPEN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD1EN	Res.	Res.
													rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANEN	Res.	Res.	MDIOSEN	Res.	Res.	Res.	Res.	Res.
							rw			rw					

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **UCPD1EN**: UCPD1 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with UCPD1ENS, and cleared with UCPD1ENC. This bit is set and reset by software.

0: UCPD1 disabled (default after reset)

1: UCPD1 enabled

Bits 17:9 Reserved, must be kept at reset value.

Bit 8 **FDCANEN**: FDCAN enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with FDCANENS, and cleared with FDCANENC. This bit is set and reset by software.

0: FDCAN disabled (default after reset)

1: FDCAN enabled

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **MDIOSEN**: MDIOS enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with MDIOSENS, and cleared with MDIOSENC. This bit is set and reset by software.

0: MDIOS disabled (default after reset)

1: MDIOS enabled

Bits 4:0 Reserved, must be kept at reset value.

14.10.86 RCC APB2 enable register (RCC_APB2ENR)

Address offset: 0x26C

Reset value: 0x0000 0000

This register is used to enable the APB2 in Run and Sleep modes (in Sleep mode, each bit of this register is AND-ed with the LPEN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI2EN	SAI1EN	SPI5EN	TIM9EN	TIM17EN	TIM16EN	TIM15EN
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIM18EN	Res.	SPI4EN	SPI1EN	Res.	Res.	Res.	Res.	USART10EN	UART9EN	USART6EN	USART1EN	Res.	Res.	TIM8EN	TIM1EN
rw		rw	rw					rw	rw	rw	rw			rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **SAI2EN**: SAI2 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with SAI2ENS, and cleared with SAI2ENC. This bit is set and reset by software.

0: SAI2 disabled (default after reset)

1: SAI2 enabled

Bit 21 **SAI1EN**: SAI1 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with SAI1EN, and cleared with SAI1ENC. This bit is set and reset by software.

0: SAI1 disabled (default after reset)

1: SAI1 enabled

Bit 20 **SPI5EN**: SPI5 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with SPI5ENS, and cleared with SPI5ENC.This bit is set and reset by software.

0: SPI5 disabled (default after reset)

1: SPI5 enabled

Bit 19 **TIM9EN**: TIM9 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM9ENS, and cleared with TIM9ENC. This bit is set and reset by software.

0: TIM9 disabled (default after reset)

1: TIM9 enabled

Bit 18 **TIM17EN**: TIM17 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM17ENS, and cleared with TIM17ENC. This bit is set and reset by software.

0: TIM17 disabled (default after reset)

1: TIM17 enabled

Bit 17 **TIM16EN**: TIM16 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM16ENS, and cleared with TIM16ENC. This bit is set and reset by software.

0: TIM16 disabled (default after reset)

1: TIM16 enabled

Bit 16 **TIM15EN**: TIM15 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit,. It can be set with TIM15ENS, and cleared with TIM15ENC. This bit is set and reset by software.

0: TIM15 disabled (default after reset)

1: TIM15 enabled

Bit 15 **TIM18EN**: TIM18 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM18ENS, and cleared with TIM18ENC. This bit is set and reset by software.

0: TIM18 disabled (default after reset)

1: TIM18 enabled

Bit 14 Reserved, must be kept at reset value.

Bit 13 **SPI4EN**: SPI4 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with SPI4ENS, and cleared with SPI4ENC. This bit is set and reset by software.

0: SPI4 disabled (default after reset)

1: SPI4 enabled

Bit 12 **SPI1EN**: SPI1 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with SPI1ENS, and cleared with SPI1ENC. This bit is set and reset by software.

0: SPI1 disabled (default after reset)

1: SPI1 enabled

Bits 11:8 Reserved, must be kept at reset value.

Bit 7 **USART10EN**: USART10 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with USART10ENS, and cleared with USART10ENC. This bit is set and reset by software.

0: USART10 disabled (default after reset)

1: USART10 enabled

Bit 6 **UART9EN**: UART9 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with UART9ENS, and cleared with UART9ENC. This bit is set and reset by software.

0: UART9 disabled (default after reset)

1: UART9 enabled

Bit 5 **USART6EN**: USART6 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with USART6ENS, and cleared with USART6ENC. This bit is set and reset by software.

0: USART6 disabled (default after reset)

1: USART6 enabled

Bit 4 **USART1EN**: USART1 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with USART1EN,S and cleared with USART1ENC. This bit is set and reset by software.

0: USART1 disabled (default after reset)

1: USART1 enabled

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **TIM8EN**: TIM8 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM8ENS, and cleared with TIM8ENC. This bit is set and reset by software.

0: TIM8 disabled (default after reset)

1: TIM8 enabled

Bit 0 **TIM1EN**: TIM1 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM1ENS, and cleared with TIM1ENC. This bit is set and reset by software.

0: TIM1 disabled (default after reset)

1: TIM1 enabled

14.10.87 RCC APB3 enable register (RCC_APB3ENR)

Address offset: 0x270

Reset value: 0x0000 0000

This register is used to enable the APB3 in Run and Sleep modes (in Sleep mode, each bit of this register is AND-ed with the LPEN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

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

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

Bit 2 **DFTEN**: DFT enable

This bit is security-protected by DFTSEC or DFTPRIV. It can be set with DFTENS, and cleared with DFTENC. This bit is set and reset by software.

0: DFT disabled (default after reset)

1: DFT enabled

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

14.10.88 RCC APB4L enable register (RCC_APB4LENR)

Address offset: 0x274

Reset value: 0x0000 0000

This register is used to enable the APB4L in Run and Sleep modes (in Sleep mode, each bit of this register is AND-ed with the LPEN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCAPBEN	RTCEN
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VREFBUFEN	Res.	Res.	LPTIM5EN	LPTIM4EN	LPTIM3EN	LPTIM2EN	Res.	I2C4EN	Res.	SPI6EN	Res.	LPUART1EN	HDPEN	Res.	Res.
rw			rw	rw	rw	rw		rw		rw		rw	rw		

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

Bit 17 **RTCAPBEN**: RTCAPB enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with RTCAPBENS, and cleared with RTCAPBENC. This bit is set and reset by software.

0: RTCAPB disabled (default after reset)

1: RTCAPB enabled

Bit 16 **RTCEN**: RTC enable

This bit is write-protected by the pwr_lock_backup_n signal. It is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. This bit can be set with RTCENS, and cleared with RTCENC.

0: RTC disabled (default after reset)

1: RTC enabled

Bit 15 **VREFBUFEN**: VREFBUF enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with VREFBUFENS, and cleared with VREFBUFENC. This bit is set and reset by software.

0: VREFBUF disabled (default after reset)

1: VREFBUF enabled

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **LPTIM5EN**: LPTIM5 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with LPTIM5ENS, and cleared with LPTIM5ENC. This bit is set and reset by software.

0: LPTIM5 disabled (default after reset)

1: LPTIM5 enabled

Bit 11 **LPTIM4EN**: LPTIM4 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with LPTIM4ENS, and cleared with LPTIM4ENC. This bit is set and reset by software.

0: LPTIM4 disabled (default after reset)

1: LPTIM4 enabled

Bit 10 **LPTIM3EN**: LPTIM3 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with LPTIM3ENS, and cleared with LPTIM3ENC. This bit is set and reset by software.

0: LPTIM3 disabled (default after reset)

1: LPTIM3 enabled

Bit 9 **LPTIM2EN**: LPTIM2 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with LPTIM2ENS, and cleared with LPTIM2ENC. This bit is set and reset by software.

0: LPTIM2 disabled (default after reset)

1: LPTIM2 enabled

Bit 8 Reserved, must be kept at reset value.

Bit 7 **I2C4EN**: I2C4 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with I2C4ENS, and cleared with I2C4ENC. This bit is set and reset by software.

0: I2C4 disabled (default after reset)

1: I2C4 enabled

Bit 6 Reserved, must be kept at reset value.

Bit 5 **SPI6EN**: SPI6 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with SPI6ENS, and cleared with SPI6ENC. This bit is set and reset by software.

0: SPI6 disabled (default after reset)

1: SPI6 enabled

Bit 4 Reserved, must be kept at reset value.

Bit 3 **LPUART1EN**: LPUART1 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with LPUART1ENS, and cleared with LPUART1ENC. This bit is set and reset by software.

0: LPUART1 disabled (default after reset)

1: LPUART1 enabled

Bit 2 **HDPEN**: HDP enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with HDPENS, and cleared with HDPENC. This bit is set and reset by software.

0: HDP disabled (default after reset)

1: HDP enabled

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

14.10.89 RCC APB4H enable register (RCC_APB4HENR)

Address offset: 0x278

Reset value: 0x0000 0002

This register is used to enable the APB4H in Run and Sleep modes (in Sleep mode, each bit of this register is AND-ed with the LPEN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

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

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

Bit 2 **DTSEN**: DTS enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with DTSENS, and cleared with DTSENC. This bit is set and reset by software.

0: DTS disabled (default after reset)

1: DTS enabled

Bit 1 **BSECEN**: BSEC enable

This bit is always security-protected. It can be set with BSECENS, and cleared with BSECENC. This bit is set and reset by software.

0: BSEC disabled

1: BSEC enabled (default after reset)

Bit 0 **SYSCFGEN**: SYSCFG enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. IT can be set with SYSCFGENS, and cleared with SYSCFGENC. This bit is set and reset by software.

0: SYSCFG disabled (default after reset)

1: SYSCFG enabled

14.10.90 RCC APB5 enable register (RCC_APB5ENR)

Address offset: 0x27C

Reset value: 0x0000 0000

This register is used to enable the APB5 in boh Run and Sleep modes (in Sleep mode, each bit of this register is AND-ed with the LPEN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

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

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CSIEN**: CSI enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with CSIENS, and cleared with CSIENC. This bit is set and reset by software.

0: CSI disabled (default after reset)

1: CSI enabled

Bit 5 **VENCEN**: VENC enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with VENCENS, and cleared with VENCENC. This bit is set and reset by software.

0: VENC disabled (default after reset)

1: VENC enabled

Bit 4 **GFXTIMEN**: GFXTIM enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with GFXTIMENS, and cleared with GFXTIMENC. This bit is set and reset by software.

0: GFXTIM disabled (default after reset)

1: GFXTIM enabled

Bit 3 Reserved, must be kept at reset value.

Bit 2 **DCMIPPEN**: DCMIPP enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with DCMIPPENS, and cleared with DCMIPPENC. This bit is set and reset by software.

0: DCMIPP disabled (default after reset)

1: DCMIPP enabled

Bit 1 **LTDCEN**: LTDC enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with LTDCENS, and cleared with LTDCENC. This bit is set and reset by software.

0: LTDC disabled (default after reset)

1: LTDC enabled

Bit 0 Reserved, must be kept at reset value.

14.10.91 RCC embedded buses sleep enable register (RCC_BUSLPENR)

Address offset: 0x284

Reset value: 0x0000 0003

This register is used to enable the embedded buses in Sleep mode (each bit in this register is AND-ed with the EN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

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

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

Bit 1 **ACLKNCLPEN**: ACLKNC enable

This bit is security-protected by ACLKNCSEC or ACLKNCPRIV, and is publicly readable if ACLKNCPUB = 1. It can be set with ACLKNCLPENS, and cleared with ACLKNCLPENC. This bit is set and reset by software.

0: ACLKNC disabled in Sleep mode

1: ACLKNC enabled in Sleep mode (default after reset)

Bit 0 **ACLKNLPEN**: ACLKN enable

This bit is security-protected by ACLKNSEC or ACLKNPRIV, and is publicly readable if ACLKNPUB = 1. It can be set with ACLKNLPENS, and cleared with ACLKNLPENC. This bit is set and reset by software.

0: ACLKN disabled in Sleep mode

1: ACLKN enabled in Sleep mode (default after reset)

14.10.92 RCC miscellaneous configurations sleep enable register (RCC_MISCLPENR)

Address offset: 0x288

Reset value: 0x0000 0000

This register is used to enable the miscellaneous configurations in Sleep mode (each bit of this register is AND-ed with the EN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

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

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **PERLPEN**: PER enable

This bit is security-protected by PERSEC or PERPRIV. It can be set with PERLPENS, and cleared with PERLPENC. This bit is set and reset by software.

0: PER disabled in Sleep mode (default after reset)

1: PER enabled in Sleep mode

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **XSPIPHYCOMPLPEN**: XSPIPHYCOMP enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with XSPIPHYCOMPLPENS, and cleared with XSPIPHYCOMPLPENC. This bit is set and reset by software.

0: XSPIPHYCOMP disabled in Sleep mode (default after reset)

1: XSPIPHYCOMP enabled in Sleep mode

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **DBGLPEN**: DBG enable

This bit is always security-protected. It can be set with DBGLPENS, and cleared with DBGLPENC. This bit is set and reset by software.

0: DBG disabled in Sleep mode (default after reset)

1: DBG enabled in Sleep mode

14.10.93 RCC embedded memories sleep enable register (RCC_MEMLPENR)

Address offset: 0x28C

Reset value: 0x0000 0000

This register is used to enable the embedded memories in Sleep mode (each bit of this register is AND-ed with the EN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	BOOTROMLPEN	VENCRAMLPEN	CACHEAXIRAMLPEN	FLEXRAMLPEN	AXISRAM2LPEN	AXISRAM1LPEN	BKPSRAMLPEN	AHBSRAM2LPEN	AHBSRAM1LPEN	AXISRAM6LPEN	AXISRAM5LPEN	AXISRAM4LPEN	AXISRAM3LPEN
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **BOOTROMLPEN**: BootROM enable

This bit is always security-protected. It can be set with BOOTROMLPENS, and cleared with BOOTROMLPENC. This bit is set and reset by software.

0: BootROM disabled in Sleep mode (default after reset)

1: BootROM enabled in Sleep mode

Bit 11 **VENCRAMLPEN**: VENCRAM enable

This bit is security-protected by a SEC or a PRIV signal from RIFSC, and is publicly readable if VENCRAMPUB = 1. It can be set with VENCRAMLPENS, and cleared with VENCRAMLPENC. This bit is set and reset by software.

0: VENCRAM disabled in Sleep mode (default after reset)

1: VENCRAM enabled in Sleep mode

Bit 10 **CACHEAXIRAMLPEN**: CACHEAXIRAM enable

This bit is security-protected by a SEC or a PRIV signal from RIFSC, and is publicly readable if CACHEAXIRAMPUB = 1. It can be set with CACHEAXIRAMLPENS and cleared with CACHEAXIRAMLPENC. This bit is set and reset by software.

0: CACHEAXIRAM disabled in Sleep mode (default after reset)

1: CACHEAXIRAM enabled in Sleep mode

Bit 9 **FLEXRAMLPEN**: FLEXRAM enable

This bit is security-protected by a SEC or a PRIV signal from RIFSC, and is publicly readable if FLEXRAMPUB = 1. It can be set with FLEXRAMLPENS, and cleared with FLEXRAMLPENC. This bit is set and reset by software.

0: FLEXRAM disabled in Sleep mode (default after reset)

1: FLEXRAM enabled in Sleep mode

Bit 8 **AXISRAM2LPEN**: AXISRAM2 enable

This bit is security-protected by a SEC or a PRIV signal from RIFSC, and is publicly readable if AXISRAM2PUB = 1. It can be set with AXISRAM2LPENS, and cleared with AXISRAM2LPENC.This bit is set and reset by software.

0: AXISRAM2 disabled in Sleep mode (default after reset)

1: AXISRAM2 enabled in Sleep mode

Bit 7 **AXISRAM1LPEN**: AXISRAM1 enable

This bit is security-protected by a SEC or a PRIV signal from RIFSC, and is publicly readable if AXISRAM1PUB = 1. It can be set with AXISRAM1LPENS, and cleared with AXISRAM1LPENC.This bit is set and reset by software.

0: AXISRAM1 disabled in Sleep mode (default after reset)

1: AXISRAM1 enabled in Sleep mode

Bit 6 **BKPSRAMLPEN**: BKPSRAM enable

This bit is security-protected by a SEC or a PRIV signal from RIFSC, and is publicly readable if BKPSRAMPUB = 1. It can be set with BKPSRAMLPENS, and cleared with BKPSRAMLPENC. This bit is set and reset by software.

0: BKPSRAM disabled in Sleep mode (default after reset)

1: BKPSRAM enabled in Sleep mode

Bit 5 **AHBSRAM2LPEN**: AHBSRAM2 enable

This bit is security-protected by a SEC or a PRIV signal from RIFSC, and is publicly readable if AHBSRAM2PUB = 1. It can be set with AHBSRAM2LPENS, and cleared with AHBSRAM2LPENC. This bit is set and reset by software.

0: AHBSRAM2 disabled in Sleep mode (default after reset)

1: AHBSRAM2 enabled in Sleep mode

Bit 4 **AHBSRAM1LPEN**: AHBSRAM1 enable

This bit is security-protected by a SEC or a PRIV signal from RIFSC, and is publicly readable if AHBSRAM1PUB = 1. It can be set with AHBSRAM1LPENS, and cleared with AHBSRAM1LPENC. This bit is set and reset by software.

0: AHBSRAM1 disabled in Sleep mode (default after reset)

1: AHBSRAM1 enabled in Sleep mode

Bit 3 **AXISRAM6LPEN**: AXISRAM6 enable

This bit is security-protected by a SEC or a PRIV signal from RIFSC, and is publicly readable if AXISRAM6PUB = 1. It can be set with AXISRAM6LPENS, and cleared with AXISRAM6LPENC. This bit is set and reset by software.

0: AXISRAM6 disabled in Sleep mode (default after reset)

1: AXISRAM6 enabled in Sleep mode

Bit 2 **AXISRAM5LPEN**: AXISRAM5 enable

This bit is security-protected by a SEC or a PRIV signal from RIFSC, and is publicly readable if AXISRAM5PUB = 1. It can be set with AXISRAM5LPENS, and cleared with AXISRAM5LPENC. This bit is set and reset by software.

0: AXISRAM5 disabled in Sleep mode (default after reset)

1: AXISRAM5 enabled in Sleep mode

Bit 1 **AXISRAM4LPEN**: AXISRAM4 enable

This bit is security-protected by a SEC or a PRIV signal from RIFSC, and is publicly readable if AXISRAM4PUB = 1. It can be set with AXISRAM4LPENS, and cleared with AXISRAM4LPENC. This bit is set and reset by software.

0: AXISRAM4 disabled in Sleep mode (default after reset)

1: AXISRAM4 enabled in Sleep mode

Bit 0 **AXISRAM3LPEN**: AXISRAM3 enable

This bit is security-protected by a SEC or a PRIV signal from RIFSC, and is publicly readable if AXISRAM3PUB = 1. It can be set with AXISRAM3LPENS, and cleared with AXISRAM3LPENC. This bit is set and reset by software.

0: AXISRAM3 disabled in Sleep mode (default after reset)

1: AXISRAM3 enabled in Sleep mode

14.10.94 RCC AHB1 sleep enable register (RCC_AHB1LPENR)

Address offset: 0x290

Reset value: 0x0000 0000

This register is used to enable the AHB1 in Sleep mode (each bit in this register is AND-ed with the EN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC12LPEN	GPDMA1LPEN	Res.	Res.	Res.	Res.
										rw	rw				

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

Bit 5 **ADC12LPEN**: ADC12 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with ADC12LPENS, and cleared with ADC12LPENC. This bit is set and reset by software.

0: ADC12 disabled in Sleep mode (default after reset)

1: ADC12 enabled in Sleep mode

Bit 4 **GPDMA1LPEN**: GPDMA1 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with GPDMA1LPENS, and cleared with GPDMA1LPENC. This bit is set and reset by software.

0: GPDMA1 disabled in Sleep mode (default after reset)

1: GPDMA1 enabled in Sleep mode

Bits 3:0 Reserved, must be kept at reset value.

14.10.95 RCC AHB2 sleep enable register (RCC_AHB2LPENR)

Address offset: 0x294

Reset value: 0x0000 0000

This register is used to enable the AHB2 in Sleep mode (each bit in this register is AND-ed with the EN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADF1LPEN	MDF1LPEN
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	RAMCFGLPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			rw												

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

Bit 17 **ADF1LPEN**: ADF1 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with ADF1LPENS, and cleared with ADF1LPENC. This bit is set and reset by software.

0: ADF1 disabled in Sleep mode (default after reset)

1: ADF1 enabled in Sleep mode

Bit 16 **MDF1LPEN**: MDF1 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with MDF1LPENS, and cleared with MDF1LPENC. This bit is set and reset by software.

0: MDF1 disabled in Sleep mode (default after reset)

1: MDF1 enabled in Sleep mode

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **RAMCFGLPEN**: RAMCFG enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with RAMCFGLPENS, and cleared with RAMCFGLPENC. This bit is set and reset by software.

0: RAMCFG disabled in Sleep mode (default after reset)

1: RAMCFG enabled in Sleep mode

Bits 11:0 Reserved, must be kept at reset value.

14.10.96 RCC AHB3 sleep enable register (RCC_AHB3LPENR)

Address offset: 0x298

Reset value: 0x0000 0400

This register is used to enable the AHB3 in Sleep mode (each bit of this register is AND-ed with the EN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RISAFLPEN	Res.	Res.	Res.	IACLPEN	RIFSCLPEN	PKALPEN	Res.	Res.	Res.	SAESLPEN	Res.	CRYPLPEN	HASHLPEN	RNGLPEN
	rw				rw	rw	rw				rw		rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **RISAFLPEN**: RISAF enable

This bit is always security-protected. It can be set with RISAFLPENS, and cleared with RISAFLPENC. This bit is set and reset by software.

0: RISAF disabled in Sleep mode (default after reset)

1: RISAF enabled in Sleep mode

Bits 13:11 Reserved, must be kept at reset value.

Bit 10 **IACLPEN**: IAC enable

This bit is always security-protected. It can be set with IACLPENS, and cleared with IACLPENC. This bit is set and reset by software.

0: IAC disabled in Sleep mode

1: IAC enabled in Sleep mode (default after reset)

Bit 9 **RIFSCLPEN**: RIFSC enable

This bit is always security-protected. It can be set with RIFSCLPENS, and cleared with RIFSCLPENC. This bit is set and reset by software.

0: RIFSC disabled in Sleep mode (default after reset)

1: RIFSC enabled in Sleep mode

Bit 8 **PKALPEN**: PKA enable

This bit is security-protected by SYSSEC or SYSPRIV. It can be set with PKALPENS, and cleared with PKALPENC. This bit is set and reset by software.

0: PKA disabled in Sleep mode (default after reset)

1: PKA enabled in Sleep mode

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **SAESLPEN**: SAES enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with SAESLPENS, and cleared with SAESLPENC. This bit is set and reset by software.

0: SAES disabled in Sleep mode (default after reset)

1: SAES enabled in Sleep mode

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CRYPLPEN**: CRYP enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with CRYPLPENS, and cleared with CRYPLPENC. This bit is set and reset by software.

0: CRYP disabled in Sleep mode (default after reset)

1: CRYP enabled in Sleep mode

Bit 1 **HASHLPEN**: HASH enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with HASHLPENS, and cleared with HASHLPENC. This bit is set and reset by software.

0: HASH disabled in Sleep mode (default after reset)

1: HASH enabled in Sleep mode

Bit 0 **RNGLPEN**: RNG enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with RNGLPENS, and cleared with RNGLPENC. This bit is set and reset by software.

0: RNG disabled in Sleep mode (default after reset)

1: RNG enabled in Sleep mode

14.10.97 RCC AHB4 sleep enable register (RCC_AHB4LPENR)

Address offset: 0x29C

Reset value: 0x0004 0000

This register is used to enable the AHB4 in Sleep mode (each bit of this register is AND-ed with the EN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCLPEN	PWRLPEN	Res.	GPIOQLPEN
												rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIOPLPEN	GPIOOLPEN	GPIONLPEN	Res.	Res.	Res.	Res.	Res.	GPIOHLPEN	GPIOGLPEN	GPIOFLPEN	GPIOELPEN	GPIODLPEN	GPIOCLPEN	GPIOBLPEN	GPIOALPEN
rw	rw	rw						rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **CRCLPEN**: CRC enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with CRCLPENS, and cleared with CRCLPENC. This bit is set and reset by software.

0: CRC disabled in Sleep mode (default after reset)

1: CRC enabled in Sleep mode

Bit 18 **PWRLPEN**: PWR enable

This bit is always security-protected. It can be set with PWRLPENS, and cleared with PWRLPENC.This bit is set and reset by software.

0: PWR disabled in Sleep mode

1: PWR enabled in Sleep mode (default after reset)

Bit 17 Reserved, must be kept at reset value.

Bit 16 **GPIOQLPEN**: GPIO Q enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with GPIOQLPENS, and cleared with GPIOQLPENC. This bit is set and reset by software.

0: GPIO Q disabled in Sleep mode (default after reset)

1: GPIO Q enabled in Sleep mode

Bit 15 **GPIOPLPEN**: GPIO P enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with GPIOPLPENS, and cleared with GPIOPLPENC. This bit is set and reset by software.

0: GPIO P disabled in Sleep mode (default after reset)

1: GPIO P enabled in Sleep mode

Bit 14 **GPIOOLPEN**: GPIO O enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with GPIOOLPENS, and cleared with GPIOOLPENC. This bit is set and reset by software.

0: GPIO O disabled in Sleep mode (default after reset)/

1: GPIO O enabled in Sleep mode

Bit 13 **GPIONLPEN**: GPIO N enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with GPIONLPENS, and cleared with GPIONLPENC. This bit is set and reset by software.

0: GPIO N disabled in Sleep mode (default after reset)

1: GPIO N enabled in Sleep mode

Bits 12:8 Reserved, must be kept at reset value.

Bit 7 **GPIOHLPEN**: GPIO H enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with GPIOHLPENS, and cleared with GPIOHLPENC. This bit is set and reset by software.

0: GPIO H disabled in Sleep mode (default after reset)

1: GPIO H enabled in Sleep mode

Bit 6 **GPIOGLPEN**: GPIO G enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with GPIOGLPENS, and cleared with GPIOGLPENC. This bit is set and reset by software.

0: GPIO G disabled in Sleep mode (default after reset)

1: GPIO G enabled in Sleep mode

Bit 5 **GPIOFLPEN**: GPIO F enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with GPIOFLPENS, and cleared with GPIOFLPENC. This bit is set and reset by software.

0: GPIO F disabled in Sleep mode (default after reset)

1: GPIO F enabled in Sleep mode

Bit 4 **GPIOELPEN**: GPIO E enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with GPIOELPENS, and cleared with GPIOELPENC. This bit is set and reset by software.

0: GPIO E disabled in Sleep mode (default after reset)

1: GPIO E enabled in Sleep mode

Bit 3 **GPIODLPEN**: GPIO D enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. it can be set with GPIODLPENS, and cleared with GPIODLPENC. This bit is set and reset by software.

0: GPIO D disabled in Sleep mode (default after reset)

1: GPIO D enabled in Sleep mode

Bit 2 **GPIOCLPEN**: GPIO C enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with GPIOCLPENS, and cleared with GPIOCLPENC. This bit is set and reset by software.

0: GPIO C disabled in Sleep mode (default after reset)

1: GPIO C enabled in Sleep mode

Bit 1 **GPIOBLPEN**: GPIO B enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with GPIOBLPENS, and cleared with GPIOBLPENC. This bit is set and reset by software.

0: GPIO B disabled in Sleep mode (default after reset)

1: GPIO B enabled in Sleep mode

Bit 0 **GPIOALPEN**: GPIO A enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with GPIOALPENS, and cleared with GPIOALPENC. This bit is set and reset by software.

0: GPIO A disabled in Sleep mode (default after reset)

1: GPIO A enabled in Sleep mode

14.10.98 RCC AHB5 sleep enable register (RCC_AHB5LPENR)

Address offset: 0x2A0

Reset value: 0x0000 0000

This register is used to enable the AHB5 in Sleep mode (each bit of this register is AND-ed with the EN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NPULPEN	CACHEAXILPEN	OTG2LPEN	OTGPHY2LPEN	OTGPHY1LPEN	OTG1LPEN	ETH1LPEN	ETH1RXLPEN	ETH1TXLPEN	ETH1MACLPEN	Res.	GPU2DLPEN	GFXMMULPEN	MCE4LPEN	XSPI3LPEN	MCE3LPEN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCE2LPEN	MCE1LPEN	XSPIMLPEN	XSPI2LPEN	Res.	Res.	Res.	SDMMC1LPEN	SDMMC2LPEN	PSSILPEN	XSPI1LPEN	FMCLPEN	JPEGLPEN	Res.	DMA2DLPEN	HPDMA1LPEN
rw	rw	rw	rw				rw	rw	rw	rw	rw	rw		rw	rw

Bit 31 **NPULPEN**: NPU enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with NPULPENS, and cleared with NPULPENC. This bit is set and reset by software.

0: NPU disabled in Sleep mode (default after reset)

1: NPU enabled in Sleep mode

Bit 30 **CACHEAXILPEN**: CACHEAXI enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with CACHEAXILPENS, and cleared with CACHEAXILPENC. This bit is set and reset by software.

0: CACHEAXI disabled in Sleep mode (default after reset)

1: CACHEAXI enabled in Sleep mode

Bit 29 **OTG2LPEN**: OTG2 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with OTG2LPENS, and cleared with OTG2LPENC. This bit is set and reset by software.

0: OTG2 disabled in Sleep mode (default after reset)

1: OTG2 enabled in Sleep mode

Bit 28 **OTGPHY2LPEN**: OTGPHY2 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with OTGPHY2LPENS, and cleared with OTGPHY2LPENC. This bit is set and reset by software.

0: OTGPHY2 disabled in Sleep mode (default after reset)

1: OTGPHY2 enabled in Sleep mode

Bit 27 **OTGPHY1LPEN**: OTGPHY1 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with OTGPHY1LPENS, and cleared with OTGPHY1LPENC. This bit is set and reset by software.

0: OTGPHY1 disabled in Sleep mode (default after reset)

1: OTGPHY1 enabled in Sleep mode

Bit 26 **OTG1LPEN**: OTG1 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with OTG1LPENS, and cleared with OTG1LPENC. This bit is set and reset by software.

0: OTG1 disabled in Sleep mode (default after reset)

1: OTG1 enabled in Sleep mode

Bit 25 **ETH1LPEN**: ETH1 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with ETH1LPENS, and cleared with ETH1LPENC. This bit is set and reset by software.

0: ETH1 disabled in Sleep mode (default after reset)

1: ETH1 enabled in Sleep mode

Bit 24 **ETH1RXLPEN**: ETH1RX enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. it can be set with ETH1RXLPENS, and cleared with ETH1RXLPENC. This bit is set and reset by software.

0: ETH1RX disabled in Sleep mode (default after reset)

1: ETH1RX enabled in Sleep mode

Bit 23 **ETH1TXLPEN**: ETH1TX enable

This bit is security-protected by a SEC signal from RIFSC, th,e PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with ETH1TXLPENS and cleared with ETH1TXLPENC. This bit is set and reset by software.

0: ETH1TX disabled in Sleep mode (default after reset)

1: ETH1TX enabled in Sleep mode

Bit 22 **ETH1MACLPEN**: ETH1MAC enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with ETH1MACLPENS, and cleared with ETH1MACLPENC. This bit is set and reset by software.

0: ETH1MAC disabled in Sleep mode (default after reset)

1: ETH1MAC enabled in Sleep mode

Bit 21 Reserved, must be kept at reset value.

Bit 20 **GPU2DLPEN**: GPU2D enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. it can be set with GPU2DLPENS, and cleared with GPU2DLPENC. This bit is set and reset by software.

0: GPU2D disabled in Sleep mode (default after reset)

1: GPU2D enabled in Sleep mode

Bit 19 **GFXMMULPEN**: GFXMMU enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. it can be set with GFXMMULPENS, and cleared with GFXMMULPENC. This bit is set and reset by software.

0: GFXMMU disabled in Sleep mode (default after reset)

1: GFXMMU enabled in Sleep mode

Bit 18 **MCE4LPEN**: MCE4 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. it can be set with MCE4LPENS, and cleared with MCE4LPENC. This bit is set and reset by software.

0: MCE4 disabled in Sleep mode (default after reset)

1: MCE4 enabled in Sleep mode

Bit 17 **XSPI3LPEN**: XSPI3 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with XSPI3LPENS, and cleared with XSPI3LPENC. This bit is set and reset by software.

0: XSPI3 disabled in Sleep mode (default after reset)

1: XSPI3 enabled in Sleep mode

Bit 16 **MCE3LPEN**: MCE3 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. it can be set with MCE3LPENS, and cleared with MCE3LPENC. This bit is set and reset by software.

0: MCE3 disabled in Sleep mode (default after reset)

1: MCE3 enabled in Sleep mode

Bit 15 **MCE2LPEN**: MCE2 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with MCE2LPENS, and cleared with MCE2LPENC. This bit is set and reset by software.

0: MCE2 disabled in Sleep mode (default after reset)

1: MCE2 enabled in Sleep mode

Bit 14 **MCE1LPEN**: MCE1 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with MCE1LPENS, and cleared with MCE1LPENC. This bit is set and reset by software.

0: MCE1 disabled in Sleep mode (default after reset)

1: MCE1 enabled in Sleep mode

Bit 13 **XSPIMLPEN**: XSPIM enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. it can be set with XSPIMLPENS, and cleared with XSPIMLPENC.This bit is set and reset by software.

0: XSPIM disabled in Sleep mode (default after reset)

1: XSPIM enabled in sleep mode

Bit 12 **XSPI2LPEN**: XSPI2 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with XSPI2LPENS, and cleared with XSPI2LPENC. This bit is set and reset by software.

0: XSPI2 disabled in Sleep mode (default after reset)

1: XSPI2 enabled in Sleep mode

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **SDMMC1LPEN**: SDMMC1 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. it can be set with SDMMC1LPENS, and cleared with SDMMC1LPENC. This bit is set and reset by software.

0: SDMMC1 disabled in Sleep mode (default after reset)

1: SDMMC1 enabled in Sleep mode

Bit 7 **SDMMC2LPEN**: SDMMC2 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with SDMMC2LPENS, and cleared with SDMMC2LPENC. This bit is set and reset by software.

0: SDMMC2 disabled in Sleep mode (default after reset)

1: SDMMC2 enabled in Sleep mode

Bit 6 **PSSILPEN**: PSSI enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. it can be set with PSSILPENS, and cleared with PSSILPENC. This bit is set and reset by software.

0: PSSI disabled in Sleep mode (default after reset)

1: PSSI enabled in Sleep mode

Bit 5 **XSPI1LPEN**: XSPI1 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with XSPI1LPENS, and cleared with XSPI1LPENC. This bit is set and reset by software.

0: XSPI1 disabled in Sleep mode (default after reset)

1: XSPI1 enabled in Sleep mode

Bit 4 **FMCLPEN**: FMC enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. it can be set with FMCLPENS, and cleared with FMCLPENC. This bit is set and reset by software.

0: FMC disabled in Sleep mode (default after reset)

1: FMC enabled in Sleep mode

Bit 3 **JPEGLPEN**: JPEG enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with JPEGLPENS, and cleared with JPEGLPENC. This bit is set and reset by software.

0: JPEG disabled in Sleep mode (default after reset)

1: JPEG enabled in Sleep mode

Bit 2 Reserved, must be kept at reset value.

Bit 1 **DMA2DLPEN**: DMA2D enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with DMA2DLPENS, and cleared with DMA2DLPENC. This bit is set and reset by software.

0: DMA2D disabled in Sleep mode (default after reset)

1: DMA2D enabled in Sleep mode

Bit 0 **HPDMA1LPEN**: HPDMA1 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. it can be set with HPDMA1LPENS, and cleared with HPDMA1LPENC. This bit is set and reset by software.

0: HPDMA1 disabled in Sleep mode (default after reset)

1: HPDMA1 enabled in Sleep mode

14.10.99 RCC APB1L sleep enable register (RCC_APB1LLPENR)

Address offset: 0x2A4

Reset value: 0x0000 0000

This register is used to enable the APB1L in Sleep mode (each bit of this register is AND-ed with the EN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UART8 LPEN	UART7 LPEN	Res.	Res.	Res.	Res.	I3C2LP EN	I3C1LP EN	I2C3LP EN	I2C2LP EN	I2C1LP EN	UART5 LPEN	UART4 LPEN	USART 3LPEN	USART 2LPEN	SPDIF RX1LP EN
rw	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3LP EN	SPI2LP EN	TIM11L PEN	TIM10L PEN	WWDG LPEN	Res.	LPTIM1 LPEN	TIM14L PEN	TIM13L PEN	TIM12L PEN	TIM7LP EN	TIM6LP EN	TIM5LP EN	TIM4LP EN	TIM3LP EN	TIM2LP EN
rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UART8LPEN**: UART8 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. it can be set with UART8LPENS, and cleared with UART8LPENC. This bit is set and reset by software.

0: UART8 disabled in Sleep mode (default after reset)

1: UART8 enabled in Sleep mode

Bit 30 **UART7LPEN**: UART7 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. it can be set with UART7LPENS, and cleared with UART7LPENC. This bit is set and reset by software.

0: UART7 disabled in Sleep mode (default after reset)

1: UART7 enabled in Sleep mode

Bits 29:26 Reserved, must be kept at reset value.

Bit 25 **I3C2LPEN**: I3C2 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. it can be set with I3C2LPENS, and cleared with I3C2LPENC. This bit is set and reset by software.

0: I3C2 disabled in Sleep mode (default after reset)

1: I3C2 enabled in Sleep mode

Bit 24 **I3C1LPEN**: I3C1 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with I3C1LPENS, and cleared with I3C1LPENC. This bit is set and reset by software.

0: I3C1 disabled in Sleep mode (default after reset)

1: I3C1 enabled in Sleep mode

Bit 23 **I2C3LPEN**: I2C3 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with I2C3LPENS, and cleared with I2C3LPENC. This bit is set and reset by software.

0: I2C3 disabled in Sleep mode (default after reset)

1: I2C3 enabled in Sleep mode

Bit 22 **I2C2LPEN**: I2C2 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with I2C2LPENS, and cleared with I2C2LPENC. This bit is set and reset by software.

0: I2C2 disabled in Sleep mode (default after reset)

1: I2C2 enabled in Sleep mode

Bit 21 **I2C1LPEN**: I2C1 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with I2C1LPENS, and cleared with I2C1LPENC. This bit is set and reset by software.

0: I2C1 disabled in Sleep mode (default after reset)

1: I2C1 enabled in Sleep mode

Bit 20 **UART5LPEN**: UART5 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with UART5LPENS, and cleared with UART5LPENC. This bit is set and reset by software.

0: UART5 disabled in Sleep mode (default after reset)

1: UART5 enabled in Sleep mode

Bit 19 **UART4LPEN**: UART4 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with UART4LPENS, and cleared with UART4LPENC. This bit is set and reset by software.

0: UART4 disabled in Sleep mode (default after reset)

1: UART4 enabled in Sleep mode

Bit 18 **USART3LPEN**: USART3 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with USART3LPENS, and cleared with USART3LPENC. This bit is set and reset by software.

0: USART3 disabled in Sleep mode (default after reset)

1: USART3 enabled in Sleep mode

Bit 17 **USART2LPEN**: USART2 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with USART2LPENS, and cleared with USART2LPENC. This bit is set and reset by software.

0: USART2 disabled in Sleep mode (default after reset)

1: USART2 enabled in Sleep mode

Bit 16 **SPDIFRX1LPEN**: SPDIFRX1 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with SPDIFRX1LPENS, and cleared with SPDIFRX1LPENC. This bit is set and reset by software.

0: SPDIFRX1 disabled in Sleep mode (default after reset)

1: SPDIFRX1 enabled in Sleep mode

Bit 15 **SPI3LPEN**: SPI3 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with SPI3LPENS, and cleared with SPI3LPENC. This bit is set and reset by software.

0: SPI3 disabled in Sleep mode (default after reset)

1: SPI3 enabled in Sleep mode

Bit 14 **SPI2LPEN**: SPI2 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with SPI2LPENS, and cleared with SPI2LPENC. This bit is set and reset by software.

0: SPI2 disabled in Sleep mode (default after reset)

1: SPI2 enabled in Sleep mode

Bit 13 **TIM11LPEN**: TIM11 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM11LPENS, and cleared with TIM11LPENC. This bit is set and reset by software.

0: TIM11 disabled in Sleep mode (default after reset)

1: TIM11 enabled in Sleep mode

Bit 12 **TIM10LPEN**: TIM10 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM10LPENS, and cleared with TIM10LPENC. This bit is set and reset by software.

0: TIM10 disabled in Sleep mode (default after reset)

1: TIM10 enabled in Sleep mode

Bit 11 **WWDGLPEN**: WWDG enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with WWDGLPENS, and cleared with WWDGLPENC. This bit is set and reset by software.

0: WWDG disabled in Sleep mode (default after reset)

1: WWDG enabled in Sleep mode

Bit 10 Reserved, must be kept at reset value.

Bit 9 **LPTIM1LPEN**: LPTIM1 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with LPTIM1LPENS, and cleared with LPTIM1LPENC. This bit is set and reset by software.

0: LPTIM1 disabled in Sleep mode (default after reset)

1: LPTIM1 enabled in Sleep mode

Bit 8 **TIM14LPEN**: TIM14 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM14LPENS, and cleared with TIM14LPENC. This bit is set and reset by software.

0: TIM14 disabled in Sleep mode (default after reset)

1: TIM14 enabled in Sleep mode

Bit 7 **TIM13LPEN**: TIM13 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM13LPEN,S and cleared with TIM13LPENC. This bit is set and reset by software.

0: TIM13 disabled in Sleep mode (default after reset)

1: TIM13 enabled in Sleep mode

Bit 6 **TIM12LPEN**: TIM12 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM12LPENS, and cleared with TIM12LPENC. This bit is set and reset by software.

0: TIM12 disabled in Sleep mode (default after reset)

1: TIM12 enabled in Sleep mode

Bit 5 **TIM7LPEN**: TIM7 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM7LPENS, and cleared with TIM7LPENC. This bit is set and reset by software.

0: TIM7 disabled in Sleep mode (default after reset)

1: TIM7 enabled in Sleep mode

Bit 4 **TIM6LPEN**: TIM6 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM6LPENS, and cleared with TIM6LPENC. This bit is set and reset by software.

0: TIM6 disabled in Sleep mode (default after reset)

1: TIM6 enabled in Sleep mode

Bit 3 **TIM5LPEN**: TIM5 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM5LPENS, and cleared with TIM5LPENC. This bit is set and reset by software.

0: TIM5 disabled in Sleep mode (default after reset)

1: TIM5 enabled in Sleep mode

Bit 2 **TIM4LPEN**: TIM4 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM4LPENS, and cleared with TIM4LPENC. This bit is set and reset by software.

0: TIM4 disabled in Sleep mode (default after reset)

1: TIM4 enabled in Sleep mode

Bit 1 **TIM3LPEN**: TIM3 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM3LPENS, and cleared with TIM3LPENC. This bit is set and reset by software.

0: TIM3 disabled in Sleep mode (default after reset)

1: TIM3 enabled in Sleep mode

Bit 0 **TIM2LPEN**: TIM2 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM2LPENS, and cleared with TIM2LPENC. This bit is set and reset by software.

0: TIM2 disabled in Sleep mode (default after reset)

1: TIM2 enabled in Sleep mode

14.10.100 RCC APB1H sleep enable register (RCC_APB1HLPENR)

Address offset: 0x2A8

Reset value: 0x0000 0000

This register is used to enable the APB1H in Sleep mode (each bit of this register is AND-ed with the EN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD1 LPEN	Res.	Res.
													rw		

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

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **UCPD1LPEN**: UCPD1 enable in Sleep mode

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with UCPD1LPENS, and cleared with UCPD1LPENC. This bit is set and reset by software.

0: UCPD1 disabled in Sleep mode (default after reset)

1: UCPD1 enabled in Sleep mode

Bits 17:9 Reserved, must be kept at reset value.

Bit 8 **FDCANLPEN**: FDCAN enablein Sleep mode

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with FDCANLPENS, and cleared with FDCANLPENC. This bit is set and reset by software.

0: FDCAN disabled in Sleep mode (default after reset)

1: FDCAN enabled in Sleep mode

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **MDIOSLPEN**: MDIOS enable in Sleep mode

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with MDIOSLPENS, and cleared with MDIOSLPENC.This bit is set and reset by software.

0: MDIOS disabled in Sleep mode (default after reset)

1: MDIOS enabled in Sleep mode

Bits 4:0 Reserved, must be kept at reset value.

14.10.101 RCC APB2 sleep enable register (RCC_APB2LPENR)

Address offset: 0x2AC

Reset value: 0x0000 0000

This register is used to enable the APB2 in Sleep mode (each bit of this register is AND-ed with the EN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI2LPEN	SAI1LPEN	SPI5LPEN	TIM9LPEN	TIM17LPEN	TIM16LPEN	TIM15LPEN
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIM18LPEN	Res.	SPI4LPEN	SPI1LPEN	Res.	Res.	Res.	Res.	USART10LPEN	UART9LPEN	USART6LPEN	USART1LPEN	Res.	Res.	TIM8LPEN	TIM1LPEN
rw		rw	rw					rw	rw	rw	rw			rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **SAI2LPEN**: SAI2 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. it can be set with SAI2LPENS, and cleared with SAI2LPENC. This bit is set and reset by software.

0: SAI2 disabled in Sleep mode (default after reset)

1: SAI2 enabled in Sleep mode

Bit 21 **SAI1LPEN**: SAI1 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with SAI1LPENS, and cleared with SAI1LPENC. This bit is set and reset by software.

0: SAI1 disabled in Sleep mode (default after reset)

1: SAI1 enabled in Sleep mode

Bit 20 **SPI5LPEN**: SPI5 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with SPI5LPENS, and cleared with SPI5LPENC. This bit is set and reset by software.

0: SPI5 disabled in Sleep mode (default after reset)

1: SPI5 enabled in Sleep mode

Bit 19 **TIM9LPEN**: TIM9 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM9LPENS, and cleared with TIM9LPENC. This bit is set and reset by software.

0: TIM9 disabled in Sleep mode (default after reset)

1: TIM9 enabled in Sleep mode

Bit 18 **TIM17LPEN**: TIM17 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM17LPENS, and cleared with TIM17LPENC. This bit is set and reset by software.

0: TIM17 disabled in Sleep mode (default after reset)

1: TIM17 enabled in Sleep mode

Bit 17 **TIM16LPEN**: TIM16 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM16LPENS, and cleared with TIM16LPENC. This bit is set and reset by software.

0: TIM16 disabled in Sleep mode (default after reset)

1: TIM16 enabled in Sleep mode

Bit 16 **TIM15LPEN**: TIM15 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM15LPENS, and cleared with TIM15LPENC. This bit is set and reset by software.

0: TIM15 disabled in Sleep mode (default after reset)

1: TIM15 enabled in Sleep mode

Bit 15 **TIM18LPEN**: TIM18 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM18LPENS, and cleared with TIM18LPENC. This bit is set and reset by software.

0: TIM18 disabled in Sleep mode (default after reset)

1: TIM18 enabled in Sleep mode

Bit 14 Reserved, must be kept at reset value.

Bit 13 **SPI4LPEN**: SPI4 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with SPI4LPENS, and cleared with SPI4LPENC. This bit is set and reset by software.

0: SPI4 disabled in Sleep mode (default after reset)

1: SPI4 enabled in Sleep mode

Bit 12 **SPI1LPEN**: SPI1 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with SPI1LPENS, and cleared with SPI1LPENC. This bit is set and reset by software.

0: SPI1 disabled in Sleep mode (default after reset)

1: SPI1 enabled in Sleep mode

Bits 11:8 Reserved, must be kept at reset value.

Bit 7 **USART10LPEN**: USART10 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with USART10LPENS, and cleared with USART10LPENC. This bit is set and reset by software.

0: USART10 disabled in Sleep mode (default after reset)

1: USART10 enabled in Sleep mode

Bit 6 **UART9LPEN**: UART9 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with UART9LPENS, and cleared with UART9LPENC. This bit is set and reset by software.

0: UART9 disabled in Sleep mode (default after reset)

1: UART9 enabled in Sleep mode

Bit 5 **USART6LPEN**: USART6 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with USART6LPENS, and cleared with USART6LPENC. This bit is set and reset by software.

0: USART6 disabled in Sleep mode (default after reset)

1: USART6 enabled in Sleep mode

Bit 4 **USART1LPEN**: USART1 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with USART1LPENS, and cleared with USART1LPENC. This bit is set and reset by software.

0: USART1 disabled in Sleep mode (default after reset)

1: USART1 enabled in Sleep mode

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **TIM8LPEN**: TIM8 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM8LPENS, and cleared with TIM8LPENC. This bit is set and reset by software.

0: TIM8 disabled in Sleep mode (default after reset)

1: TIM8 enabled in Sleep mode

Bit 0 **TIM1LPEN**: TIM1 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with TIM1LPENS, and cleared with TIM1LPENC. This bit is set and reset by software.

0: TIM1 disabled in Sleep mode (default after reset)

1: TIM1 enabled in Sleep mode

14.10.102 RCC APB3 sleep enable register (RCC_APB3LPENR)

Address offset: 0x2B0

Reset value: 0x0000 0000

This register is used to enable the APB3 in Sleep mode (each bit of this register is AND-ed with the EN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

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

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

Bit 2 **DFTLPEN**: DFT enable

This bit is security-protected by DFTSEC or DFTPRIV. It can be set with DFTLPENS, and cleared with DFTLPENC. This bit is set and reset by software.

0: DFT disabled in Sleep mode (default after reset)

1: DFT enabled in Sleep mode

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

14.10.103 RCC APB4L sleep enable register (RCC_APB4LLPENR)

Address offset: 0x2B4

Reset value: 0x0000 0000

This register is used to enable the APB4L in Sleep mode (each bit of this register is AND-ed with the EN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCAPBLPEN	RTCLPEN
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VREFBUFLPEN	Res.	Res.	LPTIM5LPEN	LPTIM4LPEN	LPTIM3LPEN	LPTIM2LPEN	Res.	I2C4LPEN	Res.	SPI6LPEN	Res.	LPUART1LPEN	HDPLPEN	Res.	Res.
rw			rw	rw	rw	rw		rw		rw		rw	rw		

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

Bit 17 **RTCAPBLPEN**: RTCAPB enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. it can be set with RTCAPBLPENS, and cleared with RTCAPBLPENC. This bit is set and reset by software.

0: RTCAPB disabled in Sleep mode (default after reset)

1: RTCAPB enabled in Sleep mode

Bit 16 **RTCLPEN**: RTC enable

This bit is write-protected by the pwr_lock_backup_n signal. It is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. This bit can be set with RTCLPENS, and cleared with RTCLPENC.

0: RTC disabled in Sleep mode (default after reset)

1: RTC enabled in Sleep mode

Bit 15 **VREFBUFLPEN**: VREFBUF enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with VREFBUFLPENS, and cleared with VREFBUFLPENC.This bit is set and reset by software.

0: VREFBUF disabled in Sleep mode (default after reset)

1: VREFBUF enabled in Sleep mode

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **LPTIM5LPEN**: LPTIM5 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with LPTIM5LPENS, and cleared with LPTIM5LPENC. This bit is set and reset by software.

0: LPTIM5 disabled in Sleep mode (default after reset)

1: LPTIM5 enabled in Sleep mode

Bit 11 **LPTIM4LPEN**: LPTIM4 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with LPTIM4LPENS, and cleared with LPTIM4LPENC. This bit is set and reset by software.

0: LPTIM4 disabled in Sleep mode (default after reset)

1: LPTIM4 enabled in Sleep mode

Bit 10 **LPTIM3LPEN**: LPTIM3 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with LPTIM3LPENS, and cleared with LPTIM3LPENC. This bit is set and reset by software.

0: LPTIM3 disabled in Sleep mode (default after reset)

1: LPTIM3 enabled in Sleep mode

Bit 9 **LPTIM2LPEN**: LPTIM2 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with LPTIM2LPENS, and cleared with LPTIM2LPENC. This bit is set and reset by software.

0: LPTIM2 disabled in Sleep mode (default after reset)

1: LPTIM2 enabled in Sleep mode

Bit 8 Reserved, must be kept at reset value.

Bit 7 **I2C4LPEN**: I2C4 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with I2C4LPENS, and cleared with I2C4LPENC. This bit is set and reset by software.

0: I2C4 disabled in Sleep mode (default after reset)

1: I2C4 enabled in Sleep mode

Bit 6 Reserved, must be kept at reset value.

Bit 5 **SPI6LPEN**: SPI6 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with SPI6LPENS, and cleared with SPI6LPENC. This bit is set and reset by software.

0: SPI6 disabled in Sleep mode (default after reset)

1: SPI6 enabled in Sleep mode

Bit 4 Reserved, must be kept at reset value.

Bit 3 **LPUART1LPEN**: LPUART1 enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with LPUART1LPENS, and cleared with LPUART1LPENC. This bit is set and reset by software.

0: LPUART1 disabled in Sleep mode (default after reset)

1: LPUART1 enabled in Sleep mode

Bit 2 **HDPLPEN**: HDP enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with HDPLPENS, and cleared with HDPLPENC. This bit is set and reset by software.

0: HDP disabled in Sleep mode (default after reset)

1: HDP enabled in Sleep mode

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

14.10.104 RCC APB4H sleep enable register (RCC_APB4HLPENR)

Address offset: 0x2B8

Reset value: 0x0000 0002

This register is used to enable the APB4H in Sleep mode (each bit of this register is AND-ed with the EN bit). It is reset by sys_rst, and is in the V_{CORE} voltage domain.

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

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

Bit 2 **DTSLPEN**: DTS enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. it can be set with DTSLPENS, and cleared with DTSLPENC. This bit is set and reset by software.

0: DTS disabled in Sleep mode (default after reset)

1: DTS enabled in Sleep mode

Bit 1 **BSECLPEN**: BSEC enable

This bit is always security-protected. it can be set with BSECLPENS, and cleared with BSECLPENC. This bit is set and reset by software.

0: BSEC disabled in Sleep mode

1: BSEC enabled in Sleep mode (default after reset)

Bit 0 **SYSCFGLPEN**: SYSCFG enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with SYSCFGLPENS, and cleared with SYSCFGLPENC. This bit IS Set and reset by software.

0: SYSCFG disabled in Sleep mode (default after reset)

1: SYSCFG enabled in Sleep mode

14.10.105 RCC APB5 sleep enable register (RCC_APB5LPENR)

Address offset: 0x2BC

Reset value: 0x0000 0000

This register is used to enable the APB5 in Sleep mode (each bit of this register is AND-ed with the EN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

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

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CSILPEN**: CSI enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with CSILPENS, and cleared with CSILPENC. This bit is set and reset by software.

0: CSI disabled in Sleep mode (default after reset)

1: CSI enabled in Sleep mode

Bit 5 **VENCLPEN**: VENC enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with VENCLPENS, and cleared with VENCLPENC. This bit is set and reset by software.

0: VENC disabled in Sleep mode (default after reset)

1: VENC enabled in Sleep mode

Bit 4 **GFXTIMLPEN**: GFXTIM enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with GFXTIMLPENS, and cleared with GFXTIMLPENC. This bit is set and reset by software.

0: GFXTIM disabled in Sleep mode (default after reset)

1: GFXTIM enabled in Sleep mode

Bit 3 Reserved, must be kept at reset value.

Bit 2 **DCMIPPLPEN**: DCMIPP enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with DCMIPPLPENS, and cleared with DCMIPPLPENC. This bit is set and reset by software.

0: DCMIPP disabled in Sleep mode (default after reset)

1: DCMIPP enabled in Sleep mode

Bit 1 **LTDCLPEN**: LTDC enable

This bit is security-protected by a SEC signal from RIFSC, the PERSEC bit, a PRIV signal from RIFSC, or the PERPRIV bit. It can be set with LTDCLPENS, and cleared with LTDCLPENC. This bit is set and reset by software.

0: LTDC disabled in Sleep mode (default after reset)

1: LTDC enabled in Sleep mode

Bit 0 Reserved, must be kept at reset value.

14.10.106 RCC reset duration control register (RCC_RDCR)

Address offset: 0x44C

Reset value: 0x0600 0000

This register is used to control the minimum sys_rstn active duration. It is reset by pwr_por_rstn, and is in the V_{RET} voltage domain.

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

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **EADLY[3:0]**: External access delay

The bit field is security-protected by the 1 bit. Set and reset by software.
Time to wait before the BOOTROM performs any external device access
0000: No extra delay added by the BOOTROM
0001: 100 µs
0010: 200 µs
0011: 500 µs
0100: 1 ms
0101: 2 ms
0110: 5 ms (default after reset)
0111: 10 ms
1000: 20 ms
1001: 50 ms
1010: 100 ms
1011: 200 ms
1100: 500 ms
1101: 1 s
1110: 2 s
1111: 5 s

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:16 **MRD[4:0]**: Minimum reset duration

This bit is alway security-protected. It is set and reset by software. It defines the minimum guaranteed duration of the NRST assertion. The LSI oscillator is automatically enabled when needed by the RPCTL.
0x00: NRST duration is guaranteed by the pulse stretcher of the PAD. The RPCTL is bypassed (default after reset).
0x01: The guaranteed NRST duration is about 1 ms (1 x 32 lsi_ck cycles).
0x02: The guaranteed NRST duration is about 2 ms (2 x 32 lsi_ck cycles).
{v}: guaranteed NRST duration is about {v} ms ({v} x 32 lsi_ck cycles).

Bits 15:0 Reserved, must be kept at reset value.

14.10.107 RCC oscillator secure configuration register 0 (RCC_SECCFGR0)

Address offset: 0x780

Reset value: 0x0000 0000

This register is used to control the secure access rights to the configuration register of the oscillators. It is reset by sys_rstn, and is in the V_{CORE} voltage domain. Each xxSEC bit defines the secure protection for the configuration registers of the oscillator: a write access is denied if the access is nonsecure while the respective bit is set.

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

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

Bit 4 **HSESEC**: Secure protection of HSE oscillator configuration bits
This bit is set and reset by secure privileged software only. it can be read by any software.
0: HSE configuration bits are accessible by nonsecure software only (default after reset).
1: HSE configuration bits are accessible by secure software only.

Bit 3 **HSISEC**: Secure protection of HSI oscillator configuration bits
This bit it set and reset by secure privileged software only. It can be read by any software.
0: HSI configuration bits are accessible by nonsecure software only (default after reset).
1: HSI configuration bits are accessible by secure software only.

Bit 2 **MSISEC**: Secure protection of MSI oscillator configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: MSI configuration bits are accessible by nonsecure software only (default after reset).
1: MSI configuration bits are accessible by secure software only.

Bit 1 **LSESEC**: Secure protection of LSE oscillator configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: LSE configuration bits are accessible by nonsecure software only (default after reset).
1: LSE configuration bits are accessible by secure software only.

Bit 0 **LSISEC**: Secure protection of LSI oscillator configuration bits.
This bit is set and reset by secure privileged software only. it can be read by any software.
0: LSI configuration bits are accessible by nonsecure software only (default after reset).
1: LSI configuration bits are accessible by secure software only.

14.10.108 RCC oscillator privilege configuration register 0 (RCC_PRIVCFGR0)

Address offset: 0x784

Reset value: 0x0000 0000

This register is used to control the privilege access rights to the configuration register of the oscillators. It is reset by sys_rstn, and is in the V_{CORE} voltage domain. Each xxPRIV bit

defines the privileged protection for the configuration registers of the oscillator: a write access is denied if the access is unprivileged while the respective bit is set.

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

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

Bit 4 **HSEPRIV**: Privileged protection of HSE oscillator configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: HSE configuration bits are accessible by unprivileged software only (default after reset).
1: HSE configuration bits are accessible by privileged software only.

Bit 3 **HSIPRIV**: Privileged protection of HSI oscillator configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: HSI configuration bits are accessible by unprivileged software only (default after reset).
1: HSI configuration bits are accessible by privileged software only.

Bit 2 **MSIPRIV**: Privileged protection of MSI oscillator configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: MSI configuration bits are accessible by unprivileged software only (default after reset).
1: MSI configuration bits are accessible by privileged software only.

Bit 1 **LSEPRIV**: Privileged protection of LSE oscillator configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: LSE configuration bits are accessible by unprivileged software only (default after reset).
1: LSE configuration bits are accessible by privileged software only.

Bit 0 **LSIPRIV**: Privileged protection of LSI oscillator configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: LSI configuration bits are accessible by unprivileged software only (default after reset.)
1: LSI configuration bits are accessible by privileged software only.

14.10.109 RCC oscillator lock configuration register 0 (RCC_LOCKCFGR0)

Address offset: 0x788

Reset value: 0x0000 0000

This register is used to control the locked access rights to the configuration register of the oscillators. It is reset by sys_rst, n and is in the V_{CORE} voltage domain.Each xxLOCK bit defines the lock protection for the configuration registers of the oscillator: a write access is denied if the access is unlocked while the respective bit is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSE LOCK	HSI LOCK	MSI LOCK	LSE LOCK	LSI LOCK
											w	w	w	w	w

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

Bit 4 **HSELOCK**: Locked protection of HSE oscillator configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: HSE configuration bits are accessible by unlocked software only (default after reset).
1: HSE configuration bits are accessible by locked software only.

Bit 3 **HSILOCK**: Locked protection of HSI oscillator configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: HSI configuration bits are accessible by unlocked software only (default after reset).
1: HSI configuration bits are accessible by locked software only.

Bit 2 **MSILOCK**: Locked protection of MSI oscillator configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: MSI configuration bits are accessible by unlocked software only (default after reset).
1: MSI configuration bits are accessible by locked software only.

Bit 1 **LSELOCK**: Locked protection of LSE oscillator configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: LSE configuration bits are accessible by unlocked software only (default after reset).
1: LSE configuration bits are accessible by locked software only.

Bit 0 **LSILOCK**: Locked protection of LSI oscillator configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: LSI configuration bits are accessible by unlocked software only (default after reset).
1: LSI configuration bits are accessible by locked software only.

14.10.110 RCC oscillator public configuration register 0 (RCC_PUBCFGR0)

Address offset: 0x78C

Reset value: 0x0000 0000

This register is used to control the public access rights to the configuration register of the oscillators. It is reset by sys_rstn, and is in the V_{CORE} voltage domain. Each xxPUB bit defines the public protection for the configuration registers of the oscillator: a write access is denied if the access is non-public while the respective bit is set.

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

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

Bit 4 **HSEPUB**: Public protection of HSE oscillator configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: HSE configuration bits are accessible by non-public software only (default after reset).
1: HSE configuration bits are accessible by public software only.

Bit 3 **HSIPUB**: Public protection of HSI oscillator configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: HSI configuration bits are accessible by non-public software only (default after reset).
1: HSI configuration bits are accessible by public software only.

Bit 2 **MSIPUB**: Public protection of MSI oscillator configuration bits
This bit is set and reset by secure privileged software only. it can be read by any software.
0: MSI configuration bits are accessible by non-public software only (default after reset).
1: MSI configuration bits are accessible by public software only.

Bit 1 **LSEPUB**: Public protection of LSE oscillator configuration bits.
This bit is set and reset by secure privileged software only. It can be read by any software.
0: LSE configuration bits are accessible by non-public software only (default after reset).
1: LSE configuration bits are accessible by public software only.

Bit 0 **LSIPUB**: Public protection of LSI oscillator configuration bits.
This bit is set and reset by secure privileged software only. It can be read by any software.
0: LSI configuration bits are accessible by non-public software only (default after reset).
1: LSI configuration bits are accessible by public software only.

14.10.111 RCC PLL secure configuration register 1 (RCC_SECCFGR1)

Address offset: 0x790

Reset value: 0x0000 0000

This register is used to control the secure access rights to the configuration register of the PLLs. It is reset by sys_rstn, and is in the V_{CORE} voltage domain. Each xxSEC bit defines the secure protection for the configuration registers of the PLL: a write access is denied if the access is nonsecure while the respective bit here is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL4 SEC	PLL3 SEC	PLL2 SEC	PLL1 SEC
												rw	rw	rw	rw

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

Bit 3 **PLL4SEC**: Secure protection of PLL4 configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: PLL4 configuration bits are accessible by nonsecure software only (default after reset).
1: PLL4 configuration bits are accessible by secure software only.

Bit 2 **PLL3SEC**: Secure protection of PLL3 configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: PLL3 configuration bits are accessible by nonsecure software only (default after reset).
1: PLL3 configuration bits are accessible by secure software only.

Bit 1 **PLL2SEC**: Secure protection of PLL2 configuration bits
This bit is set and reset by secure privileged software only. It be read by any software.
0: PLL2 configuration bits are accessible by nonsecure software only (default after reset).
1: PLL2 configuration bits are accessible by secure software only.

Bit 0 **PLL1SEC**: Secure protection of PLL1 configuration bits.
This bit is set and reset by secure privileged software only. It can be read by any software.
0: PLL1 configuration bits are accessible by nonsecure software only (default after reset).
1: PLL1 configuration bits are accessible by secure software only.

14.10.112 RCC PLL privilege configuration register 1 (RCC_PRIVCFGR1)

Address offset: 0x794

Reset value: 0x0000 0000

This register is used to control the privileged access rights to the configuration register of the PLLs. It is reset by sys_rstn, and is in the V_{CORE} voltage domain. Each xxPRIV defines the privileged protection for the configuration registers of the PLL: a write access is denied if the access is unprivileged while the respective bit is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL4PRIV	PLL3PRIV	PLL2PRIV	PLL1PRIV
												rw	rw	rw	rw

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

Bit 3 **PLL4PRIV**: Privileged protection of PLL4 configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: PLL4 configuration bits are accessible by unprivileged software only (default after reset).
1: PLL4 configuration bits are accessible by privileged software only.

Bit 2 **PLL3PRIV**: Privileged protection of PLL3 configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: PLL3 configuration bits are accessible by unprivileged software only (default after reset).
1: PLL3 configuration bits are accessible by privileged software only.

Bit 1 **PLL2PRIV**: Privileged protection of PLL2 configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: PLL2 configuration bits are accessible by unprivileged software only (default after reset).
1: PLL2 configuration bits are accessible by privileged software only.

Bit 0 **PLL1PRIV**: Privileged protection of PLL1 configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: PLL1 configuration bits are accessible by unprivileged software only (default after reset).
1: PLL1 configuration bits are accessible by privileged software only.

14.10.113 RCC PLL lock configuration register 1 (RCC_LOCKCFGR1)

Address offset: 0x798

Reset value: 0x0000 0000

This register is used to control the locked access rights to the configuration register of the PLLs. It is reset by sys_rstn, and is in the V_{CORE} voltage domain. Each xxLOCK bit defines the locked protection for the configuration registers of the PLL: a write access is denied if the access is unlocked while the respective bit is set.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL4 LOCK	PLL3 LOCK	PLL2 LOCK	PLL1 LOCK
												w	w	w	w

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

Bit 3 **PLL4LOCK**: Locked protection of PLL4 configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: PLL4 configuration bits are accessible by unlocked software only (default after reset).
1: PLL4 configuration bits are accessible by locked software only.

Bit 2 **PLL3LOCK**: Locked protection of PLL3 configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: PLL3 configuration bits are accessible by unlocked software only (default after reset).
1: PLL3 configuration bits are accessible by locked software only.

Bit 1 **PLL2LOCK**: Locked protection of PLL2 configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: PLL2 configuration bits are accessible by unlocked software only (default after reset).
1: PLL2 configuration bits are accessible by locked software only.

Bit 0 **PLL1LOCK**: Locked protection of PLL1 configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: PLL1 configuration bits are accessible by unlocked software only (default after reset).
1: PLL1 configuration bits are accessible by locked software only.

14.10.114 RCC PLL public configuration register1 (RCC_PUBCFGR1)

Address offset: 0x79C

Reset value: 0x0000 0000

This register is used to control the public access rights to the configuration register of the PLLs. It is reset by sys_rstn, and is in the V_{CORE} voltage domain. Each xxPUB defines the public protection for the configuration registers of the PLL: a write access is denied if the access is unlocked while the respective bit is set.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL4 PUB	PLL3 PUB	PLL2 PUB	PLL1 PUB
												rw	rw	rw	rw

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

Bit 3 **PLL4PUB**: Public protection of PLL4 configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: PLL4 configuration bits are accessible by non-public software. only (default after reset).
1: PLL4 configuration bits are accessible by public software only.

Bit 2 **PLL3PUB**: Public protection of PLL3 configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: PLL3 configuration bits are accessible by non-public software only (default after reset).
1: PLL3 configuration bits are accessible by public software only.

Bit 1 **PLL2PUB**: Public protection of PLL2 configuration bits
This bit is set and reset by secure privileged software only. It can my be read by any software.
0: PLL2 configuration bits are accessible by non-public software only (default after reset).
1: PLL2 configuration bits are accessible by public software only.

Bit 0 **PLL1PUB**: Public protection of PLL1 configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: PLL1 configuration bits are accessible by non-public software only (default after reset).
1: PLL1 configuration bits are accessible by public software only.

14.10.115 RCC divider secure configuration register 2 (RCC_SECCFGR2)

Address offset: 0x7A0

Reset value: 0x0000 0000

This register is used to control the secure access rights to the configuration register of the dividers. It is reset by sys_rstn, and is in the V_{CORE} voltage domain. Each xxSEC bit defines the secure protection for the configuration registers of the divider: a write access is denied if the access is nonsecure while the respective bit is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC20SEC	IC19SEC	IC18SEC	IC17SEC
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC16SEC	IC15SEC	IC14SEC	IC13SEC	IC12SEC	IC11SEC	IC10SEC	IC9SEC	IC8SEC	IC7SEC	IC6SEC	IC5SEC	IC4SEC	IC3SEC	IC2SEC	IC1SEC
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **IC20SEC**: Secure protection of IC20 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC20 configuration bits are accessible by nonsecure software only (default after reset).
1: IC20 configuration bits are accessible by secure software only.

Bit 18 **IC19SEC**: Secure protection of IC19 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC19 configuration bits are accessible by nonsecure software only (default after reset).
1: IC19 configuration bits are accessible by secure software only.

Bit 17 **IC18SEC**: Secure protection of IC18 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC18 configuration bits are accessible by nonsecure software only (default after reset).
1: IC18 configuration bits are accessible by secure software only.

Bit 16 **IC17SEC**: Secure protection of IC17 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC17 configuration bits are accessible by nonsecure software only (default after reset).
1: IC17 configuration bits are accessible by secure software only.

Bit 15 **IC16SEC**: Secure protection of IC16 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC16 configuration bits are accessible by nonsecure software only (default after reset).
1: IC16 configuration bits are accessible by secure software only.

Bit 14 **IC15SEC**: Secure protection of IC15 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC15 configuration bits are accessible by nonsecure software only (default after reset).
1: IC15 configuration bits are accessible by secure software only.

Bit 13 **IC14SEC**: Secure protection of IC14 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC14 configuration bits are accessible by nonsecure software only (default after reset).
1: IC14 configuration bits are accessible by secure software only.

Bit 12 **IC13SEC**: Secure protection of IC13 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC13 configuration bits are accessible by nonsecure software only (default after reset).
1: IC13 configuration bits are accessible by secure software only.

Bit 11 **IC12SEC**: Secure protection of IC12 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC12 configuration bits are accessible by nonsecure software only (default after reset).
1: IC12 configuration bits are accessible by secure software only.

Bit 10 **IC11SEC**: Secure protection of IC11 divider configuration bits
Set and reset by secure privileged software only. It can read by any software.
0: IC11 configuration bits are accessible by nonsecure software only (default after reset).
1: IC11 configuration bits are accessible by secure software only.

Bit 9 **IC10SEC**: Secure protection of IC10 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC10 configuration bits are accessible by nonsecure software only (default after reset)
1: IC10 configuration bits are accessible by secure software only

Bit 8 **IC9SEC**: Secure protection of IC9 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC9 configuration bits are accessible by nonsecure software only (default after reset).
1: IC9 configuration bits are accessible by secure software only.

Bit 7 **IC8SEC**: Secure protection of IC8 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC8 configuration bits are accessible by nonsecure software only (default after reset).
1: IC8 configuration bits are accessible by secure software only.

Bit 6 **IC7SEC**: Secure protection of IC7 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC7 configuration bits are accessible by nonsecure software only (default after reset).
1: IC7 configuration bits are accessible by secure software only.

Bit 5 **IC6SEC**: Secure protection of IC6 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC6 configuration bits are accessible by nonsecure software only (default after reset).
1: IC6 configuration bits are accessible by secure software only.

Bit 4 **IC5SEC**: Secure protection of IC5 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC5 configuration bits are accessible by nonsecure software only (default after reset)
1: IC5 configuration bits are accessible by secure software only

Bit 3 **IC4SEC**: Secure protection of IC4 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC4 configuration bits are accessible by nonsecure software only (default after reset).
1: IC4 configuration bits are accessible by secure software only.

Bit 2 **IC3SEC**: Secure protection of IC3 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC3 configuration bits are accessible by nonsecure software only (default after reset).
1: IC3 configuration bits are accessible by secure software only.

Bit 1 **IC2SEC**: Secure protection of IC2 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC2 configuration bits are accessible by nonsecure software only (default after reset).
1: IC2 configuration bits are accessible by secure software only.

Bit 0 **IC1SEC**: Secure protection of IC1 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC1 configuration bits are accessible by nonsecure software only (default after reset).
1: IC1 configuration bits are accessible by secure software only.

14.10.116 RCC divider privilege configuration register 2 (RCC_PRIVCFGR2)

Address offset: 0x7A4

Reset value: 0x0000 0000

This register is used to control the privileged access rights to the configuration register of the dividers. It is reset by sys_rstn, and is in the V_{CORE} voltage domain. Each xxPRIV bit defines the privileged protection for the configuration registers of the divider: a write access is denied if the access is unprivileged while the respective bit is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC20 PRIV	IC19 PRIV	IC18 PRIV	IC17 PRIV
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC16 PRIV	IC15 PRIV	IC14 PRIV	IC13 PRIV	IC12 PRIV	IC11 PRIV	IC10 PRIV	IC9 PRIV	IC8 PRIV	IC7 PRIV	IC6 PRIV	IC5 PRIV	IC4 PRIV	IC3 PRIV	IC2 PRIV	IC1 PRIV
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **IC20PRIV**: Privileged protection of IC20 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC20 configuration bits are accessible by unprivileged software only (default after reset).
1: IC20 configuration bits are accessible by privileged software only.

Bit 18 **IC19PRIV**: Privileged protection of IC19 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC19 configuration bits are accessible by unprivileged software only (default after reset).
1: IC19 configuration bits are accessible by privileged software only.

Bit 17 **IC18PRIV**: Privilege protection of IC18 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC18 configuration bits are accessible by unprivileged software only (default after reset).
1: IC18 configuration bits are accessible by privileged software only.

Bit 16 **IC17PRIV**: Privileges protection of IC17 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC17 configuration bits are accessible by unprivileged software only (default after reset).
1: IC17 configuration bits are accessible by privileged software only.

Bit 15 **IC16PRIV**: Privileged protection of IC16 divider configuration bits.
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC16 configuration bits are accessible by unprivileged software only (default after reset).
1: IC16 configuration bits are accessible by privileged software only.

Bit 14 **IC15PRIV**: Privileged protection of IC15 divider configuration bits.
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC15 configuration bits are accessible by unprivileged software only (default after reset).
1: IC15 configuration bits are accessible by privileged software only.

Bit 13 **IC14PRIV**: Privileged protection of IC14 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC14 configuration bits are accessible by unprivileged software only (default after reset).
1: IC14 configuration bits are accessible by privileged software only.

Bit 12 **IC13PRIV**: Privileged protection of IC13 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC13 configuration bits are accessible by unprivileged software only (default after reset).
1: IC13 configuration bits are accessible by privileged software only.

Bit 11 **IC12PRIV**: Privileged protection of IC12 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC12 configuration bits are accessible by unprivileged software only (default after reset).
1: IC12 configuration bits are accessible by privileged software only.

Bit 10 **IC11PRIV**: Privileged protection of IC11 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC11 configuration bits are accessible by unprivileged software only (default after reset).
1: IC11 configuration bits are accessible by privileged software only.

Bit 9 **IC10PRIV**: Privileged protection of IC10 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC10 configuration bits are accessible by unprivileged software only (default after reset).
1: IC10 configuration bits are accessible by privileged software only.

Bit 8 **IC9PRIV**: Privileged protection of IC9 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC9 configuration bits are accessible by unprivileged software only (default after reset).
1: IC9 configuration bits are accessible by privileged software only.

Bit 7 **IC8PRIV**: Privileged protection of IC8 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC8 configuration bits are accessible by unprivileged software only (default after reset).
1: IC8 configuration bits are accessible by privileged software only.

Bit 6 **IC7PRIV**: Privileged protection of IC7 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC7 configuration bits are accessible by unprivileged software only (default after reset).
1: IC7 configuration bits are accessible by privileged software only.

Bit 5 **IC6PRIV**: Privileged protection of IC6 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC6 configuration bits are accessible by unprivileged software only (default after reset).
1: IC6 configuration bits are accessible by privileged software only.

Bit 4 **IC5PRIV**: Privileged protection of IC5 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC5 configuration bits are accessible by unprivileged software only (default after reset).
1: IC5 configuration bits are accessible by privileged software only.

Bit 3 **IC4PRIV**: Privileged protection of IC4 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC4 configuration bits are accessible by unprivileged software only (default after reset).
1: IC4 configuration bits are accessible by privileged software only.

Bit 2 **IC3PRIV**: Privileged protection of IC3 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC3 configuration bits are accessible by unprivileged software only (default after reset).
1: IC3 configuration bits are accessible by privileged software only.

Bit 1 **IC2PRIV**: Privileged protection of IC2 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC2 configuration bits are accessible by unprivileged software only (default after reset).
1: IC2 configuration bits are accessible by privileged software only.

Bit 0 **IC1PRIV**: Privileged protection of IC1 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC1 configuration bits are accessible by unprivileged software only (default after reset).
1: IC1 configuration bits are accessible by privileged software only.

14.10.117 RCC divider lock configuration register 2 (RCC_LOCKCFGR2)

Address offset: 0x7A8

Reset value: 0x0000 0000

This register is used to control the locked access rights to the configuration register of the dividers. It is reset by sys_rstn, and is in the V_{CORE} voltage domain. Each xxLOCK bit defines the locked protection for the configuration registers of the divider: a write access is denied if the access is unlocked while the respective bit is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC20LOCK	IC19LOCK	IC18LOCK	IC17LOCK
												w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC16LOCK	IC15LOCK	IC14LOCK	IC13LOCK	IC12LOCK	IC11LOCK	IC10LOCK	IC9LOCK	IC8LOCK	IC7LOCK	IC6LOCK	IC5LOCK	IC4LOCK	IC3LOCK	IC2LOCK	IC1LOCK
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **IC20LOCK**: Locked protection of IC20 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC20 configuration bits are accessible by unlocked software only (default after reset).
1: IC20 configuration bits are accessible by locked software only.

Bit 18 **IC19LOCK**: Locked protection of IC19 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC19 configuration bits are accessible by unlocked software only (default after reset).
1: IC19 configuration bits are accessible by locked software only.

Bit 17 **IC18LOCK**: Locked protection of IC18 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC18 configuration bits are accessible by unlocked software only (default after reset).
1: IC18 configuration bits are accessible by locked software only.

Bit 16 **IC17LOCK**: Locked protection of IC17 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC17 configuration bits are accessible by unlocked software only (default after reset).
1: IC17 configuration bits are accessible by locked software only.

Bit 15 **IC16LOCK**: Locked protection of IC16 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC16 configuration bits are accessible by unlocked software only (default after reset).
1: IC16 configuration bits are accessible by locked software only.

Bit 14 **IC15LOCK**: Locked protection of IC15 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC15 configuration bits are accessible by unlocked software only (default after reset).
1: IC15 configuration bits are accessible by locked software only.

Bit 13 **IC14LOCK**: Locked protection of IC14 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC14 configuration bits are accessible by unlocked software only (default after reset).
1: IC14 configuration bits are accessible by locked software only.

Bit 12 **IC13LOCK**: Locked protection of IC13 divider configuration bits.
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC13 configuration bits are accessible by unlocked software only (default after reset).
1: IC13 configuration bits are accessible by locked software only.

Bit 11 **IC12LOCK**: Locked protection of IC12 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC12 configuration bits are accessible by unlocked software only (default after reset).
1: IC12 configuration bits are accessible by locked software only.

Bit 10 **IC11LOCK**: Locked protection of IC11 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC11 configuration bits are accessible by unlocked software only (default after reset).
1: IC11 configuration bits are accessible by locked software only.

Bit 9 **IC10LOCK**: Locked protection of IC10 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC10 configuration bits are accessible by unlocked software only (default after reset).
1: IC10 configuration bits are accessible by locked software only.

Bit 8 **IC9LOCK**: Locked protection of IC9 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC9 configuration bits are accessible by unlocked software only (default after reset).
1: IC9 configuration bits are accessible by locked software only.

Bit 7 **IC8LOCK**: Locked protection of IC8 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC8 configuration bits are accessible by unlocked software only (default after reset).
1: IC8 configuration bits are accessible by locked software only.

Bit 6 **IC7LOCK**: Locked protection of IC7 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC7 configuration bits are accessible by unlocked software only (default after reset).
1: IC7 configuration bits are accessible by locked software only.

Bit 5 **IC6LOCK**: Locked protection of IC6 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC6 configuration bits are accessible by unlocked software only (default after reset).
1: IC6 configuration bits are accessible by locked software only.

Bit 4 **IC5LOCK**: Locked protection of IC5 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC5 configuration bits are accessible by unlocked software only (default after reset).
1: IC5 configuration bits are accessible by locked software only.

Bit 3 **IC4LOCK**: Locked protection of IC4 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC4 configuration bits are accessible by unlocked software only (default after reset).
1: IC4 configuration bits are accessible by locked software only.

Bit 2 **IC3LOCK**: Locked protection of IC3 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC3 configuration bits are accessible by unlocked software only (default after reset).
1: IC3 configuration bits are accessible by locked software only.

Bit 1 **IC2LOCK**: Locked protection of IC2 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC2 configuration bits are accessible by unlocked software only (default after reset).
1: IC2 configuration bits are accessible by locked software only.

Bit 0 **IC1LOCK**: Locked protection of IC1 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC1 configuration bits are accessible by unlocked software only (default after reset).
1: IC1 configuration bits are accessible by locked software only.

14.10.118 RCC divider public configuration register 2 (RCC_PUBCFGR2)

Address offset: 0x7AC

Reset value: 0x0000 0000

This register is used to control the public access rights to the configuration register of the dividers. It is reset by sys_rstn, and is in the V_{CORE} voltage domain. Each xxPUB bit defines the public protection for the configuration registers of the divider: a write access is denied if the access is non-public while the respective bit here is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC20PUB	IC19PUB	IC18PUB	IC17PUB
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC16PUB	IC15PUB	IC14PUB	IC13PUB	IC12PUB	IC11PUB	IC10PUB	IC9PUB	IC8PUB	IC7PUB	IC6PUB	IC5PUB	IC4PUB	IC3PUB	IC2PUB	IC1PUB
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **IC20PUB**: Public protection of IC20 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC20 configuration bits are accessible by non-public software only (default after reset).
1: IC20 configuration bits are accessible by public software only.

Bit 18 **IC19PUB**: Public protection of IC19 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC19 configuration bits are accessible by non-public software only (default after reset).
1: IC19 configuration bits are accessible by public software only.

Bit 17 **IC18PUB**: Public protection of IC18 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC18 configuration bits are accessible by non-public software only (default after reset).
1: IC18 configuration bits are accessible by public software only.

Bit 16 **IC17PUB**: Public protection of IC17 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC17 configuration bits are accessible by non-public software only (default after reset).
1: IC17 configuration bits are accessible by public software only.

Bit 15 **IC16PUB**: Public protection of IC16 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC16 configuration bits are accessible by non-public software only (default after reset).
1: IC16 configuration bits are accessible by public software only.

Bit 14 **IC15PUB**: Public protection of IC15 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC15 configuration bits are accessible by non-public software only (default after reset).
1: IC15 configuration bits are accessible by public software only.

Bit 13 **IC14PUB**: Public protection of IC14 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC14 configuration bits are accessible by non-public software only (default after reset).
1: IC14 configuration bits are accessible by public software only.

Bit 12 **IC13PUB**: Public protection of IC13 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC13 configuration bits are accessible by non-public software only (default after reset).
1: IC13 configuration bits are accessible by public software only.

Bit 11 **IC12PUB**: Public protection of IC12 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC12 configuration bits are accessible by non-public software only (default after reset).
1: IC12 configuration bits are accessible by public software only.

Bit 10 **IC11PUB**: Public protection of IC11 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC11 configuration bits are accessible by non-public software only (default after reset).
1: IC11 configuration bits are accessible by public software only.

Bit 9 **IC10PUB**: Public protection of IC10 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC10 configuration bits are accessible by non-public software only (default after reset).
1: IC10 configuration bits are accessible by public software only.

Bit 8 **IC9PUB**: Public protection of IC9 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC9 configuration bits are accessible by non-public software only (default after reset).
1: IC9 configuration bits are accessible by public software only.

Bit 7 **IC8PUB**: Public protection of IC8 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC8 configuration bits are accessible by non-public software only (default after reset).
1: IC8 configuration bits are accessible by public software only.

Bit 6 **IC7PUB**: Public protection of IC7 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC7 configuration bits are accessible by non-public software only (default after reset).
1: IC7 configuration bits are accessible by public software only.

Bit 5 **IC6PUB**: Public protection of IC6 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC6 configuration bits are accessible by non-public software only (default after reset).
1: IC6 configuration bits are accessible by public software only.

Bit 4 **IC5PUB**: Public protection of IC5 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC5 configuration bits are accessible by non-public software only (default after reset).
1: IC5 configuration bits are accessible by public software only.

Bit 3 **IC4PUB**: Public protection of IC4 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC4 configuration bits are accessible by non-public software only (default after reset).
1: IC4 configuration bits are accessible by public software only.

Bit 2 **IC3PUB**: Public protection of IC3 divider configuration bits.
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC3 configuration bits are accessible by non-public software only (default after reset).
1: IC3 configuration bits are accessible by public software only.

Bit 1 **IC2PUB**: Public protection of IC2 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC2 configuration bits are accessible by non-public software only (default after reset).
1: IC2 configuration bits are accessible by public software only.

Bit 0 **IC1PUB**: Public protection of IC1 divider configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: IC1 configuration bits are accessible by non-public software only (default after reset).
1: IC1 configuration bits are accessible by public software only.

14.10.119 RCC system secure configuration register 3 (RCC_SECCFGR3)

Address offset: 0x7B0

Reset value: 0x0000 0000

This register is used to control the secure access rights to the system configuration registers. It is reset by sys_rstn, and is in the V_{CORE} voltage domain. Each xxSEC bit defines the secure protection for the configuration registers: a write access is denied if the access is nonsecure while the respective bit is set.

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

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

Bit 5 **RSTSEC**: Secure protection of RST system configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: RST configuration bits are accessible by nonsecure software only (default after reset).
1: RST configuration bits are accessible by secure software only.

Bit 4 **INTSEC**: Secure protection of INT system configuration bits.
This bit is set and reset by secure privileged software only. It can be read by any software.
0: INT configuration bits are accessible by nonsecure software only (default after reset).
1: INT configuration bits are accessible by secure software only.

Bit 3 **PERSEC**: Secure protection of PER system configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: PER configuration bits are accessible by nonsecure software only (default after reset).
1: PER configuration bits are accessible by secure software only.

Bit 2 **BUSSEC**: Secure protection of BUS system configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: BUS configuration bits are accessible by nonsecure software only (default after reset).
1: BUS configuration bits are accessible by secure software only.

Bit 1 **SYSSEC**: Secure protection of SYS system configuration bit.
This bit is set and reset by secure privileged software only. It can be read by any software.
0: SYS configuration bits are accessible by nonsecure software only (default after reset).
1: SYS configuration bits are accessible by secure software only.

Bit 0 **MODSEC**: Secure protection of MOD system configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: MOD configuration bits are accessible by nonsecure software only (default after reset).
1: MOD configuration bits are accessible by secure software only.

14.10.120 RCC system privilege configuration register3 (RCC_PRIVCFGR3)

Address offset: 0x7B4

Reset value: 0x0000 0000

This register is used to control the privileged access rights to the system configuration registers. It is reset by sys_rstn, and is in the V_{CORE} voltage domain. Each xxPRIV bit defines the privileged protection for the system configuration registers: a write access is denied if the access is unprivileged while the respective bit is set.

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

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

Bit 5 **RSTPRIV**: Privileged protection of RST system configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: RST configuration bits are accessible by unprivileged software only (default after reset).
1: RST configuration bits are accessible by privileged software only.

Bit 4 **INTPRIV**: Privileged protection of INT system configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: INT configuration bits are accessible by unprivileged software only (default after reset).
1: INT configuration bits are accessible by privileged software only.

Bit 3 **PERPRIV**: Privileged protection of PER system configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: PER configuration bits are accessible by unprivileged software only (default after reset).
1: PER configuration bits are accessible by privileged software only.

Bit 2 **BUSPRIV**: Privileged protection of BUS system configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: BUS configuration bits are accessible by unprivileged software only (default after reset.)
1: BUS configuration bits are accessible by privileged software only.

Bit 1 **SYSPRIV**: Privileged protection of SYS system configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: SYS configuration bits are accessible by unprivileged software only (default after reset).
1: SYS configuration bits are accessible by privileged software only.

Bit 0 **MODPRIV**: Privileged protection of MOD system configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: MOD configuration bits are accessible by unprivileged software only (default after reset).
1: MOD configuration bits are accessible by privileged software only.

14.10.121 RCC system lock configuration register 3 (RCC_LOCKCFGR3)

Address offset: 0x7B8

Reset value: 0x0000 0000

This register is used to control the locked access rights to the system configuration registers. It is reset by sys_rstn, and is in the V_{CORE} voltage domain. Each xxLOCK bit defines the locked protection for the system configuration registers: a write access is denied if the access is unlocked while the respective bit is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RST LOCK	INT LOCK	PER LOCK	BUS LOCK	SYS LOCK	MOD LOCK
										w	w	w	w	w	w

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

Bit 5 **RSTLOCK**: Locked protection of RST system configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: RST configuration bits are accessible by unlocked software only (default after reset).
1: RST configuration bits are accessible by locked software only.

Bit 4 **INTLOCK**: Locked protection of INT system configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: INT configuration bits are accessible by unlocked software only (default after reset).
1: INT configuration bits are accessible by locked software only.

Bit 3 **PERLOCK**: Locked protection of PER system configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: PER configuration bits are accessible by unlocked software only (default after reset).
1: PER configuration bits are accessible by locked software only.

Bit 2 **BUSLOCK**: Locked protection of BUS system configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: BUS configuration bits are accessible by unlocked software only (default after reset).
1: BUS configuration bits are accessible by locked software only.

Bit 1 **SYSLOCK**: Locked protection of SYS system configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: SYS configuration bits are accessible by unlocked software only (default after reset).
1: SYS configuration bits are accessible by locked software only.

Bit 0 **MODLOCK**: Locked protection of MOD system configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: MOD configuration bits are accessible by unlocked software only (default after reset).
1: MOD configuration bits are accessible by locked software only.

14.10.122 RCC system public configuration register 3 (RCC_PUBCFGR3)

Address offset: 0x7BC

Reset value: 0x0000 0000

This register is used to control the public access rights to the system configuration registers. It is reset by sys_rstn, and is in the V_{CORE} voltage domain. Each xxPUB defines the public protection for the system configuration registers: a write access is denied if the access is non-public while the respective bit is set.

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

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

Bit 5 **RSTPUB**: Public protection of RST system configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: RST configuration bits are accessible by non-public software only (default after reset).
1: RST configuration bits are accessible by public software only.

Bit 4 **INTPUB**: Public protection of INT system configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: INT configuration bits are accessible by non-public software only (default after reset).
1: INT configuration bits are accessible by public software only.

Bit 3 **PERPUB**: Public protection of PER system configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: PER configuration bits are accessible by non-public software only (default after reset).
1: PER configuration bits are accessible by public software only.

Bit 2 **BUSPUB**: Public protection of BUS system configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: BUS configuration bits are accessible by non-public software only (default after reset).
1: BUS configuration bits are accessible by public software only.

Bit 1 **SYSPUB**: Public protection of SYS system configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: SYS configuration bits are accessible by non-public software only (default after reset).
1: SYS configuration bits are accessible by public software only.

Bit 0 **MODPUB**: Public protection of MOD system configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: MOD configuration bits are accessible by non-public software only (default after reset).
1: MOD configuration bits are accessible by public software only.

14.10.123 RCC bus secure configuration register 4 (RCC_SECCFGR4)

Reset value: 0x0000 0000

Address offset: 0x7C0

This register is used to control the secure access rights to the configuration register of the buses. It is reset by sys_rstn, and is in the V_{CORE} voltage domain. Each xxSEC pub defines the secure protection for the configuration registers of the bus: a write access is denied if the access is nonsecure while the respective bit is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	NOC SEC	APB5 SEC	APB4 SEC	APB3 SEC	APB2 SEC	APB1 SEC	AHB5 SEC	AHB4 SEC	AHB3 SEC	AHB2 SEC	AHB1 SEC	AHBM SEC	ACLKNC SEC	ACLKN SEC
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **NOCSEC**: Secure protection of NOC bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: NOC configuration bits are accessible by nonsecure software only (default after reset).
1: NOC configuration bits are accessible by secure software only.

Bit 12 **APB5SEC**: Secure protection of APB5 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: APB5 configuration bits are accessible by nonsecure software only (default after reset).
1: APB5 configuration bits are accessible by secure software only.

Bit 11 **APB4SEC**: Secure protection of APB4 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: APB4 configuration bits are accessible by nonsecure software only (default after reset).
1: APB4 configuration bits are accessible by secure software only.

Bit 10 **APB3SEC**: Secure protection of APB3 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: APB3 configuration bits are accessible by nonsecure software only (default after reset).
1: APB3 configuration bits are accessible by secure software only.

Bit 9 **APB2SEC**: Secure protection of APB2 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: APB2 configuration bits are accessible by nonsecure software only (default after reset).
1: APB2 configuration bits are accessible by secure software only.

Bit 8 **APB1SEC**: Secure protection of APB1 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: APB1 configuration bits are accessible by nonsecure software only (default after reset).
1: APB1 configuration bits are accessible by secure software only.

Bit 7 **AHB5SEC**: Secure protection of AHB5 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: AHB5 configuration bits are accessible by nonsecure software only (default after reset).
1: AHB5 configuration bits are accessible by secure software only.

Bit 6 **AHB4SEC**: Secure protection of AHB4 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: AHB4 configuration bits are accessible by nonsecure software only (default after reset).
1: AHB4 configuration bits are accessible by secure software only.

Bit 5 **AHB3SEC**: Secure protection of AHB3 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: AHB3 configuration bits are accessible by nonsecure software only (default after reset).
1: AHB3 configuration bits are accessible by secure software only.

Bit 4 **AHB2SEC**: Secure protection of AHB2 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: AHB2 configuration bits are accessible by nonsecure software only (default after reset).
1: AHB2 configuration bits are accessible by secure software only.

Bit 3 **AHB1SEC**: Secure protection of AHB1 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: AHB1 configuration bits are accessible by nonsecure software only (default after reset).
1: AHB1 configuration bits are accessible by secure software only.

Bit 2 **AHBMSEC**: Secure protection of AHBM bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: AHBM configuration bits are accessible by nonsecure software only (default after reset).
1: AHBM configuration bits are accessible by secure software only.

Bit 1 **ACLKNCSEC**: Secure protection of ACLKNC bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: ACLKNC configuration bits are accessible by nonsecure software only (default after reset).
1: ACLKNC configuration bits are accessible by secure software only.

Bit 0 **ACLKNSEC**: Secure protection of ACLKN bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: ACLKN configuration bits are accessible by nonsecure software only (default after reset).
1: ACLKN configuration bits are accessible by secure software only.

14.10.124 RCC bus privilege configuration register 4 (RCC_PRIVCFGR4)

Address offset: 0x7C4

Reset value: 0x0000 0000

This register is used to control the privileged access rights to the configuration register of the bus. It is reset by sys_rstn, and is in the V_{CORE} voltage domain. Each xxPRIV bit defines the privileged protection for the configuration registers of the bus: a write access is denied if the access is unprivileges while the respective bit is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	NOC PRIV	APB5 PRIV	APB4 PRIV	APB3 PRIV	APB2 PRIV	APB1 PRIV	AHB5 PRIV	AHB4 PRIV	AHB3 PRIV	AHB2 PRIV	AHB1 PRIV	AHBM PRIV	ACLKNC PRIV	ACLKN PRIV
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **NOCPRIV**: Privileged protection of NOC bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: NOC configuration bits are accessible by unprivileged software only (default after reset).
1: NOC configuration bits are accessible by privileged software only.

Bit 12 **APB5PRIV**: Privileged protection of APB5 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: APB5 configuration bits are accessible by unprivileged software only (default after reset).
1: APB5 configuration bits are accessible by privileged software only.

Bit 11 **APB4PRIV**: Privileged protection of APB4 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: APB4 configuration bits are accessible by unprivileged software only (default after reset).
1: APB4 configuration bits are accessible by privileged software only.

Bit 10 **APB3PRIV**: Privileged protection of APB3 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: APB3 configuration bits are accessible by unprivileged software only (default after reset).
1: APB3 configuration bits are accessible by privileged software only.

Bit 9 **APB2PRIV**: Privileged protection of APB2 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: APB2 configuration bits are accessible by unprivileged software only (default after reset).
1: APB2 configuration bits are accessible by privileged software only.

Bit 8 **APB1PRIV**: Privileged protection of APB1 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: APB1 configuration bits are accessible by unprivileged software only (default after reset).
1: APB1 configuration bits are accessible by privileged software only.

Bit 7 **AHB5PRIV**: Privileged protection of AHB5 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: AHB5 configuration bits are accessible by unprivileged software only (default after reset).
1: AHB5 configuration bits are accessible by privileged software only.

Bit 6 **AHB4PRIV**: Privileged protection of AHB4 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: AHB4 configuration bits are accessible by unprivileged software only (default after reset).
1: AHB4 configuration bits are accessible by privileged software only.

Bit 5 **AHB3PRIV**: Privileged protection of AHB3 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: AHB3 configuration bits are accessible by unprivileged software only (default after reset).
1: AHB3 configuration bits are accessible by privileged software only.

Bit 4 **AHB2PRIV**: Privileged protection of AHB2 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: AHB2 configuration bits are accessible by unprivileged software only (default after reset).
1: AHB2 configuration bits are accessible by privileged software only.

Bit 3 **AHB1PRIV**: Privileged protection of AHB1 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: AHB1 configuration bits are accessible by unprivileged software only (default after reset).
1: AHB1 configuration bits are accessible by privileged software only.

Bit 2 **AHBMPRIV**: Privileged protection of AHBM bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: AHBM configuration bits are accessible by unprivileged software only (default after reset).
1: AHBM configuration bits are accessible by privileged software only.

Bit 1 **ACLKNCPRIV**: Privileged protection of ACLKNC bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: ACLKNC configuration bits are accessible by unprivileged software only (default after reset).
1: ACLKNC configuration bits are accessible by privileged software only.

Bit 0 **ACLKNPRIV**: Privileged protection of ACLKN bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: ACLKN configuration bits are accessible by unprivileged software only (default after reset).
1: ACLKN configuration bits are accessible by privileged software only.

14.10.125 RCC bus lock configuration register 4 (RCC_LOCKCFGR4)

Address offset: 0x7C8

Reset value: 0x0000 0000

This register is used to control the locked access rights to the configuration register of the bus. It is reset by sys_rstn, and is in the V_{CORE} voltage domain. Each xxLOCK defines the locked protection for the configuration registers of the bus: a write access is denied if the access is unlocked while the respective bit is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	NOCLOCK	APB5LOCK	APB4LOCK	APB3LOCK	APB2LOCK	APB1LOCK	AHB5LOCK	AHB4LOCK	AHB3LOCK	AHB2LOCK	AHB1LOCK	AHBMLOCK	ACLKNCLOCK	ACLKNLOCK
		w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **NOCLOCK**: Locked protection of NOC bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: NOC configuration bits are accessible by unlocked software only (default after reset).
1: NOC configuration bits are accessible by locked software only.

Bit 12 **APB5LOCK**: Locked protection of APB5 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: APB5 configuration bits are accessible by unlocked software only (default after reset).
1: APB5 configuration bits are accessible by locked software only.

Bit 11 **APB4LOCK**: Locked protection of APB4 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: APB4 configuration bits are accessible by unlocked software only (default after reset).
1: APB4 configuration bits are accessible by lock software only.

Bit 10 **APB3LOCK**: Locked protection of APB3 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: APB3 configuration bits are accessible by unlocked software only (default after reset).
1: APB3 configuration bits are accessible by locked software only.

Bit 9 **APB2LOCK**: Locked protection of APB2 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: APB2 configuration bits are accessible by unlocked software only (default after reset).
1: APB2 configuration bits are accessible by locked software only.

Bit 8 **APB1LOCK**: Locked protection of APB1 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: APB1 configuration bits are accessible by unlocked software only (default after reset).
1: APB1 configuration bits are accessible by locked software only.

Bit 7 **AHB5LOCK**: Locked protection of AHB5 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: AHB5 configuration bits are accessible by unlocked software only (default after reset).
1: AHB5 configuration bits are accessible by locked software only.

Bit 6 **AHB4LOCK**: Locked protection of AHB4 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: AHB4 configuration bits are accessible by unlocked software only (default after reset).
1: AHB4 configuration bits are accessible by locked software only.

Bit 5 **AHB3LOCK**: Locked protection of AHB3 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: AHB3 configuration bits are accessible by unlocked software only (default after reset).
1: AHB3 configuration bits are accessible by locked software only.

Bit 4 **AHB2LOCK**: Locked protection of AHB2 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: AHB2 configuration bits are accessible by unlocked software only (default after reset).
1: AHB2 configuration bits are accessible by locked software only.

Bit 3 **AHB1LOCK**: Locked protection of AHB1 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: AHB1 configuration bits are accessible by unlocked software only (default after reset).
1: AHB1 configuration bits are accessible by locked software only.

Bit 2 **AHBMLOCK**: Locked protection of AHBM bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: AHBM configuration bits are accessible by unlocked software only (default after reset).
1: AHBM configuration bits are accessible by locked software only.

Bit 1 **ACLKNCLOCK**: Locked protection of ACLKNC bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: ACLKNC configuration bits are accessible by unlocked software only (default after reset).
1: ACLKNC configuration bits are accessible by locked software only.

Bit 0 **ACLKNLOCK**: Locked protection of ACLKN bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: ACLKN configuration bits are accessible by unlocked software only (default after reset).
1: ACLKN configuration bits are accessible by locked software only.

14.10.126 RCC bus public configuration register 4 (RCC_PUBCFGR4)

Address offset: 0x7CC

Reset value: 0x0000 0000

This register is used to control the public access rights to the configuration register of the bus. It is reset by sys_rstn, and is in the V_{CORE} voltage domain. Each xxPUB defines the public protection for the configuration registers of the bus: a write access is denied if the access is non-public while the respective bit is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	NOCPUB	APB5PUB	APB4PUB	APB3PUB	APB2PUB	APB1PUB	AHB5PUB	AHB4PUB	AHB3PUB	AHB2PUB	AHB1PUB	AHBMPUB	ACLKNCPUB	ACLKNPUB
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **NOCPUB**: Public protection of NOC bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: NOC configuration bits are accessible by non-public software only (default after reset).
1: NOC configuration bits are accessible by public software only.

Bit 12 **APB5PUB**: Public protection of APB5 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: APB5 configuration bits are accessible by non-public software only (default after reset)
1: APB5 configuration bits are accessible by public software only

Bit 11 **APB4PUB**: Public protection of APB4 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: APB4 configuration bits are accessible by non-public software only (default after reset).
1: APB4 configuration bits are accessible by public software only.

Bit 10 **APB3PUB**: Public protection of APB3 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: APB3 configuration bits are accessible by non-public software only (default after reset).
1: APB3 configuration bits are accessible by public software only.

Bit 9 **APB2PUB**: Public protection of APB2 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: APB2 configuration bits are accessible by non-public software only (default after reset).
1: APB2 configuration bits are accessible by public software only.

Bit 8 **APB1PUB**: Public protection of APB1 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: APB1 configuration bits are accessible by non-public software only (default after reset).
1: APB1 configuration bits are accessible by public software only.

Bit 7 **AHB5PUB**: Public protection of AHB5 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: AHB5 configuration bits are accessible by non-public software only (default after reset).
1: AHB5 configuration bits are accessible by public software only.

Bit 6 **AHB4PUB**: Public protection of AHB4 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: AHB4 configuration bits are accessible by non-public software only (default after reset).
1: AHB4 configuration bits are accessible by public software only.

Bit 5 **AHB3PUB**: Public protection of AHB3 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: AHB3 configuration bits are accessible by non-public software only (default after reset).
1: AHB3 configuration bits are accessible by public software only.

Bit 4 **AHB2PUB**: Public protection of AHB2 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: AHB2 configuration bits are accessible by non-public software only (default after reset).
1: AHB2 configuration bits are accessible by public software only.

Bit 3 **AHB1PUB**: Public protection of AHB1 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: AHB1 configuration bits are accessible by non-public software only (default after reset).
1: AHB1 configuration bits are accessible by public software only.

Bit 2 **AHBMPUB**: Public protection of AHBM bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: AHBM configuration bits are accessible by non-public software only (default after reset).
1: AHBM configuration bits are accessible by public software only.

Bit 1 **ACLKNCPUB**: Public protection of ACLKNC bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: ACLKNC configuration bits are accessible by non-public software only (default after reset).
1: ACLKNC configuration bits are accessible by public software only.

Bit 0 **ACLKNPUB**: Public protection of the ACLKN bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: ACLKN configuration bits are accessible by non-public software only (default after reset).
1: ACLKN configuration bits are accessible by public software only.

14.10.127 RCC bus public configuration register 4 (RCC_PUBCFGR5)

Address offset: 0x7D0

Reset value: 0x0000 0000

This register is used to control the public access rights to the configuration register of the bus. It is reset by sys_rstn, and is in the V_{CORE} voltage domain. Each xxPUB bit defines the public protection for the configuration registers of the bus: a write access is denied if the access is non-public while the respective bit is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	VENCRAMPUB	CACHEAXIRAMPUB	FLEXRAMPUB	AXISRAM2PUB	AXISRAM1PUB	BKPSRAMPUB	AHBSRAM2PUB	AHBSRAM1PUB	AXISRAM6PUB	AXISRAM5PUB	AXISRAM4PUB	AXISRAM3PUB
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **VENCRAMPUB**: Public protection of VENCRAM bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: VENCRAM configuration bits are accessible by non-public software only (default after reset).
1: VENCRAM configuration bits are accessible by public software only.

Bit 10 **CACHEAXIRAMPUB**: Public protection of CACHEAXIRAM bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: CACHEAXIRAM configuration bits are accessible by non-public software only (default after reset).
1: CACHEAXIRAM configuration bits are accessible by public software only.

Bit 9 **FLEXRAMPUB**: Public protection of FLEXRAM bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: FLEXRAM configuration bits are accessible by non-public software only (default after reset).
1: FLEXRAM configuration bits are accessible by public software only.

Bit 8 **AXISRAM2PUB**: Public protection of AXISRAM2 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: AXISRAM2 configuration bits are accessible by non-public software only (default after reset).
1: AXISRAM2 configuration bits are accessible by public software only.

Bit 7 **AXISRAM1PUB**: Public protection of AXISRAM1 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: AXISRAM1 configuration bits are accessible by non-public software only (default after reset).
1: AXISRAM1 configuration bits are accessible by public software only.

Bit 6 **BKPSRAMPUB**: Public protection of BKPSRAM bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: BKPSRAM configuration bits are accessible by non-public software only (default after reset).
1: BKPSRAM configuration bits are accessible by public software only.

Bit 5 **AHBSRAM2PUB**: Public protection of AHBSRAM2 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: AHBSRAM2 configuration bits are accessible by non-public software only (default after reset).
1: AHBSRAM2 configuration bits are accessible by public software only.

Bit 4 **AHBSRAM1PUB**: Public protection of AHBSRAM1 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: AHBSRAM1 configuration bits are accessible by non-public software only (default after reset).
1: AHBSRAM1 configuration bits are accessible by public software only.

Bit 3 **AXISRAM6PUB**: Public protection of AXISRAM6 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: AXISRAM6 configuration bits are accessible by non-public software only (default after reset).
1: AXISRAM6 configuration bits are accessible by public software only.

Bit 2 **AXISRAM5PUB**: Public protection of AXISRAM5 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: AXISRAM5 configuration bits are accessible by non-public software only (default after reset).
1: AXISRAM5 configuration bits are accessible by public software only.

Bit 1 **AXISRAM4PUB**: Public protection of AXISRAM4 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: AXISRAM4 configuration bits are accessible by non-public software only (default after reset).
1: AXISRAM4 configuration bits are accessible by public software only.

Bit 0 **AXISRAM3PUB**: Public protection of AXISRAM3 bus configuration bits
This bit is set and reset by secure privileged software only. It can be read by any software.
0: AXISRAM3 configuration bits are accessible by non-public software only (default after reset).
1: AXISRAM3 configuration bits are accessible by public software only.

14.10.128 RCC control set register (RCC_CSR)

Address offset: 0x800

Reset value: 0x0000 0000

This register is used to enable the RCC oscillators and PLLs in Run or Sleep mode. It is reset by nreset_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	PLL4ONS	PLL3ONS	PLL2ONS	PLL1ONS	Res.	Res.	Res.	HSEONS	HSIONS	MSIONS	LSEONS	LSIONS
				w	w	w	w				w	w	w	w	w

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **PLL4ONS**: PLL4 oscillator enable
Written at 1 to set PLL4ON.

Bit 10 **PLL3ONS**: PLL3 oscillator enable
Written at 1 to set PLL3ON.

Bit 9 **PLL2ONS**: PLL2 oscillator enable
Written at 1 to set PLL2ON.

Bit 8 **PLL1ONS**: PLL1 oscillator enable
Written at 1 to set PLL1ON.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **HSEONS**: HSE oscillator enable
Written at 1 to set HSEON.

Bit 3 **HSIONS**: HSI oscillator enable
Written at 1 to set HSION.

Bit 2 **MSIONS**: MSI oscillator enable
Written at 1 to set MSION.

Bit 1 **LSEONS**: LSE oscillator enable
Written at 1 to set LSEON.

Bit 0 **LSIONS**: LSI oscillator enable
Written at 1 to set LSION.

14.10.129 RCC Stop mode configuration set register (RCC_STOPCSR)

Address offset: 0x808

Reset value: 0x0000 0000

This register is used to enable the RCC oscillators and PLLs in Stop mode. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

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

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

Bit 1 **HSISTOPENS**: HSI oscillator enable
Written at 1 to set HSISTOPEN.

Bit 0 **MSISTOPENS**: MSI oscillator enable
Written at 1 to set MSISTOPEN.

14.10.130 RCC miscellaneous reset register (RCC_MISCRSTSR)

Address offset: 0xA08

Reset value: 0x0000 0000

This register is used to reset the RCC miscellaneous. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC2DLLRSTS	SDMMC1DLLRSTS	Res.	XSPIPHY2RSTS	XSPIPHY1RSTS	Res.	Res.	Res.	DBGRSTS
							w	w		w	w				w

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **SDMMC2DLLRSTS**: SDMMC2DLL reset
Written at 1 to set SDMMC2DLLRST.

Bit 7 **SDMMC1DLLRSTS**: SDMMC1DLL reset
Written at 1 to set SDMMC1DLLRST.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **XSPIPHY2RSTS**: XSPIPHY2 reset
Written at 1 to set XSPIPHY2RST.

Bit 4 **XSPIPHY1RSTS**: XSPIPHY1 reset
Written at 1 to set XSPIPHY1RST.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **DBGRSTS**: DBG reset
Written at 1 to set DBGRST.

14.10.131 RCC memory reset register (RCC_MEMRSTSR)

Address offset: 0xA0C

Reset value: 0x0000 0000

This register is used to reset the RCC memories. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	BOOTROMRSTS	VENCRAMRSTS	CACHEAXIRAMRSTS	FLEXRAMRSTS	AXISRAM2RSTS	AXISRAM1RSTS	Res.	AHBSRAM2RSTS	AHBSRAM1RSTS	AXISRAM6RSTS	AXISRAM5RSTS	AXISRAM4RSTS	AXISRAM3RSTS
			w	w	w	w	w	w		w	w	w	w	w	w

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **BOOTROMRSTS**: BootROM reset
Written at 1 to set BOOTROMRST.

Bit 11 **VENCRAMRSTS**: VENCRAM reset
Written at 1 to set VENCRAMRST.

Bit 10 **CACHEAXIRAMRSTS**: CACHEAXIRAM reset
Written at 1 to set CACHEAXIRAM.

Bit 9 **FLEXRAMRSTS**: FLEXRAM reset
Written at 1 to set FLEXRAMRST.

Bit 8 **AXISRAM2RSTS**: AXISRAM2 reset
Written at 1 to set AXISRAM2RST.

Bit 7 **AXISRAM1RSTS**: AXISRAM1 reset
Written at 1 to set AXISRAM1RST.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **AHBSRAM2RSTS**: AHBSRAM2 reset
Written at 1 to set AHBSRAM2RST.

Bit 4 **AHBSRAM1RSTS**: AHBSRAM1 reset
Written at 1 to set AHBSRAM1RST.

Bit 3 **AXISRAM6RSTS**: AXISRAM6 reset
Written at 1 to set AXISRAM6RST.

Bit 2 **AXISRAM5RSTS**: AXISRAM5 reset
Written at 1 to set AXISRAM5RST.

Bit 1 **AXISRAM4RSTS**: AXISRAM4 reset
Written at 1 to set AXISRAM4RST.

Bit 0 **AXISRAM3RSTS**: AXISRAM3 reset
Written at 1 to set AXISRAM3RST.

14.10.132 RCC AHB1 reset register (RCC_AHB1RSTSR)

Address offset: 0xA10

Reset value: 0x0000 0000

This register is used to reset the RCC AHB1. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC12RSTS	GPDMA1RSTS	Res.	Res.	Res.	Res.
										w	w				

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

Bit 5 **ADC12RSTS**: ADC12 reset
Written at 1 to set ADC12RST.

Bit 4 **GPDMA1RSTS**: GPDMA1 reset
Written at 1 to set GPDMA1RST.

Bits 3:0 Reserved, must be kept at reset value.

14.10.133 RCC AHB2 reset register (RCC_AHB2RSTSR)

Address offset: 0xA14

Reset value: 0x0000 0000

This register is used to reset the RCC AHB2. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADF1 RSTS	MDF1 RSTS
														w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	RAMCFG RSTS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			w												

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

Bit 17 **ADF1RSTS**: ADF1 reset
Written at 1 to set ADF1RST.

Bit 16 **MDF1RSTS**: MDF1 reset
Written at 1 to set MDF1RST.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **RAMCFGRSTS**: RAMCFG reset
Written at 1 to set RAMCFGRST.

Bits 11:0 Reserved, must be kept at reset value.

14.10.134 RCC AHB3 reset register (RCC_AHB3RSTSR)

Address offset: 0xA18

Reset value: 0x0000 0000

This register is used to reset the RCC AHB3. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	IACRS TS	Res.	PKARS TS	Res.	Res.	Res.	SAESR STS	Res.	CRYPR STS	HASHR STS	RNGR STS
					w		w				w		w	w	w

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **IACRSTS**: IAC reset
Written at 1 to set IACRST.

Bit 9 Reserved, must be kept at reset value.

Bit 8 **PKARSTS**: PKA reset
Written at 1 to set PKARST.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **SAESRSTS**: SAES reset
Written at 1 to set SAESRST.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CRYPRSTS**: CRYP reset
Written at 1 to set CRYPRST.

Bit 1 **HASHRSTS**: HASH reset
Written at 1 to set HASHRST.

Bit 0 **RNGRSTS**: RNG reset
Written at 1 to set RNGRST.

14.10.135 RCC AHB4 reset register (RCC_AHB4RSTSR)

Address offset: 0xA1C

Reset value: 0x0000 0000

This register is used to reset the RCC AHB4. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCRSTS	PWRRSTS	Res.	GPIOQRSTS
												w	w		w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIOPRSTS	GPIOORSTS	GPIONRSTS	Res.	Res.	Res.	Res.	Res.	GPIOHRSTS	GPIOGRSTS	GPIOFRSTS	GPIOERSTS	GPIODRSTS	GPIOCRSTS	GPIOBRSTS	GPIOARSTS
w	w	w						w	w	w	w	w	w	w	w

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **CRCRSTS**: CRC reset
Written at 1 to set CRCRST.

Bit 18 **PWRRSTS**: PWR reset
Written at 1 to set PWRRST.

Bit 17 Reserved, must be kept at reset value.

Bit 16 **GPIOQRSTS**: GPIO Q reset
Written at 1 to set GPIOQRST.

Bit 15 **GPIOPRSTS**: GPIO P reset
Written at 1 to set GPIOPRST.

Bit 14 **GPIOORSTS**: GPIO O reset
Written at 1 to set GPIOORST.

Bit 13 **GPIONRSTS**: GPIO N reset
Written at 1 to set GPIONRST.

Bits 12:8 Reserved, must be kept at reset value.

Bit 7 **GPIOHRSTS**: GPIO H reset
Written at 1 to set GPIOHRST.

Bit 6 **GPIOGRSTS**: GPIO G reset
Written at 1 to set GPIOGRST.

Bit 5 **GPIOFRSTS**: GPIO F reset
Written at 1 to set GPIOFRST.

Bit 4 **GPIOERSTS**: GPIO E reset
Written at 1 to set GPIOERST.

Bit 3 **GPIODRSTS**: GPIO D reset
Written at 1 to set GPIODRST.

Bit 2 **GPIOCRSTS**: GPIO C reset
Written at 1 to set GPIOCRST.

Bit 1 **GPIOBRSTS**: GPIO B reset
Written at 1 to set GPIOBRST.

Bit 0 **GPIOARSTS**: GPIO A reset
Written at 1 to set GPIOARST.

14.10.136 RCC AHB5 reset register (RCC_AHB5RSTSR)

Address offset: 0xA20

Reset value: 0x0000 0000

This register is used to reset the RCC AHB5. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NPURSTS	CACHEAXIRSTS	OTG2RSTS	OTGPHY2RSTS	OTGPHY1RSTS	OTG1RSTS	ETH1RSTS	OTG2PHYCTLRSTS	OTG1PHYCTLRSTS	Res.	Res.	GPU2DRSTS	GFXMMURSTS	Res.	XSPI3RSTS	Res.
w	w	w	w	w	w	w	w	w			w	w		w	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	XSPIMRSTS	XSPI2RSTS	Res.	Res.	Res.	SDMMC1RSTS	SDMMC2RSTS	PSSIRSTS	XSPI1RSTS	FMCRSTS	JPEGRSTS	Res.	DMA2DRSTS	HPDMA1RSTS
		w	w				w	w	w	w	w	w		w	w

Bit 31 **NPURSTS**: NPU reset
Written at 1 to set NPURST.

Bit 30 **CACHEAXIRSTS**: CACHEAXI reset
Written at 1 to set CACHEAXIRST.

Bit 29 **OTG2RSTS**: OTG2 reset
Written at 1 to set OTG2RST.

Bit 28 **OTGPHY2RSTS**: OTGPHY2 reset
Written at 1 to set OTGPHY2RST.

Bit 27 **OTGPHY1RSTS**: OTGPHY1 reset
Written at 1 to set OTGPHY1RST.

Bit 26 **OTG1RSTS**: OTG1 reset
Written at 1 to set OTG1RST.

Bit 25 **ETH1RSTS**: ETH1 reset
Written at 1 to set ETH1RST.

Bit 24 **OTG2PHYCTLRSTS**: OTG2PHYCTL reset
Written at 1 to set OTG2PHYCTLRST.

Bit 23 **OTG1PHYCTLRSTS**: OTG1PHYCTL reset
Written at 1 to set OTG1PHYCTLRST.

Bits 22:21 Reserved, must be kept at reset value.

Bit 20 **GPU2DRSTS**: GPU2D reset
Written at 1 to set GPU2DRST.

Bit 19 **GFXMMURSTS**: GFXMMU reset
Written at 1 to set GFXMMURST.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **XSPI3RSTS**: XSPI3 reset
Written at 1 to set XSPI3RST.

Bits 16:14 Reserved, must be kept at reset value.

Bit 13 **XSPIMRSTS**: XSPIM reset
Written at 1 to set XSPIMRST.

Bit 12 **XSPI2RSTS**: XSPI2 reset
Written at 1 to set XSPI2RST.

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **SDMMC1RSTS**: SDMMC1 reset
Written at 1 to set SDMMC1RST.

Bit 7 **SDMMC2RSTS**: SDMMC2 reset
Written at 1 to set SDMMC2RST.

Bit 6 **PSSIRSTS**: PSSI reset
Written at 1 to set PSSIRST.

Bit 5 **XSPI1RSTS**: XSPI1 reset
Written at 1 to set XSPI1RST.

Bit 4 **FMCRSTS**: FMC reset
Written at 1 to set FMCRST.

Bit 3 **JPEGRSTS**: JPEG reset
Written at 1 to set JPEGRST.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **DMA2DRSTS**: DMA2D reset
Written at 1 to set DMA2DRST.

Bit 0 **HPDMA1RSTS**: HPDMA1 reset
Written at 1 to set HPDMA1RST.

14.10.137 RCC APB1L reset register (RCC_APB1LRSTSR)

Address offset: 0xA24

Reset value: 0x0000 0000

This register is used to reset the RCC APB1L. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UART8 RSTS	UART7 RSTS	Res.	Res.	Res.	Res.	I3C2RS TS	I3C1RS TS	I2C3RS TS	I2C2RS TS	I2C1RS TS	UART5 RSTS	UART4 RSTS	USART 3RSTS	USART 2RSTS	SPDIF RX1RS TS
w	w					w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3R STS	SPI2R STS	TIM11R STS	TIM10 RSTS	WWDG RSTS	Res.	LPTIM1 RSTS	TIM14 RSTS	TIM13 RSTS	TIM12 RSTS	TIM7R STS	TIM6R STS	TIM5R STS	TIM4R STS	TIM3R STS	TIM2R STS
w	w	w	w	w		w	w	w	w	w	w	w	w	w	w

Bit 31 **UART8RSTS**: UART8 reset
Written at 1 to set UART8RST.

Bit 30 **UART7RSTS**: UART7 reset
Written at 1 to set UART7RST.

Bits 29:26 Reserved, must be kept at reset value.

Bit 25 **I3C2RSTS**: I3C2 reset
Written at 1 to set I3C2RST.

Bit 24 **I3C1RSTS**: I3C1 reset
Written at 1 to set I3C1RST.

Bit 23 **I2C3RSTS**: I2C3 reset
Written at 1 to set I2C3RST.

Bit 22 **I2C2RSTS**: I2C2 reset
Written at 1 to set I2C2RST.

Bit 21 **I2C1RSTS**: I2C1 reset
Written at 1 to set I2C1RST.

Bit 20 **UART5RSTS**: UART5 reset
Written at 1 to set UART5RST.

Bit 19 **UART4RSTS**: UART4 reset
Written at 1 to set UART4RST.

Bit 18 **USART3RSTS**: USART3 reset
Written at 1 to set USART3RST.

Bit 17 **USART2RSTS**: USART2 reset
Written at 1 to set USART2RST.

Bit 16 **SPDIFRX1RSTS**: SPDIFRX1 reset
Written at 1 to set SPDIFRX1RST.

Bit 15 **SPI3RSTS**: SPI3 reset
Written at 1 to set SPI3RST.

Bit 14 **SPI2RSTS**: SPI2 reset
Written at 1 to set SPI2RST.

Bit 13 **TIM11RSTS**: TIM11 reset
Written at 1 to set TIM11RST.

Bit 12 **TIM10RSTS**: TIM10 reset
Written at 1 to set TIM10RST.

Bit 11 **WWDGRSTS**: WWDG reset
Written at 1 to set WWDGRST.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **LPTIM1RSTS**: LPTIM1 reset
Written at 1 to set LPTIM1RST.

Bit 8 **TIM14RSTS**: TIM14 reset
Written at 1 to set TIM14RST.

Bit 7 **TIM13RSTS**: TIM13 reset
Written at 1 to set TIM13RST.

Bit 6 **TIM12RSTS**: TIM12 reset
Written at 1 to set TIM12RST.

Bit 5 **TIM7RSTS**: TIM7 reset
Written at 1 to set TIM7RST.

Bit 4 **TIM6RSTS**: TIM6 reset
Written at 1 to set TIM6RST.

Bit 3 **TIM5RSTS**: TIM5 reset
Written at 1 to set TIM5RST.

Bit 2 **TIM4RSTS**: TIM4 reset
Written at 1 to set TIM4RST.

Bit 1 **TIM3RSTS**: TIM3 reset
Written at 1 to set TIM3RST.

Bit 0 **TIM2RSTS**: TIM2 reset
Written at 1 to set TIM2RST.

14.10.138 RCC APB1H reset register (RCC_APB1HRSTSR)

Address offset: 0xA28

Reset value: 0x0000 0000

This register is used to reset the RCC APB1H. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD1 RSTS	Res.	Res.
													w		

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

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **UCPD1RSTS**: UCPD1 reset
Written at 1 to set UCPD1RST.

Bits 17:9 Reserved, must be kept at reset value.

Bit 8 **FDCANRSTS**: FDCAN reset
Written at 1 to set FDCANRST.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **MDIOSRSTS**: MDIOS reset
Written at 1 to set MDIOSRST.

Bits 4:0 Reserved, must be kept at reset value.

14.10.139 RCC APB2 reset register (RCC_APB2RSTSR)

Address offset: 0xA2C

Reset value: 0x0000 0000

This register is used to reset the RCC APB2. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI2RSTS	SAI1RSTS	SPI5RSTS	TIM9RSTS	TIM17RSTS	TIM16RSTS	TIM15RSTS
									w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIM18RSTS	Res.	SPI4RSTS	SPI1RSTS	Res.	Res.	Res.	Res.	USART10RSTS	UART9RSTS	USART6RSTS	USART1RSTS	Res.	Res.	TIM8RSTS	TIM1RSTS
w		w	w					w	w	w	w			w	w

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **SAI2RSTS**: SAI2 reset
Written at 1 to set SAI2RST.

Bit 21 **SAI1RSTS**: SAI1 reset
Written at 1 to set SAI1RST.

Bit 20 **SPI5RSTS**: SPI5 reset
Written at 1 to set SPI5RST.

Bit 19 **TIM9RSTS**: TIM9 reset
Written at 1 to set TIM9RST.

Bit 18 **TIM17RSTS**: TIM17 reset
Written at 1 to set TIM17RST.

Bit 17 **TIM16RSTS**: TIM16 reset
Written at 1 to set TIM16RST.

Bit 16 **TIM15RSTS**: TIM15 reset
Written at 1 to set TIM15RST.

Bit 15 **TIM18RSTS**: TIM18 reset
Written at 1 to set TIM18RST.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **SPI4RSTS**: SPI4 reset
Written at 1 to set SPI4RST.

Bit 12 **SPI1RSTS**: SPI1 reset
Written at 1 to set SPI1RST.

Bits 11:8 Reserved, must be kept at reset value.

Bit 7 **USART10RSTS**: USART10 reset
Written at 1 to set USART10RST.

Bit 6 **UART9RSTS**: UART9 reset
Written at 1 to set UART9RST.

Bit 5 **USART6RSTS**: USART6 reset
Written at 1 to set USART6RST.

Bit 4 **USART1RSTS**: USART1 reset
Written at 1 to set USART1RST.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **TIM8RSTS**: TIM8 reset
Written at 1 to set TIM8RST.

Bit 0 **TIM1RSTS**: TIM1 reset
Written at 1 to set TIM1RST.

14.10.140 RCC APB4L reset register (RCC_APB4LRSTSR)

Address offset: 0xA34

Reset value: 0x0000 0000

This register is used to reset the RCC APB4L. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCRSTS
															w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VREFBUFRSTS	Res.	Res.	LPTIM5RSTS	LPTIM4RSTS	LPTIM3RSTS	LPTIM2RSTS	Res.	I2C4RSTS	Res.	SPI6RSTS	Res.	LPUART1RSTS	HDPRSTS	Res.	Res.
w			w	w	w	w		w		w		w	w		

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **RTCRSTS**: RTC reset
Written at 1 to set RTCRST.

Bit 15 **VREFBUFRSTS**: VREFBUF reset
Written at 1 to set VREFBUFRST.

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **LPTIM5RSTS**: LPTIM5 reset
Written at 1 to set LPTIM5RST.

Bit 11 **LPTIM4RSTS**: LPTIM4 reset
Written at 1 to set LPTIM4RST.

Bit 10 **LPTIM3RSTS**: LPTIM3 reset
Written at 1 to set LPTIM3RST.

Bit 9 **LPTIM2RSTS**: LPTIM2 reset
Written at 1 to set LPTIM2RST.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **I2C4RSTS**: I2C4 reset
Written at 1 to set I2C4RST.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **SPI6RSTS**: SPI6 reset
Written at 1 to set SPI6RST.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **LPUART1RSTS**: LPUART1 reset
Written at 1 to set LPUART1RST.

Bit 2 **HDPRSTS**: HDP reset
Written at 1 to set HDPRST.

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

14.10.141 RCC APB4H reset register (RCC_APB4HRSTSR)

Address offset: 0xA38

Reset value: 0x0000 0000

This register is used to reset the RCC APB4H. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

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

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

Bit 2 **DTSRSTS**: DTS reset
Written at 1 to set DTSRST.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **SYSCFGRSTS**: SYSCFG reset
Written at 1 to set SYSCFGRST.

14.10.142 RCC APB5 reset register (RCC_APB5RSTSR)

Address offset: 0xA3C

Reset value: 0x0000 0000

This register is used to reset the RCC APB5. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

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

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CSIRSTS**: CSI reset
Written at 1 to set CSIRST.

Bit 5 **VENCRSTS**: VENC reset
Written at 1 to set VENCRST.

Bit 4 **GFXTIMRSTS**: GFXTIM reset
Written at 1 to set GFXTIMRST.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **DCMIPPRSTS**: DCMIPP reset
Written at 1 to set DCMIPPRST.

Bit 1 **LTDCRSTS**: LTDC reset
Written at 1 to set LTDCRST.

Bit 0 Reserved, must be kept at reset value.

14.10.143 RCC divider enable register (RCC_DIVENSR)

Address offset: 0xA40

Reset value: 0x0000 0000

This register is used to enable the RCC IC dividers in Run, Sleep, or Stop mode. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC20ENS	IC19ENS	IC18ENS	IC17ENS
												w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC16ENS	IC15ENS	IC14ENS	IC13ENS	IC12ENS	IC11ENS	IC10ENS	IC9ENS	IC8ENS	IC7ENS	IC6ENS	IC5ENS	IC4ENS	IC3ENS	IC2ENS	IC1ENS
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **IC20ENS**: IC20 enable
Written at 1 to set IC20EN.

Bit 18 **IC19ENS**: IC19 enable
Written at 1 to set IC19EN.

Bit 17 **IC18ENS**: IC18 enable
Written at 1 to set IC18EN.

Bit 16 **IC17ENS**: IC17 enable
Written at 1 to set IC17EN.

Bit 15 **IC16ENS**: IC16 enable
Written at 1 to set IC16EN.

Bit 14 **IC15ENS**: IC15 enable
Written at 1 to set IC15EN.

Bit 13 **IC14ENS**: IC14 enable
Written at 1 to set IC14EN.

Bit 12 **IC13ENS**: IC13 enable
Written at 1 to set IC13EN.

Bit 11 **IC12ENS**: IC12 enable
Written at 1 to set IC12EN.

Bit 10 **IC11ENS**: IC11 enable
Written at 1 to set IC11EN.

Bit 9 **IC10ENS**: IC10 enable
Written at 1 to set IC10EN.

Bit 8 **IC9ENS**: IC9 enable
Written at 1 to set IC9EN.

Bit 7 **IC8ENS**: IC8 enable
Written at 1 to set IC8EN.

Bit 6 **IC7ENS**: IC7 enable
Written at 1 to set IC7EN.

Bit 5 **IC6ENS**: IC6 enable
Written at 1 to set IC6EN.

Bit 4 **IC5ENS**: IC5 enable
Written at 1 to set IC5EN.

Bit 3 **IC4ENS**: IC4 enable
Written at 1 to set IC4EN.

Bit 2 **IC3ENS**: IC3 enable
Written at 1 to set IC3EN.

Bit 1 **IC2ENS**: IC2 enable
Written at 1 to set IC2EN.

Bit 0 **IC1ENS**: IC1 enable
Written at 1 to set IC1EN.

14.10.144 RCC bus enable register (RCC_BUSENSR)

Address offset: 0xA44

Reset value: 0x0000 0000

This register is used to enable the RCC bus in Run and Sleep modes (in Sleep mode, each bit of this register is AND-ed with the LPEN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

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

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

Bit 1 **ACLKNCENS**: ACLKNC enable
Written at 1 to set ACLKNCEN.

Bit 0 **ACLKNENS**: ACLKN enable
Written at 1 to set ACLKNEN.

14.10.145 RCC miscellaneous enable register (RCC_MISCENSR)

Address offset: 0xA48

Reset value: 0x0000 0000

This register is used to enable the RCC miscellaneous in Run and Sleep modes (in Sleep mode, each bit of this register is AND-ed with the LPEN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERENS	Res.	Res.	XSPIPHYCOMPENS	MCO2ENS	MCO1ENS	DBGENS
									w			w	w	w	w

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **PERENS**: PER enable
Written at 1 to set PEREN.

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **XSPIPHYCOMPENS**: XSPIPHYCOMP enable
Written at 1 to set XSPIPHYCOMPEN.

Bit 2 **MCO2ENS**: MCO2 enable
Written at 1 to set MCO2EN.

Bit 1 **MCO1ENS**: MCO1 enable
Written at 1 to set MCO1EN.

Bit 0 **DBGENS**: DBG enable
Written at 1 to set DBGEN.

14.10.146 RCC memory enable register (RCC_MEMENSR)

Address offset: 0xA4C

Reset value: 0x0000 0000

This register is used to enable the RCC memory in Run and Sleep modes (in Sleep mode, each bit of this register is AND-ed with the LPEN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	BOOTROMENS	VENCRAMENS	CACHEAXIRAMENS	FLEXRAMENS	AXISRAM2ENS	AXISRAM1ENS	BKPSRAMENS	AHBSRAM2ENS	AHBSRAM1ENS	AXISRAM6ENS	AXISRAM5ENS	AXISRAM4ENS	AXISRAM3ENS
			w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **BOOTROMENS**: BootROM enable
Written at 1 to set BOOTROMEN.

Bit 11 **VENCRAMENS**: VENCRAM enable
Written at 1 to set VENCRAMEN.

Bit 10 **CACHEAXIRAMENS**: CACHEAXIRAM enable
Written at 1 to set CACHEAXIRAMEN.

Bit 9 **FLEXRAMENS**: FLEXRAM enable
Written at 1 to set FLEXRAMEN.

Bit 8 **AXISRAM2ENS**: AXISRAM2 enable
Written at 1 to set AXISRAM2EN.

Bit 7 **AXISRAM1ENS**: AXISRAM1 enable
Written at 1 to set AXISRAM1EN.

Bit 6 **BKPSRAMENS**: BKPSRAM enable
Written at 1 to set BKPSRAMEN.

Bit 5 **AHBSRAM2ENS**: AHBSRAM2 enable
Written at 1 to set AHBSRAM2EN.

Bit 4 **AHBSRAM1ENS**: AHBSRAM1 enable
Written at 1 to set AHBSRAM1EN.

Bit 3 **AXISRAM6ENS**: AXISRAM6 enable
Written at 1 to set AXISRAM6EN.

Bit 2 **AXISRAM5ENS**: AXISRAM5 enable
Written at 1 to set AXISRAM5EN.

Bit 1 **AXISRAM4ENS**: AXISRAM4 enable
Written at 1 to set AXISRAM4EN.

Bit 0 **AXISRAM3ENS**: AXISRAM3 enable
Written at 1 to set AXISRAM3EN.

14.10.147 RCC AHB1 enable register (RCC_AHB1ENSR)

Address offset: 0xA50

Reset value: 0x0000 0000

This register is used to enable the RCC AHB1 in Run and Sleep modes (in Sleep mode, each bit of this register is AND-ed with the LPEN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC12ENS	GPDMA1ENS	Res.	Res.	Res.	Res.
										w	w				

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

Bit 5 **ADC12ENS**: ADC12 enable
Written at 1 to set ADC12EN.

Bit 4 **GPDMA1ENS**: GPDMA1 enable
Written at 1 to set GPDMA1EN.

Bits 3:0 Reserved, must be kept at reset value.

14.10.148 RCC AHB2 enable register (RCC_AHB2ENSR)

Address offset: 0xA54

Reset value: 0x0000 0000

This register is used to enable the RCC AHB2 in Run and Sleep modes (in Sleep mode, each bit of this registers is AND-ed with the LPEN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADF1ENS	MDF1ENS
														w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	RAMCFGENS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			w												

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

Bit 17 **ADF1ENS**: ADF1 enable
Written at 1 to set ADF1EN.

Bit 16 **MDF1ENS**: MDF1 enable
Written at 1 to set MDF1EN.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **RAMCFGENS**: RAMCFG enable
Written at 1 to set RAMCFGEN.

Bits 11:0 Reserved, must be kept at reset value.

14.10.149 RCC AHB3 enable register (RCC_AHB3ENSR)

Address offset: 0xA58

Reset value: 0x0000 0000

This register is used to enable the RCC AHB3 in Run and Sleep modes (in Sleep mode, each bit of this register is AND-ed with the LPEN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RISAFENS	Res.	Res.	Res.	IACENS	RIFSCENS	PKAENS	Res.	Res.	Res.	SAESENS	Res.	CRYPENS	HASHENS	RNGENS
	w				w	w	w				w		w	w	w

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **RISAFENS**: RISAF enable
Written at 1 to set RISAFEN.

Bits 13:11 Reserved, must be kept at reset value.

Bit 10 **IACENS**: IAC enable
Written at 1 to set IACEN.

Bit 9 **RIFSCENS**: RIFSC enable
Written at 1 to set RIFSCEN.

Bit 8 **PKAENS**: PKA enable
Written at 1 to set PKAEN.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **SAESENS**: SAES enable
Written at 1 to set SAESEN.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CRYPENS**: CRYP enable
Written at 1 to set CRYPEN.

Bit 1 **HASHENS**: HASH enable
Written at 1 to set HASHEN.

Bit 0 **RNGENS**: RNG enable
Written at 1 to set RNGEN.

14.10.150 RCC AHB4 enable register (RCC_AHB4ENSR)

Address offset: 0xA5C

Reset value: 0x0000 0000

This register is used to enable the RCC AHB4 in Run and Sleep modes (in Sleep mode, each bit of this register is AND-ed with the LPEN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCENS	PWRENS	Res.	GPIOQENS
												w	w		w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIOPENS	GPIOOENS	GPIONENS	Res.	Res.	Res.	Res.	Res.	GPIOHENS	GPIOGENS	GPIOFENS	GPIOEENS	GPIODENS	GPIOCENS	GPIOBENS	GPIOAENS
w	w	w						w	w	w	w	w	w	w	w

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **CRCENS**: CRC enable
Written at 1 to set CRCEN.

Bit 18 **PWRENS**: PWR enable
Written at 1 to set PWREN.

Bit 17 Reserved, must be kept at reset value.

Bit 16 **GPIOQENS**: GPIO Q enable
Written at 1 to set GPIOQEN.

Bit 15 **GPIOPENS**: GPIO P enable
Written at 1 to set GPIOPEN.

Bit 14 **GPIOOENS**: GPIO O enable
Written at 1 to set GPIOOEN.

Bit 13 **GPIONENS**: GPIO N enable
Written at 1 to set GPIONEN.

Bits 12:8 Reserved, must be kept at reset value.

Bit 7 **GPIOHENS**: GPIO H enable
Written at 1 to set GPIOHEN.

Bit 6 **GPIOGENS**: GPIO G enable
Written at 1 to set GPIOGEN.

Bit 5 **GPIOFENS**: GPIO F enable
Written at 1 to set GPIOFEN.

Bit 4 **GPIOEENS**: GPIO E enable
Written at 1 to set GPIOEEN.

Bit 3 **GPIODENS**: GPIO D enable
Written at 1 to set GPIODEN.

Bit 2 **GPIOCENS**: GPIO C enable
Written at 1 to set GPIOCEN.

Bit 1 **GPIOBENS**: GPIO B enable
Written at 1 to set GPIOBEN.

Bit 0 **GPIOAENS**: GPIO A enable
Written at 1 to set GPIOAEN.

14.10.151 RCC AHB5 enable register (RCC_AHB5ENSR)

Address offset: 0xA60

Reset value: 0x0000 0000

This register is used to enable the RCC AHB5 in Run and Sleep modes (in Sleep mode, each bit of this register is AND-ed with the LPEN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NPUENS	CACHEAXIENS	OTG2ENS	OTGPHY2ENS	OTGPHY1ENS	OTG1ENS	ETH1ENS	ETH1RXENS	ETH1TXENS	ETH1MACENS	Res.	GPU2DENS	GFXMMUENS	MCE4ENS	XSPI3ENS	MCE3ENS
w	w	w	w	w	w	w	w	w	w		w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCE2ENS	MCE1ENS	XSPIMENS	XSPI2ENS	Res.	Res.	Res.	SDMMC1ENS	SDMMC2ENS	PSSIENS	XSPI1ENS	FMCENS	JPEGENS	Res.	DMA2DENS	HPDMA1ENS
w	w	w	w				w	w	w	w	w	w		w	w

Bit 31 **NPUENS**: NPU enable
Written at 1 to set NPUEN.

Bit 30 **CACHEAXIENS**: CACHEAXI enable
Written at 1 to set CACHEAXIEN.

Bit 29 **OTG2ENS**: OTG2 enable
Written at 1 to set OTG2EN.

Bit 28 **OTGPHY2ENS**: OTGPHY2 enable
Written at 1 to set OTGPHY2EN.

Bit 27 **OTGPHY1ENS**: OTGPHY1 enable
Written at 1 to set OTGPHY1EN.

Bit 26 **OTG1ENS**: OTG1 enable
Written at 1 to set OTG1EN.

Bit 25 **ETH1ENS**: ETH1 enable
Written at 1 to set ETH1EN.

Bit 24 **ETH1RXENS**: ETH1RX enable
Written at 1 to set ETH1RXEN.

Bit 23 **ETH1TXENS**: ETH1TX enable
Written at 1 to set ETH1TXEN.

Bit 22 **ETH1MACENS**: ETH1MAC enable
Written at 1 to set ETH1MACEN.

Bit 21 Reserved, must be kept at reset value.

Bit 20 **GPU2DENS**: GPU2D enable
Written at 1 to set GPU2DEN.

Bit 19 **GFXMMUENS**: GFXMMU enable
Written at 1 to set GFXMMUEN.

Bit 18 **MCE4ENS**: MCE4 enable
Written at 1 to set MCE4EN.

Bit 17 **XSPI3ENS**: XSPI3 enable
Written at 1 to set XSPI3EN.

Bit 16 **MCE3ENS**: MCE3 enable
Written at 1 to set MCE3EN.

Bit 15 **MCE2ENS**: MCE2 enable
Written at 1 to set MCE2EN.

Bit 14 **MCE1ENS**: MCE1 enable
Written at 1 to set MCE1EN.

Bit 13 **XSPIMENS**: XSPIM enable
Written at 1 to set XSPIMEN.

Bit 12 **XSPI2ENS**: XSPI2 enable
Written at 1 to set XSPI2EN.

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **SDMMC1ENS**: SDMMC1 enable
Written at 1 to set SDMMC1EN.

Bit 7 **SDMMC2ENS**: SDMMC2 enable
Written at 1 to set SDMMC2EN.

Bit 6 **PSSIENS**: PSSI enable
Written at 1 to set PSSIEN.

Bit 5 **XSPI1ENS**: XSPI1 enable
Written at 1 to set XSPI1EN.

Bit 4 **FMCENS**: FMC enable
Written at 1 to set FMCEN.

Bit 3 **JPEGENS**: JPEG enable
Written at 1 to set JPEGEN.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **DMA2DENS**: DMA2D enable
Written at 1 to set DMA2DEN.

Bit 0 **HPDMA1ENS**: HPDMA1 enable
Written at 1 to set HPDMA1EN.

14.10.152 RCC APB1L enable register (RCC_APB1LENSR)

Address offset: 0xA64

Reset value: 0x0000 0000

This register is used to enable the RCC APB1L in Run and Sleep modes (in Sleep mode, each bit of this register is AND-ed with the LPEN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UART8ENS	UART7ENS	Res.	Res.	Res.	Res.	I3C2ENS	I3C1ENS	I2C3ENS	I2C2ENS	I2C1ENS	UART5ENS	UART4ENS	USART3ENS	USART2ENS	SPDIFRX1ENS
w	w					w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3ENS	SPI2ENS	TIM11ENS	TIM10ENS	WWDGENS	Res.	LPTIM1ENS	TIM14ENS	TIM13ENS	TIM12ENS	TIM7ENS	TIM6ENS	TIM5ENS	TIM4ENS	TIM3ENS	TIM2ENS
w	w	w	w	w		w	w	w	w	w	w	w	w	w	w

Bit 31 **UART8ENS**: UART8 enable
Written at 1 to set UART8EN.

Bit 30 **UART7ENS**: UART7 enable
Written at 1 to set UART7EN.

Bits 29:26 Reserved, must be kept at reset value.

Bit 25 **I3C2ENS**: I3C2 enable
Written at 1 to set I3C2EN.

Bit 24 **I3C1ENS**: I3C1 enable
Written at 1 to set I3C1EN.

Bit 23 **I2C3ENS**: I2C3 enable
Written at 1 to set I2C3EN.

Bit 22 **I2C2ENS**: I2C2 enable
Written at 1 to set I2C2EN.

Bit 21 **I2C1ENS**: I2C1 enable
Written at 1 to set I2C1EN.

Bit 20 **UART5ENS**: UART5 enable
Written at 1 to set UART5EN.

Bit 19 **UART4ENS**: UART4 enable
Written at 1 to set UART4EN.

Bit 18 **USART3ENS**: USART3 enable
Written at 1 to set USART3EN.

Bit 17 **USART2ENS**: USART2 enable
Written at 1 to set USART2EN.

Bit 16 **SPDIFRX1ENS**: SPDIFRX1 enable
Written at 1 to set SPDIFRX1EN.

Bit 15 **SPI3ENS**: SPI3 enable
Written at 1 to set SPI3EN.

Bit 14 **SPI2ENS**: SPI2 enable
Written at 1 to set SPI2EN.

Bit 13 **TIM11ENS**: TIM11 enable
Written at 1 to set TIM11EN.

Bit 12 **TIM10ENS**: TIM10 enable
Written at 1 to set TIM10EN.

Bit 11 **WWDGENS**: WWDG enable
Written at 1 to set WWDGEN.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **LPTIM1ENS**: LPTIM1 enable
Written at 1 to set LPTIM1EN.

Bit 8 **TIM14ENS**: TIM14 enable
Written at 1 to set TIM14EN.

Bit 7 **TIM13ENS**: TIM13 enable
Written at 1 to set TIM13EN.

Bit 6 **TIM12ENS**: TIM12 enable
Written at 1 to set TIM12EN.

Bit 5 **TIM7ENS**: TIM7 enable
Written at 1 to set TIM7EN.

Bit 4 **TIM6ENS**: TIM6 enable
Written at 1 to set TIM6EN.

Bit 3 **TIM5ENS**: TIM5 enable
Written at 1 to set TIM5EN.

Bit 2 **TIM4ENS**: TIM4 enable
Written at 1 to set TIM4EN.

Bit 1 **TIM3ENS**: TIM3 enable
Written at 1 to set TIM3EN.

Bit 0 **TIM2ENS**: TIM2 enable
Written at 1 to set TIM2EN.

14.10.153 RCC APB1H enable register (RCC_APB1HENSR)

Address offset: 0xA68

Reset value: 0x0000 0000

This register is used to enable the RCC APB1H in Run and Sleep modes (in Sleep mode, each bit of this register is AND-ed with the LPEN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD1ENS	Res.	Res.
													w		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANENS	Res.	Res.	MDIOSENS	Res.	Res.	Res.	Res.	Res.
							w			w					

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **UCPD1ENS**: UCPD1 enable
Written at 1 to set UCPD1EN.

Bits 17:9 Reserved, must be kept at reset value.

Bit 8 **FDCANENS**: FDCAN enable
Written at 1 to set FDCANEN.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **MDIOSENS**: MDIOS enable
Written at 1 to set MDIOSEN.

Bits 4:0 Reserved, must be kept at reset value.

14.10.154 RCC APB2 enable register (RCC_APB2ENSR)

Address offset: 0xA6C

Reset value: 0x0000 0000

This register is used to enable the RCC APB2 in Run and Sleep modes (in Sleep mode, each bit in this register is AND-ed with the LPEN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI2ENS	SAI1ENS	SPI5ENS	TIM9ENS	TIM17ENS	TIM16ENS	TIM15ENS
									w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIM18ENS	Res.	SPI4ENS	SPI1ENS	Res.	Res.	Res.	Res.	USART10ENS	UART9ENS	USART6ENS	USART1ENS	Res.	Res.	TIM8ENS	TIM1ENS
w		w	w					w	w	w	w			w	w

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **SAI2ENS**: SAI2 enable
Written at 1 to set SAI2EN.

Bit 21 **SAI1ENS**: SAI1 enable
Written at 1 to set SAI1EN.

Bit 20 **SPI5ENS**: SPI5 enable
Written at 1 to set SPI5EN.

Bit 19 **TIM9ENS**: TIM9 enable
Written at 1 to set TIM9EN.

Bit 18 **TIM17ENS**: TIM17 enable
Written at 1 to set TIM17EN.

Bit 17 **TIM16ENS**: TIM16 enable
Written at 1 to set TIM16EN.

Bit 16 **TIM15ENS**: TIM15 enable
Written at 1 to set TIM15EN.

Bit 15 **TIM18ENS**: TIM18 enable
Written at 1 to set TIM18EN.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **SPI4ENS**: SPI4 enable
Written at 1 to set SPI4EN.

Bit 12 **SPI1ENS**: SPI1 enable
Written at 1 to set SPI1EN.

Bits 11:8 Reserved, must be kept at reset value.

Bit 7 **USART10ENS**: USART10 enable
Written at 1 to set USART10EN.

Bit 6 **UART9ENS**: UART9 enable
Written at 1 to set UART9EN.

Bit 5 **USART6ENS**: USART6 enable
Written at 1 to set USART6EN.

Bit 4 **USART1ENS**: USART1 enable
Written at 1 to set USART1EN.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **TIM8ENS**: TIM8 enable
Written at 1 to set TIM8EN.

Bit 0 **TIM1ENS**: TIM1 enable
Written at 1 to set TIM1EN.

14.10.155 RCC APB3 enable register (RCC_APB3ENSR)

Address offset: 0xA70

Reset value: 0x0000 0000

This register is used to enable the RCC APB3 in Run and Sleep modes (in Sleep mode, each bit of this register is AND-ed with the LPEN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

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

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

Bit 2 **DFTENS**: DFT enable
Written at 1 to set DFTEN.

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

14.10.156 RCC APB4L enable register (RCC_APB4LENSR)

Address offset: 0xA74

Reset value: 0x0000 0000

This register is used to enable the RCC APB4L in Run and Sleep modes (in Sleep mode, each bit of this register is AND-ed with the LPEN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCAPBENS	RTCENS
														w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VREFBUFENS	Res.	Res.	LPTIM5ENS	LPTIM4ENS	LPTIM3ENS	LPTIM2ENS	Res.	I2C4ENS	Res.	SPI6ENS	Res.	LPUART1ENS	HDPENS	Res.	Res.
w			w	w	w	w		w		w		w	w		

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

Bit 17 **RTCAPBENS**: RTCAPB enable
Written at 1 to set RTCAPBEN.

Bit 16 **RTCENS**: RTC enable
Written at 1 to set RTCEN.

Bit 15 **VREFBUFENS**: VREFBUF enable
Written at 1 to set VREFBUFEN.

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **LPTIM5ENS**: LPTIM5 enable
Written at 1 to set LPTIM5EN.

Bit 11 **LPTIM4ENS**: LPTIM4 enable
Written at 1 to set LPTIM4EN.

Bit 10 **LPTIM3ENS**: LPTIM3 enable
Written at 1 to set LPTIM3EN.

Bit 9 **LPTIM2ENS**: LPTIM2 enable
Written at 1 to set LPTIM2EN.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **I2C4ENS**: I2C4 enable
Written at 1 to set I2C4EN.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **SPI6ENS**: SPI6 enable
Written at 1 to set SPI6EN.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **LPUART1ENS**: LPUART1 enable
Written at 1 to set LPUART1EN.

Bit 2 **HDPENS**: HDP enable
Written at 1 to set HDPEN.

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

14.10.157 RCC APB4H enable register (RCC_APB4HENSR)

Address offset: 0xA78

Reset value: 0x0000 0000

This register is used to enable the RCC APB4H in Run and Sleep modes (in Sleep mode, each bit of this register is AND-ed with the LPEN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

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

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

Bit 2 **DTSENS**: DTS enable
Written at 1 to set DTSEN.

Bit 1 **BSECENS**: BSEC enable
Written at 1 to set BSECEN.

Bit 0 **SYSCFGENS**: SYSCFG enable
Written at 1 to set SYSCFGEN.

14.10.158 RCC APB5 enable register (RCC_APB5ENSR)

Address offset: 0xA7C

Reset value: 0x0000 0000

This register is used to enable the RCC APB5 in Run and Sleep modes (in Sleep mode, each bit of this register is AND-ed with the LPEN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

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

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CSIENS**: CSI enable
Written at 1 to set CSIEN.

Bit 5 **VENCENS**: VENC enable
Written at 1 to set VENCEN.

Bit 4 **GFXTIMENS**: GFXTIM enable
Written at 1 to set GFXTIMEN.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **DCMIPPENS**: DCMIPP enable
Written at 1 to set DCMIPPEN.

Bit 1 **LTDCENS**: LTDC enable
Written at 1 to set LTDCEN.

Bit 0 Reserved, must be kept at reset value.

14.10.159 RCC bus sleep enable register (RCC_BUSLPENSR)

Address offset: 0xA84

Reset value: 0x0000 0000

This register is used to enable the RCC bus in Sleep mode (each bit of this register is AND-ed with the EN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

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

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

Bit 1 **ACLKNCLPENS**: ACLKNC enable
Written at 1 to set ACLKNCLPEN.

Bit 0 **ACLKNLPENS**: ACLKN enable
Written at 1 to set ACLKNLPEN.

14.10.160 RCC miscellaneous sleep enable register (RCC_MISCLPENSR)

Address offset: 0xA88

Reset value: 0x0000 0000

This register is used to enable the RCC miscellaneous in Sleep mode (each bit of this register is AND-ed with the EN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

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

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **PERLPENS**: PER enable
Written at 1 to set PERLPEN.

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **XSPIPHYCOMPLPENS**: XSPIPHYCOMP enable
Written at 1 to set XSPIPHYCOMPLPEN.

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **DBGLPENS**: DBG enable
Written at 1 to set DBGLPEN.

14.10.161 RCC memory sleep enable register (RCC_MEMLPENSR)

Address offset: 0xA8C

Reset value: 0x0000 0000

This register is used to enable the RCC memory in Sleep mode (each bit of this register is AND-ed with the EN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	BOOTROMLPENS	VENCRAMLPENS	CACHEAXIRAMLPENS	FLEXRAMLPENS	AXISRAM2LPENS	AXISRAM1LPENS	BKPSRAMLPENS	AHBSRAM2LPENS	AHBSRAM1LPENS	AXISRAM6LPENS	AXISRAM5LPENS	AXISRAM4LPENS	AXISRAM3LPENS
			w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **BOOTROMLPENS**: BootROM enable
Written at 1 to set BOOTROMLPEN.

Bit 11 **VENCRAMLPENS**: VENCRAM enable
Written at 1 to set VENCRAMLPEN.

Bit 10 **CACHEAXIRAMLPENS**: CACHEAXIRAM enable
Written at 1 to set CACHEAXIRAMLPEN.

Bit 9 **FLEXRAMLPENS**: FLEXRAM enable
Written at 1 to set FLEXRAMLPEN.

Bit 8 **AXISRAM2LPENS**: AXISRAM2 enable
Written at 1 to set AXISRAM2LPEN.

Bit 7 **AXISRAM1LPENS**: AXISRAM1 enable
Written at 1 to set AXISRAM1LPEN.

Bit 6 **BKPSRAMLPENS**: BKPSRAM enable
Written at 1 to set BKPSRAMLPEN.

Bit 5 **AHBSRAM2LPENS**: AHBSRAM2 enable
Written at 1 to set AHBSRAM2LPEN.

Bit 4 **AHBSRAM1LPENS**: AHBSRAM1 enable
Written at 1 to set AHBSRAM1LPEN.

Bit 3 **AXISRAM6LPENS**: AXISRAM6 enable
Written at 1 to set AXISRAM6LPEN.

Bit 2 **AXISRAM5LPENS**: AXISRAM5 enable
Written at 1 to set AXISRAM5LPEN.

Bit 1 **AXISRAM4LPENS**: AXISRAM4 enable
Written at 1 to set AXISRAM4LPEN.

Bit 0 **AXISRAM3LPENS**: AXISRAM3 enable
Written at 1 to set AXISRAM3LPEN.

14.10.162 RCC AHB1 sleep enable register (RCC_AHB1LPENSR)

Address offset: 0xA90

Reset value: 0x0000 0000

This register is used to enable the RCC AHB1 in Sleep mode (each bit of this register is AND-ed with the EN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC12 LPENS	GPDMA1 LPENS	Res.	Res.	Res.	Res.
										w	w				

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

Bit 5 **ADC12LPENS**: ADC12 enable
Written at 1 to set ADC12LPEN.

Bit 4 **GPDMA1LPENS**: GPDMA1 enable
Written at 1 to set GPDMA1LPEN.

Bits 3:0 Reserved, must be kept at reset value.

14.10.163 RCC AHB2 sleep enable register (RCC_AHB2LPENSR)

Address offset: 0xA94

Reset value: 0x0000 0000

This register is used to enable the RCC AHB2 in Sleep mode (each bit of this register is AND-ed with the EN bit).It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADF1L PENS	MDF1L PENS
														w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	RAMCF GLPENS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			w												

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

Bit 17 **ADF1LPENS**: ADF1 enable
Written at 1 to set ADF1LPEN.

Bit 16 **MDF1LPENS**: MDF1 enable
Written at 1 to set MDF1LPEN.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **RAMCFGLPENS**: RAMCFG enable
Written at 1 to set RAMCFGLPEN.

Bits 11:0 Reserved, must be kept at reset value.

14.10.164 RCC AHB3 sleep enable register (RCC_AHB3LPENSR)

Address offset: 0xA98

Reset value: 0x0000 0000

This register is used to enable the RCC AHB3 in Sleep mode (each bit of this register is AND-ed with the EN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RISAFLPENS	Res.	Res.	Res.	IACLPENS	RIFSCLPENS	PKALPENS	Res.	Res.	Res.	SAESLPENS	Res.	CRYPLPENS	HASHLPENS	RNGLPENS
	w				w	w	w				w		w	w	w

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **RISAFLPENS**: RISAF enable
Written at 1 to set RISAFLPEN.

Bits 13:11 Reserved, must be kept at reset value.

Bit 10 **IACLPENS**: IAC enable in Sleep mode
Written at 1 to set IACLPEN.

Bit 9 **RIFSCLPENS**: RIFSC enable
Written at 1 to set RIFSCLPEN.

Bit 8 **PKALPENS**: PKA enable
Written at 1 to set PKALPEN.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **SAESLPENS**: SAES enable
Written at 1 to set SAESLPEN.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CRYPLPENS**: CRYP enable
Written at 1 to set CRYPLPEN.

Bit 1 **HASHLPENS**: HASH enable
Written at 1 to set HASHLPEN.

Bit 0 **RNGLPENS**: RNG enable
Written at 1 to set RNGLPEN.

14.10.165 RCC AHB4 sleep enable register (RCC_AHB4LPENSR)

Address offset: 0xA9C

Reset value: 0x0000 0000

This register is used to enable the RCC AHB4 in Sleep mode (each bit of this register is AND-ed with the EN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCLPENS	PWRLPENS	Res.	GPIOQLPENS
												w	w		w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIOPLPENS	GPIOOLPENS	GPIONLPENS	Res.	Res.	Res.	Res.	Res.	GPIOHLPENS	GPIOGLPENS	GPIOFLPENS	GPIOELPENS	GPIODLPENS	GPIOCLPENS	GPIOBLPENS	GPIOALPENS
w	w	w						w	w	w	w	w	w	w	w

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **CRCLPENS**: CRC enable
Written at 1 to set CRCLPEN.

Bit 18 **PWRLPENS**: PWR enable
Written at 1 to set PWRLPEN.

Bit 17 Reserved, must be kept at reset value.

Bit 16 **GPIOQLPENS**: GPIO Q enable
Written at 1 to set GPIOQLPEN.

Bit 15 **GPIOPLPENS**: GPIO P enable
Written at 1 to set GPIOPLPEN.

Bit 14 **GPIOOLPENS**: GPIO O enable
Written at 1 to set GPIOOLPEN.

Bit 13 **GPIONLPENS**: GPIO N enable
Written at 1 to set GPIONLPEN.

Bits 12:8 Reserved, must be kept at reset value.

Bit 7 **GPIOHLPENS**: GPIO H enable
Written at 1 to set GPIOHLPEN.

Bit 6 **GPIOGLPENS**: GPIO G enable
Written at 1 to set GPIOGLPEN.

Bit 5 **GPIOFLPENS**: GPIO F enable
Written at 1 to set GPIOFLPEN.

Bit 4 **GPIOELPENS**: GPIO E enable
Written at 1 to set GPIOELPEN.

Bit 3 **GPIODLPENS**: GPIO D enable
Written at 1 to set GPIODLPEN.

Bit 2 **GPIOCLPENS**: GPIO C enable
Written at 1 to set GPIOCLPEN.

Bit 1 **GPIOBLPENS**: GPIO B enable
Written at 1 to set GPIOBLPEN.

Bit 0 **GPIOALPENS**: GPIO A enable
Written at 1 to set GPIOALPEN.

14.10.166 RCC AHB5 sleep enable register (RCC_AHB5LPENSR)

Address offset: 0xAA0

Reset value: 0x0000 0000

This register is used to enable the RCC AHB5 in Sleep mode (each bit of this register is AND-ed with the EN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NPULPENS	CACHEAXILPENS	OTG2LPENS	OTGPHY2LPENS	OTGPHY1LPENS	OTG1LPENS	ETH1LPENS	ETH1RXLPENS	ETH1TXLPENS	ETH1MACLPENS	Res.	GPU2DLPENS	GFXMMULPENS	MCE4LPENS	XSPI3LPENS	MCE3LPENS
w	w	w	w	w	w	w	w	w	w		w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCE2LPENS	MCE1LPENS	XSPIMLPENS	XSPI2LPENS	Res.	Res.	Res.	SDMMC1LPENS	SDMMC2LPENS	PSSILPENS	XSPI1LPENS	FMCLPENS	JPEGLPENS	Res.	DMA2DLPENS	HPDMA1LPENS
w	w	w	w				w	w	w	w	w	w		w	w

Bit 31 **NPULPENS**: NPU enable
Written at 1 to set NPULPEN.

Bit 30 **CACHEAXILPENS**: CACHEAXI enable
Written at 1 to set CACHEAXILPEN.

Bit 29 **OTG2LPENS**: OTG2 enable
Written at 1 to set OTG2LPEN.

Bit 28 **OTGPHY2LPENS**: OTGPHY2 enable
Written at 1 to set OTGPHY2LPEN.

Bit 27 **OTGPHY1LPENS**: OTGPHY1 enable
Written at 1 to set OTGPHY1LPEN.

Bit 26 **OTG1LPENS**: OTG1 enable
Written at 1 to set OTG1LPEN.

Bit 25 **ETH1LPENS**: ETH1 enable
Written at 1 to set ETH1LPEN.

Bit 24 **ETH1RXLPENS**: ETH1RX enable
Written at 1 to set ETH1RXLPEN.

Bit 23 **ETH1TXLPENS**: ETH1TX enable
Written at 1 to set ETH1TXLPEN.

Bit 22 **ETH1MACLPENS**: ETH1MAC enable
Written at 1 to set ETH1MACLPEN.

Bit 21 Reserved, must be kept at reset value.

Bit 20 **GPU2DLPENS**: GPU2D enable
Written at 1 to set GPU2DLPEN.

Bit 19 **GFXMMULPENS**: GFXMMU enable
Written at 1 to set GFXMMULPEN.

Bit 18 **MCE4LPENS**: MCE4 enable
Written at 1 to set MCE4LPEN.

Bit 17 **XSPI3LPENS**: XSPI3 enable
Written at 1 to set XSPI3LPEN.

Bit 16 **MCE3LPENS**: MCE3 enable
Written at 1 to set MCE3LPEN.

Bit 15 **MCE2LPENS**: MCE2 enable
Written at 1 to set MCE2LPEN.

Bit 14 **MCE1LPENS**: MCE1 enable
Written at 1 to set MCE1LPEN.

Bit 13 **XSPIMLPENS**: XSPIM enable
Written at 1 to set XSPIMLPEN.

Bit 12 **XSPI2LPENS**: XSPI2 enable
Written at 1 to set XSPI2LPEN.

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **SDMMC1LPENS**: SDMMC1 enable
Written at 1 to set SDMMC1LPEN.

Bit 7 **SDMMC2LPENS**: SDMMC2 enable
Written at 1 to set SDMMC2LPEN.

Bit 6 **PSSILPENS**: PSSI enable
Written at 1 to set PSSILPEN.

Bit 5 **XSPI1LPENS**: XSPI1 enable
Written at 1 to set XSPI1LPEN.

Bit 4 **FMCLPENS**: FMC enable
Written at 1 to set FMCLPEN.

Bit 3 **JPEGLPENS**: JPEG enable
Written at 1 to set JPEGLPEN.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **DMA2DLPENS**: DMA2D enable
Written at 1 to set DMA2DLPEN.

Bit 0 **HPDMA1LPENS**: HPDMA1 enable
Written at 1 to set HPDMA1LPEN.

14.10.167 RCC APB1L sleep enable register (RCC_APB1LLPENSR)

Address offset: 0xAA4

Reset value: 0x0000 0000

This register is used to enable the RCC APB1L in Sleep mode (each bit of this register is AND-ed with the EN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UART8 LPENS	UART7 LPENS	Res.	Res.	Res.	Res.	I3C2LP ENS	I3C1LP ENS	I2C3LP ENS	I2C2LP ENS	I2C1LP ENS	UART5 LPENS	UART4 LPENS	USART 3LPEN S	USART 2LPEN S	SPDIF RX1LP ENS
w	w					w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3LP ENS	SPI2LP ENS	TIM11L PENS	TIM10L PENS	WWDG LPENS	Res.	LPTIM1 LPENS	TIM14L PENS	TIM13L PENS	TIM12L PENS	TIM7LP ENS	TIM6LP ENS	TIM5LP ENS	TIM4LP ENS	TIM3LP ENS	TIM2LP ENS
w	w	w	w	w		w	w	w	w	w	w	w	w	w	w

Bit 31 **UART8LPENS**: UART8 enable
Written at 1 to set UART8LPEN.

Bit 30 **UART7LPENS**: UART7 enable
Written at 1 to set UART7LPEN.

Bits 29:26 Reserved, must be kept at reset value.

Bit 25 **I3C2LPENS**: I3C2 enable
Written at 1 to set I3C2LPEN.

Bit 24 **I3C1LPENS**: I3C1 enable
Written at 1 to set I3C1LPEN.

Bit 23 **I2C3LPENS**: I2C3 enable
Written at 1 to set I2C3LPEN.

Bit 22 **I2C2LPENS**: I2C2 enable
Written at 1 to set I2C2LPEN.

Bit 21 **I2C1LPENS**: I2C1 enable
Written at 1 to set I2C1LPEN.

Bit 20 **UART5LPENS**: UART5 enable
Written at 1 to set UART5LPEN.

Bit 19 **UART4LPENS**: UART4 enable
Written at 1 to set UART4LPEN.

Bit 18 **USART3LPENS**: USART3 enable
Written at 1 to set USART3LPEN.

Bit 17 **USART2LPENS**: USART2 enable
Written at 1 to set USART2LPEN.

Bit 16 **SPDIFRX1LPENS**: SPDIFRX1 enable
Written at 1 to set SPDIFRX1LPEN.

Bit 15 **SPI3LPENS**: SPI3 enable
Written at 1 to set SPI3LPEN.

Bit 14 **SPI2LPENS**: SPI2 enable
Written at 1 to set SPI2LPEN.

Bit 13 **TIM11LPENS**: TIM11 enable
Written at 1 to set TIM11LPEN.

Bit 12 **TIM10LPENS**: TIM10 enable
Written at 1 to set TIM10LPEN.

Bit 11 **WWDGLPENS**: WWDG enable
Written at 1 to set WWDGLPEN.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **LPTIM1LPENS**: LPTIM1 enable
Written at 1 to set LPTIM1LPEN.

Bit 8 **TIM14LPENS**: TIM14 enable
Written at 1 to set TIM14LPEN.

Bit 7 **TIM13LPENS**: TIM13 enable
Written at 1 to set TIM13LPEN.

Bit 6 **TIM12LPENS**: TIM12 enable
Written at 1 to set TIM12LPEN.

Bit 5 **TIM7LPENS**: TIM7 enable
Written at 1 to set TIM7LPEN.

Bit 4 **TIM6LPENS**: TIM6 enable
Written at 1 to set TIM6LPEN.

Bit 3 **TIM5LPENS**: TIM5 enable
Written at 1 to set TIM5LPEN.

Bit 2 **TIM4LPENS**: TIM4 enable
Written at 1 to set TIM4LPEN.

Bit 1 **TIM3LPENS**: TIM3 enable
Written at 1 to set TIM3LPEN.

Bit 0 **TIM2LPENS**: TIM2 enable
Written at 1 to set TIM2LPEN.

14.10.168 RCC APB1H sleep enable register (RCC_APB1HLPENSR)

Address offset: 0xAA8

Reset value: 0x0000 0000

This register is used to enable the RCC APB1H in Sleep mode (each bit of this register is AND-ed with the EN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD1 LPENS	Res.	Res.
													w		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCAN LPENS	Res.	Res.	MDIOS LPENS	Res.	Res.	Res.	Res.	Res.
							w			w					

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **UCPD1LPENS**: UCPD1 enable
Written at 1 to set UCPD1LPEN.

Bits 17:9 Reserved, must be kept at reset value.

Bit 8 **FDCANLPENS**: FDCAN enable
Written at 1 to set FDCANLPEN.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **MDIOSLPENS**: MDIOS enable
Written at 1 to set MDIOSLPEN.

Bits 4:0 Reserved, must be kept at reset value.

14.10.169 RCC APB2 sleep enable register (RCC_APB2LPENSR)

Address offset: 0xAAC

Reset value: 0x0000 0000

This register is used to enable the RCC APB2 in Sleep mode (each bit of this register is AND-ed with the EN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI2LPENS	SAI1LPENS	SPI5LPENS	TIM9LPENS	TIM17LPENS	TIM16LPENS	TIM15LPENS
									w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIM18LPENS	Res.	SPI4LPENS	SPI1LPENS	Res.	Res.	Res.	Res.	USART10LPENS	UART9LPENS	USART6LPENS	USART1LPENS	Res.	Res.	TIM8LPENS	TIM1LPENS
w		w	w					w	w	w	w			w	w

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **SAI2LPENS**: SAI2 enable
Written at 1 to set SAI2LPEN.

Bit 21 **SAI1LPENS**: SAI1 enable
Written at 1 to set SAI1LPEN.

Bit 20 **SPI5LPENS**: SPI5 enable
Written at 1 to set SPI5LPEN.

Bit 19 **TIM9LPENS**: TIM9 enable
Written at 1 to set TIM9LPEN.

Bit 18 **TIM17LPENS**: TIM17 enable
Written at 1 to set TIM17LPEN.

Bit 17 **TIM16LPENS**: TIM16 enable
Written at 1 to set TIM16LPEN.

Bit 16 **TIM15LPENS**: TIM15 enable
Written at 1 to set TIM15LPEN.

Bit 15 **TIM18LPENS**: TIM18 enable
Written at 1 to set TIM18LPEN.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **SPI4LPENS**: SPI4 enable
Written at 1 to set SPI4LPEN.

Bit 12 **SPI1LPENS**: SPI1 enable
Written at 1 to set SPI1LPEN.

Bits 11:8 Reserved, must be kept at reset value.

Bit 7 **USART10LPENS**: USART10 enable
Written at 1 to set USART10LPEN.

Bit 6 **UART9LPENS**: UART9 enable
Written at 1 to set UART9LPEN.

Bit 5 **USART6LPENS**: USART6 enable
Written at 1 to set USART6LPEN.

Bit 4 **USART1LPENS**: USART1 enable
Written at 1 to set USART1LPEN.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **TIM8LPENS**: TIM8 enable
Written at 1 to set TIM8LPEN.

Bit 0 **TIM1LPENS**: TIM1 enable
Written at 1 to set TIM1LPEN.

14.10.170 RCC APB3 sleep enable register (RCC_APB3LPENSR)

Address offset: 0xAB0

Reset value: 0x0000 0000

This register is used to enable the RCC APB3 in Sleep mode (each bit of this register is AND-ed with the EN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

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

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

Bit 2 **DFTLPENS**: DFT enable
Written at 1 to set DFTLPEN.

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

14.10.171 RCC APB4L sleep enable register (RCC_APB4LLPENSR)

Address offset: 0xAB4

Reset value: 0x0000 0000

This register is used to enable the RCC APB4L in Sleep mode (each bit of this register is AND-ed with the EN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCAPBLPENS	RTCLPENS
														w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VREFBUFLPENS	Res.	Res.	LPTIM5LPENS	LPTIM4LPENS	LPTIM3LPENS	LPTIM2LPENS	Res.	I2C4LPENS	Res.	SPI6LPENS	Res.	LPUART1LPENS	HDPLPENS	Res.	Res.
w			w	w	w	w		w		w		w	w		

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

Bit 17 **RTCAPBLPENS**: RTCAPB enable
Written at 1 to set RTCAPBLPEN.

Bit 16 **RTCLPENS**: RTC enable
Written at 1 to set RTCLPEN.

Bit 15 **VREFBUFLPENS**: VREFBUF enable
Written at 1 to set VREFBUFLPEN.

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **LPTIM5LPENS**: LPTIM5 enable
Written at 1 to set LPTIM5LPEN.

Bit 11 **LPTIM4LPENS**: LPTIM4 enable
Written at 1 to set LPTIM4LPEN.

Bit 10 **LPTIM3LPENS**: LPTIM3 enable
Written at 1 to set LPTIM3LPEN.

Bit 9 **LPTIM2LPENS**: LPTIM2 enable
Written at 1 to set LPTIM2LPEN.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **I2C4LPENS**: I2C4 enable
Written at 1 to set I2C4LPEN.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **SPI6LPENS**: SPI6 enable
Written at 1 to set SPI6LPEN.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **LPUART1LPENS**: LPUART1 enable
Written at 1 to set LPUART1LPEN.

Bit 2 **HDPLPENS**: HDP enable
Written at 1 to set HDPLPEN.

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

14.10.172 RCC APB4H sleep enable register (RCC_APB4HLPENSR)

Address offset: 0xAB8

Reset value: 0x0000 0000

This register is used to enable the RCC APB4H in Sleep mode (each bit of this register is AND-ed with the EN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

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

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

Bit 2 **DTSLPENS**: DTS enable
Written at 1 to set DTSLPEN.

Bit 1 **BSECLPENS**: BSEC enable
Written at 1 to set BSECLPEN.

Bit 0 **SYSCFGLPENS**: SYSCFG enable
Written at 1 to set SYSCFGLPEN.

14.10.173 RCC APB5 sleep enable register (RCC_APB5LPENSR)

Address offset: 0xABC

Reset value: 0x0000 0000

This register is used to enable the RCC APB5 in Sleep mode (each bit of this register is AND-ed with the EN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

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

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CSILPENS**: CSI enable
Written at 1 to set CSILPEN.

Bit 5 **VENCLPENS**: VENC enable
Written at 1 to set VENCLPEN.

Bit 4 **GFXTIMLPENS**: GFXTIM enable
Written at 1 to set GFXTIMLPEN.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **DCMIPPLPENS**: DCMIPP enable
Written at 1 to set DCMIPPLPEN.

Bit 1 **LTDCLPENS**: LTDC enable
Written at 1 to set LTDCLPEN.

Bit 0 Reserved, must be kept at reset value.

14.10.174 RCC oscillator privilege configuration set register 0 (RCC_PRIVCFGSR0)

Address offset: 0xF84

Reset value: 0x0000 0000

This register is used to control the privileged access rights to the configuration register of the oscillators. It is reset by sys_rstn, and is in the V_{CORE} voltage domain. Each xxPRIV bit defines the privileged protection for the configuration registers of the oscillator: a write access is denied if the access is unprivileged while the respective bit is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSE PRIVS	HSIP RIVS	MSI PRIVS	LSE PRIVS	LSI PRIVS
											w	w	w	w	w

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

Bit 4 **HSEPRIVS**: Privileged protection of HSE configuration bits (enable, ready, divider)
Written at 1 to set HSEPRIV by secure privileged software only. It can be read by any software.

Bit 3 **HSIPRIVS**: Privileged protection of HSI configuration bits (enable, ready, divider)
Written at 1 to set HSIPRIV by secure privileged software only. It can be read by any software.

Bit 2 **MSIPRIVS**: Privileged protection of MSI configuration bits (enable, ready, divider)
Written at 1 to set MSIPRIV by secure privileged software only. It can be read by any software.

Bit 1 **LSEPRIVS**: Privileged protection of LSE configuration bits (enable, ready, divider)
Written at 1 to set LSEPRIV by secure privileged software only. It can be read by any software.

Bit 0 **LSIPRIVS**: Privileged protection of the LSI configuration bits (enable, ready, divider)
Written at 1 to set LSIPRIV by secure privileged software only. It can be read by any software.

14.10.175 RCC oscillator public configuration set register 0 (RCC_PUBCFGSR0)

Address offset: 0xF8C

Reset value: 0x0000 0000

This register is used to control the public access rights to the configuration register of the oscillators. It is reset by sys_rstn, and is in the V_{CORE} voltage domain. Each xxPUB bit defines the public protection for the configuration registers of the oscillator: a write access is denied if the access is non-public while the respective bit is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSE PUBS	HSI PUBS	MSI PUBS	LSE PUBS	LSI PUBS
											w	w	w	w	w

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

Bit 4 **HSEPUBS**: Public protection of he HSE configuration bits (enable, ready, divider)
Written at 1 to set HSEPUB by secure privileged software only. It can be read by any software.

Bit 3 **HSIPUBS**: Public protection of HSI configuration bits (enable, ready, divider)
Written at 1 to set HSIPUB by secure privileged software only. It can be read by any software.

Bit 2 **MSIPUBS**: Public protection of MSI configuration bits (enable, ready, divider)
Written at 1 to set MSIPUB by secure privileged software only. It can be read by any software.

Bit 1 **LSEPUBS**: Public protection of LSE configuration bits (enable, ready, divider)
Written at 1 to set LSEPUB by secure privileged software only. It can be read by any software.

Bit 0 **LSIPUBS**: Public protection of LSI configuration bits (enable, ready, divider)
Written at 1 to set LSIPUB by secure privileged software only. It can be read by any software.

14.10.176 RCC PLL privilege configuration set register 1 (RCC_PRIVCFGSR1)

Address offset: 0xF94

Reset value: 0x0000 0000

This register is used to control the privileged access rights to the configuration register of the PLLs. It is reset by sys_rst, and is in the V_{CORE} voltage domain. Each xxPRIV bit defines the privileged protection for the configuration registers of the PLL: a write access is denied if the access is unprivilege while the respective bit is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL4PRIVS	PLL3PRIVS	PLL2PRIVS	PLL1PRIVS
												w	w	w	w

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

Bit 3 **PLL4PRIVS**: Privileged protection of PLL4 configuration bits (enable, ready, divider)
Written at 1 to set PLL4PRIV by secure privileged software only. It can be read by any software.

Bit 2 **PLL3PRIVS**: Privileged protection of PLL3 configuration bits (enable, ready, divider)
Written at 1 to set PLL3PRIV by secure privileged software only. It can be read by any software.

Bit 1 **PLL2PRIVS**: Privileged protection of PLL2 configuration bits (enable, ready, divider)
Written at 1 to set PLL2PRIV by secure privileged software only. It can be read by any software.

Bit 0 **PLL1PRIVS**: Privileged protection of PLL1 configuration bits (enable, ready, divider)
Written at 1 to set PLL1PRIV by secure privileged software only. It can be read by any software.

14.10.177 RCC PLL public configuration set register 1 (RCC_PUBCFGSR1)

Address offset: 0xF9C

Reset value: 0x0000 0000

This register is used to control the public access rights to the configuration register of the PLLs. It is reset by sys_rstn, and is in the V_{CORE} voltage domain. Each xxPUB bit defines the public protection for the configuration registers of the PLL: a write access is denied if the access is non-public while the respective bit is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL4PUBS	PLL3PUBS	PLL2PUBS	PLL1PUBS
												w	w	w	w

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

Bit 3 **PLL4PUBS**: Public protection of PLL4 configuration bits (enable, ready, divider)
Written at 1 to set PLL4PUB by secure privileged software only. It can be read by any software.

Bit 2 **PLL3PUBS**: Public protection of PLL3 configuration bits (enable, ready, divider)
Written at 1 to set PLL3PUB by secure privileged software only. It can be read by any software.

Bit 1 **PLL2PUBS**: Public protection of PLL2 configuration bits (enable, ready, divider)
Written at 1 to set PLL2PUB by secure privileged software only. It can be read by any software.

Bit 0 **PLL1PUBS**: Public protection of PLL1 configuration bits (enable, ready, divider)
Written at 1 to set PLL1PUB by secure privileged software only. It can be read by any software.

14.10.178 RCC divider privilege configuration set register 2 (RCC_PRIVCFGSR2)

Address offset: 0xFA4

Reset value: 0x0000 0000

This register is used to control the privileged access rights to the configuration register of the dividers. It is reset by sys_rstn, and is in the V_{CORE} voltage domain. Each xxPRIV bit defines the privileged protection for the configuration registers of the divider: a write access is denied if the access is unprivileged while the respective bit is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC20PRIVS	IC19PRIVS	IC18PRIVS	IC17PRIVS
												w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC16PRIVS	IC15PRIVS	IC14PRIVS	IC13PRIVS	IC12PRIVS	IC11PRIVS	IC10PRIVS	IC9PRIVS	IC8PRIVS	IC7PRIVS	IC6PRIVS	IC5PRIVS	IC4PRIVS	IC3PRIVS	IC2PRIVS	IC1PRIVS
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **IC20PRIVS**: Privileged protection of IC20 configuration bits (enable, ready, divider)
Written at 1 to set IC20PRIV by secure privileged software only. It can be read by any software.

Bit 18 **IC19PRIVS**: Privileged protection of IC19 configuration bits (enable, ready, divider)
Written at 1 to set IC19PRIV by secure privileged software only. It can be read by any software.

Bit 17 **IC18PRIVS**: Privileged protection of IC18 configuration bits (enable, ready, divider)
Written at 1 to set IC18PRIV by secure privileged software only. It can be read by any software.

Bit 16 **IC17PRIVS**: Privileged protection of IC17 configuration bits (enable, ready, divider)
Written at 1 to set IC17PRIV by secure privileged software only. It can be read by any software.

Bit 15 **IC16PRIVS**: Privileged protection of IC16 configuration bits (enable, ready, divider)
Written at 1 to set IC16PRIV by secure privileged software only. It can rbe ead by any software.

Bit 14 **IC15PRIVS**: Privileged protection of IC15 configuration bits (enable, ready, divider)
Written at 1 to set IC15PRIV by secure privileged software only. It can be read by any software.

Bit 13 **IC14PRIVS**: Privileged protection of IC14 configuration bits (enable, ready, divider)
Written at 1 to set IC14PRIV by secure privileged software only. It can be read by any software.

Bit 12 **IC13PRIVS**: Privileged protection of IC13 configuration bits (enable, ready, divider)
Written at 1 to set IC13PRIV by secure privileged software only. It can be read by any software.

Bit 11 **IC12PRIVS**: Privileged protection of IC12 configuration bits (enable, ready, divider)
Written at 1 to set IC12PRIV by secure privileged software only. It can be read by any software.

Bit 10 **IC11PRIVS**: Privileged protection of IC11 configuration bits (enable, ready, divider)
Written at 1 to set IC11PRIV by secure privileged software only. It can be read by any software.

Bit 9 **IC10PRIVS**: Privileged protection of IC10 configuration bits (enable, ready, divider)
Written at 1 to set IC10PRIV by secure privileged software only. It can be read by any software.

Bit 8 **IC9PRIVS**: Privileged protection of IC9 configuration bits (enable, ready, divider)
Written at 1 to set IC9PRIV by secure privileged software only. It can be read by any software.

Bit 7 **IC8PRIVS**: Privileged protection of IC8 configuration bits (enable, ready, divider)
Written at 1 to set IC8PRIV by secure privileged software only. It can be read by any software.

Bit 6 **IC7PRIVS**: Privileged protection of IC7 configuration bits (enable, ready, divider)
Written at 1 to set IC7PRIV by secure privileged software only. It can be read by any software.

Bit 5 **IC6PRIVS**: Privileged protection of IC6 configuration bits (enable, ready, divider)
Written at 1 to set IC6PRIV by secure privileged software only. It can be read by any software.

Bit 4 **IC5PRIVS**: Privileged protection of IC5 configuration bits (enable, ready, divider)
Written at 1 to set IC5PRIV by secure privileged software only. It can be read by any software.

Bit 3 **IC4PRIVS**: Privileged protection of IC4 configuration bits (enable, ready, divider)
Written at 1 to set IC4PRIV by secure privileged software only. It can be read by any software.

Bit 2 **IC3PRIVS**: Privileged protection of IC3 configuration bits (enable, ready, divider)
Written at 1 to set IC3PRIV by secure privileged software only. It can be read by any software.

Bit 1 **IC2PRIVS**: Privileged protection of IC2 configuration bits (enable, ready, divider)
Written at 1 to set IC2PRIV by secure privileged software only. It can be read by any software.

Bit 0 **IC1PRIVS**: Privileged protection of IC1 configuration bits (enable, ready, divider)
Written at 1 to set IC1PRIV by secure privileged software only. It can be read by any software.

14.10.179 RCC divider public configuration set register 2 (RCC_PUBCFGSR2)

Address offset: 0xFAC

Reset value: 0x0000 0000

This register is used to control the public access rights to the configuration register of the dividers. It is reset by sys_rstn, and is in the V_{CORE} voltage domain. Each xxPUB bit defines

the public protection for the configuration registers of the divider: a write access is denied if the access is non-public while the respective bit is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC20PUBS	IC19PUBS	IC18PUBS	IC17PUBS
												w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC16PUBS	IC15PUBS	IC14PUBS	IC13PUBS	IC12PUBS	IC11PUBS	IC10PUBS	IC9PUBS	IC8PUBS	IC7PUBS	IC6PUBS	IC5PUBS	IC4PUBS	IC3PUBS	IC2PUBS	IC1PUBS
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **IC20PUBS**: Public protection of IC20 configuration bits (enable, ready, divider)
Written at 1 to set IC20PUB by secure privileged software only. It can be read by any software.

Bit 18 **IC19PUBS**: Public protection of IC19 configuration bits (enable, ready, divider)
Written at 1 to set IC19PUB by secure privileged software only. It can be read by any software.

Bit 17 **IC18PUBS**: Public protection of IC18 configuration bits (enable, ready, divider)
Written at 1 to set IC18PUB by secure privileged software only. It can be read by any software.

Bit 16 **IC17PUBS**: Public protection of IC17 configuration bits (enable, ready, divider)
Written at 1 to set IC17PUB by secure privileged software only. It can be read by any software.

Bit 15 **IC16PUBS**: Public protection of th IC16 configuration bits (enable, ready, divider
Written at 1 to set IC16PUB by secure privileged software only. It can be read by any software.

Bit 14 **IC15PUBS**: Public protection of IC15 configuration bits (enable, ready, divider)
Written at 1 to set IC15PUB by secure privileged software only. It can be read by any software.

Bit 13 **IC14PUBS**: Public protection of IC14 configuration bits (enable, ready, divider)
Written at 1 to set IC14PUB by secure privileged software only. It can be read by any software.

Bit 12 **IC13PUBS**: Public protection of IC13 configuration bits (enable, ready, divider)
Written at 1 to set IC13PUB by secure privileged software only. It can be read by any software.

Bit 11 **IC12PUBS**: Public protection of IC12 configuration bits (enable, ready, divider)
Written at 1 to set IC12PUB by secure privileged software only. It can be read by any software.

Bit 10 **IC11PUBS**: Public protection of IC11 configuration bits (enable, ready, divider)
Written at 1 to set IC11PUB by secure privileged software only. It can be read by any software.

Bit 9 **IC10PUBS**: Public protection of IC10 configuration bits (enable, ready, divider)
Written at 1 to set IC10PUB by secure privileged software only. It can be read by any software.

Bit 8 **IC9PUBS**: Public protection of IC9 configuration bits (enable, ready, divider)
Written at 1 to set IC9PUB by secure privileged software only. It can be read by any software.

Bit 7 **IC8PUBS**: Public protection of IC8 configuration bits (enable, ready, divider)
Written at 1 to set IC8PUB by secure privileged software only. It can be read by any software.

Bit 6 **IC7PUBS**: Public protection of IC7 configuration bits (enable, ready, divider)
Written at 1 to set IC7PUB by secure privileged software only. It can be read by any software.

Bit 5 **IC6PUBS**: Public protection of IC6 configuration bits (enable, ready, divider)
Written at 1 to set IC6PUB by secure privileged software only. It can be read by any software.

Bit 4 **IC5PUBS**: Public protection of IC5 configuration bits (enable, ready, divider)
Written at 1 to set IC5PUB by secure privileged software only. It can be read by any software.

Bit 3 **IC4PUBS**: Public protection of IC4 configuration bits (enable, ready, divider)
Written at 1 to set IC4PUB by secure privileged software only. It can be read by any software.

Bit 2 **IC3PUBS**: Public protection of IC3 configuration bits (enable, ready, divider)
Written at 1 to set IC3PUB by secure privileged software only. It can be read by any software.

Bit 1 **IC2PUBS**: Public protection of IC2 configuration bits (enable, ready, divider)
Written at 1 to set IC2PUB by secure privileged software only. It can be read by any software.

Bit 0 **IC1PUBS**: Public protection of IC1 configuration bits (enable, ready, divider)
Written at 1 to set IC1PUB by secure privileged software only. It can be read by any software.

14.10.180 RCC system privilege configuration set register 3 (RCC_PRIVCFGSR3)

Address offset: 0xFB4

Reset value: 0x0000 0000

This register is used to control the privileged access rights to the configuration register of the system. It is reset by sys_rstn, and is in the V_{CORE} voltage domain. Each xxPRIV bit defines the privilege protection for the configuration registers of the system: a write access is denied if the access is unprivileged while the respective bit is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFTPRIVS	RSTPRIVS	INTPRIVS	PERPRIVS	BUSPRIVS	SYSPRIVS	MODPRIVS
									w	w	w	w	w	w	w

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **DFTPRIVS**: Privileged protection of DFT configuration bits (enable, ready, divider)
Written at 1 to set DFTPRIV by secure privileged software only. It can be read by any software.

Bit 5 **RSTPRIVS**: Privileged protection of RST configuration bits (enable, ready, divider)
Written at 1 to set RSTPRIV by secure privileged software only. It can be read by any software.

Bit 4 **INTPRIVS**: Privileged protection of INT configuration bits (enable, ready, divider)
Written at 1 to set INTPRIV by secure privileged software only. It can be read by any software.

Bit 3 **PERPRIVS**: Privileged protection of PER configuration bits (enable, ready, divider)
Written at 1 to set PERPRIV by secure privileged software only. It can be read by any software.

Bit 2 **BUSPRIVS**: Privileged protection of BUS configuration bits (enable, ready, divider)
Written at 1 to set BUSPRIV by secure privileged software only. It can be read by any software.

Bit 1 **SYSPRIVS**: Privileged protection of SYS configuration bits (enable, ready, divider)
Written at 1 to set SYSPRIV by secure privileged software only. It can be read by any software.

Bit 0 **MODPRIVS**: Privileged protection of MOD configuration bits (enable, ready, divider)
Written at 1 to set MODPRIV by secure privileged software only. It can be read by any software.

14.10.181 RCC system public configuration set register 3 (RCC_PUBCFGSR3)

Address offset: 0xFBC

Reset value: 0x0000 0000

This register is used to control the public access rights to the configuration register of the system. It is reset by sys_rstn, and is in the V_{CORE} voltage domain. Each xxPUB bit defines the public protection for the configuration registers of the system: a write access is denied if the access is non-public while the respective bit is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTPUBS	INTPUBS	PERPUBS	BUSPUBS	SYSPUBS	MODPUBS
										w	w	w	w	w	w

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

Bit 5 **RSTPUBS**: Public protection of RST configuration bits (enable, ready, divider)
Written at 1 to set RSTPUB by secure privileged software only. It can be read by any software.

Bit 4 **INTPUBS**: Public protection of INT configuration bits (enable, ready, divider)
Written at 1 to set INTPUB by secure privileged software only. It can be read by any software.

Bit 3 **PERPUBS**: Public protection of PER configuration bits (enable, ready, divider)
Written at 1 to set PERPUB by secure privileged software only. It can be read by any software.

Bit 2 **BUSPUBS**: Public protection of BUS configuration bits (enable, ready, divider)
Written at 1 to set BUSPUB by secure privileged software only. It can be read by any software.

Bit 1 **SYSPUBS**: Public protection of SYS configuration bits (enable, ready, divider)
Written at 1 to set SYSPUB by secure privileged software only. It can be read by any software.

Bit 0 **MODPUBS**: Public protection of MOD configuration bits (enable, ready, divider)
Written at 1 to set MODPUB by secure privileged software only. It can be read by any software.

14.10.182 RCC privilege configuration set register 4 (RCC_PRIVCFGSR4)

Address offset: 0xFC4

Reset value: 0x0000 0000

This register is used to control the privileged access rights to the configuration register of the buses. It is reset by sys_rstn, and is in the V_{CORE} voltage domain. Each xxPRIV bit defines the privileged protection for the configuration registers of each bus: a write access is denied if the access is unprivileged while the respective bit is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	NOCPRIVS	APB5PRIVS	APB4PRIVS	APB3PRIVS	APB2PRIVS	APB1PRIVS	AHB5PRIVS	AHB4PRIVS	AHB3PRIVS	AHB2PRIVS	AHB1PRIVS	AHBMPRIVS	ACLKNCPRIVS	ACLKNPRIVS
		w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **NOCPRIVS**: Privileged protection of NOC configuration bits (enable, ready, divider)
Written at 1 to set NOCPRIV by secure privileged software only. It can be read by any software.

Bit 12 **APB5PRIVS**: Privileged protection of APB5 configuration bits (enable, ready, divider)
Written at 1 to set APB5PRIV by secure privileged software only. It can be read by any software.

Bit 11 **APB4PRIVS**: Privileged protection of APB4 configuration bits (enable, ready, divider)
Written at 1 to set APB4PRIV by secure privileged software only. It can be read by any software.

Bit 10 **APB3PRIVS**: Privileged protection of APB3 configuration bits (enable, ready, divider)
Written at 1 to set APB3PRIV by secure privileged software only. It can be read by any software.

Bit 9 **APB2PRIVS**: Privileged protection of APB2 configuration bits (enable, ready, divider)
Written at 1 to set APB2PRIV by secure privileged software only. It can be read by any software.

Bit 8 **APB1PRIVS**: Privileged protection of APB1 configuration bits (enable, ready, divider)
Written at 1 to set APB1PRIV by secure privileged software only. It can be read by any software.

Bit 7 **AHB5PRIVS**: Privileged protection of AHB5 configuration bits (enable, ready, divider)
Written at 1 to set AHB5PRIV by secure privileged software only. It can be read by any software.

Bit 6 **AHB4PRIVS**: Privileged protection of AHB4 configuration bits (enable, ready, divider)
Written at 1 to set AHB4PRIV by secure privileged software only. It can be read by any software.

Bit 5 **AHB3PRIVS**: Privileged protection of AHB3 configuration bits (enable, ready, divider)
Written at 1 to set AHB3PRIV by secure privileged software only. It can be read by any software.

Bit 4 **AHB2PRIVS**: Privileged protection of AHB2 configuration bits (enable, ready, divider)
Written at 1 to set AHB2PRIV by secure privileged software only. It can be read by any software.

Bit 3 **AHB1PRIVS**: Privileged protection of AHB1 configuration bits (enable, ready, divider)
Written at 1 to set AHB1PRIV by secure privileged software only. It can be read by any software.

Bit 2 **AHBMPRIVS**: Privileged protection of AHBM configuration bits (enable, ready, divider)
Written at 1 to set AHBMPRIV by secure privileged software only. It can be read by any software.

Bit 1 **ACLKNCPRIVS**: Privileged protection of th ACLKNC configuration bits (enable, ready, divider)
Written at 1 to set ACLKNCPRIV by secure privileged software only. It can be read by any software.

Bit 0 **ACLKNPRIVS**: Privileged protection of ACLKN configuration bits (enable, ready, divider)
Written at 1 to set ACLKNPRIV by secure privileged software only. It can be read by any software.

14.10.183 RCC public configuration set register 4 (RCC_PUBCFGSR4)

Address offset: 0xFCC

Reset value: 0x0000 0000

This register is used to control the public access rights to the configuration register of the buses. It is reset by sys_rstn, and is in the V_{CORE} voltage domain. Each xxPUB bit defines the public protection for the configuration registers of each bus: a write access is denied if the access is non-public while the respective bit is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	NOCPUBS	APB5PUBS	APB4PUBS	APB3PUBS	APB2PUBS	APB1PUBS	AHB5PUBS	AHB4PUBS	AHB3PUBS	AHB2PUBS	AHB1PUBS	AHBMPUBS	ACLKNCPUBS	ACLKNPUBS
		w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **NOCPUBS**: Public protection of NOC configuration bits (enable, ready, divider)
Written at 1 to set NOCPUB by secure privileged software only. It can be read by any software.

Bit 12 **APB5PUBS**: Public protection of APB5 configuration bits (enable, ready, divider)
Written at 1 to set APB5PUB by secure privileged software only. It can be read by any software.

Bit 11 **APB4PUBS**: Public protection of APB4 configuration bits (enable, ready, divider)
Written at 1 to set APB4PUB by secure privileged software only. It can be read by any software.

Bit 10 **APB3PUBS**: Public protection of APB3 configuration bits (enable, ready, divider)
Written at 1 to set APB3PUB by secure privileged software only. It can be read by any software.

Bit 9 **APB2PUBS**: Public protection of APB2 configuration bits (enable, ready, divider)
Written at 1 to set APB2PUB by secure privileged software only. It can be read by any software.

Bit 8 **APB1PUBS**: Public protection of APB1 configuration bits (enable, ready, divider)
Written at 1 to set APB1PUB by secure privileged software only. It can be read by any software.

Bit 7 **AHB5PUBS**: Public protection of AHB5 configuration bits (enable, ready, divider)
Written at 1 to set AHB5PUB by secure privileged software only. It can be read by any software.

Bit 6 **AHB4PUBS**: Public protection of AHB4 configuration bits (enable, ready, divider)
Written at 1 to set AHB4PUB by secure privileged software only. It can be read by any software.

Bit 5 **AHB3PUBS**: Public protection of AHB3 configuration bits (enable, ready, divider)
Written at 1 to set AHB3PUB by secure privileged software only. It can be read by any software.

Bit 4 **AHB2PUBS**: Public protection of AHB2 configuration bits (enable, ready, divider)
Written at 1 to set AHB2PUB by secure privileged software only. It can be read by any software.

Bit 3 **AHB1PUBS**: Public protection of AHB1 configuration bits (enable, ready, divider)
Written at 1 to set AHB1PUB by secure privileged software only. It can be read by any software.

Bit 2 **AHBMPUBS**: Public protection of AHBM configuration bits (enable, ready, divider)
Written at 1 to set AHBMPUB by secure privileged software only. It can be read by any software.

Bit 1 **ACLKNCPUBS**: Public protection of ACLKNC configuration bits (enable, ready, divider)
Written at 1 to set ACLKNCPUB by secure privileged software only. It can be read by any software.

Bit 0 **ACLKNPUBS**: Public protection of ACLKN configuration bits (enable, ready, divider)
Written at 1 to set ACLKNPUB by secure privileged software only. It can be read by any software.

14.10.184 RCC public configuration set register 5 (RCC_PUBCFGSR5)

Address offset: 0xFD0

Reset value: 0x0000 0000

This register is used to control the public access rights to the configuration register of the SRAMs. It is reset by sys_rstn, and is in the V_{CORE} voltage domain. Each xxPUB bit defines the public protection for the configuration registers of each SRAM: a write access is denied if the access is non-public while the respective bit is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	VENCRAMPUBS	CACHEAXIRAMPUBS	FLEXRAMPUBS	AXISRAM2PUBS	AXISRAM1PUBS	BKPSRAMPUBS	AHBSRAM2PUBS	AHBSRAM1PUBS	AXISRAM6PUBS	AXISRAM5PUBS	AXISRAM4PUBS	AXISRAM3PUBS
				w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **VENCRAMPUBS**: Public protection of VENCRAM configuration bits (enable, ready, divider)
Written at 1 to set VENCRAMPUB by secure privileged software only. It can be read by any software.

Bit 10 **CACHEAXIRAMPUBS**: Public protection of CACHEAXIRAM configuration bits (enable, ready, divider)
Written at 1 to set CACHEAXIRAMPUB by secure privileged software only. It can be read by any software.

Bit 9 **FLEXRAMPUBS**: Public protection of FLEXRAM configuration bits (enable, ready, divider)
Written at 1 to set FLEXRAMPUB by secure privileged software only. It can be read by any software.

Bit 8 **AXISRAM2PUBS**: Public protection of AXISRAM2 configuration bits (enable, ready, divider)
Written at 1 to set AXISRAM2PUB by secure privileged software only. It can rbe ead by any software.

Bit 7 **AXISRAM1PUBS**: Public protection of AXISRAM1 configuration bits (enable, ready, divider)
Written at 1 to set AXISRAM1PUB by secure privileged software only. It can be read by any software.

Bit 6 **BKPSRAMPUBS**: Public protection of BKPSRAM configuration bits (enable, ready, divider)
Written at 1 to set BKPSRAMPUB by secure privileged software only. It can be read by any software.

Bit 5 **AHBSRAM2PUBS**: Public protection of AHBSRAM2 configuration bits (enable, ready, divider)
Written at 1 to set AHBSRAM2PUB by secure privileged software only. It can be read by any software.

Bit 4 **AHBSRAM1PUBS**: Public protection of AHBSRAM1 configuration bits (enable, ready, divider)
Written at 1 to set AHBSRAM1PUB by secure privileged software only. It can be read by any software.

Bit 3 **AXISRAM6PUBS**: Public protection of AXISRAM6 configuration bits (enable, ready, divider)
Written at 1 to set AXISRAM6PUB by secure privileged software only. It can be read by any software.

Bit 2 **AXISRAM5PUBS**: Public protection of AXISRAM5 configuration bits (enable, ready, divider)
Written at 1 to set AXISRAM5PUB by secure privileged software only. It can be read by any software.

Bit 1 **AXISRAM4PUBS**: Public protection of AXISRAM4 configuration bits (enable, ready, divider)
Written at 1 to set AXISRAM4PUB by secure privileged software only. It can be read by any software.

Bit 0 **AXISRAM3PUBS**: Public protection of AXISRAM3 configuration bits (enable, ready, divider)
Written at 1 to set AXISRAM3PUB by secure privileged software only. It can be read by any software.

14.10.185 RCC control clear register (RCC_CCR)

Address offset: 0x1000

Reset value: 0x0000 0000

This register is used to enable the RCC oscillators and PLLs in Run or Sleep mode. It is reset by nreset_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	PLL4 ONC	PLL3 ONC	PLL2 ONC	PLL1 ONC	Res.	Res.	Res.	HSE ONC	HSI ONC	MSI ONC	LSE ONC	LSI ONC
				w	w	w	w				w	w	w	w	w

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **PLL4ONC**: PLL4 oscillator enable
Written at 1 to clear PLL4ON.

Bit 10 **PLL3ONC**: PLL3 oscillator enable
Written at 1 to clear PLL3ON.

Bit 9 **PLL2ONC**: PLL2 oscillator enable
Written at 1 to clear PLL2ON.

Bit 8 **PLL1ONC**: PLL1 oscillator enable
Written at 1 to clear PLL1ON.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **HSEONC**: HSE oscillator enable
Written at 1 to clear HSEON.

Bit 3 **HSIONC**: HSI oscillator enable
Written at 1 to clear HSION.

Bit 2 **MSIONC**: MSI oscillator enable
Written at 1 to clear MSION.

Bit 1 **LSEONC**: LSE oscillator enable
Written at 1 to clear LSEON.

Bit 0 **LSIONC**: LSI oscillator enable
Written at 1 to clear LSION.

14.10.186 RCC Stop mode configuration clear register (RCC_STOPCCR)

Address offset: 0x1008

Reset value: 0x0000 0000

This register is used to enable the RCC oscillators and PLLs in Stop mode. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

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

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

Bit 1 **HSISTOPENC**: HSI oscillator enable
Written at 1 to clear HSISTOPEN.

Bit 0 **MSISTOPENC**: MSI oscillator enable
Written at 1 to clear MSISTOPEN.

14.10.187 RCC miscellaneous reset clear register (RCC_MISCRSTCR)

Address offset: 0x1208

Reset value: 0x0000 0000

This register is used to clear miscellaneous RCC resets. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC2DLLRSTC	SDMMC1DLLRSTC	Res.	XSPIPHY2RSTC	XSPIPHY1RSTC	Res.	Res.	Res.	DBGRSTC
							w	w		w	w				w

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **SDMMC2DLLRSTC**: SDMMC2DLL reset
Written at 1 to clear SDMMC2DLLRST.

Bit 7 **SDMMC1DLLRSTC**: SDMMC1DLL reset
Written at 1 to clear SDMMC1DLLRST.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **XSPIPHY2RSTC**: XSPIPHY2 reset
Written at 1 hto clear XSPIPHY2RST.

Bit 4 **XSPIPHY1RSTC**: XSPIPHY1 reset
Written at 1 to clear XSPIPHY1RST.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **DBGRSTC**: DBG reset
Written at 1 to clear DBGRST.

14.10.188 RCC memory reset clear register (RCC_MEMRSTCR)

Address offset: 0x120C

Reset value: 0x0000 0000

This register is used to reset the RCC memory. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	BOOTROMRSTC	VENCRAMRSTC	CACHEAXIRAMRSTC	FLEXRAMRSTC	AXISRAM2RSTC	AXISRAM1RSTC	Res.	AHBSRAM2RSTC	AHBSRAM1RSTC	AXISRAM6RSTC	AXISRAM5RSTC	AXISRAM4RSTC	AXISRAM3RSTC
			w	w	w	w	w	w		w	w	w	w	w	w

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **BOOTROMRSTC**: BootROM reset
Written at 1 to clear BOOTROMRST.

Bit 11 **VENCRAMRSTC**: VENCRAM reset
Written at 1 to clear VENCRAMRST.

Bit 10 **CACHEAXIRAMRSTC**: CACHEAXIRAM reset
Written at 1 to clear CACHEAXIRAMRST.

Bit 9 **FLEXRAMRSTC**: FLEXRAM reset
Written at 1 to clear FLEXRAMRST.

Bit 8 **AXISRAM2RSTC**: AXISRAM2 reset
Written at 1 to clear AXISRAM2RST.

Bit 7 **AXISRAM1RSTC**: AXISRAM1 reset
Written at 1 to clear AXISRAM1RST.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **AHBSRAM2RSTC**: AHBSRAM2 reset
Written at 1 to clear AHBSRAM2RST.

Bit 4 **AHBSRAM1RSTC**: AHBSRAM1 reset
Written at 1 to clear AHBSRAM1RST.

Bit 3 **AXISRAM6RSTC**: AXISRAM6 reset
Written at 1 to clear AXISRAM6RST.

Bit 2 **AXISRAM5RSTC**: AXISRAM5 reset
Written at 1 to clear AXISRAM5RST.

Bit 1 **AXISRAM4RSTC**: AXISRAM4 reset
Written at 1 to clear AXISRAM4RST.

Bit 0 **AXISRAM3RSTC**: AXISRAM3 reset
Written at 1 to clear AXISRAM3RST.

14.10.189 RCC AHB1 reset clear register (RCC_AHB1RSTCR)

Address offset: 0x1210

Reset value: 0x0000 0000

This register is used to reset the RCC AHB1. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC12 RSTC	GPDMA1 RSTC	Res.	Res.	Res.	Res.
										w	w				

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

Bit 5 **ADC12RSTC**: ADC12 reset
Written at 1 to clear ADC12RST.

Bit 4 **GPDMA1RSTC**: GPDMA1 reset
Written at 1 to clear GPDMA1RST.

Bits 3:0 Reserved, must be kept at reset value.

14.10.190 RCC AHB2 reset clear register (RCC_AHB2RSTCR)

Address offset: 0x1214

Reset value: 0x0000 0000

This register is used to reset the RCC AHB2. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADF1RSTC	MDF1RSTC
														w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	RAMCFGRSTC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			w												

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

Bit 17 **ADF1RSTC**: ADF1 reset
Written at 1 to clear ADF1RST.

Bit 16 **MDF1RSTC**: MDF1 reset
Written at 1 to clear MDF1RST.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **RAMCFGRSTC**: RAMCFG reset
Written at 1 to clear RAMCFGRST.

Bits 11:0 Reserved, must be kept at reset value.

14.10.191 RCC AHB3 reset clear register (RCC_AHB3RSTCR)

Address offset: 0x1218

Reset value: 0x0000 0000

This register is used to reset the RCC AHB3. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	IACRSTC	Res.	PKARSTC	Res.	Res.	Res.	SAESRSTC	Res.	CRYPRSTC	HASHRSTC	RNGRSTC
					w		w				w		w	w	w

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **IACRSTC**: IAC reset
Written at 1 to clear IACRST.

Bit 9 Reserved, must be kept at reset value.

Bit 8 **PKARSTC**: PKA reset
Written at 1 to clear PKARST.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **SAESRSTC**: SAES reset
Written at 1 to clear SAESRST.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CRYPRSTC**: CRYP reset
Written at 1 to clear CRYPRST.

Bit 1 **HASHRSTC**: HASH reset
Written at 1 to clear HASHRST.

Bit 0 **RNGRSTC**: RNG reset
Written at 1 to clear RNGRST.

14.10.192 RCC AHB4 reset clear register (RCC_AHB4RSTCR)

Address offset: 0x121C

Reset value: 0x0000 0000

This register is used to reset the RCC AHB4. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCRSTC	PWRRSTC	Res.	GPIOQRSTC
												w	w		w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIOPRSTC	GPIOORSTC	GPIONRSTC	Res.	Res.	Res.	Res.	Res.	GPIOHRSTC	GPIOGRSTC	GPIOFRSTC	GPIOERSTC	GPIODRSTC	GPIOCRSTC	GPIOBRSTC	GPIOARSTC
w	w	w						w	w	w	w	w	w	w	w

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **CRCRSTC**: CRC reset
Written at 1 to clear CRCRST.

Bit 18 **PWRRSTC**: PWR reset
Written at 1 to clear PWRRST.

Bit 17 Reserved, must be kept at reset value.

Bit 16 **GPIOQRSTC**: GPIO Q reset
Written at 1 to clear GPIOQRST.

Bit 15 **GPIOPRSTC**: GPIO P reset
Written at 1 to clear GPIOPRST.

Bit 14 **GPIOORSTC**: GPIO O reset
Written at 1 to clear GPIOORST.

Bit 13 **GPIONRSTC**: GPIO N reset
Written at 1 to clear GPIONRST.

Bits 12:8 Reserved, must be kept at reset value.

Bit 7 **GPIOHRSTC**: GPIO H reset
Written at 1 to clear GPIOHRST.

Bit 6 **GPIOGRSTC**: GPIO G reset
Written at 1 to clear GPIOGRST.

Bit 5 **GPIOFRSTC**: GPIO F reset
Written at 1 to clear GPIOFRST.

Bit 4 **GPIOERSTC**: GPIO E reset
Written at 1 to clear GPIOERST.

Bit 3 **GPIODRSTC**: GPIO D reset
Written at 1 to clear GPIODRST.

Bit 2 **GPIOCRSTC**: GPIO C reset
Written at 1 to clear GPIOCRST.

Bit 1 **GPIOBRSTC**: GPIO B reset
Written at 1 to clear GPIOBRST.

Bit 0 **GPIOARSTC**: GPIO A reset
Written at 1 to clear GPIOARST.

14.10.193 RCC AHB5 reset clear register (RCC_AHB5RSTCR)

Address offset: 0x1220

Reset value: 0x0000 0000

This register is used to reset the RCC AHB5. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NPU RSTC	CACHE AXI RSTC	OTG2 RSTC	OTG PHY2 RSTC	OTG PHY1 RSTC	OTG1 RSTC	ETH1 RSTC	OTG2 PHY CTL RSTC	OTG1 PHY CTL RSTC	Res.	Res.	GPU2D RSTC	GFX MMU RSTC	Res.	XSPI3 RSTC	Res.
w	w	w	w	w	w	w	w	w			w	w		w	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	XSPIM RSTC	XSPI2 RSTC	Res.	Res.	Res.	SDMM C1 RSTC	SDMM C2 RSTC	PSSI RSTC	XSPI1 RSTC	FMC RSTC	JPEG RSTC	Res.	DMA2D RSTC	HPD MA1 RSTC
		w	w				w	w	w	w	w	w		w	w

Bit 31 **NPURSTC**: NPU reset
Written at 1 to clear NPURST.

Bit 30 **CACHEAXIRSTC**: CACHEAXI reset
Written at 1 to clear CACHEAXIRST.

Bit 29 **OTG2RSTC**: OTG2 reset
Written at 1 to clear OTG2RST.

Bit 28 **OTGPHY2RSTC**: OTGPHY2 reset
Written at 1 to clear OTGPHY2RST.

Bit 27 **OTGPHY1RSTC**: OTGPHY1 reset
Written at 1 to clear OTGPHY1RST.

Bit 26 **OTG1RSTC**: OTG1 reset
Written at 1 to clear OTG1RST.

Bit 25 **ETH1RSTC**: ETH1 reset
Written at 1 to clear ETH1RST.

Bit 24 **OTG2PHYCTLRSTC**: OTG2PHYCTL reset
Written at 1 to clear OTG2PHYCTLRST.

Bit 23 **OTG1PHYCTLRSTC**: OTG1PHYCTL reset
Written at 1 to clear OTG1PHYCTLRST.

Bits 22:21 Reserved, must be kept at reset value.

Bit 20 **GPU2DRSTC**: GPU2D reset
Written at 1 to clear GPU2DRST.

Bit 19 **GFXMMURSTC**: GFXMMU reset
Written at 1 to clear GFXMMURST.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **XSPI3RSTC**: XSPI3 reset
Written at 1 to clear XSPI3RST.

Bits 16:14 Reserved, must be kept at reset value.

Bit 13 **XSPIMRSTC**: XSPIM reset
Written at 1 to clear XSPIMRST.

Bit 12 **XSPI2RSTC**: XSPI2 reset
Written at 1 to clear XSPI2RST.

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **SDMMC1RSTC**: SDMMC1 reset
Written at 1 to clear SDMMC1RST.

Bit 7 **SDMMC2RSTC**: SDMMC2 reset
Written at 1 to clear SDMMC2RST.

Bit 6 **PSSIRSTC**: PSSI reset
Written at 1 to clear PSSIRST.

Bit 5 **XSPI1RSTC**: XSPI1 reset
Written at 1 to clear XSPI1RST.

Bit 4 **FMCRSTC**: FMC reset
Written at 1 to clear FMCRST.

Bit 3 **JPEGRSTC**: JPEG reset
Written at 1 to clear JPEGRST.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **DMA2DRSTC**: DMA2D reset
Written at 1 to clear DMA2DRST.

Bit 0 **HPDMA1RSTC**: HPDMA1 reset
Written at 1 to clear HPDMA1RST.

14.10.194 RCC APB1L reset clear register (RCC_APB1LRSTCR)

Address offset: 0x1224

Reset value: 0x0000 0000

This register is used to reset the RCC APB1L. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UART8 RSTC	UART7 RSTC	Res.	Res.	Res.	Res.	I3C2RS TC	I3C1RS TC	I2C3RS TC	I2C2RS TC	I2C1RS TC	UART5 RSTC	UART4 RSTC	USART 3RSTC	USART 2RSTC	SPDIF RX1RS TC
w	w					w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3R STC	SPI2R STC	TIM11R STC	TIM10 RSTC	WWDG RSTC	Res.	LPTIM1 RSTC	TIM14 RSTC	TIM13 RSTC	TIM12 RSTC	TIM7R STC	TIM6R STC	TIM5R STC	TIM4R STC	TIM3R STC	TIM2R STC
w	w	w	w	w		w	w	w	w	w	w	w	w	w	w

Bit 31 **UART8RSTC**: UART8 reset
Written at 1 to clear UART8RST.

Bit 30 **UART7RSTC**: UART7 reset
Written at 1 to clear UART7RST.

Bits 29:26 Reserved, must be kept at reset value.

Bit 25 **I3C2RSTC**: I3C2 reset
Written at 1 to clear I3C2RST.

Bit 24 **I3C1RSTC**: I3C1 reset
Written at 1 to clear I3C1RST.

Bit 23 **I2C3RSTC**: I2C3 reset
Written at 1 to clear I2C3RST.

Bit 22 **I2C2RSTC**: I2C2 reset
Written at 1 to clear I2C2RST.

Bit 21 **I2C1RSTC**: I2C1 reset
Written at 1 to clear I2C1RST.

Bit 20 **UART5RSTC**: UART5 reset
Written at 1 to clear UART5RST.

Bit 19 **UART4RSTC**: UART4 reset
Written at 1 to clear UART4RST.

Bit 18 **USART3RSTC**: USART3 reset
Written at 1 to clear USART3RST.

Bit 17 **USART2RSTC**: USART2 reset
Written at 1 to clear USART2RST.

Bit 16 **SPDIFRX1RSTC**: SPDIFRX1 reset
Written at 1 to clear SPDIFRX1RST.

Bit 15 **SPI3RSTC**: SPI3 reset
Written at 1 to clear SPI3RST.

Bit 14 **SPI2RSTC**: SPI2 reset
Written at 1 to clear SPI2RST.

Bit 13 **TIM11RSTC**: TIM11 reset
Written at 1 to clear TIM11RST.

Bit 12 **TIM10RSTC**: TIM10 reset
Written at 1 to clear TIM10RST.

Bit 11 **WWDGRSTC**: WWDG reset
Written at 1 to clear WWDGRST.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **LPTIM1RSTC**: LPTIM1 reset
Written at 1 to clear LPTIM1RST.

Bit 8 **TIM14RSTC**: TIM14 reset
Written at 1 to clear TIM14RST.

Bit 7 **TIM13RSTC**: TIM13 reset
Written at 1 to clear TIM13RST.

Bit 6 **TIM12RSTC**: TIM12 reset
Written at 1 to clear TIM12RST.

Bit 5 **TIM7RSTC**: TIM7 reset
Written at 1 to clear TIM7RST.

Bit 4 **TIM6RSTC**: TIM6 reset
Written at 1 to clear TIM6RST.

Bit 3 **TIM5RSTC**: TIM5 reset
Written at 1 to clear TIM5RST.

Bit 2 **TIM4RSTC**: TIM4 reset
Written at 1 to clear TIM4RST.

Bit 1 **TIM3RSTC**: TIM3 reset
Written at 1 to clear TIM3RST.

Bit 0 **TIM2RSTC**: TIM2 reset
Written at 1 to clear TIM2RST.

14.10.195 RCC APB1H reset clear register (RCC_APB1HRSTCR)

Address offset: 0x1228

Reset value: 0x0000 0000

This register is used to reset the RCC APB1H. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD1 RSTC	Res.	Res.
													w		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCAN RSTC	Res.	Res.	MDIOS RSTC	Res.	Res.	Res.	Res.	Res.
							w			w					

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **UCPD1RSTC**: UCPD1 reset
Written at 1 to clear UCPD1RST.

Bits 17:9 Reserved, must be kept at reset value.

Bit 8 **FDCANRSTC**: FDCAN reset
Written at 1 to clear FDCANRST.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **MDIOSRSTC**: MDIOS reset
Written at 1 to clear MDIOSRST.

Bits 4:0 Reserved, must be kept at reset value.

14.10.196 RCC APB2 reset clear register (RCC_APB2RSTCR)

Address offset: 0x122C

Reset value: 0x0000 0000

This register is used to reset the RCC APB2. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI2RSTC	SAI1RSTC	SPI5RSTC	TIM9RSTC	TIM17RSTC	TIM16RSTC	TIM15RSTC
									w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIM18RSTC	Res.	SPI4RSTC	SPI1RSTC	Res.	Res.	Res.	Res.	USART10RSTC	UART9RSTC	USART6RSTC	USART1RSTC	Res.	Res.	TIM8RSTC	TIM1RSTC
w		w	w					w	w	w	w			w	w

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **SAI2RSTC**: SAI2 reset
Written at 1 to clear SAI2RST.

Bit 21 **SAI1RSTC**: SAI1 reset
Written at 1 to clear SAI1RST.

Bit 20 **SPI5RSTC**: SPI5 reset
Written at 1 to clear SPI5RST.

Bit 19 **TIM9RSTC**: TIM9 reset
Written at 1 to clear TIM9RST.

Bit 18 **TIM17RSTC**: TIM17 reset
Written at 1 to clear TIM17RST.

Bit 17 **TIM16RSTC**: TIM16 reset
Written at 1 to clear TIM16RST.

Bit 16 **TIM15RSTC**: TIM15 reset
Written at 1 to clear TIM15RST.

Bit 15 **TIM18RSTC**: TIM18 reset
Written at 1 to clear TIM18RST.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **SPI4RSTC**: SPI4 reset
Written at 1 to clear SPI4RST.

Bit 12 **SPI1RSTC**: SPI1 reset
Written at 1 to clear SPI1RST.

Bits 11:8 Reserved, must be kept at reset value.

Bit 7 **USART10RSTC**: USART10 reset
Written at 1 to clear USART10RST.

Bit 6 **UART9RSTC**: UART9 reset
Written at 1 to clear UART9RST.

Bit 5 **USART6RSTC**: USART6 reset
Written at 1 to clear USART6RST.

Bit 4 **USART1RSTC**: USART1 reset
Written at 1 to clear USART1RST.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **TIM8RSTC**: TIM8 reset
Written at 1 to clear TIM8RST.

Bit 0 **TIM1RSTC**: TIM1 reset
Written at 1 to clear TIM1RST.

14.10.197 RCC APB4L reset clear register (RCC_APB4LRSTCR)

Address offset: 0x1234

Reset value: 0x0000 0000

This register is used to reset the RCC APB4L. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTC RSTC
															w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VREFBUF RSTC	Res.	Res.	LPTIM5 RSTC	LPTIM4 RSTC	LPTIM3 RSTC	LPTIM2 RSTC	Res.	I2C4 RSTC	Res.	SPI6 RSTC	Res.	LPUART1 RSTC	HDP RSTC	Res.	Res.
w			w	w	w	w		w		w		w	w		

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **RTCRSTC**: RTC reset
Written at 1 to clear RTCRST.

Bit 15 **VREFBUFRSTC**: VREFBUF reset
Written at 1 to clear VREFBUFRST.

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **LPTIM5RSTC**: LPTIM5 reset
Written at 1 to clear LPTIM5RST.

Bit 11 **LPTIM4RSTC**: LPTIM4 reset
Written at 1 to clear LPTIM4RST.

Bit 10 **LPTIM3RSTC**: LPTIM3 reset
Written at 1 to clear LPTIM3RST.

Bit 9 **LPTIM2RSTC**: LPTIM2 reset
Written at 1 to clear LPTIM2RST.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **I2C4RSTC**: I2C4 reset
Written at 1 to clear I2C4RST.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **SPI6RSTC**: SPI6 reset
Written at 1 to clear SPI6RST.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **LPUART1RSTC**: LPUART1 reset
Written at 1 to clear LPUART1RST.

Bit 2 **HDPRSTC**: HDP reset
Written at 1 to clear HDPRST.

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

14.10.198 RCC APB4H reset clear register (RCC_APB4HRSTCR)

Address offset: 0x1238

Reset value: 0x0000 0000

This register is used to reset the RCC APB4H. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

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

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

Bit 2 **DTSRSTC**: DTS reset
Written at 1 to clear DTSRST.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **SYSCFGRSTC**: SYSCFG reset
Written at 1 to clear SYSCFGRST.

14.10.199 RCC APB5 reset clear register (RCC_APB5RSTCR)

Address offset: 0x123C

Reset value: 0x0000 0000

This register is used to clear the reset of APB5 peripherals. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

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

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CSIRSTC**: CSI reset
Written at 1 to clear CSIRST.

Bit 5 **VENCRSTC**: VENC reset
Written at 1 to clear VENCRST.

Bit 4 **GFXTIMRSTC**: GFXTIM reset
Written at 1 to clear GFXTIMRST.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **DCMIPPRSTC**: DCMIPP reset
Written at 1 to clear DCMIPPRST.

Bit 1 **LTDCRSTC**: LTDC reset
Written at 1 to clear LTDCRST.

Bit 0 Reserved, must be kept at reset value.

14.10.200 RCC divider enable clear register (RCC_DIVENCR)

Address offset: 0x1240

Reset value: 0x0000 0000

This register is used to enable the RCC IC dividers in Run, Sleep, or Stop mode. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC20ENC	IC19ENC	IC18ENC	IC17ENC
												w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC16ENC	IC15ENC	IC14ENC	IC13ENC	IC12ENC	IC11ENC	IC10ENC	IC9ENC	IC8ENC	IC7ENC	IC6ENC	IC5ENC	IC4ENC	IC3ENC	IC2ENC	IC1ENC
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **IC20ENC**: IC20 enable
Written at 1 to clear IC20EN.

Bit 18 **IC19ENC**: IC19 enable
Written at 1 to clear IC19EN.

Bit 17 **IC18ENC**: IC18 enable
Written at 1 to clear IC18EN.

Bit 16 **IC17ENC**: IC17 enable
Written at 1 to clear IC17EN.

Bit 15 **IC16ENC**: IC16 enable
Written at 1 to clear IC16EN.

Bit 14 **IC15ENC**: IC15 enable
Written at 1 to clear IC15EN.

Bit 13 **IC14ENC**: IC14 enable
Written at 1 to clear IC14EN.

Bit 12 **IC13ENC**: IC13 enable
Written at 1 to clear IC13EN.

Bit 11 **IC12ENC**: IC12 enable
Written at 1 to clear IC12EN.

Bit 10 **IC11ENC**: IC11 enable
Written at 1 to clear IC11EN.

Bit 9 **IC10ENC**: IC10 enable
Written at 1 to clear IC10EN.

Bit 8 **IC9ENC**: IC9 enable
Written at 1 to clear IC9EN.

Bit 7 **IC8ENC**: IC8 enable
Written at 1 to clear IC8EN.

Bit 6 **IC7ENC**: IC7 enable
Written at 1 to clear IC7EN.

Bit 5 **IC6ENC**: IC6 enable
Written at 1 to clear IC6EN.

Bit 4 **IC5ENC**: IC5 enable
Written at 1 to clear IC5EN.

Bit 3 **IC4ENC**: IC4 enable
Written at 1 to clear IC4EN.

Bit 2 **IC3ENC**: IC3 enable
Written at 1 to clear IC3EN.

Bit 1 **IC2ENC**: IC2 enable
Written at 1 to clear IC2EN.

Bit 0 **IC1ENC**: IC1 enable
Written at 1 to clear IC1EN.

14.10.201 RCC bus enable clear register (RCC_BUSENCR)

Address offset: 0x1244

Reset value: 0x0000 0000

This register is used to enable the RCC bus in Run and Sleep modes (in Sleep mode, each bit of this register is AND-ed with the LPEN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

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

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

Bit 1 **ACLKNCENC**: ACLKNC enable
Written at 1 to clear ACLKNCEN.

Bit 0 **ACLKNENC**: ACLKN enable
Written at 1 to clear ACLKNEN.

14.10.202 RCC miscellaneous enable clear register (RCC_MISCENCR)

Address offset: 0x1248

Reset value: 0x0000 0000

This register is used to enable the RCC miscellaneous in Run and Sleep modes (in Sleep mode, each bit of this register is AND-ed with the LPEN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERENC	Res.	Res.	XSPIPHYCOMPENC	MCO2ENC	MCO1ENC	DBGENC
									w			w	w	w	w

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **PERENC**: PER enable
Written at 1 to clear PEREN.

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **XSPIPHYCOMPENC**: XSPIPHYCOMP enable
Written at 1 to clear XSPIPHYCOMPEN.

Bit 2 **MCO2ENC**: MCO2 enable
Written at 1 to clear MCO2EN.

Bit 1 **MCO1ENC**: MCO1 enable
Written at 1 to clear MCO1EN.

Bit 0 **DBGENC**: DBG enable
Written at 1 to clear DBGEN.

14.10.203 RCC memory enable clear register (RCC_MEMENCR)

Address offset: 0x124C

Reset value: 0x0000 0000

This register is used to enable the RCC memory in Run and Sleep modes (in Sleep mode, each bit of this register is AND-ed with the LPEN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	BOOTROMENC	VENCRAMENC	CACHEAXIRAMENC	FLEXRAMENC	AXISRAM2ENC	AXISRAM1ENC	BKPSRAMENC	AHBSRAM2ENC	AHBSRAM1ENC	AXISRAM6ENC	AXISRAM5ENC	AXISRAM4ENC	AXISRAM3ENC
			w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **BOOTROMENC**: BootROM enable
Written at 1 to clear BOOTROMEN.

Bit 11 **VENCRAMENC**: VENCRAM enable
Written at 1 to clear VENCRAMEN.

Bit 10 **CACHEAXIRAMENC**: CACHEAXIRAM enable
Written at 1 to clear CACHEAXIRAMEN.

Bit 9 **FLEXRAMENC**: FLEXRAM enable
Written at 1 to clear FLEXRAMEN.

Bit 8 **AXISRAM2ENC**: AXISRAM2 enable
Written at 1 to clear AXISRAM2EN.

Bit 7 **AXISRAM1ENC**: AXISRAM1 enable
Written at 1 to clear AXISRAM1EN.

Bit 6 **BKPSRAMENC**: BKPSRAM enable
Written at 1 to clear BKPSRAMEN.

Bit 5 **AHBSRAM2ENC**: AHBSRAM2 enable
Written at 1 to clear AHBSRAM2EN.

Bit 4 **AHBSRAM1ENC**: AHBSRAM1 enable
Written at 1 to clear AHBSRAM1EN.

Bit 3 **AXISRAM6ENC**: AXISRAM6 enable
Written at 1 to clear AXISRAM6EN.

Bit 2 **AXISRAM5ENC**: AXISRAM5 enable
Written at 1 to clear AXISRAM5EN.

Bit 1 **AXISRAM4ENC**: AXISRAM4 enable
Written at 1 to clear AXISRAM4EN.

Bit 0 **AXISRAM3ENC**: AXISRAM3 enable
Written at 1 to clear AXISRAM3EN.

14.10.204 RCC AHB1 enable clear register (RCC_AHB1ENCR)

Address offset: 0x1250

Reset value: 0x0000 0000

This register is used to enable the RCC AHB1 in Run and Sleep modes (in Sleep mode, each bit of this register is AND-ed with the LPEN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC12 ENC	GPDM A1ENC	Res.	Res.	Res.	Res.
										w	w				

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

Bit 5 **ADC12ENC**: ADC12 enable
Written at 1 to clear ADC12EN.

Bit 4 **GPDMA1ENC**: GPDMA1 enable
Written at 1 to clear GPDMA1EN.

Bits 3:0 Reserved, must be kept at reset value.

14.10.205 RCC AHB2 enable clear register (RCC_AHB2ENCR)

Address offset: 0x1254

Reset value: 0x0000 0000

This register is used to enable the RCC AHB2 in Run and Sleep modes (in Sleep mode, each bit of this register is AND-ed with the LPEN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADF1ENC	MDF1ENC
														w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	RAMCFGENC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			w												

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

Bit 17 **ADF1ENC**: ADF1 enable
Written at 1 to clear ADF1EN.

Bit 16 **MDF1ENC**: MDF1 enable
Written at 1 to clear MDF1EN.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **RAMCFGENC**: RAMCFG enable
Written at 1 to clear RAMCFGEN.

Bits 11:0 Reserved, must be kept at reset value.

14.10.206 RCC AHB3 enable clear register (RCC_AHB3ENCR)

Address offset: 0x1258

Reset value: 0x0000 0000

This register is used to enable the RCC AHB3 in Run and Sleep modes (in Sleep mode, each bit in this register is AND-ed with the LPEN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RISAFENC	Res.	Res.	Res.	IACENC	RIFSCENC	PKAENC	Res.	Res.	Res.	SAESENC	Res.	CRYPENC	HASHENC	RNGENC
	w				w	w	w				w		w	w	w

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **RISAFENC**: RISAF enable
Written at 1 to clear RISAFEN.

Bits 13:11 Reserved, must be kept at reset value.

Bit 10 **IACENC**: IAC enable
Written at 1 to clear IACEN.

Bit 9 **RIFSCENC**: RIFSC enable
Written at 1 to clear RIFSCEN.

Bit 8 **PKAENC**: PKA enable
Written at 1 to clear PKAEN.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **SAESENC**: SAES enable
Written at 1 to clear SAESEN.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CRYPENC**: CRYP enable
Written at 1 to clear CRYPEN.

Bit 1 **HASHENC**: HASH enable
Written at 1 to clear HASHEN.

Bit 0 **RNGENC**: RNG enable
Written at 1 to clear RNGEN.

14.10.207 RCC AHB4 enable clear register (RCC_AHB4ENCR)

Address offset: 0x125C

Reset value: 0x0000 0000

This register is used to enable the RCC AHB4 in Run and Sleep modes (in Sleep mode, each bit of this register is AND-ed with the LPEN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCENC	PWRENC	Res.	GPIOQENC
												w	w		w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIOPENC	GPIOOENC	GPIONENC	Res.	Res.	Res.	Res.	Res.	GPIOHENC	GPIOGENC	GPIOFENC	GPIOEENC	GPIODENC	GPIOCENC	GPIOBENC	GPIOAENC
w	w	w						w	w	w	w	w	w	w	w

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **CRCENC**: CRC enable
Written at 1 to clear CRCEN.

Bit 18 **PWRENC**: PWR enable
Written at 1 to clear PWREN.

Bit 17 Reserved, must be kept at reset value.

Bit 16 **GPIOQENC**: GPIO Q enable
Written at 1 to clear GPIOQEN.

Bit 15 **GPIOPENC**: GPIO P enable
Written at 1 to clear GPIOPEN.

Bit 14 **GPIOOENC**: GPIO O enable
Written at 1 to clear GPIOOEN.

Bit 13 **GPIONENC**: GPIO N enable
Written at 1 to clear GPIONEN.

Bits 12:8 Reserved, must be kept at reset value.

Bit 7 **GPIOHENC**: GPIO H enable
Written at 1 to clear GPIOHEN.

Bit 6 **GPIOGENC**: GPIO G enable
Written at 1 to clear GPIOGEN.

Bit 5 **GPIOFENC**: GPIO F enable
Written at 1 to clear GPIOFEN.

Bit 4 **GPIOEENC**: GPIO E enable
Written at 1 to clear GPIOEEN.

Bit 3 **GPIODENC**: GPIO D enable
Written at 1 to clear GPIODEN.

Bit 2 **GPIOCENC**: GPIO C enable
Written at 1 to clear GPIOCEN.

Bit 1 **GPIOBENC**: GPIO B enable
Written at 1 to clear GPIOBEN.

Bit 0 **GPIOAENC**: GPIO A enable
Written at 1 to clear GPIOAEN.

14.10.208 RCC AHB5 enable clear register (RCC_AHB5ENCR)

Address offset: 0x1260

Reset value: 0x0000 0000

This register is used to enable the RCC AHB5 in Run and Sleep modes (in Sleep mode, each bit of this register is AND-ed with the LPEN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NPUENC	CACHEAXIENC	OTG2ENC	OTGPHY2ENC	OTGPHY1ENC	OTG1ENC	ETH1ENC	ETH1RXENC	ETH1TXENC	ETH1MACENC	Res.	GPU2DENC	GFXMMUENC	MCE4ENC	XSPI3ENC	MCE3ENC
w	w	w	w	w	w	w	w	w	w		w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCE2ENC	MCE1ENC	XSPIMENC	XSPI2ENC	Res.	Res.	Res.	SDMMC1ENC	SDMMC2ENC	PSSIENC	XSPI1ENC	FMCENC	JPEGENC	Res.	DMA2DENC	HPDMA1ENC
w	w	w	w				w	w	w	w	w	w		w	w

Bit 31 **NPUENC**: NPU enable
Written at 1 to clear NPUEN.

Bit 30 **CACHEAXIENC**: CACHEAXI enable
Written at 1 to clear CACHEAXIEN.

Bit 29 **OTG2ENC**: OTG2 enable
Written at 1 to clear OTG2EN.

Bit 28 **OTGPHY2ENC**: OTGPHY2 enable
Written at 1 to clear OTGPHY2EN.

Bit 27 **OTGPHY1ENC**: OTGPHY1 enable
Written at 1 to clear OTGPHY1EN.

Bit 26 **OTG1ENC**: OTG1 enable
Written at 1 to clear OTG1EN.

Bit 25 **ETH1ENC**: ETH1 enable
Written at 1 to clear ETH1EN.

Bit 24 **ETH1RXENC**: ETH1RX enable
Written at 1 to clear ETH1RXEN.

Bit 23 **ETH1TXENC**: ETH1TX enable
Written at 1 to clear ETH1TXEN.

Bit 22 **ETH1MACENC**: ETH1MAC enable
Written at 1 to clear ETH1MACEN.

Bit 21 Reserved, must be kept at reset value.

Bit 20 **GPU2DENC**: GPU2D enable
Written at 1 to clear GPU2DEN.

Bit 19 **GFXMMUENC**: GFXMMU enable
Written at 1 to clear GFXMMUEN.

Bit 18 **MCE4ENC**: MCE4 enable
Written at 1 to clear MCE4EN.

Bit 17 **XSPI3ENC**: XSPI3 enable
Written at 1 to clear XSPI3EN.

Bit 16 **MCE3ENC**: MCE3 enable
Written at 1 to clear MCE3EN.

Bit 15 **MCE2ENC**: MCE2 enable
Written at 1 to clear MCE2EN.

Bit 14 **MCE1ENC**: MCE1 enable
Written at 1 to clear MCE1EN.

Bit 13 **XSPIMENC**: XSPIM enable
Written at 1 to clear XSPIMEN.

Bit 12 **XSPI2ENC**: XSPI2 enable
Written at 1 to clear XSPI2EN.

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **SDMMC1ENC**: SDMMC1 enable
Written at 1 to clear SDMMC1EN.

Bit 7 **SDMMC2ENC**: SDMMC2 enable
Written at 1 to clear SDMMC2EN.

Bit 6 **PSSIENC**: PSSI enable
Written at 1 to clear PSSIEN.

Bit 5 **XSPI1ENC**: XSPI1 enable
Written at 1 to clear XSPI1EN.

Bit 4 **FMCENC**: FMC enable
Written at 1 to clear FMCEN.

Bit 3 **JPEGENC**: JPEG enable
Written at 1 to clear JPEGEN.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **DMA2DENC**: DMA2D enable
Written at 1 to clear DMA2DEN.

Bit 0 **HPDMA1ENC**: HPDMA1 enable
Written at 1 to clear HPDMA1EN.

14.10.209 RCC APB1L enable clear register (RCC_APB1LENCR)

Address offset: 0x1264

Reset value: 0x0000 0000

This register is used to enable the RCC APB1L in Run and Sleep modes (in Sleep mode, each bit of this register is AND-ed with the LPEN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UART8ENC	UART7ENC	Res.	Res.	Res.	Res.	I3C2ENC	I3C1ENC	I2C3ENC	I2C2ENC	I2C1ENC	UART5ENC	UART4ENC	USART3ENC	USART2ENC	SPDIFRX1ENC
w	w					w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3ENC	SPI2ENC	TIM11ENC	TIM10ENC	Res.	Res.	LPTIM1ENC	TIM14ENC	TIM13ENC	TIM12ENC	TIM7ENC	TIM6ENC	TIM5ENC	TIM4ENC	TIM3ENC	TIM2ENC
w	w	w	w			w	w	w	w	w	w	w	w	w	w

Bit 31 **UART8ENC**: UART8 enable
Written at 1 to clear UART8EN.

Bit 30 **UART7ENC**: UART7 enable
Written at 1 to clear UART7EN.

Bits 29:26 Reserved, must be kept at reset value.

Bit 25 **I3C2ENC**: I3C2 enable
Written at 1 to clear I3C2EN.

Bit 24 **I3C1ENC**: I3C1 enable
Written at 1 to clear I3C1EN.

Bit 23 **I2C3ENC**: I2C3 enable
Written at 1 to clear I2C3EN.

Bit 22 **I2C2ENC**: I2C2 enable
Written at 1 to clear I2C2EN.

Bit 21 **I2C1ENC**: I2C1 enable
Written at 1 to clear I2C1EN.

Bit 20 **UART5ENC**: UART5 enable
Written at 1 to clear UART5EN.

Bit 19 **UART4ENC**: UART4 enable
Written at 1 to clear UART4EN.

Bit 18 **USART3ENC**: USART3 enable
Written at 1 to clear USART3EN.

Bit 17 **USART2ENC**: USART2 enable
Written at 1 to clear USART2EN.

Bit 16 **SPDIFRX1ENC**: SPDIFRX1 enable
Written at 1 to clear SPDIFRX1EN.

Bit 15 **SPI3ENC**: SPI3 enable
Written at 1 to clear SPI3EN.

Bit 14 **SPI2ENC**: SPI2 enable
Written at 1 to clear SPI2EN.

Bit 13 **TIM11ENC**: TIM11 enable
Written at 1 to clear TIM11EN.

Bit 12 **TIM10ENC**: TIM10 enable
Written at 1 to clear TIM10EN.

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **LPTIM1ENC**: LPTIM1 enable
Written at 1 to clear LPTIM1EN.

Bit 8 **TIM14ENC**: TIM14 enable
Written at 1 to clear TIM14EN.

Bit 7 **TIM13ENC**: TIM13 enable
Written at 1 to clear TIM13EN.

Bit 6 **TIM12ENC**: TIM12 enable
Written at 1 to clear TIM12EN.

Bit 5 **TIM7ENC**: TIM7 enable
Written at 1 to clear TIM7EN.

Bit 4 **TIM6ENC**: TIM6 enable
Written at 1 to clear TIM6EN.

Bit 3 **TIM5ENC**: TIM5 enable
Written at 1 to clear TIM5EN.

Bit 2 **TIM4ENC**: TIM4 enable
Written at 1 to clear TIM4EN.

Bit 1 **TIM3ENC**: TIM3 enable
Written at 1 to clear TIM3EN.

Bit 0 **TIM2ENC**: TIM2 enable
Written at 1 to clear TIM2EN.

14.10.210 RCC APB1H enable clear register (RCC_APB1HENCR)

Address offset: 0x1268

Reset value: 0x0000 0000

This register is used to enable the RCC APB1H in Run and Sleep modes (in Sleep mode, each bit of this register is AND-ed with the LPEN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD1 ENC	Res.	Res.
													w		

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

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **UCPD1ENC**: UCPD1 enable
Written at 1 to clear UCPD1EN.

Bits 17:9 Reserved, must be kept at reset value.

Bit 8 **FDCANENC**: FDCAN enable
Written at 1 to clear FDCANEN.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **MDIOSENC**: MDIOS enable
Written at 1 to clear MDIOSEN.

Bits 4:0 Reserved, must be kept at reset value.

14.10.211 RCC APB2 enable clear register (RCC_APB2ENCR)

Address offset: 0x126C

Reset value: 0x0000 0000

This register is used to enable the RCC APB2 in Run and Sleep modes (in Sleep mode, each bit of this register is AND-ed with the LPEN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI2ENC	SAI1ENC	SPI5ENC	TIM9ENC	TIM17ENC	TIM16ENC	TIM15ENC
									w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIM18ENC	Res.	SPI4ENC	SPI1ENC	Res.	Res.	Res.	Res.	USART10ENC	UART9ENC	USART6ENC	USART1ENC	Res.	Res.	TIM8ENC	TIM1ENC
w		w	w					w	w	w	w			w	w

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **SAI2ENC**: SAI2 enable
Written at 1 to clear SAI2EN.

Bit 21 **SAI1ENC**: SAI1 enable
Written at 1 to clear SAI1EN.

Bit 20 **SPI5ENC**: SPI5 enable
Written at 1 to clear SPI5EN.

Bit 19 **TIM9ENC**: TIM9 enable
Written at 1 to clear TIM9EN.

Bit 18 **TIM17ENC**: TIM17 enable
Written at 1 to clear TIM17EN.

Bit 17 **TIM16ENC**: TIM16 enable
Written at 1 to clear TIM16EN.

Bit 16 **TIM15ENC**: TIM15 enable
Written at 1 to clear TIM15EN.

Bit 15 **TIM18ENC**: TIM18 enable
Written at 1 to clear TIM18EN.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **SPI4ENC**: SPI4 enable
Written at 1 to clear SPI4EN.

Bit 12 **SPI1ENC**: SPI1 enable
Written at 1 to clear SPI1EN.

Bits 11:8 Reserved, must be kept at reset value.

Bit 7 **USART10ENC**: USART10 enable
Written at 1 to clear USART10EN.

Bit 6 **UART9ENC**: UART9 enable
Written at 1 to clear UART9EN.

Bit 5 **USART6ENC**: USART6 enable
Written at 1 to clear USART6EN.

Bit 4 **USART1ENC**: USART1 enable
Written at 1 to clear USART1EN.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **TIM8ENC**: TIM8 enable
Written at 1 to clear TIM8EN.

Bit 0 **TIM1ENC**: TIM1 enable
Written at 1 to clear TIM1EN.

14.10.212 RCC APB3 enable clear register (RCC_APB3ENCR)

Address offset: 0x1270

Reset value: 0x0000 0000

This register is used to enable the RCC APB3 in Run and Sleep modes (in Sleep mode, each bit of this register is AND-ed with the LPEN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

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

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

Bit 2 **DFTENC**: DFT enable
Written at 1 to clear DFTEN.

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

14.10.213 RCC APB4L enable clear register (RCC_APB4LENCR)

Address offset: 0x1274

Reset value: 0x0000 0000

This register is used to enable the RCC APB4L in Run and Sleep modes (in Sleep mode, each bit of this register is AND-ed with the LPEN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCAPBENC	RTCENC
														w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VREFBUFENC	Res.	Res.	LPTIM5ENC	LPTIM4ENC	LPTIM3ENC	LPTIM2ENC	Res.	I2C4ENC	Res.	SPI6ENC	Res.	LPUART1ENC	HDPENC	Res.	Res.
w			w	w	w	w		w		w		w	w		

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

Bit 17 **RTCAPBENC**: RTCAPB enable
Written at 1 to clear RTCAPBEN.

Bit 16 **RTCENC**: RTC enable
Written at 1 to clear RTCEN.

Bit 15 **VREFBUFENC**: VREFBUF enable
Written at 1 to clear VREFBUFEN.

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **LPTIM5ENC**: LPTIM5 enable
Written at 1 to clear LPTIM5EN.

Bit 11 **LPTIM4ENC**: LPTIM4 enable
Written at 1 to clear LPTIM4EN.

Bit 10 **LPTIM3ENC**: LPTIM3 enable
Written at 1 to clear LPTIM3EN.

Bit 9 **LPTIM2ENC**: LPTIM2 enable
Written at 1 to clear LPTIM2EN.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **I2C4ENC**: I2C4 enable
Written at 1 to clear I2C4EN.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **SPI6ENC**: SPI6 enable
Written at 1 to clear SPI6EN.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **LPUART1ENC**: LPUART1 enable
Written at 1 to clear LPUART1EN.

Bit 2 **HDPENC**: HDP enable
Written at 1 to clear HDPEN.

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

14.10.214 RCC APB4H enable clear register (RCC_APB4HENCR)

Address offset: 0x1278

Reset value: 0x0000 0000

This register is used to enable the RCC APB4H in Run and Sleep modes (in Sleep mode, each bit of this register is AND-ed with the LPEN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

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

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

Bit 2 **DTSENC**: DTS enable
Written at 1 to clear DTSEN.

Bit 1 **BSECENC**: BSEC enable
Written at 1 to clear BSECEN.

Bit 0 **SYSCFGENC**: SYSCFG enable
Written at 1 to clear SYSCFGEN.

14.10.215 RCC APB5 enable clear register (RCC_APB5ENCR)

Address offset: 0x127C

Reset value: 0x0000 0000

This register is used to enable the RCC APB5 in Run and Sleep modes (in Sleep mode, each bit of this register is AND-ed with the LPEN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

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

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CSIENC**: CSI enable
Written at 1 to clear CSIEN.

Bit 5 **VENCENC**: VENC enable
Written at 1 to clear VENCEN.

Bit 4 **GFXTIMENC**: GFXTIM enable
Written at 1 to clear GFXTIMEN.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **DCMIPPENC**: DCMIPP enable
Written at 1 to clear DCMIPPEN.

Bit 1 **LTDCENC**: LTDC enable
Written at 1 to clear LTDCEN.

Bit 0 Reserved, must be kept at reset value.

14.10.216 RCC bus sleep enable clear register (RCC_BUSLPENCR)

Address offset: 0x1284

Reset value: 0x0000 0000

This register is used to enable the RCC ACLKN bus in Sleep mode (each bit of this register is AND-ed with the EN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

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

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

Bit 1 **ACLKNCLPENC**: ACLKNC enable in Sleep mode
Written at 1 to clear ACLKNCLPEN.

Bit 0 **ACLKNLPENC**: ACLKN enable in Sleep mode
Written at 1 to clear ACLKNLPEN.

14.10.217 RCC miscellaneous sleep enable clear register (RCC_MISCLPENCR)

Address offset: 0x1288

Reset value: 0x0000 0000

This register is used to enable the RCC DBG miscellaneous in Sleep mode (each bit of this register is AND-ed with the EN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

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

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **PERLPENC**: PER enable in Sleep mode
Written at 1 to clear PERLPEN.

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **XSPIPHYCOMPLPENC**: XSPIPHYCOMP enable in Sleep mode
Written at 1 to clear XSPIPHYCOMPLPEN.

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **DBGLPENC**: DBG enable in Sleep mode
Written at 1 to clear DBGLPEN.

14.10.218 RCC memory sleep enable clear register (RCC_MEMLPENCR)

Address offset: 0x128C

Reset value: 0x0000 0000

This register is used to enable the RCC AXISRAM3 memory in Sleep mode (each bit of this register is AND-ed with the EN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	BOOTROMLPENC	VENCRAMLPENC	CACHEAXIRAMLPENC	FLEXRAMLPENC	AXISRAM2LPENC	AXISRAM1LPENC	BKPSRAMLPENC	AHBSRAM2LPENC	AHBSRAM1LPENC	AXISRAM6LPENC	AXISRAM5LPENC	AXISRAM4LPENC	AXISRAM3LPENC
			w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **BOOTROMLPENC**: BootROM enable in Sleep mode
Written at 1 to clear BOOTROMLPEN.

Bit 11 **VENCRAMLPENC**: VENCRAM enable in Sleep mode
Written at 1 to clear VENCRAMLPEN.

Bit 10 **CACHEAXIRAMLPENC**: CACHEAXIRAM enable in Sleep mode
Written at 1 to clear CACHEAXIRAMLPEN.

Bit 9 **FLEXRAMLPENC**: FLEXRAM enable in Sleep mode
Written at 1 to clear FLEXRAMLPEN.

Bit 8 **AXISRAM2LPENC**: AXISRAM2 enable in Sleep mode
Written at 1 to clear AXISRAM2LPEN.

Bit 7 **AXISRAM1LPENC**: AXISRAM1 enable in Sleep mode
Written at 1 to clear AXISRAM1LPEN.

Bit 6 **BKPSRAMLPENC**: BKPSRAM enable in Sleep mode
Written at 1 to clear BKPSRAMLPEN.

Bit 5 **AHBSRAM2LPENC**: AHBSRAM2 enable in Sleep mode
Written at 1 to clear AHBSRAM2LPEN.

Bit 4 **AHBSRAM1LPENC**: AHBSRAM1 enable in Sleep mode
Written at 1 to clear AHBSRAM1LPEN.

Bit 3 **AXISRAM6LPENC**: AXISRAM6 enable in Sleep mode
Written at 1 to clear AXISRAM6LPEN.

Bit 2 **AXISRAM5LPENC**: AXISRAM5 enable in Sleep mode
Written at 1 to clear AXISRAM5LPEN.

Bit 1 **AXISRAM4LPENC**: AXISRAM4 enable in Sleep mode
Written at 1 to clear AXISRAM4LPEN.

Bit 0 **AXISRAM3LPENC**: AXISRAM3 enable in Sleep mode
Written at 1 to clear AXISRAM3LPEN.

14.10.219 RCC AHB1 sleep enable clear register (RCC_AHB1LPENCR)

Address offset: 0x1290

Reset value: 0x0000 0000

This register is used to enable the RCC AHB1 in Sleep mode (each bit of this register is AND-ed with the EN bit).It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC12LPENC	GPDMA1LPENC	Res.	Res.	Res.	Res.
										w	w				

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

Bit 5 **ADC12LPENC**: ADC12 enable in Sleep mode
Written at 1 to clear ADC12LPEN.

Bit 4 **GPDMA1LPENC**: GPDMA1 enable in Sleep mode
Written at 1 to clear GPDMA1LPEN.

Bits 3:0 Reserved, must be kept at reset value.

14.10.220 RCC AHB2 sleep enable clear register (RCC_AHB2LPENCR)

Address offset: 0x1294

Reset value: 0x0000 0000

This register is used to enable the RCC AHB2 in Sleep mode (each bit of this register is AND-ed with the EN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADF1LPENC	MDF1LPENC
														w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	RAMCFGLPENC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			w												

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

Bit 17 **ADF1LPENC**: ADF1 enable in Sleep mode
Written at 1 to clear ADF1LPEN.

Bit 16 **MDF1LPENC**: MDF1 enable in Sleep mode
Written at 1 to clear MDF1LPEN.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **RAMCFGLPENC**: RAMCFG enable in Sleep mode
Written at 1 to clear RAMCFGLPEN.

Bits 11:0 Reserved, must be kept at reset value.

14.10.221 RCC AHB3 sleep enable clear register (RCC_AHB3LPENCR)

Address offset: 0x1298

Reset value: 0x0000 0000

This register is used to enable the RCC RNG AHB3 in Sleep mode (each bit of this register is AND-ed with the EN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RISAFLPENC	Res.	Res.	Res.	IACLPENC	RIFSCLPENC	PKALPENC	Res.	Res.	Res.	SAESLPENC	Res.	CRYPLPENC	HASHLPENC	RNGLPENC
	w				w	w	w				w		w	w	w

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **RISAFLPENC**: RISAF enable in Sleep mode
Written at 1 to clear RISAFLPEN.

Bits 13:11 Reserved, must be kept at reset value.

Bit 10 **IACLPENC**: IAC enable in Sleep mode
Written at 1 to clear IACLPEN.

Bit 9 **RIFSCLPENC**: RIFSC enable in Sleep mode
Written at 1 to clear RIFSCLPEN.

Bit 8 **PKALPENC**: PKA enable in Sleep mode
Written at 1 to clear PKALPEN.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **SAESLPENC**: SAES enable in Sleep mode
Written at 1 to clear SAESLPEN.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CRYPLPENC**: CRYP enable in Sleep mode
Written at 1 to clear CRYPLPEN.

Bit 1 **HASHLPENC**: HASH enable in Sleep mode
Written at 1 to clear HASHLPEN.

Bit 0 **RNGLPENC**: RNG enable in Sleep mode
Written at 1 to clear RNGLPEN.

14.10.222 RCC AHB4 sleep enable clear register (RCC_AHB4LPENCR)

Address offset: 0x129C

Reset value: 0x0000 0000

This register is used to enable the RCC GPIOA AHB4 in Sleep mode (each bit of this register is AND-ed with the EN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCLPENC	PWRLPENC	Res.	GPIOQLPENC
												w	w		w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPIOPLPENC	GPIOOLPENC	GPIONLPENC	Res.	Res.	Res.	Res.	Res.	GPIOHLPENC	GPIOGLPENC	GPIOFLPENC	GPIOELPENC	GPIODLPENC	GPIOCLPENC	GPIOBLPENC	GPIOALPENC
w	w	w						w	w	w	w	w	w	w	w

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **CRCLPENC**: CRC enable in Sleep mode
Written at 1 to clear CRCLPEN.

Bit 18 **PWRLPENC**: PWR enable in Sleep mode
Written at 1 to clear PWRLPEN.

Bit 17 Reserved, must be kept at reset value.

Bit 16 **GPIOQLPENC**: GPIO Q enable in Sleep mode
Written at 1 to clear GPIOQLPEN.

Bit 15 **GPIOPLPENC**: GPIO P enable in Sleep mode
Written at 1 to clear GPIOPLPEN.

Bit 14 **GPIOOLPENC**: GPIO O enable in Sleep mode
Written at 1 to clear GPIOOLPEN.

Bit 13 **GPIONLPENC**: GPIO N enable in Sleep mode
Written at 1 to clear GPIONLPEN.

Bits 12:8 Reserved, must be kept at reset value.

Bit 7 **GPIOHLPENC**: GPIO H enable in Sleep mode
Written at 1 to clear GPIOHLPEN.

Bit 6 **GPIOGLPENC**: GPIO G enable in Sleep mode
Written at 1 to clear GPIOGLPEN.

Bit 5 **GPIOFLPENC**: GPIO F enable in Sleep mode
Written at 1 to clear GPIOFLPEN.

Bit 4 **GPIOELPENC**: GPIO E enable in Sleep mode
Written at 1 to clear GPIOELPEN.

Bit 3 **GPIODLPENC**: GPIO D enable in Sleep mode
Written at 1 to clear GPIODLPEN.

Bit 2 **GPIOCLPENC**: GPIO C enable in Sleep mode
Written at 1 to clear GPIOCLPEN.

Bit 1 **GPIOBLPENC**: GPIO B enable in Sleep mode
Written at 1 to clear GPIOBLPEN.

Bit 0 **GPIOALPENC**: GPIO A enable in Sleep mode
Written at 1 to clear GPIOALPEN.

14.10.223 RCC AHB5 sleep enable clear register (RCC_AHB5LPENCR)

Address offset: 0x12A0

Reset value: 0x0000 0000

This register is used to enable the RCC HPDMA1 AHB5 in Sleep mode (each bit of this register is AND-ed with the EN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NPULPENC	CACHEAXILPENC	OTG2LPENC	OTGPHY2LPENC	OTGPHY1LPENC	OTG1LPENC	ETH1LPENC	ETH1RXLPENC	ETH1TXLPENC	ETH1MACLPENC	Res.	GPU2DLPENC	GFXMMULPENC	MCE4LPENC	XSPI3LPENC	MCE3LPENC
w	w	w	w	w	w	w	w	w	w		w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCE2LPENC	MCE1LPENC	XSPIMLPENC	XSPI2LPENC	Res.	Res.	Res.	SDMMC1LPENC	SDMMC2LPENC	PSSILPENC	XSPI1LPENC	FMCLPENC	JPEGLPENC	Res.	DMA2DLPENC	HPDMA1LPENC
w	w	w	w				w	w	w	w	w	w		w	w

Bit 31 **NPULPENC**: NPU enable in Sleep mode
Written at 1 to clear NPULPEN.

Bit 30 **CACHEAXILPENC**: CACHEAXI enable in Sleep mode
Written at 1 to clear CACHEAXILPEN.

Bit 29 **OTG2LPENC**: OTG2 enable in Sleep mode
Written at 1 to clear OTG2LPEN.

Bit 28 **OTGPHY2LPENC**: OTGPHY2 enable in Sleep mode
Written at 1 to clear OTGPHY2LPEN.

Bit 27 **OTGPHY1LPENC**: OTGPHY1 enable in Sleep mode
Written at 1 to clear OTGPHY1LPEN.

Bit 26 **OTG1LPENC**: OTG1 enable in Sleep mode
Written at 1 to clear OTG1LPEN.

Bit 25 **ETH1LPENC**: ETH1 enable in Sleep mode
Written at 1 to clear ETH1LPEN.

Bit 24 **ETH1RXLPENC**: ETH1RX enable in Sleep mode
Written at 1 to clear ETH1RXLPEN.

Bit 23 **ETH1TXLPENC**: ETH1TX enable in Sleep mode
Written at 1 to clear ETH1TXLPEN.

Bit 22 **ETH1MACLPENC**: ETH1MAC enable in Sleep mode
Written at 1 to clear ETH1MACLPEN.

Bit 21 Reserved, must be kept at reset value.

Bit 20 **GPU2DLPENC**: GPU2D enable in Sleep mode
Written at 1 to clear GPULPEN.

Bit 19 **GFXMMULPENC**: GFXMMU enable in Sleep mode
Written at 1 to clear GFXMMULPEN.

Bit 18 **MCE4LPENC**: MCE4 enable in Sleep mode
Written at 1 to clear MCE4LPEN.

Bit 17 **XSPI3LPENC**: XSPI3 enable in Sleep mode
Written at 1 to clear XSPI3LPEN.

Bit 16 **MCE3LPENC**: MCE3 enable in Sleep mode
Written at 1 to clear MCE3LPEN.

Bit 15 **MCE2LPENC**: MCE2 enable in Sleep mode
Written at 1 to clear MCE2LPEN.

Bit 14 **MCE1LPENC**: MCE1 enable in Sleep mode
Written at 1 to clear MCE1LPEN.

Bit 13 **XSPIMLPENC**: XSPIM enable in Sleep mode
Written at 1 to clear XSPIMLPEN.

Bit 12 **XSPI2LPENC**: XSPI2 enable in Sleep mode
Written at 1 to clear XSPI2LPEN.

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **SDMMC1LPENC**: SDMMC1 enable in Sleep mode
Written at 1 to clear SDMMC1LPEN.

Bit 7 **SDMMC2LPENC**: SDMMC2 enable in Sleep mode
Written at 1 to clear SDMMC2LPEN.

Bit 6 **PSSILPENC**: PSSI enable in Sleep mode
Written at 1 to clear PSSILPEN.

Bit 5 **XSPI1LPENC**: XSPI1 enable in Sleep mode
Written at 1 to clear XSPI1LPEN.

Bit 4 **FMCLPENC**: FMC enable in Sleep mode
Written at 1 to clear FMCLPEN.

Bit 3 **JPEGLPENC**: JPEG enable in Sleep mode
Written at 1 to clear JPEGLPEN.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **DMA2DLPENC**: DMA2D enable in Sleep mode
Written at 1 to clear DMA2DLPEN.

Bit 0 **HPDMA1LPENC**: HPDMA1 enable in Sleep mode
Written at 1 to clear HPDMA1LPEN.

14.10.224 RCC APB1L sleep enable clear register (RCC_APB1LLPENCR)

Address offset: 0x12A4

Reset value: 0x0000 0000

This register is used to enable the RCC APB1L in Sleep mode (each bit in this register is AND-ed with the EN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UART8 LPENC	UART7 LPENC	Res.	Res.	Res.	Res.	I3C2LP ENC	I3C1LP ENC	I2C3LP ENC	I2C2LP ENC	I2C1LP ENC	UART5 LPENC	UART4 LPENC	USART 3LPEN C	USART 2LPEN C	SPDIF RX1LP ENC
w	w					w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI3LP ENC	SPI2LP ENC	TIM11L PENC	TIM10L PENC	WWDG LPENC	Res.	LPTIM1 LPENC	TIM14L PENC	TIM13L PENC	TIM12L PENC	TIM7LP ENC	TIM6LP ENC	TIM5LP ENC	TIM4LP ENC	TIM3LP ENC	TIM2LP ENC
w	w	w	w	w		w	w	w	w	w	w	w	w	w	w

Bit 31 **UART8LPENC**: UART8 enable
Written at 1 to clear UART8LPEN.

Bit 30 **UART7LPENC**: UART7 enable
Written at 1 to clear UART7LPEN.

Bits 29:26 Reserved, must be kept at reset value.

Bit 25 **I3C2LPENC**: I3C2 enable
Written at 1 to clear I3C2LPEN.

Bit 24 **I3C1LPENC**: I3C1 enable
Written at 1 to clear I3C1LPEN.

Bit 23 **I2C3LPENC**: I2C3 enable
Written at 1 to clear I2C3LPEN.

Bit 22 **I2C2LPENC**: I2C2 enable
Written at 1 to clear I2C2LPEN.

Bit 21 **I2C1LPENC**: I2C1 enable
Written at 1 to clear I2C1LPEN.

Bit 20 **UART5LPENC**: UART5 enable
Written at 1 to clear UART5LPEN.

Bit 19 **UART4LPENC**: UART4 enable
Written at 1 to clear UART4LPEN.

Bit 18 **USART3LPENC**: USART3 enable
Written at 1 to clear USART3LPEN.

Bit 17 **USART2LPENC**: USART2 enable
Written at 1 to clear USART2LPEN.

Bit 16 **SPDIFRX1LPENC**: SPDIFRX1 enable
Written at 1 to clear SPDIFRX1LPEN.

Bit 15 **SPI3LPENC**: SPI3 enable
Written at 1 to clear SPI3LPEN.

Bit 14 **SPI2LPENC**: SPI2 enable
Written at 1 to clear SPI2LPEN.

Bit 13 **TIM11LPENC**: TIM11 enable
Written at 1 to clear TIM11LPEN.

Bit 12 **TIM10LPENC**: TIM10 enable
Written at 1 to clear TIM10LPEN.

Bit 11 **WWDGLPENC**: WWDG enable
Written at 1 to clear WWDGLPEN.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **LPTIM1LPENC**: LPTIM1 enable
Written at 1 to clear LPTIM1LPEN.

Bit 8 **TIM14LPENC**: TIM14 enable
Written at 1 to clear TIM14LPEN.

Bit 7 **TIM13LPENC**: TIM13 enable
Written at 1 to clear TIM13LPEN.

Bit 6 **TIM12LPENC**: TIM12 enable
Written at 1 to clear TIM12LPEN.

Bit 5 **TIM7LPENC**: TIM7 enable
Written at 1 to clear TIM7LPEN.

Bit 4 **TIM6LPENC**: TIM6 enable
Written at 1 to clear TIM6LPEN.

Bit 3 **TIM5LPENC**: TIM5 enable
Written at 1 to clear TIM5LPEN.

Bit 2 **TIM4LPENC**: TIM4 enable
Written at 1 to clear TIM4LPEN.

Bit 1 **TIM3LPENC**: TIM3 enable
Written at 1 to clear TIM3LPEN.

Bit 0 **TIM2LPENC**: TIM2 enable
Written at 1 to clear TIM2LPEN.

14.10.225 RCC APB1H sleep enable clear register (RCC_APB1HLPENCR)

Address offset: 0x12A8

Reset value: 0x0000 0000

This register is used to enable the RCC APB1H in Sleep mode (each bit of this register is AND-ed with the EN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD1 LPENC	Res.	Res.
													w		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCAN LPENC	Res.	Res.	MDIOS LPENC	Res.	Res.	Res.	Res.	Res.
							w			w					

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **UCPD1LPENC**: UCPD1 enable
Written at 1 to clear UCPD1LPEN.

Bits 17:9 Reserved, must be kept at reset value.

Bit 8 **FDCANLPENC**: FDCAN enable
Written at 1 to clear FDCANLPEN.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **MDIOSLPENC**: MDIOS enable
Written at 1 to clear MDIOSLPEN.

Bits 4:0 Reserved, must be kept at reset value.

14.10.226 RCC APB2 sleep enable clear register (RCC_APB2LPENCR)

Address offset: 0x12AC

Reset value: 0x0000 0000

This register is used to enable the RCC APB2 in Sleep mode (each bit of this register is AND-ed with the EN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI2LP ENC	SAI1LP ENC	SPI5LP ENC	TIM9LP ENC	TIM17L PENC	TIM16L PENC	TIM15L PENC
									w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIM18L PENC	Res.	SPI4LP ENC	SPI1LP ENC	Res.	Res.	Res.	Res.	USART 10LPE NC	UART9 LPENC	USART 6LPEN C	USART 1LPEN C	Res.	Res.	TIM8LP ENC	TIM1LP ENC
w		w	w					w	w	w	w			w	w

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **SAI2LPENC**: SAI2 enable
Written at 1 to clear SAI2LPEN.

Bit 21 **SAI1LPENC**: SAI1 enable
Written at 1 to clear SAI1LPEN.

Bit 20 **SPI5LPENC**: SPI5 enable
Written at 1 to clear SPI5LPEN.

Bit 19 **TIM9LPENC**: TIM9 enable
Written at 1 to clear TIM9LPEN.

Bit 18 **TIM17LPENC**: TIM17 enable
Written at 1 to clear TIM17LPEN.

Bit 17 **TIM16LPENC**: TIM16 enable
Written at 1 to clear TIM16LPEN.

Bit 16 **TIM15LPENC**: TIM15 enable
Written at 1 to clear TIM15LPEN.

Bit 15 **TIM18LPENC**: TIM18 enable
Written at 1 to clear TIM18LPEN.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **SPI4LPENC**: SPI4 enable
Written at 1 to clear SPI4LPEN.

Bit 12 **SPI1LPENC**: SPI1 enable
Written at 1 to clear SPI1LPEN.

Bits 11:8 Reserved, must be kept at reset value.

Bit 7 **USART10LPENC**: USART10 enable
Written at 1 to clear USART10LPEN.

Bit 6 **UART9LPENC**: UART9 enable
Written at 1 to clear UART9LPEN.

Bit 5 **USART6LPENC**: USART6 enable
Written at 1 to clear USART6LPEN.

Bit 4 **USART1LPENC**: USART1 enable
Written at 1 to clear USART1LPEN.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **TIM8LPENC**: TIM8 enable
Written at 1 to clear TIM8LPEN.

Bit 0 **TIM1LPENC**: TIM1 enable
Written at 1 to clear TIM1LPEN.

14.10.227 RCC APB3 sleep enable clear register (RCC_APB3LPENCR)

Address offset: 0x12B0

Reset value: 0x0000 0000

This register is used to enable the RCC- APB3 in Sleep mode (each bit of this register is AND-ed with the EN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

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

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

Bit 2 **DFTLPENC**: DFT enable
Written at 1 to clear DFTLPEN.

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

14.10.228 RCC APB4L sleep enable clear register (RCC_APB4LLPENCR)

Address offset: 0x12B4

Reset value: 0x0000 0000

This register is used to enable the RCC APB4L in Sleep mode (each bit of this register is AND-ed with the EN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCAPBLPENC	RTCLPENC
														w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VREFBUFLPENC	Res.	Res.	LPTIM5LPENC	LPTIM4LPENC	LPTIM3LPENC	LPTIM2LPENC	Res.	I2C4LPENC	Res.	SPI6LPENC	Res.	LPUART1LPENC	HDPLPENC	Res.	Res.
w			w	w	w	w		w		w		w	w		

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

Bit 17 **RTCAPBLPENC**: RTCAPB enable
Written at 1 to clear RTCAPBLPEN.

Bit 16 **RTCLPENC**: RTC enable
Written at 1 to clear RTCLPEN.

Bit 15 **VREFBUFLPENC**: VREFBUF enable
Written at 1 to clear VREFBUFLPEN.

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **LPTIM5LPENC**: LPTIM5 enable
Written at 1 to clear LPTIM5LPEN.

Bit 11 **LPTIM4LPENC**: LPTIM4 enable
Written at 1 to clear LPTIM4LPEN.

Bit 10 **LPTIM3LPENC**: LPTIM3 enable
Written at 1 to clear LPTIM3LPEN.

Bit 9 **LPTIM2LPENC**: LPTIM2 enable
Written at 1 to clear LPTIM2LPEN.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **I2C4LPENC**: I2C4 enable
Written at 1 to clear I2C4LPEN.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **SPI6LPENC**: SPI6 enable
Written at 1 to clear SPI6LPEN.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **LPUART1LPENC**: LPUART1 enable
Written at 1 to clear LPUART1LPEN.

Bit 2 **HDPLPENC**: HDP enable
Written at 1 to clear HDPLPEN.

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

14.10.229 RCC APB4H sleep enable clear register (RCC_APB4HLPENCR)

Address offset: 0x12B8

Reset value: 0x0000 0000

This register is used to enable the RCC APB4H in Sleep mode (each bit of this register is AND-ed with the EN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

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

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

Bit 2 **DTSLPENC**: DTS enable
Written at 1 to clear DTSLPEN.

Bit 1 **BSECLPENC**: BSEC enable
Written at 1 to clear BSECLPEN.

Bit 0 **SYSCFGLPENC**: SYSCFG enable
Written at 1 to clear SYSCFGLPEN.

14.10.230 RCC APB5 sleep enable clear register (RCC_APB5LPENCR)

Address offset: 0x12BC

Reset value: 0x0000 0000

This register is used to enable the RCC- APB5 in Sleep mode (each bit in this register is AND-ed with the EN bit). It is reset by sys_rstn, and is in the V_{CORE} voltage domain.

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

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CSILPENC**: CSI sleep enable
Written at 1 to clear CSILPEN.

Bit 5 **VENCLPENC**: VENC sleep enable
Written at 1 to clear VENCLPEN.

Bit 4 **GFXTIMLPENC**: GFXTIM sleep enable
Written at 1 to clear GFXTIMLPEN.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **DCMIPPLPENC**: DCMIPP sleep enable
Written at 1 to clear DCMIPPLPEN.

Bit 1 **LTDCLPENC**: LTDC sleep enable
Written at 1 to clear LTDCLPEN.

Bit 0 Reserved, must be kept at reset value.

14.10.231 RCC oscillator privilege configuration clear register 0 (RCC_PRIVCFGCR0)

Address offset: 0x1784

Reset value: 0x0000 0000

This register is used to control the privileged access rights to the configuration register of the oscillators. It is reset by sys_rstn, and is in the V_{CORE} voltage domain. Each xxPRIV bit defines the privileged protection for the configuration registers of the oscillator: a write access is denied if the access is unprivileged while the respective bit is set.

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

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

Bit 4 **HSEPRIVC**: Privileged protection of HSE configuration bits (enable, ready, divider)
Written at 1 to clear HSEPRIV by secure privileged software only. It can be read by any software.

Bit 3 **HSIPRIVC**: Privileged protection of HSI configuration bits (enable, ready, divider)
Written at 1 to clear HSIPRIV by secure privileged software only. It can be read by any software.

Bit 2 **MSIPRIVC**: Privileged protection of MSI configuration bits (enable, ready, divider)
Written at 1 to clear MSIPRIV by secure privileged software only. It can be read by any software.

Bit 1 **LSEPRIVC**: Privileged protection of LSE configuration bits (enable, ready, divider)
Written at 1 to clear LSEPRIV by secure privileged software only. It can be read by any software.

Bit 0 **LSIPRIVC**: Privileged protection of LSI configuration bits (enable, ready, divider)
Written at 1 to clear LSIPRIV by secure privileged software only. It can be read by any software.

14.10.232 RCC oscillator public configuration clear register 0 (RCC_PUBCFGCR0)

Address offset: 0x178C

Reset value: 0x0000 0000

This register is used to control the public access rights to the configuration register of the oscillators. It is reset by sys_rstn and is in VCORE voltage domain. Each xxPUB bit defines the public protection for the configuration registers of the oscillator: a write access is denied if the access is non-public while the respective bit is set.

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

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

Bit 4 **HSEPUBC**: Public protection of HSE configuration bits (enable, ready, divider)
Written at 1 to clear HSEPUB by secure privileged software only. It can be read by any software.

Bit 3 **HSIPUBC**: Public protection of HSI configuration bits (enable, ready, divider)
Written at 1 to clear HSIPUB by secure privileged software only. It can be read by any software.

Bit 2 **MSIPUBC**: Public protection of MSI configuration bits (enable, ready, divider)
Written at 1 to clear MSIPUB by secure privileged software only. It can be read by any software.

Bit 1 **LSEPUBC**: Public protection of LSE configuration bits (enable, ready, divider)
Written at 1 to clear LSEPUB by secure privileged software only. It can be read by any software.

Bit 0 **LSIPUBC**: Public protection of LSI configuration bits (enable, ready, divider)
Written at 1 to clear LSIPUB by secure privileged software only. It can be read by any software.

14.10.233 RCC PLL privilege configuration clear register 1 (RCC_PRIVCFGCR1)

Address offset: 0x1794

Reset value: 0x0000 0000

This register is used to control the privileged access rights to the configuration register of the PLLs. It is reset by sys_rstn and is in VCORE voltage domain. Each xxPRIV bit defines the privileged protection for the configuration registers of the PLL: a write access is denied if the access is unprivileged while the respective bit is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL4PRIVC	PLL3PRIVC	PLL2PRIVC	PLL1PRIVC
												w	w	w	w

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

Bit 3 **PLL4PRIVC**: Privileged protection of PLL4 configuration bits (enable, ready, divider)
Written at 1 to clear PLL4PRIV by secure privileged software only. It can be read by any software.

Bit 2 **PLL3PRIVC**: Privileged protection of PLL3 configuration bits (enable, ready, divider)
Written at 1 to clear PLL3PRIV by secure privileged software only. It can be read by any software.

Bit 1 **PLL2PRIVC**: Privileged protection of PLL2 configuration bits (enable, ready, divider)
Written at 1 to clear PLL2PRIV by secure privileged software only. It can be read by any software.

Bit 0 **PLL1PRIVC**: Privileged protection of PLL1 configuration bits (enable, ready, divider)
Written at 1 to clear PLL1PRIV by secure privileged software only. It can be read by any software.

14.10.234 RCC PLL public configuration clear register 1 (RCC_PUBCFGCR1)

Address offset: 0x179C

Reset value: 0x0000 0000

This register is used to control the public access rights to the configuration register of the PLLs. It is reset by sys_rstn, and is in the V_{CORE} voltage domain. Each xxPUB bit defines the public protection for the configuration registers of the PLL: a write access is denied if the access is non-public while the respective bit is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL4PUBC	PLL3PUBC	PLL2PUBC	PLL1PUBC
												w	w	w	w

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

Bit 3 **PLL4PUBC**: Public protection of PLL4 configuration bits (enable, ready, divider)
Written at 1 to clear PLL4PUB by secure privileged software only. It can be read by any software.

Bit 2 **PLL3PUBC**: Public protection of PLL3 configuration bits (enable, ready, divider)
Written at 1 to clear PLL3PUB by secure privileged software only. It can be read by any software.

Bit 1 **PLL2PUBC**: Public protection of te PLL2 configuration bits (enable, ready, divider)
Written at 1 to clear PLL2PUB by secure privileged software only. It can be read by any software.

Bit 0 **PLL1PUBC**: Public protection of th PLL1 configuration bits (enable, ready, divider)
Written at 1 to clear PLL1PUB by secure privileged software only. It can be read by any software.

14.10.235 RCC divider privilege configuration clear register 2 (RCC_PRIVCFGCR2)

Address offset: 0x17A4

Reset value: 0x0000 0000

This register is used to control the privilege access rights to the configuration register of the dividers. It is reset by sys_rstn, and is in the V_{CORE} voltage domain. Each xxPRIV bit defines the privileged protection for the configuration registers of the divider: a write access is denied if the access is unprivileged while the respective bit is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC20PRIVC	IC19PRIVC	IC18PRIVC	IC17PRIVC
												w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC16PRIVC	IC15PRIVC	IC14PRIVC	IC13PRIVC	IC12PRIVC	IC11PRIVC	IC10PRIVC	IC9PRIVC	IC8PRIVC	IC7PRIVC	IC6PRIVC	IC5PRIVC	IC4PRIVC	IC3PRIVC	IC2PRIVC	IC1PRIVC
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **IC20PRIVC**: Privileged protection of IC20 configuration bits (enable, ready, divider)
Written at 1 to clear IC20PRIV by secure privileged software only. It can be read by any software.

Bit 18 **IC19PRIVC**: Privileged protection of IC19 configuration bits (enable, ready, divider)
Written at 1 to clear IC19PRIV by secure privileged software only. It can be read by any software.

Bit 17 **IC18PRIVC**: Privileged protection of IC18 configuration bits (enable, ready, divider)
Written at 1 to clear IC18PRIV by secure privileged software only. It can be read by any software.

Bit 16 **IC17PRIVC**: Privileged protection of IC17 configuration bits (enable, ready, divider)
Written at 1 to clear IC17PRIV by secure privileged software only. It can be read by any software.

Bit 15 **IC16PRIVC**: Privileged protection of IC16 configuration bits (enable, ready, divider)
Written at 1 to clear IC16PRIV by secure privileged software only. It can be read by any software.

Bit 14 **IC15PRIVC**: Privileged protection of IC15 configuration bits (enable, ready, divider)
Written at 1 to clear IC15PRIV by secure privileged software only. It can be read by any software.

Bit 13 **IC14PRIVC**: Privileged protection of IC14 configuration bits (enable, ready, divider)
Written at 1 to clear IC14PRIV by secure privileged software only. It can be read by any software.

Bit 12 **IC13PRIVC**: Privileged protection of IC13 configuration bits (enable, ready, divider)
Written at 1 to clear IC13PRIV by secure privileged software only. It can be read by any software.

Bit 11 **IC12PRIVC**: Privileged protection of IC12 configuration bits (enable, ready, divider)
Written at 1 to clear IC12PRIV by secure privileged software only. It can be read by any software.

Bit 10 **IC11PRIVC**: Privileged protection of IC11 configuration bits (enable, ready, divider)
Written at 1 to clear IC11PRIV by secure privileged software only. It can be read by any software.

Bit 9 **IC10PRIVC**: Privileged protection of IC10 configuration bits (enable, ready, divider)
Written at 1 to clear IC10PRIV by secure privileged software only. It can be read by any software.

Bit 8 **IC9PRIVC**: Privileged protection of IC9 configuration bits (enable, ready, divider)
Written at 1 to clear IC9PRIV by secure privileged software only. It can be read by any software.

Bit 7 **IC8PRIVC**: Privileged protection of IC8 configuration bits (enable, ready, divider)
Written at 1 to clear IC8PRIV by secure privileged software only. It can be read by any software.

Bit 6 **IC7PRIVC**: Privileged protection of IC7 configuration bits (enable, ready, divider)
Written at 1 to clear IC7PRIV by secure privileged software only. It can be read by any software.

Bit 5 **IC6PRIVC**: Privileged protection of IC6 configuration bits (enable, ready, divider)
Written at 1 to clear IC6PRIV by secure privileged software only. It can be read by any software.

Bit 4 **IC5PRIVC**: Privileged protection of the IC5 configuration bits (enable, ready, divider).
Written at 1 to clear IC5PRIV by secure privileged software only. It can read by any software.

Bit 3 **IC4PRIVC**: Privileged protection of the IC4 configuration bits (enable, ready, divider).
Written at 1 to clear IC4PRIV by secure privileged software only. It can read by any software.

Bit 2 **IC3PRIVC**: Privileged protection of the IC3 configuration bits (enable, ready, divider).
Written at 1 to clear IC3PRIV by secure privileged software only. It can read by any software.

Bit 1 **IC2PRIVC**: Privileged protection of the IC2 configuration bits (enable, ready, divider).
Written at 1 to clear IC2PRIV by secure privileged software only. It can read by any software.

Bit 0 **IC1PRIVC**: Privileged protection of the IC1 configuration bits (enable, ready, divider).
Written at 1 to clear IC1PRIV by secure privileged software only. It can read by any software.

14.10.236 RCC divider public configuration clear register 2 (RCC_PUBCFGCR2)

Address offset: 0x17AC

Reset value: 0x0000 0000

This register is used to control the public access rights to the configuration register of the dividers. It is reset by sys_rstn, and is in the V_{CORE} voltage domain.Each xxPUB bit defines the public protection for the configuration registers of the divider: a write access is denied if the access is non-public while the respective bit here is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC20PUBC	IC19PUBC	IC18PUBC	IC17PUBC
												w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC16PUBC	IC15PUBC	IC14PUBC	IC13PUBC	IC12PUBC	IC11PUBC	IC10PUBC	IC9PUBC	IC8PUBC	IC7PUBC	IC6PUBC	IC5PUBC	IC4PUBC	IC3PUBC	IC2PUBC	IC1PUBC
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **IC20PUBC**: Public protection of IC20 configuration bits (enable, ready, divider)
Written at 1 to clear IC20PUB by secure privileged software only. It can be read by any software.

Bit 18 **IC19PUBC**: Public protection of IC19 configuration bits (enable, ready, divider)
Written at 1 to clear IC19PUB by secure privileged software only. It can be read by any software.

Bit 17 **IC18PUBC**: Public protection of IC18 configuration bits (enable, ready, divider)
Written at 1 to clear IC18PUB by secure privileged software only. It can be read by any software.

Bit 16 **IC17PUBC**: Public protection of IC17 configuration bits (enable, ready, divider)
Written at 1 to clear IC17PUB by secure privileged software only. It can be read by any software.

Bit 15 **IC16PUBC**: Public protection of IC16 configuration bits (enable, ready, divider)
Written at 1 to clear IC16PUB by secure privileged software only. It can be read by any software.

Bit 14 **IC15PUBC**: Public protection of IC15 configuration bits (enable, ready, divider)
Written at 1 to clear IC15PUB by secure privileged software only. It can be read by any software.

Bit 13 **IC14PUBC**: Public protection of IC14 configuration bits (enable, ready, divider)
Written at 1 to clear IC14PUB by secure privileged software only. It can be read by any software.

Bit 12 **IC13PUBC**: Public protection of IC13 configuration bits (enable, ready, divider)
Written at 1 to clear IC13PUB by secure privileged software only. It can be read by any software.

Bit 11 **IC12PUBC**: Public protection of IC12 configuration bits (enable, ready, divider)
Written at 1 to clear IC12PUB by secure privileged software only. It can be read by any software.

Bit 10 **IC11PUBC**: Public protection of IC11 configuration bits (enable, ready, divider)
Written at 1 to clear IC11PUB by secure privileged software only. It can be read by any software.

Bit 9 **IC10PUBC**: Public protection of IC10 configuration bits (enable, ready, divider)
Written at 1 to clear IC10PUB by secure privileged software only. It can be read by any software.

Bit 8 **IC9PUBC**: Public protection of IC9 configuration bits (enable, ready, divider)
Written at 1 to clear IC9PUB by secure privileged software only. It can be read by any software.

Bit 7 **IC8PUBC**: Public protection of IC8 configuration bits (enable, ready, divider)
Written at 1 to clear IC8PUB by secure privileged software only. It can be read by any software.

Bit 6 **IC7PUBC**: Public protection of IC7 configuration bits (enable, ready, divider)
Written at 1 to clear IC7PUB by secure privileged software only. It can be read by any software.

Bit 5 **IC6PUBC**: Public protection of IC6 configuration bits (enable, ready, divider)
Written at 1 to clear IC6PUB by secure privileged software only. It can be read by any software.

Bit 4 **IC5PUBC**: Public protection of IC5 configuration bits (enable, ready, divider)
Written at 1 to clear IC5PUB by secure privileged software only. It can be read by any software.

Bit 3 **IC4PUBC**: Public protection of IC4 configuration bits (enable, ready, divider)
Written at 1 to clear IC4PUB by secure privileged software only. It can be read by any software.

Bit 2 **IC3PUBC**: Public protection of IC3 configuration bits (enable, ready, divider)
Written at 1 to clear IC3PUB by secure privileged software only. It can be read by any software.

Bit 1 **IC2PUBC**: Public protection of IC2 configuration bits (enable, ready, divider)
Written at 1 to clear IC2PUB by secure privileged software only. It can be read by any software.

Bit 0 **IC1PUBC**: Public protection of IC1 configuration bits (enable, ready, divider)
Written at 1 to clear IC1PUB by secure privileged software only. It can be read by any software.

14.10.237 RCC system privilege configuration clear register 3 (RCC_PRIVCFGCR3)

Address offset: 0x17B4

Reset value: 0x0000 0000

This register is used to control the privilege access rights to the configuration register of the system. It is reset by sys_rstn, and is in the V_{CORE} voltage domain. Each xxPRIV bit defines the privileged protection for the configuration registers of the system: a write access is denied if the access is unprivileged while the respective bit is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFTPRIVC	RSTPRIVC	INTPRIVC	PERPRIVC	BUSPRIVC	SYSPRIVC	MODPRIVC
									w	w	w	w	w	w	w

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **DFTPRIVC**: Privileged protection of DFT configuration bits (enable, ready, divider)
Written at 1 to clear DFTPRIV by secure privileged software only. It can be read by any software.

Bit 5 **RSTPRIVC**: Privileged protection of RST configuration bits (enable, ready, divider)
Written at 1 to clear RSTPRIV by secure privileged software only. It can be read by any software.

Bit 4 **INTPRIVC**: Privileged protection of INT configuration bits (enable, ready, divider)
Written at 1 to clear INTPRIV by secure privileged software only. It can be read by any software.

Bit 3 **PERPRIVC**: Privileged protection of PER configuration bits (enable, ready, divider)
Written at 1 to clear PERPRIV by secure privileged software only. It can be read by any software.

Bit 2 **BUSPRIVC**: Privileged protection of BUS configuration bits (enable, ready, divider)
Written at 1 to clear BUSPRIV by secure privileged software only. It can be read by any software.

Bit 1 **SYSPRIVC**: Privileged protection of SYS configuration bits (enable, ready, divider)
Written at 1 to clear SYSPRIV by secure privileged software only. It can be read by any software.

Bit 0 **MODPRIVC**: Privileged protection of MOD configuration bits (enable, ready, divider)
Written at 1 to clear MODPRIV by secure privileged software only. It can be read by any software.

14.10.238 RCC system public configuration clear register 3 (RCC_PUBCFGCR3)

Address offset: 0x17BC

Reset value: 0x0000 0000

This register is used to control the public access rights to the configuration register of the system. It is reset by sys_rstn, and is in the V_{CORE} voltage domain. Each xxPUB defines the public protection for the configuration registers of the system: a write access is denied if the access is non-public while the respective bit is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTPUBC	INTPUBC	PERPUBC	BUSPUBC	SYSPUBC	MODPUBC
										w	w	w	w	w	w

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

Bit 5 **RSTPUBC**: Public protection of RST configuration bits (enable, ready, divider)
Written at 1 to clear RSTPUB by secure privileged software only. It can be read by any software.

Bit 4 **INTPUBC**: Public protection of INT configuration bits (enable, ready, divider)
Written at 1 to clear INTPUB by secure privileged software only. It can be read by any software.

Bit 3 **PERPUBC**: Public protection of PER configuration bits (enable, ready, divider)
Written at 1 to clear PERPUB by secure privileged software only. It can be read by any software.

Bit 2 **BUSPUBC**: Public protection of BUS configuration bits (enable, ready, divider)
Written at 1 to clear BUSPUB by secure privileged software only. It can be read by any software.

Bit 1 **SYSPUBC**: Public protection of SYS configuration bits (enable, ready, divider)
Written at 1 to clear SYSPUB by secure privileged software only. It can be read by any software.

Bit 0 **MODPUBC**: Public protection of MOD configuration bits (enable, ready, divider)
Written at 1 to clear MODPUB by secure privileged software only. It can be read by any software.

14.10.239 RCC privilege configuration clear register 4 (RCC_PRIVCFGCR4)

Address offset: 0x17C4

Reset value: 0x0000 0000

This register is used to control the privileged access rights to the configuration register of the buses. It is reset by sys_rstn, and is in the V_{CORE} voltage domain. Each xxPRIV bit defines the privileged protection for the configuration registers of each bus: a write access is denied if the access is unprivileged while the respective bit here is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	NOCPRIVC	APB5PRIVC	APB4PRIVC	APB3PRIVC	APB2PRIVC	APB1PRIVC	AHB5PRIVC	AHB4PRIVC	AHB3PRIVC	AHB2PRIVC	AHB1PRIVC	AHBMPRIVC	ACLKNCPRIVC	ACLKNPRIVC
		w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **NOCPRIVC**: Privileged protection of NOC configuration bits (enable, ready, divider)
Written at 1 to clear NOCPRIV by secure privileged software only. It can be read by any software.

Bit 12 **APB5PRIVC**: Privileged protection of APB5 configuration bits (enable, ready, divider)
Written at 1 to clear APB5PRIV by secure privileged software only. It can be read by any software.

Bit 11 **APB4PRIVC**: Privileged protection of APB4 configuration bits (enable, ready, divider)
Written at 1 to clear APB4PRIV by secure privileged software only. It can be read by any software.

Bit 10 **APB3PRIVC**: Privileged protection of APB3 configuration bits (enable, ready, divider)
Written at 1 to clear APB3PRIV by secure privileged software only. It can be read by any software.

Bit 9 **APB2PRIVC**: Privileged protection of APB2 configuration bits (enable, ready, divider)
Written at 1 to clear APB2PRIV by secure privileged software only. It can be read by any software.

Bit 8 **APB1PRIVC**: Privileged protection of APB1 configuration bits (enable, ready, divider)
Written at 1 to clear APB1PRIV by secure privileged software only. It can be read by any software.

Bit 7 **AHB5PRIVC**: Privileged protection of AHB5 configuration bits (enable, ready, divider)
Written at 1 to clear AHB5PRIV by secure privileged software only. It can be read by any software.

Bit 6 **AHB4PRIVC**: Privileged protection of AHB4 configuration bits (enable, ready, divider)
Written at 1 to clear AHB4PRIV by secure privileged software only. It can be read by any software.

Bit 5 **AHB3PRIVC**: Privileged protection of AHB3 configuration bits (enable, ready, divider)
Written at 1 to clear AHB3PRIV by secure privileged software only. It can be read by any software.

Bit 4 **AHB2PRIVC**: Privileged protection of AHB2 configuration bits (enable, ready, divider)
Written at 1 to clear AHB2PRIV by secure privileged software only. It can be read by any software.

Bit 3 **AHB1PRIVC**: Privileged protection of AHB1 configuration bits (enable, ready, divider)
Written at 1 to clear AHB1PRIV by secure privileged software only. It can be read by any software.

Bit 2 **AHBMPRIVC**: Privileged protection of AHBM configuration bits (enable, ready, divider)
Written at 1 to clear AHBMPRIV by secure privileged software only. It can be read by any software.

Bit 1 **ACLKNCPRIVC**: Privileged protection of ACLKNC configuration bits (enable, ready, divider)
Written at 1 to clear ACLKNCPRIV by secure privileged software only. It can read by any software.

Bit 0 **ACLKNPRIVC**: Privileged protection of ACLKN configuration bits (enable, ready, divider)
Written at 1 to clear ACLKNPRIV by secure privileged software only. It can read by any software.

14.10.240 RCC public configuration clear register 4 (RCC_PUBCFGCR4)

Address offset: 0x17CC

Reset value: 0x0000 0000

This register is used to control the public access rights to the configuration register of the buses. It is reset by sys_rstn, and is in the V_{CORE} voltage domain. Each xxPUB bit defines the public protection for the configuration registers of each bus: a write access is denied if the access is non-public while the respective bit is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	NOCP UBC	APB5P UBC	APB4P UBC	APB3P UBC	APB2P UBC	APB1P UBC	AHB5P UBC	AHB4P UBC	AHB3P UBC	AHB2P UBC	AHB1P UBC	AHBM PUBC	ACLKN CPUBC	ACLKN PUBC
		w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **NOCPUBC**: Public protection of NOC configuration bits (enable, ready, divider)
Written at 1 to clear NOCPUB by secure privileged software only. It can be read by any software.

Bit 12 **APB5PUBC**: Public protection of APB5 configuration bits (enable, ready, divider)
Written at 1 to clear APB5PUB by secure privileged software only. It can be read by any software.

Bit 11 **APB4PUBC**: Public protection of APB4 configuration bits (enable, ready, divider)
Written at 1 to clear APB4PUB by secure privileged software only. It can be read by any software.

Bit 10 **APB3PUBC**: Public protection of APB3 configuration bits (enable, ready, divider)
Written at 1 to clear APB3PUB by secure privileged software only. It can be read by any software.

Bit 9 **APB2PUBC**: Public protection of APB2 configuration bits (enable, ready, divider)
Written at 1 to clear APB2PUB by secure privileged software only. It can be read by any software.

Bit 8 **APB1PUBC**: Public protection of APB1 configuration bits (enable, ready, divider)
Written at 1 to clear APB1PUB by secure privileged software only. It can be read by any software.

Bit 7 **AHB5PUBC**: Public protection of AHB5 configuration bits (enable, ready, divider)
Written at 1 to clear AHB5PUB by secure privileged software only. It can be read by any software.

Bit 6 **AHB4PUBC**: Public protection of AHB4 configuration bits (enable, ready, divider)
Written at 1 to clear AHB4PUB by secure privileged software only. It can be read by any software.

Bit 5 **AHB3PUBC**: Public protection of AHB3 configuration bits (enable, ready, divider)
Written at 1 to clear AHB3PUB by secure privileged software only. It can be read by any software.

Bit 4 **AHB2PUBC**: Public protection of AHB2 configuration bits (enable, ready, divider)
Written at 1 to clear AHB2PUB by secure privileged software only. It can be read by any software.

Bit 3 **AHB1PUBC**: Public protection of AHB1 configuration bits (enable, ready, divider)
Written at 1 to clear AHB1PUB by secure privileged software only. It can be read by any software.

Bit 2 **AHBMPUBC**: Public protection of AHBM configuration bits (enable, ready, divider)
Written at 1 to clear AHBMPUB by secure privileged software only. It can be read by any software.

Bit 1 **ACLKNCPUBC**: Public protection of ACLKNC configuration bits (enable, ready, divider)
Written at 1 to clear ACLKNCPUB by secure privileged software only. It can be read by any software.

Bit 0 **ACLKNPUBC**: Public protection of ACLKN configuration bits (enable, ready, divider)
Written at 1 to clear ACLKNPUB by secure privileged software only. It can be read by any software.

14.10.241 RCC public configuration clear register 4 (RCC_PUBCFGCR5)

Address offset: 0x17D0

Reset value: 0x0000 0000

This register is used to control the public access rights to the configuration register of the SRAMs. It is reset by sys_rstn, and is in the V_{CORE} voltage domain. Each xxPUB bit defines the public protection for the configuration registers of each SRAM: a write access is denied if the access is non-public while the respective bit is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	VENCRAMPUBC	CACHEAXIRAMPUBC	FLEXRAMPUBC	AXISRAM2PUBC	AXISRAM1PUBC	BKPSRAMPUBC	AHBSRAM2PUBC	AHBSRAM1PUBC	AXISRAM6PUBC	AXISRAM5PUBC	AXISRAM4PUBC	AXISRAM3PUBC
				w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **VENCRAMPUBC**: Public protection of VENCRAM configuration bits (enable, ready, divider)
This bit is written to 1 to clear VENCRAMPUB by secure privileged software only. It can be read by any software.

Bit 10 **CACHEAXIRAMPUBC**: Public protection of CACHEAXIRAM configuration bits (enable, ready, divider)
This bit is written to 1 to clear CACHEEXIRAMPUB by secure privileged software only. It can be read by any software.

Bit 9 **FLEXRAMPUBC**: Public protection of FLEXRAM configuration bits (enable, ready, divider)
This bit is written to 1 to clear FLEXRAMPUB by secure privileged software only. It can be read by any software.

Bit 8 **AXISRAM2PUBC**: Public protection of AXISRAM2 configuration bits (enable, ready, divider)
This bit is written to 1 to clear AXISRAM2PUB by secure privileged software only. It can be read by any software.

Bit 7 **AXISRAM1PUBC**: Public protection of AXISRAM1 configuration bits (enable, ready, divider)
This bit is written to 1 to clear AXISRAM1PUB by secure privileged software only. It can be read by any software.

Bit 6 **BKPSRAMPUBC**: Public protection of BKPSRAM configuration bits (enable, ready, divider)
This bit is written to 1 to clear BKPSRAMPUB by secure privileged software only. It can be read by any software.

Bit 5 **AHBSRAM2PUBC**: Public protection of AHBSRAM2 configuration bits (enable, ready, divider)
This bit is written to 1 to clear AHBSRAM2PUB by secure privileged software only. It can be read by any software.

Bit 4 **AHBSRAM1PUBC**: Public protection of AHBSRAM1 configuration bits (enable, ready, divider)
This bit is written to 1 to clear AHBSRAM1PUB by secure privileged software only. It can be read by any software.

Bit 3 **AXISRAM6PUBC**: Public protection of AXISRAM6 configuration bits (enable, ready, divider)
This bit is written to 1 to clear AXISRAM6PUB by secure privileged software only. It can be read by any software.

Bit 2 **AXISRAM5PUBC**: Public protection of AXISRAM5 configuration bits (enable, ready, divider)
Written at 1 to clear AXISRAM5PUB by secure privileged software only. It can be read by any software.

Bit 1 **AXISRAM4PUBC**: Public protection of AXISRAM4 configuration bits (enable, ready, divider)
Written at 1 to clear AXISRAM4PUB by secure privileged software only. It can be read by any software.

Bit 0 **AXISRAM3PUBC**: Public protection of AXISRAM3 configuration bits (enable, ready, divider)
Written at 1 to clear AXISRAM3PUB by secure privileged software only. It can be read by any software.

14.10.242 RCC register map

Table 77. RCC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	RCC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL4ON	PLL3ON	PLL2ON	PLL1ON	Res.	Res.	Res.	HSEON	HSION	MSION	LSEON	LSION
	Reset value																					0	0	0	0				0	1	0	0	0
0x004	RCC_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL4RDY	PLL3RDY	PLL2RDY	PLL1RDY	Res.	Res.	Res.	HSERDY	HSIRDY	MSIRDY	LSERDY	LSIRDY
	Reset value																					0	0	0	0				0	0	0	0	0
0x008	RCC_STOPCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSISTOPEN	MSISTOPEN
	Reset value																															1	0
0x00C-0x01C	Reserved	Reserved																															
0x020	RCC_CFGR1	Res.	Res.	SYSSWS [1:0]		Res.	Res.	SYSSW [1:0]		Res.	Res.	CPUSWS [1:0]		Res.	Res.	CPUSW [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STOPWUCK
	Reset value			0	0			0	0			0	0			0	0																0
0x024	RCC_CFGR2	Res.	Res.	Res.	Res.	Res.	Res.	TIMPRE [1:0]		Res.	HPRE [2:0]			Res.	PPRE5 [2:0]			Res.	PPRE4 [2:0]			Res.	Res.	Res.	Res.	Res.	PPRE2 [2:0]			Res.	PPRE1 [2:0]		
	Reset value							0	0		0	0	1		0	0	0		0	0	0						0	0	0		0	0	0
0x028	Reserved	Reserved																															
0x02C	RCC_BDCR	VSWRST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0																															
0x030	RCC_HWRSR	Res.	LPWRRSTF	Res.	WWDGRSTF	Res.	IWDGRSTF	Res.	SFTRSTF	PORRSTF	PINRSTF	BORRSTF	Res.	Res.	Res.	LCKRSTF	RMVF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value		0		0		0		0	1	0	1				0	0																
0x034	RCC_RSR	Res.	LPWRRSTF	Res.	WWDGRSTF	Res.	IWDGRSTF	Res.	SFTRSTF	PORRSTF	PINRSTF	BORRSTF	Res.	Res.	Res.	LCKRSTF	RMVF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value		0		0		0		0	1	0	1				0	0																
0x038-0x03C	Reserved	Reserved																															

Table 77. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x040	RCC_LSECFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSEDRV[1:0]		LSEGFON	LSEEXT	LSEBYP	Res.	Res.	Res.	Res.	Res.	LSECSSD	LSECSSRA	LSECSSON	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value													0	0	0	0	0						0	0	0							
0x044	RCC_MSICFGR	Res.	MSICAL[7:0]								Res.	Res.	MSITRIM[4:0]						Res.	Res.	Res.	Res.	Res.	MSIFREQSE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value		0	0	0	0	0	0	0	0			0	0	0	0	0							0									
0x048	RCC_HSICFGR	HSICAL[8:0]									HSITRIM[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSIDIV [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								0	0							
0x04C	RCC_HSIMCR	HSIMONEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSIDEV[5:0]						Res.	Res.	Res.	Res.	Res.	HSIREF[10:0]										
	Reset value	0										0	1	1	1	1	1						1	1	1	1	0	1	0	0	0	0	1
0x050	RCC_HSIMSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSIVAL[10:0]										
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x054	RCC_HSECFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSEDRV [1:0]		HSEGFON	HSEEXT	HSEBYP	HSECSSBPRE [3:0]				HSECSSBYP	HSECSSD	HSECSSRA	HSECSSON	HSEDIV2SEL	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value													0	0	0	0	0	0	0	0	1	0	0	0	0	0						
0x058-0x07C	Reserved	Reserved																															
0x080	RCC_PLL1CFGR1	Res.	PLL1SEL [2:0]			PLL1BYP	Res.	PLL1DIVM[5:0]						PLL1DIVN[11:0]													Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value		0	0	0	1		0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0	1							
0x084	RCC_PLL1CFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL1DIVNFRAC[23:0]																							
	Reset value									1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x088	RCC_PLL1CFGR3	Res.	PLL1PDIVEN	PLL1PDIV1 [2:0]			PLL1PDIV2 [2:0]			Res.	Res.	Res.	PLL1MODSPR [4:0]					Res.	Res.	Res.	Res.	PLL1MODDIV [3:0]				Res.	Res.	Res.	PLL1MODSPRDW	PLL1MODDSEN	PLL1MODSSDIS	PLL1DACEN	PLL1MODSSRST
	Reset value		1	0	0	1	0	0	1				0	0	0	0	0					0	0	0	0				0	1	1	0	1
0x08C	Reserved	Reserved																															
0x090	RCC_PLL2CFGR1	Res.	PLL2SEL [2:0]			PLL2BYP	Res.	PLL2DIVM[5:0]						PLL2DIVN[11:0]													Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value		0	0	0	1		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							

Table 77. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x094	RCC_PLL2CFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL2DIVNFRAC[23:0]																							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x098	RCC_PLL2CFGR3	Res.	PLL2PDIVEN	PLL2PDIV1 [2:0]			PLL2PDIV2 [2:0]			Res.	Res.	Res.	PLL2MODSPR [4:0]					Res.	Res.	Res.	Res.	PLL2MODDIV [3:0]				Res.	Res.	Res.	PLL2MODSPRDW	PLL2MODDSEN	PLL2MODSSDIS	PLL2DACEN	PLL2MODSSRST
	Reset value		1	0	0	1	0	0	1				0	0	0	0	0					0	0	0	0				0	0	1	0	1
0x09C	Reserved	Reserved																															
0x0A0	RCC_PLL3CFGR1	Res.	PLL3SEL [2:0]			PLL3BYP	Res.	PLL3DIVM[5:0]								PLL3DIVN[11:0]										Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value		0	0	0	1		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								
0x0A4	RCC_PLL3CFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL3DIVNFRAC[23:0]																							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A8	RCC_PLL3CFGR3	Res.	PLL3PDIVEN	PLL3PDIV1 [2:0]			PLL3PDIV2 [2:0]			Res.	Res.	Res.	PLL3MODSPR [4:0]					Res.	Res.	Res.	Res.	PLL3MODDIV [3:0]				Res.	Res.	Res.	PLL3MODSPRDW	PLL3MODDSEN	PLL3MODSSDIS	PLL3DACEN	PLL3MODSSRST
	Reset value		1	0	0	1	0	0	1				0	0	0	0	0					0	0	0	0				0	0	1	0	1
0x0AC	Reserved	Reserved																															
0x0B0	RCC_PLL4CFGR1	Res.	PLL4SEL [2:0]			PLL4BYP	Res.	PLL4DIVM[5:0]								PLL4DIVN[11:0]										Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value		0	0	0	1		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								
0x0B4	RCC_PLL4CFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL4DIVNFRAC[23:0]																							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B8	RCC_PLL4CFGR3	Res.	PLL4PDIVEN	PLL4PDIV1 [2:0]			PLL4PDIV2 [2:0]			Res.	Res.	Res.	PLL4MODSPR [4:0]					Res.	Res.	Res.	Res.	PLL4MODDIV [3:0]				Res.	Res.	Res.	PLL4MODSPRDW	PLL4MODDSEN	PLL4MODSSDIS	PLL4DACEN	PLL4MODSSRST
	Reset value		1	0	0	1	0	0	1				0	0	0	0	0					0	0	0	0				0	0	1	0	1
0x0BC-0x0C0	Reserved	Reserved																															
0x0C4	RCC_IC1CFGR	Res.	Res.	IC1SEL[1:0]		Res.	Res.	Res.	Res.	IC1INT[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value			0	0					0	0	0	0	0	0	1	0																

Table 77. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0C8	RCC_IC2CFGR	Res.	Res.	IC2SEL[1:0]		Res.	Res.	Res.	Res.	IC2INT[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value			0	0					0	0	0	0	0	0	1	1																
0x0CC	RCC_IC3CFGR	Res.	Res.	IC3SEL[1:0]		Res.	Res.	Res.	Res.	IC3INT[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value			0	0					0	0	0	0	0	0	0	0																
0x0D0	RCC_IC4CFGR	Res.	Res.	IC4SEL[1:0]		Res.	Res.	Res.	Res.	IC4INT[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value			0	0					0	0	0	0	0	0	0	0																
0x0D4	RCC_IC5CFGR	Res.	Res.	IC5SEL[1:0]		Res.	Res.	Res.	Res.	IC5INT[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value			0	0					0	0	0	0	0	0	0	0																
0x0D8	RCC_IC6CFGR	Res.	Res.	IC6SEL[1:0]		Res.	Res.	Res.	Res.	IC6INT[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value			0	0					0	0	0	0	0	0	1	1																
0x0DC	RCC_IC7CFGR	Res.	Res.	IC7SEL[1:0]		Res.	Res.	Res.	Res.	IC7INT[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value			0	1					0	0	0	0	0	0	0	0																
0x0E0	RCC_IC8CFGR	Res.	Res.	IC8SEL[1:0]		Res.	Res.	Res.	Res.	IC8INT[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value			0	1					0	0	0	0	0	0	0	0																
0x0E4	RCC_IC9CFGR	Res.	Res.	IC9SEL[1:0]		Res.	Res.	Res.	Res.	IC9INT[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value			0	1					0	0	0	0	0	0	0	0																
0x0E8	RCC_IC10CFGR	Res.	Res.	IC10SEL [1:0]		Res.	Res.	Res.	Res.	IC10INT[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value			0	1					0	0	0	0	0	0	0	0																
0x0EC	RCC_IC11CFGR	Res.	Res.	IC11SEL [1:0]		Res.	Res.	Res.	Res.	IC11INT[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value			0	0					0	0	0	0	0	0	1	1																

Table 77. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0F0	RCC_IC12CFGR	Res.	Res.	IC12SEL[1:0]		Res.	Res.	Res.	Res.	IC12INT[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value			1	0					0	0	0	0	0	0	0	0																
0x0F4	RCC_IC13CFGR	Res.	Res.	IC13SEL[1:0]		Res.	Res.	Res.	Res.	IC13INT[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value			1	0					0	0	0	0	0	0	0	0																
0x0F8	RCC_IC14CFGR	Res.	Res.	IC14SEL[1:0]		Res.	Res.	Res.	Res.	IC14INT[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value			1	0					0	0	0	0	0	0	0	0																
0x0FC	RCC_IC15CFGR	Res.	Res.	IC15SEL[1:0]		Res.	Res.	Res.	Res.	IC15INT[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value			1	0					0	0	0	0	0	0	0	0																
0x100	RCC_IC16CFGR	Res.	Res.	IC16SEL[1:0]		Res.	Res.	Res.	Res.	IC16INT[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value			1	1					0	0	0	0	0	0	0	0																
0x104	RCC_IC17CFGR	Res.	Res.	IC17SEL[1:0]		Res.	Res.	Res.	Res.	IC17INT[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value			1	1					0	0	0	0	0	0	0	0																
0x108	RCC_IC18CFGR	Res.	Res.	IC18SEL[1:0]		Res.	Res.	Res.	Res.	IC18INT[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value			1	1					0	0	0	0	0	0	0	0																
0x10C	RCC_IC19CFGR	Res.	Res.	IC19SEL[1:0]		Res.	Res.	Res.	Res.	IC19INT[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value			1	1					0	0	0	0	0	0	0	0																
0x110	RCC_IC20CFGR	Res.	Res.	IC20SEL[1:0]		Res.	Res.	Res.	Res.	IC20INT[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value			1	1					0	0	0	0	0	0	0	0																
0x114-0x120	Reserved	Reserved																															
0x124	RCC_CIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPIE	Res.	Res.	Res.	Res.	Res.	Res.	HSECSSIE	LSECSSIE	Res.	Res.	Res.	Res.	PLL4RDYIE	PLL3RDYIE	PLL2RDYIE	PLL1RDYIE	Res.	Res.	Res.	HSERDYIE	HSIRDYIE	MSIRDYIE	LSERDYIE	LSIRDYIE
	Reset value								0							1	0					0	0	0	0				0	0	0	0	0
0x128	RCC_CIFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPF	Res.	Res.	Res.	Res.	Res.	Res.	HSECSSF	LSECSSF	Res.	Res.	Res.	Res.	PLL4RDYF	PLL3RDYF	PLL2RDYF	PLL1RDYF	Res.	Res.	Res.	HSERDYF	HSIRDYF	MSIRDYF	LSERDYF	LSIRDYF
	Reset value								0							0	0					0	0	0	0				0	0	0	0	0

Table 77. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x12C	RCC_CICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WKUPFC	Res.	Res.	Res.	Res.	Res.	Res.	HSECSSC	LSECSSC	Res.	Res.	Res.	Res.	PLL4RDYC	PLL3RDYC	PLL2RDYC	PLL1RDYC	Res.	Res.	Res.	HSERDYC	HSIRDYC	MSIRDYC	LSERDYC	LSIRDYC
	Reset value								0							0	0					0	0	0	0				0	0	0	0	0
0x130-0x140	Reserved	Reserved																															
0x144	RCC_CCIPR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCMIPPSEL [1:0]		Res.	Res.	Res.	Res.	ADCPRE[7:0]								Res.	ADC12SEL [2:0]			Res.	ADF1SEL [2:0]		
	Reset value											0	0					0	0	0	0	0	0	0	0		0	0	0		0	0	0
0x148	RCC_CCIPR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETH1GTXCLKSEL	Res.	Res.	Res.	ETH1REFCLKSEL	Res.	ETH1SEL[2:0]			Res.	Res.	ETH1CLKSEL[1:0]		Res.	Res.	Res.	ETH1PWRDOWNACK	ETH1PTPDIV[3:0]			Res.	Res.	ETH1PTPSEL[1:0]		
	Reset value								0				0		0	0	0			0	0				0	0	0	0			0	0	
0x14C	RCC_CCIPR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FMCSEL [1:0]		Res.	Res.	FDCANSEL [1:0]	
	Reset value																											0	0			1	1
0x150	RCC_CCIPR4	Res.	Res.	Res.	Res.	Res.	Res.	LTDCSEL [1:0]		Res.	I3C2SEL [2:0]			Res.	I3C1SEL [2:0]			Res.	I2C4SEL [2:0]			Res.	I2C3SEL [2:0]			Res.	I2C2SEL [2:0]			Res.	I2C1SEL [2:0]		
	Reset value							0	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0
0x154	RCC_CCIPR5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MDF1SEL [2:0]			MCO2PRE [3:0]				Res.	MCO2SEL [2:0]			MCO1PRE [3:0]				Res.	MCO1SEL [2:0]		
	Reset value														0	0	0	1	1	1	1		0	0	0	1	1	1	1		0	0	0
0x158	RCC_CCIPR6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTGPHY2CKREFSEL	Res.	Res.	OTGPHY2SEL[1:0]		Res.	Res.	Res.	OTGPHY1CKREFSEL	Res.	Res.	OTGPHY1SEL[1:0]		Res.	Res.	XSPI3SEL[1:0]		Res.	Res.	XSPI2SEL[1:0]		Res.	Res.	XSPI1SEL[1:0]	
	Reset value								0			0	0				0			0	0			0	0			0	0			0	0
0x15C	RCC_CCIPR7	Res.	Res.	Res.	Res.	Res.	SAI2SEL [2:0]			Res.	SAI1SEL [2:0]			Res.	Res.	RTCPRE[5:0]						Res.	Res.	RTCSEL [1:0]		Res.	Res.	PSSISEL		Res.	PERSEL [2:0]		
	Reset value						0	0	0		0	0	0			0	0	0	0	0	0			1	0			0	0		0	0	0

Table 77. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x160	RCC_CCIPR8	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC2SEL [1:0]		Res.	Res.	SDMMC1SEL [1:0]	
	Reset value																											0	0			0	0
0x164	RCC_CCIPR9	Res.	Res.	Res.	Res.	Res.	SPI6SEL[2:0]			Res.	SPI5SEL[2:0]			Res.	SPI4SEL[2:0]			Res.	SPI3SEL[2:0]			Res.	SPI2SEL[2:0]			Res.	SPI1SEL[2:0]			Res.	SPDIFRX1SEL[2:0]		
	Reset value						0	0	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0
0x168-0x16C	Reserved	Reserved																															
0x170	RCC_CCIPR12	Res.	Res.	Res.	Res.	Res.	LPTIM5SEL[2:0]			Res.	LPTIM4SEL[2:0]			Res.	LPTIM3SEL[2:0]			Res.	LPTIM2SEL[2:0]			Res.	LPTIM1SEL[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value						0	0	0		0	0	0		0	0	0		0	0	0		0	0	0								
0x174	RCC_CCIPR13	Res.	UART8SEL[2:0]			Res.	UART7SEL[2:0]			Res.	USART6SEL[2:0]			Res.	UART5SEL[2:0]			Res.	UART4SEL[2:0]			Res.	USART3SEL[2:0]			Res.	USART2SEL[2:0]			Res.	USART1SEL[2:0]		
	Reset value		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0
0x178	RCC_CCIPR14	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPUART1SEL[2:0]			Res.	USART10SEL[2:0]			Res.	UART9SEL[2:0]		
	Reset value																						0	0	0		0	0	0		0	0	0
0x17C-0x204	Reserved	Reserved																															
0x208	RCC_MISCRSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC2DLLRS	SDMMC1DLLRS	Res.	XSPIPHY2RST	XSPIPHY1RST	Res.	Res.	Res.	DBGRST
	Reset value																								0	0		0	0				0
0x20C	RCC_MEMRSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOTROMRST	VENCRAMRST	CACHEAXIRAMRST	FLEXRAMRST	AXISRAM2RST	AXISRAM1RST	Res.	AHBSRAM2RST	AHBSRAM1RST	AXISRAM6RST	AXISRAM5RST	AXISRAM4RST	AXISRAM3RST
	Reset value																				0	0	0	0	0	0		0	0	0	0	0	0

Table 77. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x210	RCC_AHB1RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC12RST	GPDMA1RST				
	Reset value																											0	0				
0x214	RCC_AHB2RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADF1RST	MDF1RST	Res.	Res.	Res.	RAMCFGRST												
	Reset value															0	0				0												
0x218	RCC_AHB3RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IACRST		PKARST			SAESRST		CRYPRST	HASHRST	RNGRST	
	Reset value																						0		0			0		0	0	0	
0x21C	RCC_AHB4RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCRST	PWRRST	Res.	GPIOQRST	GPIOPRST	GPIOORST	GPIONRST	Res.	Res.	Res.	Res.	Res.	GPIOHRST	GPIOGRST	GPIOFRST	GPIOERST	GPIODRST	GPIOCRST	GPIOBRST	GPIOARST	
	Reset value													0	0		0	0	0	0						0	0	0	0	0	0	0	0	
0x220	RCC_AHB5RSTR	NPURST	CACHEAXIRST	OTG2RST	OTGPHY2RST	OTGPHY1RST	OTG1RST	ETH1RST	OTG2PHYCTLRST	OTG1PHYCTLRST	Res.	Res.	GPU2DRST	GFXMMURST	Res.	XSPI3RST	Res.	Res.	Res.	XSPIMRST	XSPI2RST	Res.	Res.	Res.	Res.	SDMMC1RST	SDMMC2RST	PSSIRST	XSPI1RST	FMCRST	JPEGRST	Res.	DMA2DRST	HPDMA1RST	
	Reset value	0	0	0	0	0	0	0	0	0			0	0		0				0	0				0	0	0	0	0	0		0	0	
0x224	RCC_APB1LRSTR	UART8RST	UART7RST	Res.	Res.	Res.	Res.	I3C2RST	I3C1RST	I2C3RST	I2C2RST	I2C1RST	UART5RST	UART4RST	USART3RST	USART2RST	SPDIFRX1RST	SPI3RST	SPI2RST	TIM11RST	TIM10RST	WWDGRST	Res.	LPTIM1RST	TIM14RST	TIM13RST	TIM12RST	TIM7RST	TIM6RST	TIM5RST	TIM4RST	TIM3RST	TIM2RST	
	Reset value	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0
0x228	RCC_APB1HRSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD1RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANRS		Res.	MDIOSRST	Res.	Res.	Res.	Res.	Res.	
	Reset value														0										0			0						
0x22C	RCC_APB2RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI2RST	SAI1RST	SPI5RST	TIM9RST	TIM17RST	TIM16RST	TIM15RST	TIM18RST	Res.	SPI4RST	SPI1RST	Res.	Res.	Res.	Res.	USART10RST	UART9RST	USART6RST	USART1RST	Res.	Res.	TIM8RST	TIM1RST	
	Reset value										0	0	0	0	0	0	0	0		0	0					0	0	0	0			0	0	
0x230	Reserved	Reserved																															

Table 77. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x234	RCC_APB4LRSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCRST	VREFBUFRST	Res.	Res.	LPTIM5RST	LPTIM4RST	LPTIM3RST	LPTIM2RST	Res.	I2C4RST	Res.	SPI6RST	Res.	LPUART1RST	HDPRST	Res.	Res.
	Reset value																0	0			0	0	0	0		0		0		0	0		
0x238	RCC_APB4HRSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTSRST	Res.	SYSCFGRST
	Reset value																														0		0
0x23C	RCC_APB5RSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSIRST	VENCRST	GFXTIMRST	Res.	DCMIPPRST	LTDCRST	Res.
	Reset value																										0	0	0		0	0	
0x240	RCC_DIVENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC20EN	IC19EN	IC18EN	IC17EN	IC16EN	IC15EN	IC14EN	IC13EN	IC12EN	IC11EN	IC10EN	IC9EN	IC8EN	IC7EN	IC6EN	IC5EN	IC4EN	IC3EN	IC2EN	IC1EN
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x244	RCC_BUSENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACLKNCEN	ACLKNEN
	Reset value																															1	1
0x248	RCC_MISCENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PEREN	Res.	Res.	XSPIPHYCOMPEN	MCO2EN	MCO1EN	DBGEN
	Reset value																										0			0	0	0	0
0x24C	RCC_MEMENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOTROMEN	VENCRAMEN	CACHEAXIRAMEN	FLEXRAMEN	AXISRAM2EN	AXISRAM1EN	BKPSRAMEN	AHBSRAM2EN	AHBSRAM1EN	AXISRAM6EN	AXISRAM5EN	AXISRAM4EN	AXISRAM3EN
	Reset value																				1	0	0	1	1	1	1	1	1	0	0	0	0
0x250	RCC_AHB1ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC12EN	GPDMA1EN	Res.	Res.	Res.	Res.
	Reset value																											0	0				
0x254	RCC_AHB2ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADF1EN	MDF1EN	Res.	Res.	Res.	RAMCFGEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value															0	0				1												

Table 77. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x258	RCC_AHB3ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RISAFEN	Res.	Res.	Res.	IACEN	RIFSCEN	PKAEN	Res.	Res.	Res.	SAESEN	Res.	CRYPEN	HASHEN	RNGEN
	Reset value																		1				1	1	0				0		0	0	0
0x25C	RCC_AHB4ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCEN	PWREN	Res.	GPIOQEN	GPIOPEN	GPIOOEN	GPIONEN	Res.	Res.	Res.	Res.	Res.	GPIOHEN	GPIOGEN	GPIOFEN	GPIOEEN	GPIODEN	GPIOCEN	GPIOBEN	GPIOAEN
	Reset value													0	1		0	0	0	0						0	0	0	0	0	0	0	0
0x260	RCC_AHB5ENR	NPUEN	CACHEAXIEN	OTG2EN	OTGPHY2EN	OTGPHY1EN	OTG1EN	ETH1EN	ETH1RXEN	ETH1TXEN	ETH1MACEN	Res.	GPU2DEN	GFXMMUEN	MCE4EN	XSPI3EN	MCE3EN	MCE2EN	MCE1EN	XSPIMEN	XSPI2EN	Res.	Res.	Res.	SDMMC1EN	SDMMC2EN	PSSIEN	XSPI1EN	FMCEN	JPEGEN	Res.	DMA2DEN	HPDMA1EN
	Reset value	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0				0	0	0	0	0	0		0	0
0x264	RCC_APB1LENR	UART8EN	UART7EN	Res.	Res.	Res.	Res.	I3C2EN	I3C1EN	I2C3EN	I2C2EN	I2C1EN	UART5EN	UART4EN	USART3EN	USART2EN	SPDIFRX1EN	SPI3EN	SPI2EN	TIM11EN	TIM10EN	WWDGEN	Res.	LPTIM1EN	TIM14EN	TIM13EN	TIM12EN	TIM7EN	TIM6EN	TIM5EN	TIM4EN	TIM3EN	TIM2EN
	Reset value	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0
0x268	RCC_APB1HENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANEN	Res.	Res.	MDIOSEN	Res.	Res.	Res.	Res.	Res.
	Reset value														0										0			0					
0x26C	RCC_APB2ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI2EN	SAI1EN	SPI5EN	TIM9EN	TIM17EN	TIM16EN	TIM15EN	TIM18EN	Res.	SPI4EN	SPI1EN	Res.	Res.	Res.	Res.	USART10EN	UART9EN	USART6EN	USART1EN	Res.	Res.	TIM8EN	TIM1EN
	Reset value										0	0	0	0	0	0	0	0		0	0					0	0	0	0			0	0
0x270	RCC_APB3ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFTEN	Res.	Res.
	Reset value																														0		
0x274	RCC_APB4LENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCAPBEN	RTCEN	VREFBUFEN	Res.	Res.	LPTIM5EN	LPTIM4EN	LPTIM3EN	LPTIM2EN	Res.	I2C4EN	Res.	SPI6EN	Res.	LPUART1EN	HDPEN	Res.	Res.
	Reset value															0	0	0			0	0	0	0		0		0		0	0		
0x278	RCC_APB4HENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTSEN	BSECEN	SYSCFGEN
	Reset value																														0	1	0
0x27C	RCC_APB5ENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSIEN	VENCEN	GFXTIMEN	Res.	DCMIPPEN	LTDCEN	Res.
	Reset value																										0	0	0		0	0	
0x280	Reserved	Reserved																															

Table 77. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x284	RCC_BUSLPENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACLKNCLPEN	ACLKNLPEN
	Reset value																															1	1
0x288	RCC_MISCLPENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERLPEN	Res.	Res.	XSPIPHYCOMPLPEN	Res.	Res.	DBGLPEN
	Reset value																										0			0			0
0x28C	RCC_MEMLPENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOTROMLPEN	VENCRAMLPEN	CACHEAXIRAMLPEN	FLEXRAMLPEN	AXISRAM2LPEN	AXISRAM1LPEN	BKPSRAMLPEN	AHBSRAM2LPEN	AHBSRAM1LPEN	AXISRAM6LPEN	AXISRAM5LPEN	AXISRAM4LPEN	AXISRAM3LPEN
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x290	RCC_AHB1LPENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC12LPEN	GPDMA1LPEN	Res.	Res.	Res.	Res.
	Reset value																											0	0				
0x294	RCC_AHB2LPENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADF1LPEN	MDF1LPEN	Res.	Res.	Res.	RAMCFGLPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value															0	0				0												
0x298	RCC_AHB3LPENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RISAFLPEN	Res.	Res.	Res.	IACLPEN	RIFSCLPEN	PKALPEN	Res.	Res.	Res.	SAESLPEN	Res.	CRYPLPEN	HASHLPEN	RNGLPEN
	Reset value																		0				1	0	0				0		0	0	0
0x29C	RCC_AHB4LPENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCLPEN	PWRLPEN	Res.	GPIOQLPEN	GPIOPLPEN	GPIOOLPEN	GPIONLPEN	Res.	Res.	Res.	Res.	Res.	GPIOHLPEN	GPIOGLPEN	GPIOFLPEN	GPIOELPEN	GPIODLPEN	GPIOCLPEN	GPIOBLPEN	GPIOALPEN
	Reset value													0	1		0	0	0	0						0	0	0	0	0	0	0	0
0x2A0	RCC_AHB5LPENR	NPULPEN	CACHEAXILPEN	OTG2LPEN	OTGPHY2LPEN	OTGPHY1LPEN	OTG1LPEN	ETH1LPEN	ETH1RXLPEN	ETH1TXLPEN	ETH1MACLPEN	Res.	GPU2DLPEN	GFXMMULPEN	MCE4LPEN	XSPI3LPEN	MCE3LPEN	MCE2LPEN	MCE1LPEN	XSPIMLPEN	XSPI2LPEN	Res.	Res.	Res.	SDMMC1LPEN	SDMMC2LPEN	PSSILPEN	XSPI1LPEN	FMCLPEN	JPEGLPEN	Res.	DMA2DLPEN	HPDMA1LPEN
	Reset value	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0				0	0	0	0	0	0		0	0

Table 77. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2A4	RCC_APB1LLPENR	UART8LPEN	UART7LPEN	Res.	Res.	Res.	Res.	I3C2LPEN	I3C1LPEN	I2C3LPEN	I2C2LPEN	I2C1LPEN	UART5LPEN	UART4LPEN	USART3LPEN	USART2LPEN	SPDIFRX1LPEN	SPI3LPEN	SPI2LPEN	TIM11LPEN	TIM10LPEN	WWDGLPEN	Res.	LPTIM1LPEN	TIM14LPEN	TIM13LPEN	TIM12LPEN	TIM7LPEN	TIM6LPEN	TIM5LPEN	TIM4LPEN	TIM3LPEN	TIM2LPEN
	Reset value	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0
0x2A8	RCC_APB1HLPENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD1LPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANLPEN	Res.	Res.	MDIOSLPEN	Res.	Res.	Res.	Res.	Res.
	Reset value														0										0			0					
0x2AC	RCC_APB2LPENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI2LPEN	SAI1LPEN	SPI5LPEN	TIM9LPEN	TIM17LPEN	TIM16LPEN	TIM15LPEN	TIM18LPEN	Res.	SPI4LPEN	SPI1LPEN	Res.	Res.	Res.	Res.	USART10LPEN	UART9LPEN	USART6LPEN	USART1LPEN	Res.	Res.	TIM8LPEN	TIM1LPEN
	Reset value										0	0	0	0	0	0	0	0		0	0					0	0	0	0			0	0
0x2B0	RCC_APB3LPENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFTLPEN	Res.	Res.
	Reset value																														0		
0x2B4	RCC_APB4LLPENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCAPBLPEN	RTCLPEN	VREFBUFLPEN	Res.	Res.	LPTIM5LPEN	LPTIM4LPEN	LPTIM3LPEN	LPTIM2LPEN	Res.	I2C4LPEN	Res.	SPI6LPEN	Res.	LPUART1LPEN	HDPLPEN	Res.	Res.
	Reset value															0	0	0			0	0	0	0		0		0		0	0		
0x2B8	RCC_APB4HLPENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTSLPEN	BSECLPEN	SYSCFGLPEN
	Reset value																														0	1	0
0x2BC	RCC_APB5LPENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSILPEN	VENCLPEN	GFXTIMLPEN	Res.	DCMIPPLPEN	LTDCLPEN	Res.
	Reset value																										0	0	0		0	0	
0x2C0-0x448	Reserved	Reserved																															
0x44C	RCC_RDCR	Res.	Res.	Res.	Res.	EADLY[3:0]				Res.	Res.	Res.	MRD[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value					0	0	0	0				0	0	0	0	0																
0x450-0x77C	Reserved	Reserved																															
0x780	RCC_SECCFGR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSESEC	HSISEC	MSISEC	LSESEC	LSISEC
	Reset value																												0	0	0	0	0

Table 77. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x784	RCC_PRIVCFGR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSEPRIV	HSIPRIV	MSIPRIV	LSEPRIV	LSIPRIV
	Reset value																												0	0	0	0	0
0x788	RCC_LOCKCFGR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSELOCK	HSILOCK	MSILOCK	LSELOCK	LSILOCK
	Reset value																												0	0	0	0	0
0x78C	RCC_PUBCFGR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSEPUB	HSIPUB	MSIPUB	LSEPUB	LSIPUB
	Reset value																												0	0	0	0	0
0x790	RCC_SECCFGR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL4SEC	PLL3SEC	PLL2SEC	PLL1SEC
	Reset value																													0	0	0	0
0x794	RCC_PRIVCFGR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL4PRIV	PLL3PRIV	PLL2PRIV	PLL1PRIV
	Reset value																													0	0	0	0
0x798	RCC_LOCKCFGR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL4LOCK	PLL3LOCK	PLL2LOCK	PLL1LOCK
	Reset value																													0	0	0	0
0x79C	RCC_PUBCFGR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL4PUB	PLL3PUB	PLL2PUB	PLL1PUB
	Reset value																													0	0	0	0
0x7A0	RCC_SECCFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC20SEC	IC19SEC	IC18SEC	IC17SEC	IC16SEC	IC15SEC	IC14SEC	IC13SEC	IC12SEC	IC11SEC	IC10SEC	IC9SEC	IC8SEC	IC7SEC	IC6SEC	IC5SEC	IC4SEC	IC3SEC	IC2SEC	IC1SEC
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x7A4	RCC_PRIVCFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC20PRIV	IC19PRIV	IC18PRIV	IC17PRIV	IC16PRIV	IC15PRIV	IC14PRIV	IC13PRIV	IC12PRIV	IC11PRIV	IC10PRIV	IC9PRIV	IC8PRIV	IC7PRIV	IC6PRIV	IC5PRIV	IC4PRIV	IC3PRIV	IC2PRIV	IC1PRIV
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x7A8	RCC_LOCKCFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC20LOCK	IC19LOCK	IC18LOCK	IC17LOCK	IC16LOCK	IC15LOCK	IC14LOCK	IC13LOCK	IC12LOCK	IC11LOCK	IC10LOCK	IC9LOCK	IC8LOCK	IC7LOCK	IC6LOCK	IC5LOCK	IC4LOCK	IC3LOCK	IC2LOCK	IC1LOCK
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x7AC	RCC_PUBCFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC20PUB	IC19PUB	IC18PUB	IC17PUB	IC16PUB	IC15PUB	IC14PUB	IC13PUB	IC12PUB	IC11PUB	IC10PUB	IC9PUB	IC8PUB	IC7PUB	IC6PUB	IC5PUB	IC4PUB	IC3PUB	IC2PUB	IC1PUB
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 77. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x7B0	RCC_SECCFGR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTSEC	INTSEC	PERSEC	BUSSEC	SYSSEC	MODSEC
	Reset value																											0	0	0	0	0	0
0x7B4	RCC_PRIVCFGR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTPRIV	INTPRIV	PERPRIV	BUSPRIV	SYSPRIV	MODPRIV
	Reset value																											0	0	0	0	0	0
0x7B8	RCC_LOCKCFGR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTLOCK	INTLOCK	PERLOCK	BUSLOCK	SYSLOCK	MODLOCK
	Reset value																											0	0	0	0	0	0
0x7BC	RCC_PUBCFGR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTPUB	INTPUB	PERPUB	BUSPUB	SYSPUB	MODPUB
	Reset value																											0	0	0	0	0	0
0x7C0	RCC_SECCFGR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NOCSEC	APB5SEC	APB4SEC	APB3SEC	APB2SEC	APB1SEC	AHB5SEC	AHB4SEC	AHB3SEC	AHB2SEC	AHB1SEC	AHBMSEC	ACLKNCSEC	ACLKNSEC
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x7C4	RCC_PRIVCFGR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NOCPRIV	APB5PRIV	APB4PRIV	APB3PRIV	APB2PRIV	APB1PRIV	AHB5PRIV	AHB4PRIV	AHB3PRIV	AHB2PRIV	AHB1PRIV	AHBMPRIV	ACLKNCPRIV	ACLKNPRIV
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x7C8	RCC_LOCKCFGR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NOCLOCK	APB5LOCK	APB4LOCK	APB3LOCK	APB2LOCK	APB1LOCK	AHB5LOCK	AHB4LOCK	AHB3LOCK	AHB2LOCK	AHB1LOCK	AHBMLOCK	ACLKNCLOCK	ACLKNLOCK
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x7CC	RCC_PUBCFGR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NOCPUB	APB5PUB	APB4PUB	APB3PUB	APB2PUB	APB1PUB	AHB5PUB	AHB4PUB	AHB3PUB	AHB2PUB	AHB1PUB	AHBMPUB	ACLKNCPUB	ACLKNPUB
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x7D0	RCC_PUBCFGR5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VENCRAMPUB	CACHEAXIRAMPUB	FLEXRAMPUB	AXISRAM2PUB	AXISRAM1PUB	BKPSRAMPUB	AHBSRAM2PUB	AHBSRAM1PUB	AXISRAM6PUB	AXISRAM5PUB	AXISRAM4PUB	AXISRAM3PUB
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x7D4-0x7FC	Reserved	Reserved																															

Table 77. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x800	RCC_CSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL4ONS	PLL3ONS	PLL2ONS	PLL1ONS	Res.	Res.	Res.	HSEONS	HSIONS	MSIONS	LSEONS	LSIONS
	Reset value																					0	0	0	0				0	0	0	0	0
0x804	Reserved	Reserved																															
0x808	RCC_STOPCSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSISTOPENS	MSISTOPENS
	Reset value																															0	0
0x80C-0xA04	Reserved	Reserved																															
0xA08	RCC_MISCRSTSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC2DLLRSTS	SDMMC1DLLRSTS	Res.	XSPIPHY2RSTS	XSPIPHY1RSTS	Res.	Res.	Res.	DBGRSTS
	Reset value																								0	0		0	0				0
0xA0C	RCC_MEMRSTSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOTROMRSTS	VENCRAMRSTS	CACHEAXIRAMRSTS	FLEXRAMRSTS	AXISRAM2RSTS	AXISRAM1RSTS	Res.	AHBSRAM2RSTS	AHBSRAM1RSTS	AXISRAM6RSTS	AXISRAM5RSTS	AXISRAM4RSTS	AXISRAM3RSTS
	Reset value																				0	0	0	0	0	0		0	0	0	0	0	0
0xA10	RCC_AHB1RSTSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC12RSTS	GPDMA1RSTS	Res.	Res.	Res.	Res.
	Reset value																											0	0				
0xA14	RCC_AHB2RSTSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADF1RSTS	MDF1RSTS	Res.	Res.	Res.	RAMCFGRSTS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value															0	0				0												
0xA18	RCC_AHB3RSTSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IACRSTS	Res.	PKARSTS	Res.	Res.	Res.	SAESRSTS	Res.	CRYPRSTS	HASHRSTS	RNGRSTS
	Reset value																						0		0				0		0	0	0
0xA1C	RCC_AHB4RSTSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCRSTS	PWRRSTS	Res.	GPIOQRSTS	GPIOPRSTS	GPIOORSTS	GPIONRSTS	Res.	Res.	Res.	Res.	Res.	GPIOHRSTS	GPIOGRSTS	GPIOFRSTS	GPIOERSTS	GPIODRSTS	GPIOCRSTS	GPIOBRSTS	GPIOARSTS
	Reset value													0	0		0	0	0	0						0	0	0	0	0	0	0	0

Table 77. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xA20	RCC_AHB5RSTSR	NPURSTS	CACHEAXIRSTS	OTG2RSTS	OTGPHY2RSTS	OTGPHY1RSTS	OTG1RSTS	ETH1RSTS	OTG2PHYCTLRSTS	OTG1PHYCTLRSTS	Res.	Res.	GPU2DRSTS	GFXMMURSTS	Res.	XSPI3RSTS	Res.	Res.	Res.	XSPIMRSTS	XSPI2RSTS	Res.	Res.	Res.	SDMMC1RSTS	SDMMC2RSTS	PSSIRSTS	XSPI1RSTS	FMCRSTS	JPEGRSTS	Res.	DMA2DRSTS	HPDMA1RSTS
	Reset value	0	0	0	0	0	0	0	0	0			0	0		0				0	0				0	0	0	0	0	0		0	0
0xA24	RCC_APB1LRSTSR	UART8RSTS	UART7RSTS	Res.	Res.	Res.	Res.	I3C2RSTS	I3C1RSTS	I2C3RSTS	I2C2RSTS	I2C1RSTS	UART5RSTS	UART4RSTS	USART3RSTS	USART2RSTS	SPDIFRX1RSTS	SPI3RSTS	SPI2RSTS	TIM11RSTS	TIM10RSTS	WWDGRSTS	Res.	LPTIM1RSTS	TIM14RSTS	TIM13RSTS	TIM12RSTS	TIM7RSTS	TIM6RSTS	TIM5RSTS	TIM4RSTS	TIM3RSTS	TIM2RSTS
	Reset value	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0
0xA28	RCC_APB1HRSTSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD1RSTS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANRSTS	Res.	Res.	MDIOSRSTS	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value														0										0			0						
0xA2C	RCC_APB2RSTSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI2RSTS	SAI1RSTS	SPI5RSTS	TIM9RSTS	TIM17RSTS	TIM16RSTS	TIM15RSTS	TIM18RSTS	Res.	SPI4RSTS	SPI1RSTS	Res.	Res.	Res.	Res.	USART10RSTS	Res.	UART9RSTS	USART6RSTS	USART1RSTS	Res.	Res.	TIM8RSTS	TIM1RSTS
	Reset value										0	0	0	0	0	0	0	0		0	0					0		0	0	0			0	0
0xA30	Reserved	Reserved																															
0xA34	RCC_APB4LRSTSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCRSTS	VREFBUFRSTS	Res.	Res.	LPTIM5RSTS	LPTIM4RSTS	LPTIM3RSTS	LPTIM2RSTS	Res.	I2C4RSTS	Res.	SPI6RSTS	Res.	LPUART1RSTS	HDPRSTS	Res.	Res.
	Reset value																0	0			0	0	0	0		0		0		0	0		
0xA38	RCC_APB4HRSTSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTSRSTS	Res.	SYSCFGRSTS
	Reset value																														0		0
0xA3C	RCC_APB5RSTSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSIRSTS	VENCRSTS	GFXTIMRSTS	Res.	DCMIPPRSTS	LTDCRSTS	Res.
	Reset value																										0	0	0		0	0	
0xA40	RCC_DIVENSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC20ENS	IC19ENS	IC18ENS	IC17ENS	IC16ENS	IC15ENS	IC14ENS	IC13ENS	IC12ENS	IC11ENS	IC10ENS	IC9ENS	IC8ENS	IC7ENS	IC6ENS	IC5ENS	IC4ENS	IC3ENS	IC2ENS	IC1ENS
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 77. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xA44	RCC_BUSENSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACLKNCENS	ACLKNENS
	Reset value																															0	0
0xA48	RCC_MISCENSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERENS	Res.	Res.	XSPIPHYCOMPENS	MCO2ENS	MCO1ENS	DBGENS
	Reset value																										0			0	0	0	0
0xA4C	RCC_MEMENSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOTROMENS	VENCRAMENS	CACHEAXIRAMENS	FLEXRAMENS	AXISRAM2ENS	AXISRAM1ENS	BKPSRAMENS	AHBSRAM2ENS	AHBSRAM1ENS	AXISRAM6ENS	AXISRAM5ENS	AXISRAM4ENS	AXISRAM3ENS
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0xA50	RCC_AHB1ENSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC12ENS	GPDMA1ENS	Res.	Res.	Res.	Res.
	Reset value																											0	0				
0xA54	RCC_AHB2ENSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADF1ENS	MDF1ENS	Res.	Res.	Res.	RAMCFGENS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value															0	0				0												
0xA58	RCC_AHB3ENSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RISAFENS	Res.	Res.	Res.	IACENS	RIFSCENS	PKAENS	Res.	Res.	SAESENS	Res.	Res.	CRYPENS	HASHENS	RNGENS
	Reset value																		0				0	0	0			0			0	0	0
0xA5C	RCC_AHB4ENSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCENS	PWRENS	Res.	GPIOQENS	GPIOPENS	GPIOOENS	GPIONENS	Res.	Res.	Res.	Res.	Res.	GPIOHENS	GPIOGENS	GPIOFENS	GPIOEENS	GPIODENS	GPIOCENS	GPIOBENS	GPIOAENS
	Reset value													0	0		0	0	0	0						0	0	0	0	0	0	0	0
0xA60	RCC_AHB5ENSR	NPUENS	CACHEAXIENS	OTG2ENS	OTGPHY2ENS	OTGPHY1ENS	OTG1ENS	ETH1ENS	ETH1RXENS	ETH1TXENS	ETH1MACENS	Res.	GPU2DENS	GFXMMUENS	MCE4ENS	XSPI3ENS	MCE3ENS	MCE2ENS	MCE1ENS	XSPIMENS	XSPI2ENS	Res.	Res.	Res.	SDMMC1ENS	SDMMC2ENS	PSSIENS	XSPI1ENS	FMCENS	JPEGENS	Res.	DMA2DENS	HPDMA1ENS
	Reset value	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0				0	0	0	0	0	0		0	0

Table 77. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xA64	RCC_APB1LENSR	UART8ENS	UART7ENS	Res.	Res.	Res.	Res.	I3C2ENS	I3C1ENS	I2C3ENS	I2C2ENS	I2C1ENS	UART5ENS	UART4ENS	USART3ENS	USART2ENS	SPDIFRX1ENS	SPI3ENS	SPI2ENS	TIM11ENS	TIM10ENS	WWDGENS	Res.	LPTIM1ENS	TIM14ENS	TIM13ENS	TIM12ENS	TIM7ENS	TIM6ENS	TIM5ENS	TIM4ENS	TIM3ENS	TIM2ENS
	Reset value	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0
0xA68	RCC_APB1HENSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD1ENS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANENS	Res.	Res.	MDIOSENS	Res.	Res.	Res.	Res.	Res.
	Reset value														0										0			0					
0xA6C	RCC_APB2ENSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI2ENS	SAI1ENS	SPI5ENS	TIM9ENS	TIM17ENS	TIM16ENS	TIM15ENS	TIM18ENS	Res.	SPI4ENS	SPI1ENS	Res.	Res.	Res.	Res.	USART10ENS	UART9ENS	USART6ENS	USART1ENS	Res.	Res.	TIM8ENS	TIM1ENS
	Reset value										0	0	0	0	0	0	0	0		0	0					0	0	0	0			0	0
0xA70	RCC_APB3ENSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFTENS	Res.	Res.
	Reset value																														0		
0xA74	RCC_APB4LENSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCAPBENS	RTCENS	VREFBUFENS	Res.	Res.	LPTIM5ENS	LPTIM4ENS	LPTIM3ENS	LPTIM2ENS	Res.	I2C4ENS	Res.	SPI6ENS	Res.	LPUART1ENS	HDPENS	Res.	Res.
	Reset value															0	0	0			0	0	0	0		0		0		0	0		
0xA78	RCC_APB4HENSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTSENS	BSECENS	SYSCFGENS
	Reset value																														0	0	0
0xA7C	RCC_APB5ENSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSIENS	VENCENS	GFXTIMENS	Res.	DCMIPPENS	LTDCENS	Res.
	Reset value																										0	0	0		0	0	
0xA80	Reserved	Reserved																															
0xA84	RCC_BUSLPENSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACLKNCLPENS	ACLKNLPENS
	Reset value																															0	0

Table 77. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xA88	RCC_MISCLPENSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERLPENS	Res.	Res.	XSPIPHYCOMPLPENS	Res.	Res.	DBGLPENS
	Reset value																										0			0			0
0xA8C	RCC_MEMLPENSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOTROMLPENS	VENCRAMLPENS	CACHEAXIRAMLPENS	FLEXRAMLPENS	AXISRAM2LPENS	AXISRAM1LPENS	BKPSRAMLPENS	AHBSRAM2LPENS	AHBSRAM1LPENS	AXISRAM6LPENS	AXISRAM5LPENS	AXISRAM4LPENS	AXISRAM3LPENS
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0xA90	RCC_AHB1LPENSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC12LPENS	GPDMA1LPENS	Res.	Res.	Res.	Res.
	Reset value																											0	0				
0xA94	RCC_AHB2LPENSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADF1LPENS	MDF1LPENS	Res.	Res.	Res.	RAMCFGLPENS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value															0	0				0												
0xA98	RCC_AHB3LPENSR	Res.	Res.	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	RISAFLPENS	Res	Res	Res	IACLPENS	RIFSCLPENS	PKALPENS	Res	Res	Res	SAESLPENS	Res	CRYPLPENS	HASHLPENS	RNGLPENS
	Reset value																		0				0	0	0				0		0	0	0
0xA9C	RCC_AHB4LPENSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCLPENS	PWRLPENS	Res.	GPIOQLPENS	GPIOPLPENS	GPIOOLPENS	GPIONLPENS	Res.	Res.	Res.	Res.	Res.	GPIOHLPENS	GPIOGLPENS	GPIOFLPENS	GPIOELPENS	GPIODLPENS	GPIOCLPENS	GPIOBLPENS	GPIOALPENS
	Reset value													0	0		0	0	0	0						0	0	0	0	0	0	0	0
0xAA0	RCC_AHB5LPENSR	NPULPENS	CACHEAXILPENS	OTG2LPENS	OTGPHY2LPENS	OTGPHY1LPENS	OTG1LPENS	ETH1LPENS	ETH1RXLPENS	ETH1TXLPENS	ETH1MACLPENS	Res.	GPU2DLPENS	GFXMMULPENS	MCE4LPENS	XSPI3LPENS	MCE3LPENS	MCE2LPENS	MCE1LPENS	XSPIMLPENS	XSPI2LPENS	Res.	Res.	Res.	SDMMC1LPENS	SDMMC2LPENS	PSSILPENS	XSPI1LPENS	FMCLPENS	JPEGLPENS	Res.	DMA2DLPENS	HPDMA1LPENS
	Reset value	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0				0	0	0	0	0	0		0	0

Table 77. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xAA4	RCC_APB1LLPENSR	UART8LPENS	UART7LPENS	Res.	Res.	Res.	Res.	I3C2LPENS	I3C1LPENS	I2C3LPENS	I2C2LPENS	I2C1LPENS	UART5LPENS	UART4LPENS	USART3LPENS	USART2LPENS	SPDIFRX1LPENS	SPI3LPENS	SPI2LPENS	TIM11LPENS	TIM10LPENS	WWDGLPENS	Res.	LPTIM1LPENS	TIM14LPENS	TIM13LPENS	TIM12LPENS	TIM7LPENS	TIM6LPENS	TIM5LPENS	TIM4LPENS	TIM3LPENS	TIM2LPENS
	Reset value	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0
0xAA8	RCC_ APB1HLPENSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD1LPENS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANLPENS	Res.	Res.	MDIOSLPENS	Res.	Res.	Res.	Res.	Res.
	Reset value														0										0			0					
0xAAC	RCC_APB2LPENSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI2LPENS	SAI1LPENS	SPI5LPENS	TIM9LPENS	TIM17LPENS	TIM16LPENS	TIM15LPENS	TIM18LPENS	Res.	SPI4LPENS	SPI1LPENS	Res.	Res.	Res.	Res.	USART10LPENS	UART9LPENS	USART6LPENS	USART1LPENS	Res.	Res.	TIM8LPENS	TIM1LPENS
	Reset value										0	0	0	0	0	0	0	0		0	0					0	0	0	0			0	0
0xAB0	RCC_APB3LPENSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFTLPENS	Res.	Res.
	Reset value																														0		
0xAB4	RCC_APB4LLPENSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCAPBLPENS	RTCLPENS	VREFBUFLPENS	Res.	Res.	LPTIM5LPENS	LPTIM4LPENS	LPTIM3LPENS	LPTIM2LPENS	Res.	I2C4LPENS	Res.	SPI6LPENS	Res.	LPUART1LPENS	HDPLPENS	Res.	Res.
	Reset value															0	0	0			0	0	0	0		0		0		0	0		
0xAB8	RCC_ APB4HLPENSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTSLPENS	BSECLPENS	SYSCFGLPENS
	Reset value																														0	0	0
0xABC	RCC_APB5LPENSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSILPENS	VENCLPENS	GFXTIMLPENS	Res.	DCMIPPLPENS	LTDCLPENS	Res.
	Reset value																										0	0	0		0	0	
0xAC0-0xF80	Reserved	Reserved																															
0xF84	RCC_PRIVCFGSR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSEPRIVS	HSIPRIVS	MSIPRIVS	LSEPRIVS	LSIPRIVS
	Reset value																												0	0	0	0	0
0xF88	Reserved	Reserved																															

Table 77. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xF8C	RCC_PUBCFGSR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSEPUBS	HSIPUBS	MSIPUBS	LSEPUBS	LSIPUBS
	Reset value																												0	0	0	0	0
0xF90	Reserved	Reserved																															
0xF94	RCC_PRIVCFGSR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL4PRIVS	PLL3PRIVS	PLL2PRIVS	PLL1PRIVS
	Reset value																													0	0	0	0
0xF98	Reserved	Reserved																															
0xF9C	RCC_PUBCFGSR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL4PUBS	PLL3PUBS	PLL2PUBS	PLL1PUBS
	Reset value																													0	0	0	0
0xFA0	Reserved	Reserved																															
0xFA4	RCC_PRIVCFGSR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC20PRIVS	IC19PRIVS	IC18PRIVS	IC17PRIVS	IC16PRIVS	IC15PRIVS	IC14PRIVS	IC13PRIVS	IC12PRIVS	IC11PRIVS	IC10PRIVS	IC9PRIVS	IC8PRIVS	IC7PRIVS	IC6PRIVS	IC5PRIVS	IC4PRIVS	IC3PRIVS	IC2PRIVS	IC1PRIVS
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xFA8	Reserved	Reserved																															
0xFAC	RCC_ PUBCFGSR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC20PUBS	IC19PUBS	IC18PUBS	IC17PUBS	IC16PUBS	IC15PUBS	IC14PUBS	IC13PUBS	IC12PUBS	IC11PUBS	IC10PUBS	IC9PUBS	IC8PUBS	IC7PUBS	IC6PUBS	IC5PUBS	IC4PUBS	IC3PUBS	IC2PUBS	IC1PUBS
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xFB0	Reserved	Reserved																															
0xFB4	RCC_ PRIVCFGSR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFTPRIVS	RSTPRIVS	INTPRIVS	PERPRIVS	BUSPRIVS	SYSPRIVS	MODPRIVS
	Reset value																										0	0	0	0	0	0	0
0xFB8	Reserved	Reserved																															
0xFBC	RCC_ PUBCFGSR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTPUBS	INTPUBS	PERPUBS	BUSPUBS	SYSPUBS	MODPUBS
	Reset value																											0	0	0	0	0	0
0xFC0	Reserved	Reserved																															
0xFC4	RCC_ PRIVCFGSR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NOCPRIVS	APB5PRIVS	APB4PRIVS	APB3PRIVS	APB2PRIVS	APB1PRIVS	AHB5PRIVS	AHB4PRIVS	AHB3PRIVS	AHB2PRIVS	AHB1PRIVS	AHBMPRIVS	ACLKNCPRIVS	ACLKNPRIVS
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xFC8	Reserved	Reserved																															

Table 77. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFCC	RCC_ PUBCFGSR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NOCPUBS	APB5PUBS	APB4PUBS	APB3PUBS	APB2PUBS	APB1PUBS	AHB5PUBS	AHB4PUBS	AHB3PUBS	AHB2PUBS	AHB1PUBS	AHBMPUBS	ACLKNCPUBS	ACLKNPUBS
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xFD0	RCC_ PUBCFGSR5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VENCRAMPUBS	CACHEAXIRAMPUBS	FLEXRAMPUBS	AXISRAM2PUBS	AXISRAM1PUBS	BKPSRAMPUBS	AHBSRAM2PUBS	AHBSRAM1PUBS	AXISRAM6PUBS	AXISRAM5PUBS	AXISRAM4PUBS	AXISRAM3PUBS
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0xFD4-0xFFC	Reserved	Reserved																															
0x1000	RCC_CCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL4ONC	PLL3ONC	PLL2ONC	PLL1ONC				HSEONC	HSIONC	MSIONC	LSEONC	LSIONC
	Reset value																					0	0	0	0				0	0	0	0	0
0x1004	Reserved	Reserved																															
0x1008	RCC_STOPCCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSISTOPENC	MSISTOPENC
	Reset value																															0	0
0x100C-0x1204	Reserved	Reserved																															
0x1208	RCC_MISCRSTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC2DLLRSTC	SDMMC1DLLRSTC	Res.	XSPIPHY2RSTC	XSPIPHY1RSTC	Res.	Res.	Res.	DBGRSTC
	Reset value																								0	0		0	0				0
0x120C	RCC_MEMRSTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOTROMRSTC	VENCRAMRSTC	CACHEAXIRAMRSTC	FLEXRAMRSTC	AXISRAM2RSTC	AXISRAM1RSTC	Res.	AHBSRAM2RSTC	AHBSRAM1RSTC	AXISRAM6RSTC	AXISRAM5RSTC	AXISRAM4RSTC	AXISRAM3RSTC
	Reset value																				0	0	0	0	0	0		0	0	0	0	0	0
0x1210	RCC_AHB1RSTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC12RSTC	GPDMA1RSTC	Res.	Res.	Res.	Res.
	Reset value																											0	0				

Table 77. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1214	RCC_AHB2RSTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADF1RSTC	MDF1RSTC	Res.	Res.	Res.	RAMCFGRSTC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value															0	0				0												
0x1218	RCC_AHB3RSTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IACRSTC	Res.	PKARSTC	Res.	Res.	Res.	SAESRSTC	Res.	CRYPRSTC	HASHRSTC	RNGRSTC
	Reset value																						0		0				0		0	0	0
0x121C	RCC_AHB4RSTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCRSTC	PWRRSTC	Res.	GPIOQRSTC	GPIOPRSTC	GPIOORSTC	GPIONRSTC	Res.	Res.	Res.	Res.	Res.	GPIOHRSTC	GPIOGRSTC	GPIOFRSTC	GPIOERSTC	GPIODRSTC	GPIOCRSTC	GPIOBRSTC	GPIOARSTC
	Reset value													0	0		0	0	0	0						0	0	0	0	0	0	0	0
0x1220	RCC_AHB5RSTCR	NPURSTC	CACHEAXIRSTC	OTG2RSTC	OTGPHY2RSTC	OTGPHY1RSTC	OTG1RSTC	ETH1RSTC	OTG2PHYCTLRSTC	OTG1PHYCTLRSTC	Res.	Res.	GPU2DRSTC	GFXMMURSTC	Res.	XSPI3RSTC	Res.	Res.	Res.	XSPIMRSTC	XSPI2RSTC	Res.	Res.	Res.	SDMMC1RSTC	SDMMC2RSTC	PSSIRSTC	XSPI1RSTC	FMCRSTC	JPEGRSTC	Res.	DMA2DRSTC	HPDMA1RSTC
	Reset value	0	0	0	0	0	0	0	0	0			0	0		0				0	0				0	0	0	0	0	0		0	0
0x1224	RCC_APB1LRSTCR	UART8RSTC	UART7RSTC	Res.	Res.	Res.	Res.	I3C2RSTC	I3C1RSTC	I2C3RSTC	I2C2RSTC	I2C1RSTC	UART5RSTC	UART4RSTC	USART3RSTC	USART2RSTC	SPDIFRX1RSTC	SPI3RSTC	SPI2RSTC	TIM11RSTC	TIM10RSTC	WWDGRSTC	Res.	LPTIM1RSTC	TIM14RSTC	TIM13RSTC	TIM12RSTC	TIM7RSTC	TIM6RSTC	TIM5RSTC	TIM4RSTC	TIM3RSTC	TIM2RSTC
	Reset value	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0
0x1228	RCC_APB1HRSTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD1RSTC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANRSTC	Res.	Res.	MDIOSRSTC	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value														0										0			0						
0x122C	RCC_APB2RSTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI2RSTC	SAI1RSTC	SPI5RSTC	TIM9RSTC	TIM17RSTC	TIM16RSTC	TIM15RSTC	TIM18RSTC	Res.	SPI4RSTC	SPI1RSTC	Res.	Res.	Res.	Res.	USART10RSTC	UART9RSTC	USART6RSTC	USART1RSTC	Res.	Res.	Res.	TIM8RSTC	TIM1RSTC
	Reset value										0	0	0	0	0	0	0	0		0	0					0	0	0	0				0	0
0x1230	Reserved	Reserved																															
0x1234	RCC_APB4LRSTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCRSTC	VREFBUFRSTC	Res.	Res.	LPTIM5RSTC	LPTIM4RSTC	LPTIM3RSTC	LPTIM2RSTC	Res.	I2C4RSTC	Res.	SPI6RSTC	Res.	LPUART1RSTC	HDPRSTC	Res.	Res.
	Reset value																0	0			0	0	0	0		0		0		0	0		

Table 77. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1238	RCC_APB4HRSTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTSRSTC	Res.	SYSCFGRSTC
	Reset value																														0		0
0x123C	RCC_APB5RSTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSIRSTC	VENCRSTC	GFXTIMRSTC	Res.	DCMIPPRSTC	LTDCRSTC	Res.
	Reset value																										0	0	0		0	0	
0x1240	RCC_DIVENCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC20ENC	IC19ENC	IC18ENC	IC17ENC	IC16ENC	IC15ENC	IC14ENC	IC13ENC	IC12ENC	IC11ENC	IC10ENC	IC9ENC	IC8ENC	IC7ENC	IC6ENC	IC5ENC	IC4ENC	IC3ENC	IC2ENC	IC1ENC
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1244	RCC_BUSENCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACLKNCENC	ACLKNENC
	Reset value																															0	0
0x1248	RCC_MISCENCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERENC	Res.	Res.	XSPIPHYCOMPENC	MCO2ENC	MCO1ENC	DBGENC
	Reset value																										0			0	0	0	0
0x124C	RCC_MEMENCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOTROMENC	VENCRAMENC	CACHEAXIRAMENC	FLEXRAMENC	AXISRAM2ENC	AXISRAM1ENC	BKPSRAMENC	AHBSRAM2ENC	AHBSRAM1ENC	AXISRAM6ENC	AXISRAM5ENC	AXISRAM4ENC	AXISRAM3ENC
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x1250	RCC_AHB1ENCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC12ENC	GPDMA1ENC	Res.	Res.	Res.	Res.
	Reset value																											0	0				
0x1254	RCC_AHB2ENCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADF1ENC	MDF1ENC	Res.	Res.	Res.	RAMCFGENC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value															0	0				0												

Table 77. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1258	RCC_AHB3ENCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RISAFENC	Res.	Res.	Res.	IACENC	RIFSCENC	PKAENC	Res.	Res.	Res.	SAESENC	Res.	CRYPENC	HASHENC	RNGENC
	Reset value																		0				0	0	0				0		0	0	0
0x125C	RCC_AHB4ENCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCENC	PWRENC	Res.	GPIOQENC	GPIOPENC	GPIOOENC	GPIONENC	Res.	Res.	Res.	Res.	Res.	GPIOHENC	GPIOGENC	GPIOFENC	GPIOEENC	GPIODENC	GPIOCENC	GPIOBENC	GPIOAENC
	Reset value													0	0		0	0	0	0						0	0	0	0	0	0	0	0
0x1260	RCC_AHB5ENCR	NPUENC	CACHEAXIENC	OTG2ENC	OTGPHY2ENC	OTGPHY1ENC	OTG1ENC	ETH1ENC	ETH1RXENC	ETH1TXENC	ETH1MACENC	Res.	GPU2DENC	GFXMMUENC	MCE4ENC	XSPI3ENC	MCE3ENC	MCE2ENC	MCE1ENC	XSPIMENC	XSPI2ENC	Res.	Res.	Res.	SDMMC1ENC	SDMMC2ENC	PSSIENC	XSPI1ENC	FMCENC	JPEGENC	Res.	DMA2DENC	HPDMA1ENC
	Reset value	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0				0	0	0	0	0	0		0	0
0x1264	RCC_APB1LENCR	UART8ENC	UART7ENC	Res.	Res.	Res.	Res.	I3C2ENC	I3C1ENC	I2C3ENC	I2C2ENC	I2C1ENC	UART5ENC	UART4ENC	USART3ENC	USART2ENC	SPDIFRX1ENC	SPI3ENC	SPI2ENC	TIM11ENC	TIM10ENC	Res.	Res.	LPTIM1ENC	TIM14ENC	TIM13ENC	TIM12ENC	TIM7ENC	TIM6ENC	TIM5ENC	TIM4ENC	TIM3ENC	TIM2ENC
	Reset value	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0
0x1268	RCC_APB1HENCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD1ENC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANENC	Res.	Res.	MDIOSENC	Res.	Res.	Res.	Res.	Res.
	Reset value														0										0			0					
0x126C	RCC_APB2ENCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI2ENC	SAI1ENC	SPI5ENC	TIM9ENC	TIM17ENC	TIM16ENC	TIM15ENC	TIM18ENC	Res.	SPI4ENC	SPI1ENC	Res.	Res.	Res.	Res.	USART10ENC	UART9ENC	USART6ENC	USART1ENC	Res.	Res.	TIM8ENC	TIM1ENC
	Reset value										0	0	0	0	0	0	0	0		0	0					0	0	0	0			0	0
0x1270	RCC_APB3ENCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFTENC	Res.	Res.
	Reset value																														0		
0x1274	RCC_APB4LENCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCAPBENC	RTCENC	VREFBUFENC	Res.	Res.	LPTIM5ENC	LPTIM4ENC	LPTIM3ENC	LPTIM2ENC	Res.	I2C4ENC	Res.	SPI6ENC	Res.	LPUART1ENC	HDPENC	Res.	Res.
	Reset value															0	0	0			0	0	0	0		0		0		0	0		
0x1278	RCC_APB4HENCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTSENC	BSECENC	SYSCFGENC
	Reset value																														0	0	0

Table 77. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x127C	RCC_APB5ENCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSIENC	VENCENC	GFXTIMENC	Res.	DCMIPPENC	LTDCENC	Res.
	Reset value																										0	0	0		0	0	
0x1280	Reserved	Reserved																															
0x1284	RCC_BUSLPENCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACLKNCLPENC	ACLKNLPENC
	Reset value																															0	0
0x1288	RCC_MISCLPENCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERLPENC	Res.	Res.	XSPIPHYCOMPLPENC	Res.	Res.	DBGLPENC
	Reset value																										0			0			0
0x128C	RCC_MEMLPENCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOTROMLPENC	VENCRAMLPENC	CACHEAXIRAMLPENC	FLEXRAMLPENC	AXISRAM2LPENC	AXISRAM1LPENC	BKPSRAMLPENC	AHBSRAM2LPENC	AHBSRAM1LPENC	AXISRAM6LPENC	AXISRAM5LPENC	AXISRAM4LPENC	AXISRAM3LPENC
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x1290	RCC_AHB1LPENCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADC12LPENC	GPDMA1LPENC	Res.	Res.	Res.	Res.
	Reset value																											0	0				
0x1294	RCC_AHB2LPENCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADF1LPENC	MDF1LPENC	Res.	Res.	Res.	RAMCFGLPENC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value															0	0				0												
0x1298	RCC_AHB3LPENCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RISAFLPENC	Res.	Res.	Res.	IACLPENC	RIFSCLPENC	PKALPENC	Res.	Res.	Res.	SAESLPENC	Res.	CRYPLPENC	HASHLPENC	RNGLPENC
	Reset value																		0				0	0	0				0		0	0	0

Table 77. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x129C	RCC_AHB4LPENCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCLPENC	PWRLPENC	Res.	GPIOQLPENC	GPIOPLPENC	GPIOOLPENC	GPIONLPENC						GPIOHLPENC	GPIOGLPENC	GPIOFLPENC	GPIOELPENC	GPIODLPENC	GPIOCLPENC	GPIOBLPENC	GPIOALPENC
	Reset value													0	0		0	0	0	0						0	0	0	0	0	0	0	0
0x12A0	RCC_AHB5LPENCR	NPULPENC	CACHEAXILPENC	OTG2LPENC	OTGPHY2LPENC	OTGPHY1LPENC	OTG1LPENC	ETH1LPENC	ETH1RXLPENC	ETH1TXLPENC	ETH1MACLPENC	Res.	GPU2DLPENC	GFXMMULPENC	MCE4LPENC	XSPI3LPENC	MCE3LPENC	MCE2LPENC	MCE1LPENC	XSPIMLPENC	XSPI2LPENC	Res.	Res.	Res.	SDMMC1LPENC	SDMMC2LPENC	PSSILPENC	XSPI1LPENC	FMCLPENC	JPEGLPENC	Res.	DMA2DLPENC	HPDMA1LPENC
	Reset value	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0				0	0	0	0	0	0		0	0
0x12A4	RCC_APB1LLPENCR	UART8LPENC	UART7LPENC	Res.	Res.	Res.	Res.	I3C2LPENC	I3C1LPENC	I2C3LPENC	I2C2LPENC	I2C1LPENC	UART5LPENC	UART4LPENC	USART3LPENC	USART2LPENC	SPDIFRX1LPENC	SPI3LPENC	SPI2LPENC	TIM11LPENC	TIM10LPENC	WWDGLPENC	Res.	LPTIM1LPENC	TIM14LPENC	TIM13LPENC	TIM12LPENC	TIM7LPENC	TIM6LPENC	TIM5LPENC	TIM4LPENC	TIM3LPENC	TIM2LPENC
	Reset value	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0
0x12A8	RCC_ APB1HLPENCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCPD1LPENC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FDCANLPENC	Res.	Res.	MDIOSLPENC	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value														0										0			0						
0x12AC	RCC_APB2LPENCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAI2LPENC	SAI1LPENC	SPI5LPENC	TIM9LPENC	TIM17LPENC	TIM16LPENC	TIM15LPENC	TIM18LPENC	Res.	SPI4LPENC	SPI1LPENC	Res.	Res.	Res.	Res.	USART10LPENC	UART9LPENC	USART6LPENC	USART1LPENC	Res.	Res.	Res.	TIM8LPENC	TIM1LPENC
	Reset value										0	0	0	0	0	0	0	0		0	0					0	0	0	0				0	0
0x12B0	RCC_APB3LPENCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFTLPENC	Res.	Res.
	Reset value																														0		
0x12B4	RCC_APB4LLPENCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTCAPBLPENC	RTCLPENC	VREFBUFLPENC	Res.	Res.	LPTIM5LPENC	LPTIM4LPENC	LPTIM3LPENC	LPTIM2LPENC	Res.	I2C4LPENC	Res.	SPI6LPENC	Res.	LPUART1LPENC	HDPLPENC	Res.	Res.
	Reset value															0	0	0			0	0	0	0		0		0		0	0		
0x12B8	RCC_ APB4HLPENCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTSLPENC	BSECLPENC	SYSCFGLPENC
	Reset value																														0	0	0

Table 77. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x12BC	RCC_APB5LPENCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSILPENC	VENCLPENC	GFXTIMLPENC	Res.	DCMIPPLPENC	LTDCLPENC	Res.
	Reset value																										0	0	0		0	0	
0x12C0-0x1780	Reserved	Reserved																															
0x1784	RCC_PRIVCFGCR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSEPRIVC	HSIPRIVC	MSIPRIVC	LSEPRIVC	LSIPRIVC
	Reset value																												0	0	0	0	0
0x1788	Reserved	Reserved																															
0x178C	RCC_PUBCFGCR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSEPUBC	HSIPUBC	MSIPUBC	LSEPUBC	LSIPUBC
	Reset value																												0	0	0	0	0
0x1790	Reserved	Reserved																															
0x1794	RCC_PRIVCFGCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL4PRIVC	PLL3PRIVC	PLL2PRIVC	PLL1PRIVC
	Reset value																													0	0	0	0
0x1798	Reserved	Reserved																															
0x179C	RCC_PUBCFGCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PLL4PUBC	PLL3PUBC	PLL2PUBC	PLL1PUBC
	Reset value																													0	0	0	0
0x17A0	Reserved	Reserved																															
0x17A4	RCC_PRIVCFGCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC20PRIVC	IC19PRIVC	IC18PRIVC	IC17PRIVC	IC16PRIVC	IC15PRIVC	IC14PRIVC	IC13PRIVC	IC12PRIVC	IC11PRIVC	IC10PRIVC	IC9PRIVC	IC8PRIVC	IC7PRIVC	IC6PRIVC	IC5PRIVC	IC4PRIVC	IC3PRIVC	IC2PRIVC	IC1PRIVC
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x17A8	Reserved	Reserved																															
0x17AC	RCC_PUBCFGCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC20PUBC	IC19PUBC	IC18PUBC	IC17PUBC	IC16PUBC	IC15PUBC	IC14PUBC	IC13PUBC	IC12PUBC	IC11PUBC	IC10PUBC	IC9PUBC	IC8PUBC	IC7PUBC	IC6PUBC	IC5PUBC	IC4PUBC	IC3PUBC	IC2PUBC	IC1PUBC
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x17B0	Reserved	Reserved																															
0x17B4	RCC_PRIVCFGCR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DFTPRIVC	RSTPRIVC	INTPRIVC	PERPRIVC	BUSPRIVC	SYSPRIVC	MODPRIVC
	Reset value																										0	0	0	0	0	0	0
0x17B8	Reserved	Reserved																															

Table 77. RCC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x17BC	RCC_PUBCFGCR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTPUBC	INTPUBC	PERPUBC	BUSPUBC	SYSPUBC	MODPUBC
	Reset value																											0	0	0	0	0	0
0x17C0	Reserved	Reserved																															
0x17C4	RCC_PRIVCFGCR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NOCPRIVC	APB5PRIVC	APB4PRIVC	APB3PRIVC	APB2PRIVC	APB1PRIVC	AHB5PRIVC	AHB4PRIVC	AHB3PRIVC	AHB2PRIVC	AHB1PRIVC	AHBMPRIVC	ACLKNCPRIVC	ACLKNPRIVC
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x17C8	Reserved	Reserved																															
0x17CC	RCC_PUBCFGCR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NOCPUBC	APB5PUBC	APB4PUBC	APB3PUBC	APB2PUBC	APB1PUBC	AHB5PUBC	AHB4PUBC	AHB3PUBC	AHB2PUBC	AHB1PUBC	AHBMPUBC	ACLKNCPUBC	ACLKNPUBC
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x17D0	RCC_PUBCFGCR5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VENCRAMPUBC	CACHEAXIRAMPUBC	FLEXRAMPUBC	AXISRAM2PUBC	AXISRAM1PUBC	BKPSRAMPUBC	AHBSRAM2PUBC	AHBSRAM1PUBC	AXISRAM6PUBC	AXISRAM5PUBC	AXISRAM4PUBC	AXISRAM3PUBC
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0

15 General-purpose I/Os (GPIO)

15.1 Introduction

Each general-purpose I/O port has four 32-bit configuration registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, and GPIOx_PUPDR), two 32-bit data registers (GPIOx_IDR and GPIOx_ODR), a 16-bit reset register (GPIOx_BRR), and a 32-bit set/reset register (GPIOx_BSRR).

In addition, all GPIOs have a 32-bit locking register (GPIOx_LCKR), two couples of 32-bit advanced configuration registers (GPIOx_DELAYRL/H, GPIOx_ADVCFGRL/H), and two 32-bit alternate function selection registers (GPIOx_AFRH and GPIOx_AFRL).

Access to each general-purpose I/O configuration bit can be restricted to secure-only and/or privileged-only.

15.2 GPIO main features

- Multiple choice of configurations per I/O port:
  - Input configuration in floating, pull-up/down, or analog state
  - Analog configuration (output buffer and Schmitt trigger input disabled)
  - Output configuration, or alternate function configuration, in push-pull or open drain state, with pull-up or pull-down activated
- Data present on the I/O pin sampled to the input data register GPIOx_IDR (input configuration) or to the peripheral (alternate function configuration)
- Output buffer on the I/O pin driven by the output data register GPIOx_ODR (output configuration) or by the peripheral (alternate function configuration)
- I/O data output atomic read/modify through GPIOx_ BSRR and GPIOx_BRR
- Speed selection for each I/O (GPIOx_OSPEEDR)
- Lock mechanism (GPIOx_LCKR) to selectively freeze the I/O port configurations
- Highly-flexible pin multiplexing, enabling the use of I/O pins as GPIOs, or as one of several possible peripheral functions
- Programmable delay to the input or the output path (GPIOx_DELAYR)
- Double edge selection, clock inversion, and optional retime (GPIOx_ADVCFGR)
- Possibility to restrict each I/O control to secure-only and/or privileged-only

15.3 GPIO functional description

The specific hardware characteristics of each I/O port are detailed in the datasheet.

Each port bit of the GPIO ports can be individually configured by software in several modes:

- input floating
- input pull-up
- input-pull-down
- analog
- output open-drain with pull-up or pull-down capability

- output push-pull with pull-up or pull-down capability
- alternate function push-pull with pull-up or pull-down capability
- alternate function open-drain with pull-up or pull-down capability

Each I/O port bit is freely programmable, however the I/O port registers must be accessed as 32-bit words, half-words, or bytes. GPIOx_BSRR and GPIOx_BRR allow atomic read/modify accesses to any of GPIOx_ODR. In this way, there is no risk of IRQ occurring between the read and the modify access.

Figure 66 shows the basic structure of a standard I/O port bit.

Figure 66. Basic structure of an I/O port bit

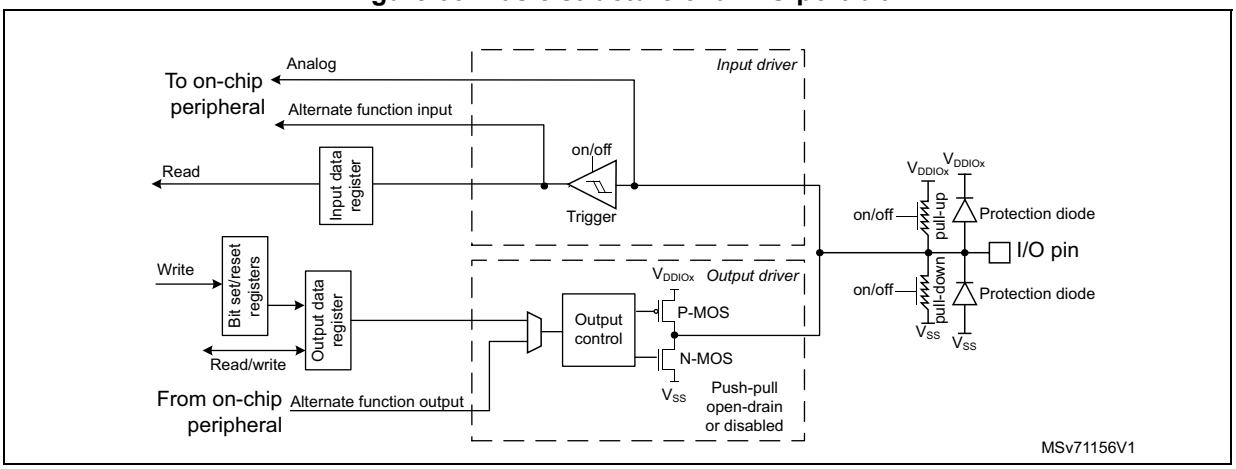


Table 78 gives the possible port bit configurations.

Table 78. Port x bit configurations

Port x bit configurations					I/O configuration(1)	
MODEx[1:0]	OTx	OSPEEDx[1:0]	PUPDx[1:0]			
01	0	SPEED[1:0]	0	0	GP output	PP
			0	1		PP + PU
			1	0		PP + PD
			1	1	Reserved (GP output PP)	
	1		0	0	GP output (open drain)	OD
			0	1		OD + PU
			1	0		OD + PD
			1	1	Reserved (GP output OD)	

Table 78. Port x bit configurations (continued)

Port x bit configurations						I/O configuration(1)	
MODEx[1:0]	OTx	OSPEEDx[1:0]		PUPDx[1:0]			
10	0	SPEED[1:0]		0	0	AF	PP
				0	1		PP + PU
				1	0		PP + PD
				1	1	Reserved (AF PP)	
	1			0	0	AF (open drain)	OD
				0	1		OD + PU
				1	0		OD + PD
				1	1	Reserved (AF OD)	
00	x	x	x	0	0	Input	Floating
	x	x	x	0	1		PU
	x	x	x	1	0		PD
	x	x	x	1	1	Reserved (input floating)	
11	x	x	x	0	0	Input/output	Analog
	x	x	x	0	1		Analog
	x	x	x	1	0		Analog + PD
	x	x	x	1	1		Analog

1. GP = general-purpose, PP = push-pull, PU = pull-up, PD = pull-down, OD = open-drain, AF = alternate function.

15.3.1 General-purpose I/O (GPIO)

During and just after reset, AFs (alternate functions) are not active, and most of the I/O ports are configured in analog mode, as defined through GPIOx_MODER register.

The debug pins are in AF pull-up/pull-down after reset:

- PA15: JTDI in pull-up
- PA14: JTCK/SWCLK in pull-down
- PA13: JTMS/SWDIO in pull-up
- PB4: NJTRST in pull-up
- PB5: JTDO/TRACESWO in floating state no pull-up/pull-down

When the pin is configured as output, the value written to GPIOx_ODR is output on the I/O pin. The output driver can be used in push-pull mode or open-drain mode (only the low level is driven, high level is Hi-Z).

GPIOx_IDR captures the data present on the I/O pin at every AHB clock cycle.

All GPIO pins have weak internal pull-up and pull-down resistors, that can be activated or not, depending on the value in GPIOx_PUPDR.

15.3.2 I/O pin AF multiplexer and mapping

The device I/O pins are connected to on-board peripherals/modules through a multiplexer used to connect only one peripheral AF to an I/O pin at a time. There is no conflict between peripherals available on the same I/O pin.

Each I/O pin has a multiplexer with up to 16 AF inputs (AF0 to AF15), which can be configured through GPIOx_AFRL (for pins 0 to 7) and GPIOx_AFRH (for pins 8 to 15):

- During and just after reset, AFs are not active and most of the I/O ports are configured in analog mode, as defined through GPIOx_MODER. When one AF is active, AF0 is selected by default.
- Specific AF assignments for each I/O pin are detailed in the device datasheet.

This flexible I/O multiplexing architecture is used to optimize the number of peripherals available in smaller packages.

To use an I/O in a given configuration, the user must proceed as follows:

- Debug function: after each device reset, these pins are assigned as AF pins immediately usable by the debugger host.
- GPIO modes: the desired I/O is configured as output, input, or analog in GPIOx_MODER.
- Peripheral AF:
  - For a given peripheral function, the ‘Alternate functions’ table in the datasheet helps to identify the AF to use with a given I/O. One of GPIOx_AFRL or GPIOx_AFRH must be used.
  - Configure the desired I/O as an AF in GPIOx_MODER. Select the type, pull-up/pull-down, and output speed via GPIOx_OTYPER, GPIOx_PUPDR, and GPIOx_OSPEEDR respectively.
- Cortex®-M55 alternate function (EVENTOUT)
  - The output EVENTOUT signal can be used by configuring the I/O pin to output at AF15. An event can be signaled through the configured pin after executing SEV instruction.
  - EVENTOUT signal can be used internally as a trigger for some peripherals (see *Section 17: Peripherals interconnect matrix*)
- Additional functions:
  - The additional functions listed in the datasheet give the peripheral functions that directly select/enable its allocated I/O port(s). Typical peripherals are RTC/TAMP, PWR (WKUPx pins), or RCC (oscillator pins). These functions have priority over the configuration in standard GPIO registers.
  - For ADCs, configure the desired I/O in analog mode in GPIOx_MODER, and configure the required function in ADC registers.

15.3.3 I/O port control registers

Each GPIO port has four 32-bit memory-mapped control registers to configure up to 16 I/Os:

- GPIOx_MODER is used to select the I/O mode (input, output, AF, analog).
- GPIOx_OTYPER and GPIOx_OSPEEDR are respectively used to select the output type (push-pull or open-drain) and the speed.
- GPIOx_PUPDR is used to select the pull-up/pull-down whatever the I/O direction.

15.3.4 I/O port data registers

Each GPIO port has two 16-bit memory-mapped data registers: GPIOx_ODR. These read/writable register store the data to be output per I/O. Data input through each I/O are stored into read-only GPIOx_IDR.

15.3.5 I/O data bitwise handling

GPIOx_BSRR is a 32-bit register that allows the application to set/reset each individual bit in GPIOx_ODR). GPIOx_BSRR has twice the size of GPIOx_ODR.

To each bit in GPIOx_ODR, correspond two control bits in GPIOx_BSRR: BS(i) and BR(i). When written to 1, BS(i) sets the corresponding ODR(i). When written to 1, BR(i) resets the corresponding ODR(i).

Writing any bit to 0 in GPIOx_BSRR does not have any effect on the corresponding bit in GPIOx_ODR. If there is an attempt to both set and reset a bit in GPIOx_BSRR, the set action takes priority.

Using GPIOx_BSRR to change the values of individual bits in GPIOx_ODR is a one-shot effect that does not lock GPIOx_ODR bits: these bits can always be accessed directly. GPIOx_BSRR provides a way of performing atomic bitwise handling.

The software does not need to disable interrupts when programming GPIOx_ODR at bit level: one or more bits can be modified in a single atomic AHB write access.

15.3.6 GPIO locking mechanism

GPIO control registers can be frozen by applying a specific write sequence to GPIOx_LCKR. Indeed, each GPIOx_LCKR bit freezes the corresponding bit in the IO ports control registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_DELAYRL/H, GPIOx_ADVCFGR, and GPIOx_AFRL/H).

To write GPIOx_LCKR, a specific write/read sequence must be applied. When the right lock sequence is applied to bit 16 in this register, the value of LCKR[15:0] is used to lock the configuration of the I/Os (during the write sequence the LCKR[15:0] value must be the same, during the write sequence, only the IOs for which LCKR[15:0] stays to 1 for all the specific write sequence are locked; other IOs configuration are not (lock sequence aborted). When the lock sequence is applied to a port bit, the value of the port bit can no longer be modified until a next device or peripheral reset. Each GPIOx_LCKR bit freezes the corresponding bit in the control registers (GPIOx_MODER, GPIOx_OTYPER, GPIOx_OSPEEDR, GPIOx_PUPDR, GPIOx_DELAYRL/H, GPIOx_ADVCFGR, and GPIOx_AFRL/H).

The LOCK sequence can be performed only using a word (32-bit long) access to GPIOx_LCKR, because GPIOx_LCKR bit 16 must be set at the same time as the [15:0] bits.

Note: A lock sequence aborts in the event of an AHB bus error (see Section 15.1), or insufficient access rights (see Section 15.3.16).

A lock sequence automatically suspends when accessing a different register from GPIOx_LCKR. It resumes with the next access to GPIOx_LCKR.

15.3.7 I/O AF input/output

Two registers are provided to select one of the AF inputs/outputs available for each I/O: GPIOx_AFRL and GPIOx_AFRH. With these registers, the user can connect an AF to some other pin, as required by the application.

A number of possible peripheral functions are multiplexed on each GPIO using these registers. The application can select any one of the possible functions for each I/O. As the AF selection signal is common to the AF input and output, a single channel is selected for the AF input/output of a given I/O.

To know which functions are multiplexed on each GPIO pin, refer to the datasheet.

15.3.8 External interrupt/wake-up lines

All ports have external interrupt capability. To use external interrupt lines, the ports can be configured in input, output, or AF mode (the ports must not be configured in analog mode). Refer to *Section 25: Extended interrupts and event controller (EXTI)* for more details.

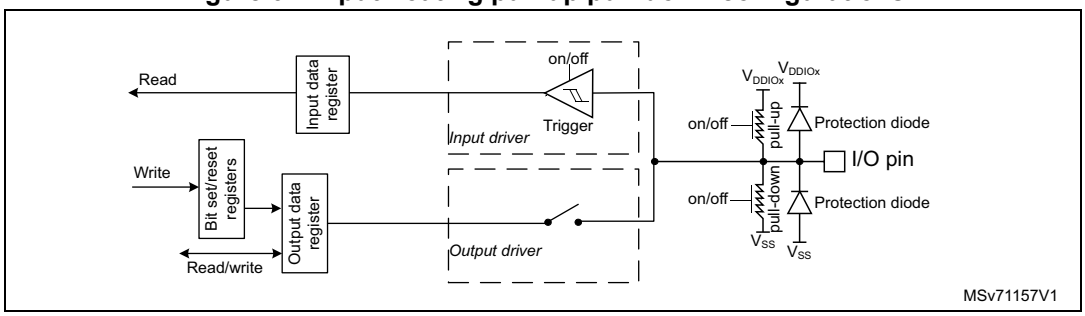
15.3.9 Input configuration

When the I/O port is programmed as input:

- the output buffer is disabled
- the Schmitt trigger input is activated
- the pull-up and pull-down resistors are activated, depending on the value in GPIOx_PUPDR.
- data present on the I/O pin are sampled into the input data register every AHB clock cycle
- a read access to the input data register provides the I/O state

Figure 67 shows the input configuration of the I/O port bit.

Figure 67. Input floating/pull-up/pull-down configurations



15.3.10 Output configuration

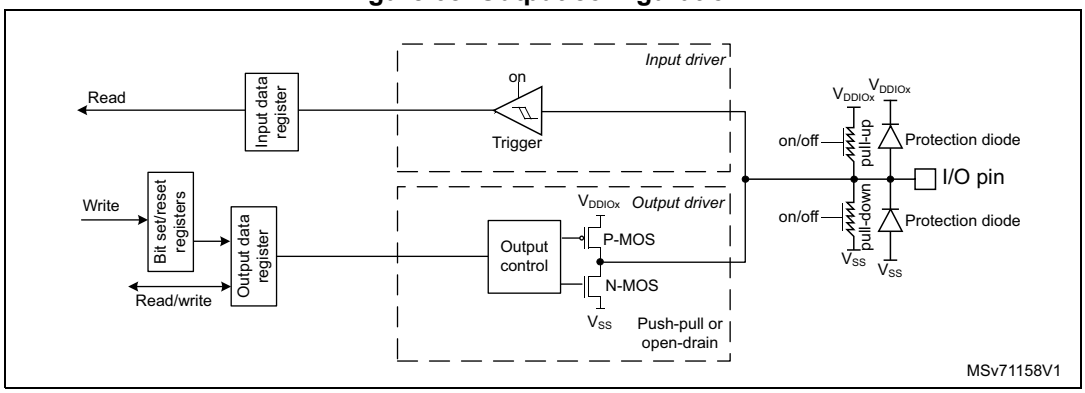
When the I/O port is programmed as output:

- The output buffer is enabled:
  - open-drain mode: a 0 in the output register activates the N-MOS, whereas a 1 in the output register leaves the port in Hi-Z (P-MOS never activated).
  - push-pull mode: a 0 in the output register activates the N-MOS, whereas a 1 in the output register activates the P-MOS.

- The Schmitt trigger input is activated.
- The pull-up and pull-down resistors are activated depending on the value in GPIOx_PUPDR.
- Data present on the I/O pin are sampled into the input data register every AHB clock cycle.
- A read access to the input data register gets the I/O state.
- A read access to the output data register gets the last written value.

Figure 68 shows the output configuration of the I/O port bit.

Figure 68. Output configuration



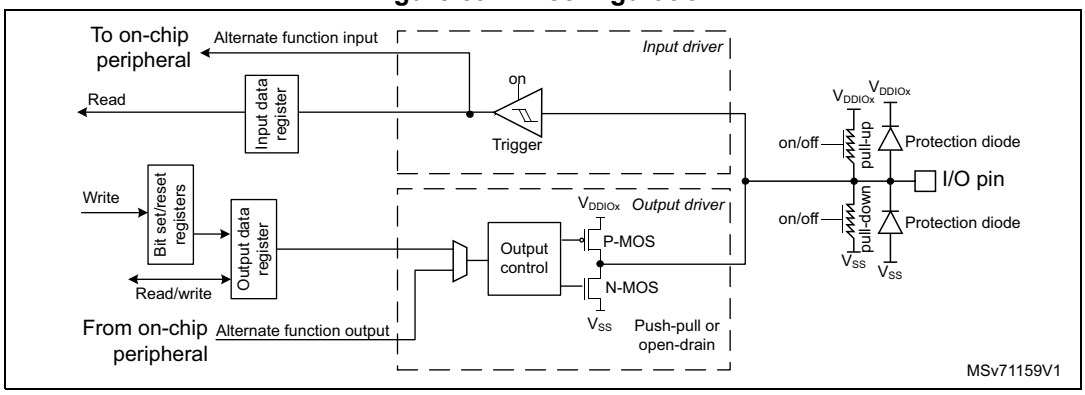
15.3.11 AF configuration

When the I/O port is programmed as AF:

- the output buffer can be configured in open-drain or push-pull mode
- the output buffer is driven by the signals coming from the peripheral (transmitter enable and data)
- the Schmitt trigger input is activated
- the weak pull-up and pull-down resistors are activated or not, depending on the value in GPIOx_PUPDR
- data present on the I/O pin are sampled into the input data register every AHB clock cycle
- aread access to the input data register gets the I/O state.

Figure 69 shows the AF configuration of the I/O port bit.

Figure 69. AF configuration



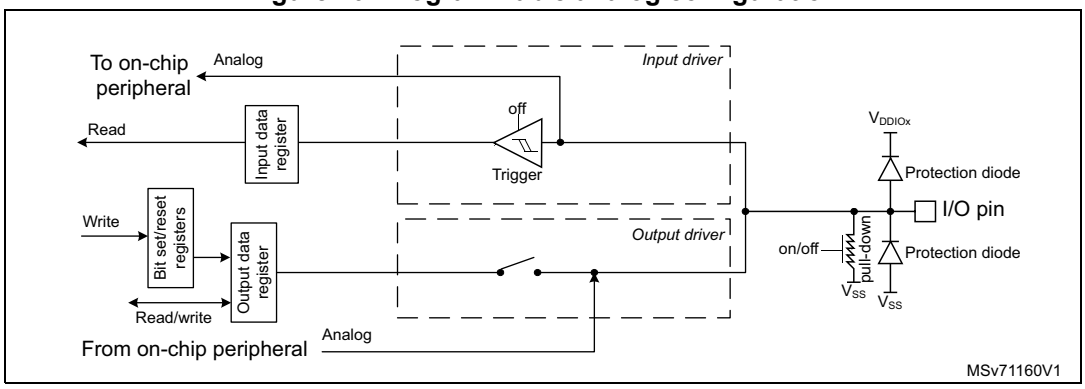
15.3.12 Analog configuration

When the I/O port is programmed as analog configuration:

- The output buffer is disabled.
- The Schmitt trigger input is deactivated, providing zero consumption for every analog value of the I/O pin. The output of the Schmitt trigger is forced to a constant value (0).
- The weak pull-up resistor is disabled by hardware. The weak pull-down resistor is activated or not, depending on the value in GPIOx_PUPDR.
- Read access to the input data register gets the value 0.

Figure 70 shows the programmable, analog-input configuration of the I/O port bits.

Figure 70. Programmable analog configuration



ADC analog configuration

The ADC voltage is 1.8 V. Refer to the product datasheet for more details on the allowed voltage on ADC input pins.

Warning: Permanent damage to the ADC can occur if the voltage on input pins exceeds 1.8 V.

15.3.13 Using HSE or LSE oscillator pins as GPIOs

When the HSE or LSE oscillator is switched off (default state after reset), the related oscillator pins can be used as normal GPIOs, but only in input mode (output mode is not supported).

When the HSE or LSE oscillator is switched on (HSEON or LSEON set in RCC_CSR), the oscillator takes control of its associated pins, and the GPIO configuration of these pins has no effect.

When the oscillator is configured in a user external clock mode, only the pin is reserved for clock input (OSC_OUT or OSC32_OUT pins can still be used, in input mode only).

15.3.14 Using GPIO pins in V_{SW} supply domain

The functionality of PC13, PC14, PC15, and PQ7 GPIOs is lost when the core supply domain is powered off (the device enters Standby mode). If their GPIO configuration is not bypassed by the RTC/TAMP configuration or by the RCC (LSE), these pins are set in analog input mode.

For details about I/O control by the RTC, refer to *Section 61.3: RTC functional description*.

15.3.15 Advanced I/O configurations

I/O compensation cell

The I/O commutation slew rate (t_{fall}/t_{rise}) can be adapted by software depending on process, voltage, and temperature conditions, to reduce the I/O noise on the power supply. Refer to *Section 16: System configuration controller (SYSCFG)* for more details.

Input/output path configurable delay

A delay programmed in GPIOx_DELAYRL/H can be applied to either input or output path, based on DLYPATH in GPIOx_ADVCFGRL/R.

Input/output data double edge selection

Input and output data can be managed as single- or double-edge, according to DE in GPIOx_ADVCFGRL/R.

Input/output data retiming

When RET is set in GPIOx_ADVCFGRL/R, input and output data are retimed to either rising or falling clock edge, depending upon INVCLK value.

15.3.16 I/O pin isolation using TrustZone

As TrustZone-aware peripheral, each I/O pin of GPIO port x can be individually configured as secure through GPIOx_SECCFGR. After reset, all GPIO ports are secure.

The I/Os connected to peripheral functions can be conditioned by the peripheral security configuration:

- Peripherals for which the I/O pin selection is done through GPIOx_AFRL/H

  If the peripheral is configured as secure, it cannot be connected to a nonsecure I/O pin. If this is not respected, input data to the secure peripheral are forced to 0 (I/O input pin value ignored), and the output pin value is forced to 0, thus avoiding any secure

information leak through nonsecure I/Os. TrustZone-aware logic around GPIO ports is summarized in *Table 79*.

- Peripherals (like ADC) with embedded analog functions that directly select/enable its allocated I/O ports using analog switches

  If the I/O is secure, this analog switch cannot be controlled by a nonsecure peripheral. If the peripheral is configured as nonsecure and the I/O is secure, the switch remains open. This prevents the redirection of secure data to a nonsecure peripheral or I/O through analog path. Refer to *Section 3: System security* for the list of peripherals using this security.
- The list of I/Os without a hardware protection linked to TrustZone is given in *Section 3: System security*. More specifically, the listed signals (input and/or outputs) are not blocked when the I/O is set as secure, and the associated peripheral is non secure. For each of these listed I/Os, a secure application must decide if a potential effect on data integrity or confidentiality is critical or not.

Refer to the pins definitions table in the datasheet for more information about AFs and additional functions mapping.

Table 79. Secure AF between peripherals and allocated I/Os

Security configuration		AF logic		Comment
Peripheral	**Allocated I/O pin**	**Input**	**Output**	
Secure	Secure	I/O data	Peripheral data	-
Nonsecure				Out of reset configuration
Secure	Nonsecure	Zero	Zero	-
Nonsecure		I/O data	Peripheral data	

Table 80 gives a summary of the I/O port y secured bits, following the setting of SECy in GPIOx_SECCFGR. The following is valid for each register in the table:

- When a bit is secured, read/write operations are only allowed by a secure access. Non-secure read or write accesses on secured bits are RAZ/WI, with no illegal access event generated.
- When a bit is nonsecure, there is no TrustZone restriction. Read/write operations are allowed by both secure and nonsecure accesses.

Table 80. GPIO secured bits

Secured bitfield(s)	Register name	Nonsecure access on secure bits
MODEy[1:0]	GPIOx_MODER	RAZ/WI
OTy	GPIOx_OTYPER	
OSPEEDy[1:0]	GPIOx_OSPEEDR	
PUPDy[1:0]	GPIOx_PUPDR	
IDy	GPIOx_IDR	
ODy	GPIOx_ODR	
BSy and BRy	GPIOx_BSRR	

Table 80. GPIO secured bits (continued)

Secured bitfield(s)	Register name	Nonsecure access on secure bits
BRy	GPIOx_BRR	RAZ/WI
LCKy	GPIOx_LCKR	
AFSELy[3:0]	GPIOx_AFRH	
	GPIOx_AFRL	
DELAYy[3:0]	GPIOx_DELAYRL	
	GPIOx_DELAYRH	
RETy, INVCLKy, DEy, and DLYPATHy	GPIOx_ADVCFGRL	
	GPIOx_ADVCFGRH	

Note: GPIOx_SECCFGR is readable by any application. Each bit in this register is write-locked until the next device reset, when setting the corresponding bit in GPIOx_RCFGLOCKR.

15.3.17 I/O pin isolation using privilege

As RIF-aware peripheral, each I/O pin of GPIO port x can be individually configured as privileged through GPIOx_PRIVCFGR. After reset, all GPIO registers can be read and written by privileged and unprivileged accesses, if security is enough and the CPU is allowed.

Table 80 gives a summary of the I/O port y protected bits, following the setting of PRIVy in GPIOx_PRIVCFGR. The following is valid for each register in the table:

- When a bit is privileged, read/write operations are only allowed by a privileged access. Unprivileged read or write accesses on privileged bits are RAZ/WI, with no illegal access event generated.
- When a bit is unprivileged, there is no privilege restriction. Read and write operations are allowed by both privileged and unprivileged accesses.

Note: GPIOx_PRIVCFGR is readable by any application, writable by only privileged applications. Each bit in GPIOx_PRIVCFGR is write locked until next device reset, when setting the corresponding bit in GPIOx_RCFGLOCKR.

15.4 GPIO registers

GPIO registers are accessible through 32-bit word, half-word, or byte single access. Double-word and instruction fetch accesses trigger an AHB bus error. Byte or half-word writes to GPIOx_LCKR are silently ignored (no bus error).

15.4.1 GPIO port x mode register (GPIOx_MODER) (x = A to H, N to Q)

Address offset:0x00

Reset value: Port A: 0xABFF EFFF

Reset value: Port B: 0xFFFF FAFF

Reset value: 0xFFFF FFFF (for the other ports)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODE15[1:0]		MODE14[1:0]		MODE13[1:0]		MODE12[1:0]		MODE11[1:0]		MODE10[1:0]		MODE9[1:0]		MODE8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODE7[1:0]		MODE6[1:0]		MODE5[1:0]		MODE4[1:0]		MODE3[1:0]		MODE2[1:0]		MODE1[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MODEy[1:0]:** Port x configuration I/O pin y (y = 15 to 0)

These bits are written by software to configure the I/O mode.

00: General-purpose input mode

01: General-purpose output mode

10: Alternate function mode (refer to device datasheet for available options)

11: Analog mode (reset state)

Note: Each bitfield is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

Successful read or write to MODEy must be secure if SECy = 1 in GPIOx_SECCFGR, and must be privileged if PRIVy = 1 in GPIOx_PRIVCFGR.

15.4.2 GPIO port x output type register (GPIOx_OTYPER) (x =A to H, N to Q)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **OTy:** Port x configuration I/O pin y (y = 15 to 0)

These bits are written by software to configure the I/O output type.

0: Output push-pull (reset state)

1: Output open-drain

Note: Each bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

Successful read or write to OTy must be secure if SECy = 1 in GPIOx_SECCFGR, and must be privileged if PRIVy = 1 in GPIOx_PRIVCFGR.

15.4.3 GPIO port x output speed register (GPIOx_OSPEEDR) (x =A to H, N to Q)

Address offset: 0x08

Reset value: Port A: 0x0C00 0000

Reset value: Port B: 0x0000 0C00

Reset value: 0x0000 0000 (for the other ports)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSPEED15[1:0]		OSPEED14[1:0]		OSPEED13[1:0]		OSPEED12[1:0]		OSPEED11[1:0]		OSPEED10[1:0]		OSPEED9[1:0]		OSPEED8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSPEED7[1:0]		OSPEED6[1:0]		OSPEED5[1:0]		OSPEED4[1:0]		OSPEED3[1:0]		OSPEED2[1:0]		OSPEED1[1:0]		OSPEED0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **OSPEEDy[1:0]**: Port x configuration I/O pin y (y = 15 to 0)

These bits are written by software to configure the I/O output speed.

00: Low speed

01: Medium speed

10: High speed

11: Very high speed

Note: Refer to the device datasheet for frequency specifications, power supply, and load conditions for each speed. Each bitfield is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

Successful read or write to OSPEEDy must be secure if SECy = 1 in GPIOx_SECCFGR, and must be privileged if PRIVy = 1 in GPIOx_PRIVCFGR.

15.4.4 GPIO port x pull-up/pull-down register (GPIOx_PUPDR) (x= A to H, N to Q)

Address offset: 0x0C

Reset value: Port A: 0x6400 0000

Reset value: Port B: 0x0000 0100

Reset value: 0x0000 0000 (for the other ports)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PUPD15[1:0]		PUPD14[1:0]		PUPD13[1:0]		PUPD12[1:0]		PUPD11[1:0]		PUPD10[1:0]		PUPD9[1:0]		PUPD8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PUPD7[1:0]		PUPD6[1:0]		PUPD5[1:0]		PUPD4[1:0]		PUPD3[1:0]		PUPD2[1:0]		PUPD1[1:0]		PUPD0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PUPDy[1:0]:** Port x configuration I/O pin y (y = 15 to 0)

These bits are written by software to configure the I/O pull-up or pull-down

00: No pull-up, pull-down

01: Pull-up

10: Pull-down

11: Reserved

Note: Each bitfield is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

Successful read or write to PUPDy must be secure if SECy = 1 in GPIOx_SECCFGR, and must be privileged if PRIVy = 1 in GPIOx_PRIVCFGR.

15.4.5 GPIO port x input data register (GPIOx_IDR) (x = A to H, N to Q)

Address offset: 0x10

Reset value: 0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **IDy:** Port x input data I/O pin y (y = 15 to 0)

These bits are read-only. They contain the input value of the corresponding I/O port.

Note: Each bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

Successful read to IDy must be secure if SECy = 1 in GPIOx_SECCFGR, and must be privileged if PRIVy = 1 in GPIOx_PRIVCFGR.

15.4.6 GPIO port x output data register (GPIOx_ODR) (x = A to H, N to Q)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ODy:** Port x output data I/O pin y (y = 15 to 0)

These bits can be read and written by software.

Note: For atomic bit set/reset, OD bits can be individually set and/or reset by writing to GPIOx_BSRR or GPIOx_BRR. Each bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

Successful read or write to ODy must be secure if SECy= 1 in GPIOx_SECCFGR, and must be privileged if PRIVy = 1 in GPIOx_PRIVCFGR.

15.4.7 GPIO port x bit set/reset register (GPIOx_BSRR) (x = A to H, N to Q)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 **BRy:** Port x reset I/O pin y (y = 15 to 0)

These bits are write-only. A read to these bits returns 0x0000.

0: No action on the corresponding ODy bit

1: Resets the corresponding ODy bit

Note: If both BSy and BRy are set, BSy has priority.

Each bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

Successful write to BRy must be secure if SECy = 1 in GPIOx_SECCFGR, and must be privileged if PRIVy = 1 in GPIOx_PRIVCFGR.

Bits 15:0 **BSy:** Port x set I/O pin y (y = 15 to 0)

These bits are write-only. A read to these bits returns 0x0000.

0: No action on the corresponding ODx bit

1: Sets the corresponding ODx bit

Note: Each bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

Successful write to BSy must be secure if SECy = 1 in GPIOx_SECCFGR, and must be privileged if PRIVy = 1 in GPIOx_PRIVCFGR.

15.4.8 GPIO port x configuration lock register (GPIOx_LCKR) (x = A to H, N to Q)

Address offset: 0x1C

Reset value: 0x0000 0000

This register is used to lock the configuration of the port bits when a correct write sequence is applied to bit 16 (LCKK). The value of bits [15:0] is used to lock the GPIO configuration. During the lock sequence, only bit port unlock can be changed.

Note: Only word access (32-bit long) is allowed during this locking sequence.

Byte or half-word writes are silently ignored (no bus error, no abort to lock sequence).

When the LOCK sequence has been applied on a port bit, the value of this port bit can no longer be modified until the next MCU reset or peripheral reset. Each lock bit freezes a specific configuration register (control and AF registers). See *Section 15.3.6* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKK
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **LCKK:** Lock key

This bit can be read any time. It can only be modified using the lock key write sequence.

0: Port configuration lock key not active

1: Port configuration lock key active. The GPIOx_LCKR register is locked until the next MCU reset or peripheral reset.

Lock key write sequence:

WR LCKR[16] = 1 + LCKR[15:0]

WR LCKR[16] = 0 + LCKR[15:0]

WR LCKR[16] = 1 + LCKR[15:0]

RD LCKR

RD LCKR[16] = 1 (this read operation is optional but it confirms that the lock is active)

Note: Any error in the sequence aborts the lock. After the first lock sequence on any bit of the port, any read access on LCKK bit returns 1 until the next device or peripheral reset.

Bits 15:0 **LCKy:** Port x lock I/O pin y (y = 15 to 0)

These bits are read/write but can only be written when LCKK = 0.

0: Port configuration not locked

1: Port configuration locked

Note: Each bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

Successful read or write to LCKy must be secure if SECy = 1 in GPIOx_SECCFGR, and must be privileged if PRIVy = 1 in GPIOx_PRIVCFGR.

15.4.9 GPIO port x AF low register (GPIOx_AFRL) (x = A to H, N to Q)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL7[3:0]				AFSEL6[3:0]				AFSEL5[3:0]				AFSEL4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL3[3:0]				AFSEL2[3:0]				AFSEL1[3:0]				AFSEL0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AFSELy[3:0]:** AF selection for port x I/O pin y (y = 7 to 0)

These bits are written by software to configure AF I/Os.

0000: AF0
0001: AF1
0010: AF2
0011: AF3
0100: AF4
0101: AF5
0110: AF6
0111: AF7
1000: AF8
1001: AF9
1010: AF10
1011: AF11
1100: AF12
1101: AF13
1110: AF14
1111: AF15

Note: Each bitfield is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

Successful read or write to the bitfield AFSELy must be secure if SECy = 1 in GPIOx_SECCFGR, and must be privileged if PRIVy =1 in GPIOx_PRIVCFGR.

15.4.10 GPIO port x AF high register (GPIOx_AFRH) (x = A to H, N to Q)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFSEL15[3:0]				AFSEL14[3:0]				AFSEL13[3:0]				AFSEL12[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFSEL11[3:0]				AFSEL10[3:0]				AFSEL9[3:0]				AFSEL8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AFSELy[3:0]:** AF selection for port x I/O pin y (y = 15 to 8)

These bits are written by software to configure AF I/Os (see GPIOx_AFRL for details).

Note: Each bitfield is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

Successful read or write to the bitfield AFRHy must be secure if SECy = 1 in GPIOx_SECCFGR, and must be privileged if PRIVy = 1 in GPIOx_PRIVCFGR.

15.4.11 GPIO port x bit reset register (GPIOx_BRR) (x = A to H, N to Q)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **BRy:** Port x reset IO pin y (y = 15 to 0)

These bits are write-only. A read to these bits returns 0x0000.

0: No action on the corresponding ODx bit

1: Reset the corresponding ODx bit

Note: Each bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

Successful write to BRy must be secure if SECy = 1 in GPIOx_SECCFGR, and must be privileged if PRIVy = 1 in GPIOx_PRIVCFGR.

15.4.12 GPIO port x secure configuration register (GPIOx_SECCFGR) (x = A to H, N to Q)

Address offset: 0x30

Reset value: Port A: 0x0000 FFFF

Reset value: Port B: 0x0000 FFFF

Reset value: Port C: 0x0000 FFFF

Reset value: Port D: 0x0000 FFFF

Reset value: Port E: 0x0000 FFFF

Reset value: Port F: 0x0000 FFFF

Reset value: Port G: 0x0000 FFFF

Reset value: Port H: 0x0000 03FF

Reset value: Port N: 0x0000 1FFF

Reset value: Port O: 0x0000 003F

Reset value: Port P: 0x0000 FFFF

Reset value: Port Q: 0x0000 00FF

This register is used to configure a selected I/O as secure. Nonsecure or unprivileged writes to this register are ignored, while any read is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **SECy**: I/O pin y of Port x security configuration (y = 15 to 0)

0: The I/O pin y is nonsecure. Secure access is also possible.

1: The I/O pin y is secure (see *Table 80* for all corresponding secure bits).

Note: Each bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

15.4.13 GPIO port x privileged configuration register (GPIOx_PRIVCFGR) (x = A to H, N to Q)

Address offset: 0x34

Reset value: Port A: 0x0000 FFFF

Reset value: Port B: 0x0000 FFFF

Reset value: Port C: 0x0000 FFFF

Reset value: Port D: 0x0000 FFFF

Reset value: Port E: 0x0000 FFFF

Reset value: Port F: 0x0000 FFFF

Reset value: Port G: 0x0000 FFFF

Reset value: Port H: 0x0000 03FF

Reset value: Port N: 0x0000 1FFF

Reset value: Port O: 0x0000 003F

Reset value: Port P: 0x0000 FFFF

Reset value: Port Q: 0x0000 00FF

Unprivileged writes to this register are ignored, while any read is allowed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PRIVy**: I/O pin y of Port x privilege configuration (y = 15 to 0)

0: The I/O pin y is unprivileged. Privileged access is also possible.

1: The I/O pin y is privileged only (see *Table 80* for all corresponding protected bits).

If the corresponding SECy = 1 in GPIOx_SECCFGR, PRIVy can only be written by a secure privileged application.

Note: Each bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

15.4.14 GPIO port x resource configuration lock register (GPIOx_RCFGLOCKR) (x = A to H, N to Q)

Address offset: 0x38

Reset value: 0x0000 0000

Secure privileged write access only. Write only by trusted domain CID. Any read access is allowed on this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RLOCK15	RLOCK14	RLOCK13	RLOCK12	RLOCK11	RLOCK10	RLOCK9	RLOCK8	RLOCK7	RLOCK6	RLOCK5	RLOCK4	RLOCK3	RLOCK2	RLOCK1	RLOCK0
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **RLOCKy**: I/O pin y of port x resource lock (y = 15 to 0)

This bit is set to lock this I/O resource. It is cleared by default, and once set, it cannot be cleared until the GPIOx is reset.

0: SECy in GPIOx_SECCFGR and PRIVy in GPIOx_PRIVCFGR are writable.

1: Writes to SECy in GPIOx_SECCFGR and PRIVy in GPIOx_PRIVCFGR are ignored.

Note: Each bit is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

15.4.15 GPIO port x delay low register (GPIOx_DELAYRL) (x = A to H, N to Q)

Address offset: 0x40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DLY7[3:0]				DLY6[3:0]				DLY5[3:0]				DLY4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DLY3[3:0]				DLY2[3:0]				DLY1[3:0]				DLY0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DLYy[3:0]:** Port x IO pin y delay setup (y = 7 to 0)

Configure delay applied to the input (data IN from the pin) or the output path (data OUT to the pin), depending on the value of the corresponding DLYPATHy in GPIOx_ADVCFGRL.

0000: No delay
0001: Delay 0.3 ns
0010: Delay 0.5 ns
0011: Delay 0.75 ns
0100: Delay 1.0 ns
0101: Delay 1.25 ns
0110: Delay 1.5 ns
0111: Delay 1.75 ns
1000: Delay 2.0 ns
1001: Delay 2.25 ns
1010: Delay 2.5 ns
1011: Delay 2.75 ns
1100: Delay 3.0 ns
1101: Delay 3.25 ns
1110 to 1111: Delay 3.25 ns (maximum)

Note: Each bitfield is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

Successful read or write to DELAYy must be secure if SECy = 1 in GPIOx_SECCFGR, and must be privileged if PRIVy = 1 in GPIOx_PRIVCFGR.

15.4.16 GPIO port x delay high register (GPIOx_DELAYRH) (x = A to H, N to Q)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DLY15[3:0]				DLY14[3:0]				DLY13[3:0]				DLY12[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DLY11[3:0]				DLY10[3:0]				DLY9[3:0]				DLY8[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DLYy[3:0]:** Port x I/O pin y delay setup (y = 15 to 8)

Configure delay on input or output data prior to retiming (see GPIOx_DELAYRL for details).

Note: Each bitfield is reserved and must be kept to reset value when the corresponding I/O is not available on the selected package.

Successful read or write to DELAYy must be secure if SECy = 1 in GPIOx_SECCFGR, and must be privileged if PRIVy = 1 in GPIOx_PRIVCFGR.

15.4.17 GPIO port x advanced configuration low register (GPIOx_ADVCFGRL) (x = A to H, N to Q)

Address offset: 0x48

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RET7	INV CLK7	DE7	DLY PATH7	RET6	INV CLK6	DE6	DLY PATH6	RET5	INV CLK5	DE5	DLY PATH5	RET4	INV CLK4	DE4	DLY PATH4
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RET3	INV CLK3	DE3	DLY PATH3	RET2	INV CLK2	DE2	DLY PATH2	RET1	INV CLK1	DE1	DLY PATH1	RET0	INV CLK0	DE0	DLY PATH0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31, 27, 23, 19, 15, 11, 7, 3 **RETy:** Data retime selection (y = 7 to 0)
0: Input and output data are not synchronized or retimed on clock edges.
1: Input and output data are retimed to rising or falling clock edge, depending upon value of INVCLKy.

Bits 30, 26, 22, 18, 14, 10, 6, 2 **INVCLKy:** Clock inversion selection (y = 7 to 0)
0: I/O clocks are not inverted. Input and output data are retimed to rising clock edge.
1: I/O clocks are inverted. Input and output data are retimed to falling clock edge

Bits 29, 25, 21, 17, 13, 9, 5, 1 **DEy:** Input/Output data double edge selection (cfg_double_edge) (y = 7 to 0)
0: Input and output data is single-edge (changing on rising or falling clock edge, but not both)
1: Input and output data is double-edge (changing on both rising and falling clock edges)

Bits 28, 24, 20, 16, 12, 8, 4, 0 **DLYPATHy:** Clock inversion selection (y = 7 to 0)
Controls which path contains the configurable delay, input or output
0: Delay is switched into the output path, while the input path is set as pass through.
1: Delay is switched into the input path, while the output path is set as pass through.

15.4.18 GPIO port x advanced configuration high register (GPIOx_ADVCFGRH) (x =A to H, N to Q)

Address offset: 0x4C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RET15	INV CLK15	DE15	DLY PATH 15	RET14	INV CLK14	DE14	DLY PATH 14	RET13	INV CLK13	DE13	DLY PATH 13	RET12	INV CLK12	DE12	DLY PATH 12
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RET11	INV CLK11	DE11	DLY PATH 11	RET10	INV CLK10	DE10	DLY PATH 10	RET9	INV CLK9	DE9	DLY PATH 9	RET8	INV CLK8	DE8	DLY PATH 8
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31, 27, 23, 19, 15, 11, 7, 3 **RETy:** Data retime selection (y = 15 to 8)
0: Input and output data are not synchronized or retimed on clock edges.
1: Input and output data are retimed to rising or falling clock edge, depending upon value of INVCLKy.

Bits 30, 26, 22, 18, 14, 10, 6, 2 **INVCLKy:** Clock inversion selection (y = 15 to 8)
0: I/O clocks are not inverted. Input and output data are retimed to rising clock edge.
1: I/O clocks are inverted. Input and output data are retimed to falling clock edge

Bits 29, 25, 21, 17, 13, 9, 5, 1 **DEy:** Input/Output data double edge selection (cfg_double_edge) (y = 15 to 8)
0: Input and output data is single-edge (changing on rising or falling clock edge, but not both)
1: Input and output data is double-edge (changing on both rising and falling clock edges)

Bits 28, 24, 20, 16, 12, 8, 4, 0 **DLYPATHy:** Clock inversion selection (y = 15 to 8)
Controls which path contains the configurable delay, input or output
0: Delay is switched into the output path, while the input path is set as pass through.
1: Delay is switched into the input path, while the output path is set as pass through.

15.4.19 GPIO register map

Table 81. GPIO register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	GPIOx_MODER (x = A to H, N to Q)	MODE15 [1:0]		MODE14 [1:0]		MODE13 [1:0]		MODE12 [1:0]		MODE11 [1:0]		MODE10 [1:0]		MODE9 [1:0]		MODE8 [1:0]		MODE7 [1:0]		MODE6 [1:0]		MODE5 [1:0]		MODE4 [1:0]		MODE3 [1:0]		MODE2 [1:0]		MODE1 [1:0]		MODE0 [1:0]	
	Reset value port A	1	0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Reset value port B	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	1	1	1	1	1	1	1	1
	Reset value others	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x004	GPIOx_OTYPER (x = A to H, N to Q)	Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.		OT15	OT14	OT13	OT12	OT11	OT10	OT9	OT8	OT7	OT6	OT5	OT4	OT3	OT2	OT1	OT0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	GPIOx_OSPEEDR (x = A to H, N to Q)	OSPEED15 [1:0]		OSPEED14 [1:0]		OSPEED13 [1:0]		OSPEED12 [1:0]		OSPEED11 [1:0]		OSPEED10 [1:0]		OSPEED9 [1:0]		OSPEED8 [1:0]		OSPEED7 [1:0]		OSPEED6 [1:0]		OSPEED5 [1:0]		OSPEED4 [1:0]		OSPEED3 [1:0]		OSPEED2 [1:0]		OSPEED 1[1:0]		OSPEED0 [1:0]	
	Reset value port A	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Reset value port B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
	Reset value others	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	GPIOx_PUPDR (x = A to H, N to Q)	PUPD15 [1:0]		PUPD14 [1:0]		PUPD13 [1:0]		PUPD12 [1:0]		PUPD11 [1:0]		PUPD10 [1:0]		PUPD9 [1:0]		PUPD8 [1:0]		PUPD7 [1:0]		PUPD6 [1:0]		PUPD5 [1:0]		PUPD4 [1:0]		PUPD3 [1:0]		PUPD2 [1:0]		PUPD1 [1:0]		PUPD0 [1:0]	
	Reset value port A	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	Reset value port B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
	Reset value others	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	GPIOx_IDR (x = A to H, N to Q)	Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.		ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
	Reset value																	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x014	GPIOx_ODR (x = A to H, N to Q)	Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.		OD15	OD14	OD13	OD12	OD11	OD10	OD9	OD8	OD7	OD6	OD5	OD4	OD3	OD2	OD1	OD0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	GPIOx_BSRR (x = A to H, N to Q)	BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0	BS15	BS14	BS13	BS12	BS11	BS10	BS9	BS8	BS7	BS6	BS5	BS4	BS3	BS2	BS1	BS0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	GPIOx_LCKR (x = A to H, N to Q)	Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.	LCKK	LCK15	LCK14	LCK13	LCK12	LCK11	LCK10	LCK9	LCK8	LCK7	LCK6	LCK5	LCK4	LCK3	LCK2	LCK1	LCK0
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	GPIOx_AFRL (x = A to H, N to Q)	AFRL7[3:0]				AFRL6[3:0]				AFRL5[3:0]				AFRL4[3:0]				AFRL3[3:0]				AFRL2[3:0]				AFRL1[3:0]				AFRL0[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	GPIOx_AFRH (x = A to H, N to Q)	AFRH15[3:0]				AFRH14[3:0]				AFRH13[3:0]				AFRH12[3:0]				AFRH11[3:0]				AFRH10[3:0]				AFRH9[3:0]				AFRH8[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	GPIOx_BRR (x = A to H, N to Q)	Res.		Res.		Res.		Res.		Res.		Res.		Res.		Res.		BR15	BR14	BR13	BR12	BR11	BR10	BR9	BR8	BR7	BR6	BR5	BR4	BR3	BR2	BR1	BR0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	Reserved	Reserved																															

Table 81. GPIO register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x030	GPIOx_SECCFGR (x = A to H, N to Q)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
	Reset value port A																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Reset value port B																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Reset value port C																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Reset value port D																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Reset value port E																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Reset value port F																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Reset value port G																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Reset value port H																	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
	Reset value port N																	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
	Reset value port O																	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
	Reset value port P																	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
	Reset value port Q																	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0x034	GPIOx_PRIVCFGR (x = A to H, N to Q)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
	Reset value port A																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Reset value port B																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Reset value port C																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Reset value port D																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Reset value port E																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Reset value port F																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Reset value port G																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	Reset value port H																	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
	Reset value port N																	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
	Reset value port O																	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
	Reset value port P																	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
	Reset value port Q																	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0x038	GPIOx_ RCFGLOCKR (x = A to H, N to Q)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RLOCK15	RLOCK14	RLOCK13	RLOCK12	RLOCK11	RLOCK10	RLOCK9	RLOCK8	RLOCK7	RLOCK6	RLOCK5	RLOCK4	RLOCK3	RLOCK2	RLOCK1	RLOCK0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	Reserved	Reserved																															
0x040	GPIOx_DELAYRL (x = A to H, N to Q)	DLY7[3:0]				DLY6[3:0]				DLY5[3:0]				DLY4[3:0]				DLY3[3:0]				DLY2[3:0]				DLY1[3:0]				DLY0[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	GPIOx_DELAYRH (x = A to H, N to Q)	DLY15 [3:0]				DLY14 [3:0]				DLY13 [3:0]				DLY12 [3:0]				DLY11 [3:0]				DLY10 [3:0]				DLY9[3:0]				DLY8[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048	GPIOx_ADVCFGRL (x = A to H, N to Q)	RET7	INVCLK7	DE7	DLYPATH7	RET6	INVCLK6	DE6	DLYPATH6	RET5	INVCLK5	DE5	DLYPATH5	RET4	INVCLK4	DE4	DLYPATH4	RET3	INVCLK3	DE3	DLYPATH3	RET2	INVCLK2	DE2	DLYPATH2	RET1	INVCLK1	DE1	DLYPATH1	RET0	INVCLK0	DE0	DLYPATH0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	GPIOx_ADVCFGRH (x = A to H, N to Q)	RET15	INVCLK15	DE15	DLYPATH15	RET14	INVCLK14	DE14	DLYPATH14	RET13	INVCLK13	DE13	DLYPATH13	RET12	INVCLK12	DE12	DLYPATH12	RET11	INVCLK11	DE11	DLYPATH11	RET10	INVCLK10	DE10	DLYPATH10	RET9	INVCLK9	DE9	DLYPATH9	RET8	INVCLK8	DE8	DLYPATH8
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3* for the register boundary addresses.

16 System configuration controller (SYSCFG)

STM32N6x5/x7xx devices feature a set of configuration registers. The SYSCFG manages:

- Cortex-M55 internal settings (such as TCM, CACHE, or vectors)
- interconnect, security, and memory settings
- compensation cells

16.0.1 I/O compensation cell

The I/O compensation cell generates an 8-bit value (4 bits for N-MOS and 4 bits for P-MOS) for the I/O buffer that depends upon PVT operating conditions (process, voltage, temperature).

These bits are used to control the current slew-rate and output impedance in the I/O buffer. Five compensation cells are embedded: one for I/Os supplied by V_{DD}, and four for I/Os supplied by V_{DDIO2}, V_{DDIO3}, V_{DDIO4}, and V_{DDIO5}.

By default, the compensation cells are disabled, and a fixed code is applied to all I/Os.

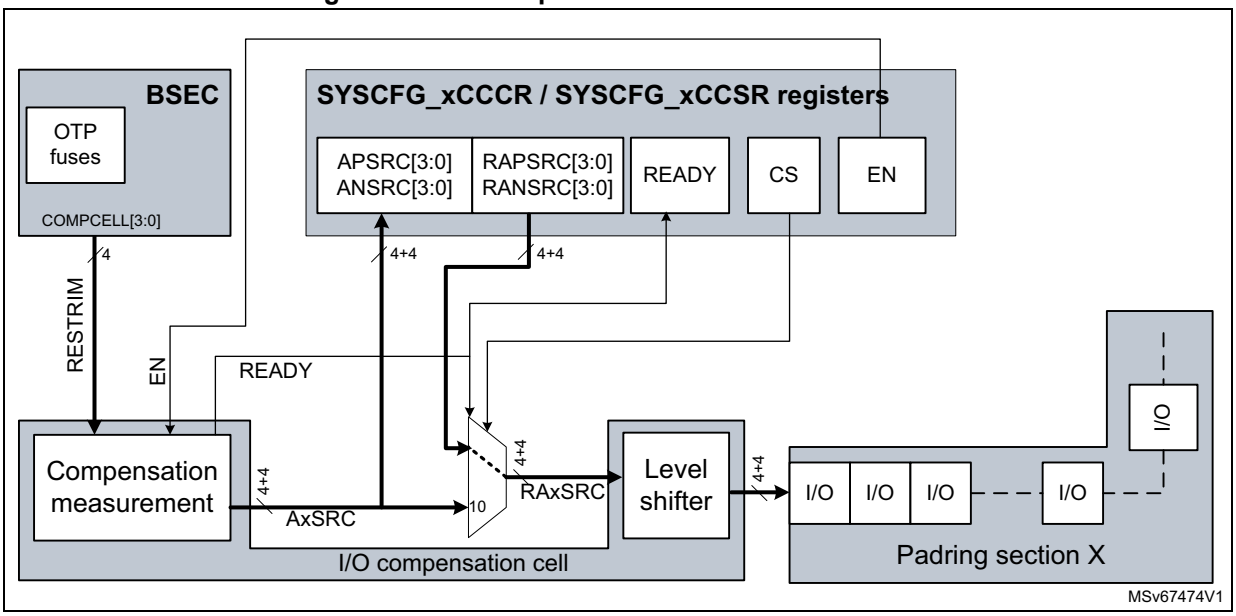
When enabled, the compensation cell tracks the PVT, and the 4-bit APSRC and ANSRC values are available in SYSCFG_xCCSR (with x = V_{DD}, V_{DDIO2}, V_{DDIO3}, V_{DDIO4}, or V_{DDIO5}) once the corresponding READY is set in SYSCFG_xCCSR.

If CS is cleared in SYSCFG_xCCCR, and READY is set, I/Os receive the APSRC and ANSRC values that result from the compensation cell.

To optimize the trimming, these values can be adjusted using RAPSRC and RANSRC in SYSCFG_xCCCR (see *Figure 71*).

To reduce the power consumption, it is recommended to copy the values from SYSCFG_xCCSR to SYSCFG_xCCCR. When the result is ready, set CS and disable the compensation cell.

Figure 71. I/O compensation cell control overview



16.1 SYSCFG registers

16.1.1 SYSCFG boot pin control register (SYSCFG_BOOTCR)

Address offset: 0x000

Reset value: 0x0000 0000

Reset by system reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOT1_PD	BOOT0_PD
														rw	rw

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

Bit 1 **BOOT1_PD**: BOOT1 pin pull-down disable
This is used to save power when BOOT1 pin is connected to V_{DD}.
0:Pull-down enabled. The BOOT1 pin can be left open and takes a value of 0 if open.
1: Pull-down disabled. The BOOT1 pin must not be left open.

Bit 0 **BOOT0_PD**: BOOT0 pin pull-down disable
This is used to save power when BOOT0 pin is connected to V_{DD}.
0: Pull-down enabled. The BOOT0 pin can be left open and takes a value of 0 if open.
1: Pull-down disabled. The BOOT0 pin must not be left open.

16.1.2 SYSCFG Cortex-M55 control register (SYSCFG_CM55CR)

Address offset: 0x004

Reset value: 0x0000 0000

Reset by system reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKDCAIC	LOCKSAU	LOCKNSMPU	LOCKSMPU	LOCKNSVTOR	LOCKSVTAIRCR
										rw	rw	rw	rw	rw	rw

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

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

Bit 21 **LOCKDCAIC**: Disable access to the instruction cache direct cache access registers DCAICLR and DCAICRR.

Bit 20 **LOCKSAU**: Prevent changes to secure SAU memory regions already programmed.

Bit 19 **LOCKNSMPU**: Prevent changes to nonsecure MPU memory regions already programmed.

Bit 18 **LOCKSMPU**: Prevent changes to programmed secure MPU memory regions.

Bit 17 **LOCKNSVTOR**: Prevent changes to the nonsecure vector table base address.

Bit 16 **LOCKSVTAIRCR**: Prevent changes to:
- the secure vector table base address
- handling of secure interrupt priority
- BusFault, HardFault, and NMI security target settings in the processor

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:0 **FPU_IT_EN[5:0]**: Enable FPU exception
bit 5: Input abnormal interrupt enable
bit 4: Inexact interrupt enable
bit 3: Underflow operation interrupt enable
bit 2: Overflow interrupt enable
bit 1: Divide-by-zero interrupt enable
bit 0: Invalid operation interrupt enable

16.1.3 SYSCFG Cortex-M55 TCM control register (SYSCFG_CM55TCMCR)

Address offset: 0x008

Reset value: 0x0000 0087

Reset on pwr_vsw_rstn.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTCMWSDISABLE	ITCMWSDISABLE	Res.	Res.	Res.	Res.	LOCKDTGU	LOCKITGU	LOCKTCM
							rw	rw					rw	rw	rw

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

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DTCMWSDISABLE**: Disable wait-state applied by default on extended DTCM memory.

Bit 23 **ITCMWSDISABLE**: Disable wait-state applied by default on extended ITCM memory.

Bits 22:19 Reserved, must be kept at reset value.

Bit 18 **LOCKDTGU**: Disable writes to registers associated with the DTCM interface security gating.

Bit 17 **LOCKITGU**: Disable writes to registers associated with the ITCM interface security gating.

Bit 16 **LOCKTCM**: Disable writes to registers associated with the TCM region

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:4 **CFGDTCMSZ[3:0]**: Select DTCM memory size
0x8: 128 Kbytes (default value)
0x9: 256 Kbytes
Others: Reserved

Bits 3:0 **CFGITCMSZ[3:0]**: Select ITCM memory size
0x7: 64 Kbytes (default value)
0x8: 128 Kbytes
0x9: 256 Kbytes
Others: Reserved

Note: CFGDTCMSZ and CFGITCMSZ must be written in the same data access, because they are write-once bitfields locked together. Writing only one of them blocks write-access to the other up to the next power-on reset.

16.1.4 SYSCFG Cortex-CM55 memory RW margin register (SYSCFG_CM55RWMCR)

Address offset: 0x00C

Reset value: 0x0000 1020

Reset by system reset.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	BC2_CACHE	BC1_CACHE	RM_CACHE[3:0]				RME_CACHE	BC2_TCM	BC1_TCM	RM_TCM[3:0]				RME_TCM
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **BC2_CACHE**: Biasing level adjust input recommended for Vnom + 10%
Its setting gives a higher value of rise on V_{SS}-core voltage to give enough head room for retention.

Bit 12 **BC1_CACHE**: Biasing level adjust input recommended for Vnom.
Its setting gives a smaller value of rise on V_{SS}-core voltage to give enough head room for retention.

Bits 11:8 **RM_CACHE[3:0]**: External read/write (RW) margin inputs for caches memories

Bit 7 **RME_CACHE**: RW margin enable input for caches memories
0: Default RW margin settings
1: Use external pin RW margin setting

Bit 6 **BC2_TCM**: Biasing level adjust input recommended for Vnom + 10%
Its setting gives a higher value of rise on V_{SS}-core voltage to give enough head room for retention.

Bit 5 **BC1_TCM**: Biasing level adjust input recommended for Vnom
Its setting gives a smaller value of rise on V_{SS}-core voltage to give enough head room for retention.

Bits 4:1 **RM_TCM[3:0]**: External RW margin inputs for TCM memories

Bit 0 **RME_TCM**: RW margin enable input for TCM memories
0: Default RW margin settings
1: Use external pin RW margin setting

16.1.5 SYSCFG Cortex-M55 SVTOR control register (SYSCFG_INITSVTORCR)

Address offset: 0x010

Reset value: 0x1800 0000

Reset by system reset.

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

Bits 31:7 **SVTOR_ADDR[24:0]**: Secure vector table base address

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

16.1.6 SYSCFG Cortex-M55 NSVTOR control register (SYSCFG_INITNSVTORCR)

Address offset: 0x014

Reset value: 0x0800 0000

Reset by system reset.

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

Bits 31:7 **NSVTOR_ADDR[24:0]**: Nonsecure vector table base address

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

16.1.7 SYSCFG Cortex-M55 reset type control register (SYSCFG_CM55RSTCR)

Address offset: 0x018

Reset value: 0x0000 0000

Reset by system reset.

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

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

Bit 2 **LOCKUP_NMI_EN**: Select action to perform on a lockup state on the core
- 0: Lockup state must be recovered from NVIC interrupt (default value)
- 1: Lockup generates a NMI on the core.

Bit 1 **LOCKUP_RST_EN**: Select action to perform on a lockup state on the core
- 0: Lockup state must be recovered from interrupt (default value)
- 1: Lockup requests a warm reset to the RCC.

Bit 0 **CORE_RESET_TYPE**: Select reset to apply on core upon SYSRESETREQ
- 0: Warm reset (default value)
- 1: Power-on reset

16.1.8 SYSCFG Cortex-M55 P-AHB write posting control register (SYSCFG_CM55PAHBWPR)

Address offset: 0x01C

Reset value: 0x0000 0000

Reset by system reset.

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

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

Bit 0 **PAHB_ERROR_ACK**: Error capture in write posting buffer
0: Error capture
1: Clean error

16.1.9 SYSCFG VENCRAM control register (SYSCFG_VENCRAMCR)

Address offset: 0x020

Reset value: 0x0000 0000

Reset by system reset.

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

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

Bit 0 **VENCRAM_EN**: VENCRAM allocation VENC if active, or to system (if VENC inactive)
0: VENCRAM reserved for the VENC
1: VENCRAM available for the system (VENC inactive)

16.1.10 SYSCFG potential tamper reset register (SYSCFG_POTTAMPRSTCR)

Address offset: 0x024

Reset value: 0x0000 0000

Reset by system reset.

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

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

Bit 0 **POTTAMPERSETMASK**:

This bit can be set by software to mask PKA, SAES, CRYP, and HASH reset, in case of potential tamper.

0: PKA, SAES, CRYP, and HASH reset in case of potential tamper

1: PKA, SAES, CRYP, and HASH not reset in case of potential tamper

16.1.11 SYSCFG NPUNIC QoS control register (SYSCFG_NPUNICQOSCR)

Address offset: 0x028

Reset value: 0x0000 0000

Reset by system reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPUSS_AWQOS[3:0]				CPUSS_ARQOS[3:0]			
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NPU2_AWQOS[3:0]				NPU2_ARQOS[3:0]				NPU1_AWQOS[3:0]				NPU1_ARQOS[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:20 **CPUSS_AWQOS[3:0]**: NPUNIC write QoS information for master port from CPUSS

Bits 19:16 **CPUSS_ARQOS[3:0]**: NPUNIC read QoS information for master port from CPUSS

Bits 15:12 **NPU2_AWQOS[3:0]**: NPUNIC write QoS information for NPU2 master port

Bits 11:8 **NPU2_ARQOS[3:0]**: NPUNIC read QoS information for NPU2 master port

Bits 7:4 **NPU1_AWQOS[3:0]**: NPUNIC write QoS information for NPU1 master port

Bits 3:0 **NPU1_ARQOS[3:0]**: NPUNIC read QoS information for NPU1 master port

16.1.12 SYSCFG AHB-AXI bridge early write response control register (SYSCFG_ICNEWRCR)

Address offset: 0x034

Reset value: 0x0000 0000

Reset by system reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USB2_EARLY_WR_RSP_ENABLE	USB1_EARLY_WR_RSP_ENABLE	SDMMC2_EARLY_WR_RSP_ENABLE	SDMMC1_EARLY_WR_RSP_ENABLE
												rw	rw	rw	rw

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

Bit 3 **USB2_EARLY_WR_RSP_ENABLE**:
0: Early-write response disabled. The last AHB write data beat receives the AXI buffered response for the complete AHB transaction.
1: Early-write response enabled. AHB-Lite write data beats receive an automatic OK response from the AHB-to-AXI bridge, whatever the B-channel AXI response.

Bit 2 **USB1_EARLY_WR_RSP_ENABLE**:
0: Early-write response disabled. The last AHB write data beat receives the AXI buffered response for the complete AHB transaction.
1: Early-write response enabled. AHB-Lite write data beats receive an automatic OK response from the AHB-to-AXI bridge, whatever the B-channel AXI response.

Bit 1 **SDMMC2_EARLY_WR_RSP_ENABLE**:
0: Early-write response disabled. The last AHB write data beat receives the AXI buffered response for the complete AHB transaction.
1: Early-write response enabled. AHB-Lite write data beats receive an automatic OK response from the AHB-to-AXI bridge, whatever the B-channel AXI response.

Bit 0 **SDMMC1_EARLY_WR_RSP_ENABLE**:
0: Early-write response disabled. The last AHB write data beat receives the AXI buffered response for the complete AHB transaction.
1: Early-write response enabled. AHB-Lite write data beats receive an automatic OK response from the AHB-to-AXI bridge, whatever the B-channel AXI response.

16.1.13 SYSCFG ICN clock gating control register (SYSCFG_ICNCGCR)

Address offset: 0x038

Reset value: 0x0000 0000

Reset by system reset.

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

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

Bit 2 **CPU_NOC_CG_DISABLE**: CPU_NOC clock gating disable
0: CPU_NOC clock gating enabled (default value)
1: CPU_NOC clock gating disabled

Bit 1 **CPU_NIC_CG_DISABLE**: CPU_NIC clock gating disable
0: CPU_NIC clock gating enabled (default value)
1: CPU_NIC clock gating disabled

Bit 0 **NPU_NOC_CG_DISABLE**: NPU_NOC clock gating disable
0: NPU_NOC clock gating enabled (default value)
1: NPU_NOC clock gating disabled

16.1.14 SYSCFG V_{DDIO4} compensation cell control register (SYSCFG_VDDIO4CCCR)

Address offset: 0x044

Reset value: 0x0000 0078

Reset by system reset.

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

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

Bit 9 **CS**: Selects the code to be applied for the compensation cell of I/Os supplied by V_{DDIO4}.
0: V_{DDIO4} code from the cell (available in the SYSCFG_VDDIO4CCSR)
1: V_{DDIO4} I/O code from RANSRC[3:0] and RAPSRC[3:0] in this register
Note: CS = 0 is not taken into account until READY = 1. Whenever CS, RANSRC[3:0], and RAPSRC[3:0] are used for I/O compensation, the compensation cell must be not enabled (EN = 0, which is the case after a reset).

Bit 8 **EN**: Enables the compensation cell of I/Os supplied by VDDIOx.
0: V_{DDIO4} I/O compensation cell disabled
1: V_{DDIO4} I/O compensation cell enabled
Note: The HSI oscillator must be enabled and ready (controlled in RCC) before EN can be set to 1. The HSI oscillator can be disabled only if EN is set to 0.

Bits 7:4 **RAPSRC[3:0]**: These bits are written by software to define an I/O compensation code for PMOS transistors. This code is applied to the I/O compensation cell when CS = 1.

Bits 3:0 **RANSRC[3:0]**: These bits are written by software to define an I/O compensation code for NMOS transistors. This code is applied to the I/O compensation cell when CS = 1.

16.1.15 SYSCFG V_{DDIO4} compensation cell status register (SYSCFG_VDDIO4CCSR)

Address offset: 0x048

Reset value: 0x0000 0000

Reset by system reset.

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

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **READY**: Provides the compensation cell status of I/Os supplied by V_{DDIO4}
0: V_{DDIO4} I/O compensation cell not ready
1: V_{DDIO4} I/O compensation cell ready

Bits 7:4 **APSRC[3:0]**: This value is provided by the cell, and can be used by the CPU to compute an I/O compensation cell code for PMOS transistors.
This code is applied to the I/O compensation cell when CS = 0 in SYSCFG_VDDIO4CCCR, and READY = 1 in this register.

Bits 3:0 **ANSRC[3:0]**: This value is provided by the cell, and can be used by the CPU to compute an I/O compensation cell code for NMOS transistors.
This code is applied to the I/O compensation cell when CS = 0 in SYSCFG_VDDIO4CCCR, and READY = 1 in this register.

16.1.16 SYSCFG V_{DDIO5} compensation cell control register (SYSCFG_VDDIO5CCCR)

Address offset: 0x04C

Reset value: 0x0000 0078

Reset by system reset.

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

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

Bit 9 **CS**: Selects the code to be applied for the compensation cell of I/Os supplied by V_{DDIO5}.
0: V_{DDIO5} I/O code from the cell (available in the SYSCFG_VDDIO5CCSR)
1: V_{DDIO5} I/O code from RANSRC[3:0] and RAPSRC[3:0] in this register
Note: CS = 0 is not taken into account until READY = 1. Whenever CS, RANSRC[3:0], and RAPSRC[3:0] are used for I/O compensation, the compensation cell must be not enabled (EN = 0, which is the case after a reset).

Bit 8 **EN**: Enables the compensation cell of I/Os supplied by V_{DDIO5}.
0: V_{DDIO5} I/O compensation cell disabled
1: V_{DDIO5} I/O compensation cell enabled
Note: The HSI oscillator must be enabled and ready (controlled in RCC) before EN can be set to 1. The HSI oscillator can be disabled only if EN is set to 0.

Bits 7:4 **RAPSRC[3:0]**: These bits are written by software to define an I/O compensation code for PMOS transistors. This code is applied to the I/O compensation cell when CS = 1.

Bits 3:0 **RANSRC[3:0]**: These bits are written by software to define an I/O compensation code for NMOS transistors. This code is applied to the I/O compensation cell when CS = 1.

16.1.17 SYSCFG V_{DDIO5} compensation cell status register (SYSCFG_VDDIO5CCSR)

Address offset: 0x050

Reset value: 0x0000 0000

Reset by system reset.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	READY	APSRC[3:0]				ANSRC[3:0]			
							r	r	r	r	r	r	r	r	r

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **READY**: Provides the compensation cell status of I/Os supplied by V_{DDIO5}
0: V_{DDIO5} I/O compensation cell not ready
1: V_{DDIO5} I/O compensation cell ready

Bits 7:4 **APSRC[3:0]**: This value is provided by the cell, and can be used by the CPU to compute an I/O compensation cell code for PMOS transistors.
This code is applied to the I/O compensation cell when CS = 0 in SYSCFG_VDDIO5CCCR, and READY = 1 in this register.

Bits 3:0 **ANSRC[3:0]**: This value is provided by the cell, and can be used by the CPU to compute an I/O compensation cell code for NMOS transistors.
This code is applied to the I/O compensation cell when CS = 0 in SYSCFG_VDDIO5CCCR, and READY = 1 in this register.

16.1.18 SYSCFG V_{DDIO2} compensation cell control register (SYSCFG_VDDIO2CCCR)

Address offset: 0x054

Reset value: 0x0000 0078

Reset by system reset.

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

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

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

Bit 9 **CS**: Selects the code to apply for the compensation cell of I/Os supplied by VDDIO2.
0: V_{DDIO2} I/O code from the cell (available in the SYSCFG_VDDIO2CCSR)
1: V_{DDIO2} I/O code from RANSRC[3:0] and RAPSRC[3:0] in this register
Note: CS = 0 is not taken into account until READY = 1. Whenever CS, RANSRC[3:0], and RAPSRC[3:0] are used for I/O compensation, the compensation cell must be not enabled (EN = 0, which is the case after a reset).

Bit 8 **EN**: Enables the compensation cell of I/Os supplied by VDDIO2.
0: V_{DDIO2} I/O compensation cell disabled
1: V_{DDIO2} I/O compensation cell enabled
Note: The HSI oscillator must be enabled and ready (controlled in RCC) before EN can be set to 1. The HSI oscillator can be disabled only if EN is set to 0.

Bits 7:4 **RAPSRC[3:0]**: These bits are written by software to define an I/O compensation code for PMOS transistors. This code is applied to the I/O compensation cell when CS = 1.

Bits 3:0 **RANSRC[3:0]**: These bits are written by software to define an I/O compensation code for NMOS transistors. This code is applied to the I/O compensation cell when CS = 1.

16.1.19 SYSCFG V_{DDIO2} compensation cell status register (SYSCFG_VDDIO2CCSR)

Address offset: 0x058

Reset value: 0x0000 0000

Reset by system reset.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	READY	APSRC[3:0]				ANSRC[3:0]			
							r	r	r	r	r	r	r	r	r

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **READY**: Provides the compensation cell status of I/Os supplied by V_{DDIO2}
0: V_{DDIO2} I/O compensation cell not ready
1: V_{DDIO2} I/O compensation cell ready

Bits 7:4 **APSRC[3:0]**: This value is provided by the cell, and can be used by the CPU to compute an I/O compensation cell code for PMOS transistors.
This code is applied to the I/O compensation cell when CS = 0 in SYSCFG_VDDIO2CCCR, and READY = 1 in this register.

Bits 3:0 **ANSRC[3:0]**: This value is provided by the cell, and can be used by the CPU to compute an I/O compensation cell code for NMOS transistors.
This code is applied to the I/O compensation cell when CS = 0 in SYSCFG_VDDIO2CCCR, and READY = 1 in this register.

16.1.20 SYSCFG V_{DDIO3} compensation cell control register (SYSCFG_VDDIO3CCCR)

Address offset: 0x05C

Reset value: 0x0000 0078

Reset by system reset.

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

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

Bit 9 **CS**: Selects the code to be applied for the compensation cell of I/Os supplied by VDDIO3.
0: V_{DDIO3} I/O code from the cell (available in the SYSCFG_VDDIO3CCSR)
1: V_{DDIO3} I/O code from RANSRC[3:0] and RAPSRC[3:0] in this register
Note: CS = 0 is not taken into account until READY = 1. Whenever CS, RANSRC[3:0], and RAPSRC[3:0] are used for I/O compensation, the compensation cell must be not enabled (EN = 0, which is the case after a reset).

Bit 8 **EN**: Enables the compensation cell of I/Os supplied by VDDIOx.
0: V_{DDIO3} I/O compensation cell disabled
1: V_{DDIO3} I/O compensation cell enabled
Note: The HSI oscillator must be enabled and ready (controlled in RCC) before EN can be set to 1. The HSI oscillator can be disabled only if EN is set to 0.

Bits 7:4 **RAPSRC[3:0]**: These bits are written by software to define an I/O compensation code for PMOS transistors. This code is applied to the I/O compensation cell when CS = 1.

Bits 3:0 **RANSRC[3:0]**: These bits are written by software to define an I/O compensation code for NMOS transistors. This code is applied to the I/O compensation cell when CS = 1.

16.1.21 SYSCFG V_{DDIO3} compensation cell status register (SYSCFG_VDDIO3CCSR)

Address offset: 0x060

Reset value: 0x0000 0000

Reset by system reset.

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

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **READY**: Provides the compensation cell status of I/Os supplied by V_{DDIO3}
0: V_{DDIO3} I/O compensation cell not ready
1: V_{DDIO3} I/O compensation cell ready

Bits 7:4 **APSRC[3:0]**: This value is provided by the cell, and can be used by the CPU to compute an I/O compensation cell code for PMOS transistors.
This code is applied to the I/O compensation cell when the CS = 0 in SYSCFG_VDDIO3CCCR, and READY = 1 in this register.

Bits 3:0 **ANSRC[3:0]**: This value is provided by the cell, and can be used by the CPU to compute an I/O compensation cell code for NMOS transistors.
This code is applied to the I/O compensation cell when the CS = 0 in SYSCFG_VDDIO3CCCR, and READY = 1 in this register.

16.1.22 SYSCFG V_{DD} compensation cell control register (SYSCFG_VDDCCCR)

Address offset: 0x064

Reset value: 0x0000 0078

Reset by system reset.

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

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

Bit 9 **CS**: Selects the code to be applied for the compensation cell of I/Os supplied by V_{DD}.
0: V_{DD} I/O code from the cell (available in the SYSCFG_VDDCCSR)
1: V_{DD} I/O code from RANSRC[3:0] and RAPSRC[3:0]
Note: CS = 0 is not taken into account until READY = 1. Whenever CS, RANSRC[3:0] and RAPSRC[3:0] are used for I/O compensation when the compensation cell is not enabled (EN = 0, which is the case after a reset).

Bit 8 **EN**: Enables the compensation cell of I/Os supplied by V_{DD}.
0: V_{DD} I/O compensation cell disabled
1: V_{DD} I/O compensation cell enabled
Note: The HSI oscillator must be enabled and ready (controlled in RCC) before EN can be set to 1. The HSI oscillator can be disabled only if EN is set to 0.

Bits 7:4 **RAPSRC[3:0]**: These bits are written by software to define an I/O compensation code for PMOS transistors. This code is applied to the I/O compensation cell when CS = 1.

Bits 3:0 **RANSRC[3:0]**: These bits are written by software to define an I/O compensation code for NMOS transistors. This code is applied to the I/O compensation cell when CS = 1.

16.1.23 SYSCFG V_{DD} compensation cell status register (SYSCFG_VDDCCSR)

Address offset: 0x068

Reset value: 0x0000 0000

Reset by system reset.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	READY	APSRC[3:0]				ANSRC[3:0]			
							r	r	r	r	r	r	r	r	r

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **READY**: Provides the compensation cell status of I/Os supplied by V_{DD}
0: V_{DD} I/O compensation cell not ready
1: V_{DD} I/O compensation cell ready

Bits 7:4 **APSRC[3:0]**: This value is provided by the cell, and can be used by the CPU to compute an I/O compensation cell code for PMOS transistors.
This code is applied to the I/O compensation cell when CS = 0 in SYSCFG_VDDCCCR, and READY = 1.

Bits 3:0 **ANSRC[3:0]**: This value is provided by the cell, and can be used by the CPU to compute an I/O compensation cell code for NMOS transistors.
This code is applied to the I/O compensation cell when CS = 0 in SYSCFG_VDDCCCR, and READY = 1.

16.1.24 SYSCFG control timer break register (SYSCFG_CBR)

Address offset: 0x06C

Reset value: 0x0000 0000

Reset by system reset.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CM55TCML	CM55CACHEL	BKPRAML	Res.	Res.	Res.	Res.	Res.	PVDEN	Res.	CM55L
					rw	rw	rw						rw		rw

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **CM55TCML**: Cortex-M55 TCM double ECC error lock

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the Cortex-M55 TCM double ECC error signal connection to TIM1/8/15/16/17 break inputs.

0: Cortex-M55 TCM double ECC error signal disconnected from TIM1/8/15/16/17 break inputs

1: Cortex-M55 TCM double ECC error signal connected to TIM1/8/15/16/17 break inputs

Bit 9 **CM55CACHEL**: Cortex-M55 cache double ECC error lock

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the Cortex-M55 cache double ECC error signal connection to TIM1/8/15/16/1break inputs.

0: Cortex-M55 cache double ECC error signal disconnected from TIM1/8/15/16/17 break inputs

1: Cortex-M55 cache double ECC error signal connected to TIM1/8/15/16/17 break inputs

Bit 8 **BKPRAML**: Backup SRAM double ECC error lock

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the backup SRAM double ECC error signal connection to TIM1/8/15/16/17 break inputs.

0: Backup SRAM double ECC error signal disconnected from TIM1/8/15/16/17 break inputs

1: Backup SRAM double ECC error signal connected to TIM1/8/15/16/17 break inputs

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **PVDEN**: PVD lock enable

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the PVD connection to TIM1/8/15/16/17 break input, as well as the PVDE in PWR_CR2.

0: PVD interrupt disconnected from TIM1/8/15/16/17 break input. PVDE bits can be programmed by the application.

1: PVD interrupt connected to TIM1/8/15/16/17 break input. PVDE and bits are read only.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CM55L**: CM55 lockup lock enable

This bit is set by software and cleared only by a system reset. It can be used to enable and lock the Cortex-M55 lockup (HardFault) output connection to TIM1/8/15/16/17 break inputs.

0: Cortex-M55 lockup output disconnected from TIM1/8/15/16/17 break inputs

1: Cortex-M55 lockup output disconnected from TIM1/8/15/16/17 break inputs

16.1.25 SYSCFG DMA CID secure control register (SYSCFG_SEC_AIDCR)

Address offset: 0x070

Reset value: 0x0000 0001

Reset by system reset.

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

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

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

Bits 2:0 **DMACID_SEC[2:0]**: Secure user accesses to the DMA present this programmed CID.

16.1.26 SYSCFG FMC retiming logic control register (SYSCFG_FMC_RETIMECR)

Address offset: 0x074

Reset value: 0x0000 0000

Reset by system reset.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDFBCLK_180	CFG_RETIME_TX	CFG_RETIME_RX
													rw	rw	rw

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

Bit 2 **SDFBCLK_180**: Delay on feedback clock
- 0: No delay on the feedback clock
- 1: Half a cycle delay on the feedback clock

Bit 1 **CFG_RETIME_TX**: Retiming on Tx path
- 0: No retiming on Tx path
- 1: Retiming on Tx path

Bit 0 **CFG_RETIME_RX**: Retiming on Rx path
- 0: No retiming on Rx path
- 1: Retiming on Rx path

16.1.27 SYSCFG NPU RAM interleaving control register (SYSCFG_NPU_ICNCR)

Address offset: 0x078

Reset value: 0x0000 0000

Reset by system reset.

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

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

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

Bit 0 **INTERLEAVING_ACTIVE**: Control interleaving on NPU RAMs
0: Interleaving disabled
1: Interleaving enabled

16.1.28 SYSCFG boot pin status register (SYSCFG_BOOTSR)

Address offset: 0x100

Reset value: 0x0000 0000

Reset by system reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOT1	BOOT0
														r	r

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

Bit 1 **BOOT1**: BOOT1 pin value
0: BOOT1 pin connected to V_{SS} (or left open if BOOT1_PD = 0 in SYSCFG_BOOTCR)
1: BOOT1 pin connected to V_{DD}

Bit 0 **BOOT0**: BOOT0 pin value
0: BOOT0 pin connected to V_{SS} (or left open if BOOT0_PD = 0)
1: BOOT0 pin connected to V_{DD}

16.1.29 SYSCFG AHB write posting address error register (SYSCFG_AHBWP_ERROR_SR)

Address offset: 0x104

Reset value: 0x0000 0000

Reset by system reset.

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

Bits 31:0 **PAHB_ERROR_ADDR[31:0]**: Reports address of the first error in P-AHB write-posting buffer

16.1.30 SYSCFG SMPS observable signals through HDP selection configuration register (SYSCFG_SMPSHDPCR)

Address offset: 0x400

Reset value: 0x0000 0000

Reset by system reset.

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

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

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

Bits 3:0 **SMPSHDPSEL[3:0]**:
0000: Standard run mode (no HDP)
1100: Analyze fsm mode analysis
1101: Analyze fsm mos analysis
1110: Analyze fsm rampe analysis
1111: Analyze fsm mode analysis
Others: Reserved

16.1.31 SYSCFG DMA CID nonsecure control register (SYSCFG_SECPRIV_AIDCR)

Address offset: 0x800

Reset value: 0x0000 0001

Reset by system reset.

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

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

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

Bits 2:0 **DMACID_SECPRIV[2:0]**: Secure privilege accesses to the DMA present this programmed CID.

16.1.32 SYSCFG device ID register (SYSCFG_DEVICEID)

Address offset: 0x0FF0

Reset value: 0xXXXX 6XXX

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

Bits 31:16 **REV_ID[15:0]**: Silicon revision
0x1000: Revision A
0x2000: Revision B

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

Bits 11:0 **DEV_ID[11:0]**: Device identifier
0x486: STM32N6x5/x7xx

16.1.33 SYSCFG register map

Table 82. SYSCFG register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	SYSCFG_BOOTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOT1	BOOT0
	Reset value																															0	0
0x004	SYSCFG_CM55CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKDCAIC	LOCKSAU	LOCKNSMPU	LOCKSMPU	LOCKNSVTOR	LOCKSVTAIRCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPU_IT_EN[5:0]					
	Reset value											0	0	0	0	0	0											0	0	0	0	0	0
0x008	SYSCFG_ CM55TCMCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTCMWSDISABLE	ITCMWSDISABLE	Res.	Res.	Res.	Res.	LOCKDTGU	LOCKITGU	LOCKTCM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFGDTCMSZ [3:0]				CFGITCMSZ [3:0]			
	Reset value								0	0					0	0	0									1	0	0	0	0	1	1	1
0x00C	SYSCFG_ CM55RWMCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BC2_CACHE	BC1_CACHE	RM_CACHE [3:0]				RME_CACHE	BC2_TCM	BC1_TCM	RM_TCM [3:0]				RME_TCM
	Reset value																			0	1	0	0	0	0	0	0	1	0	0	0	0	0
0x010	SYSCFG_ _INITSVTORCR	SVTOR_ADDR[24:0]																								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								
0x014	SYSCFG_ _INITNSVTORCR	NSVTOR_ADDR[24:0]																								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								

Table 82. SYSCFG register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x018	SYSCFG_ CM55RSTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKUP_NMI_EN	LOCKUP_RST_EN	CORE_RESET_TYPE
	Reset value																														0	0	0
0x01C	SYSCFG_ CM55PAHBWPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PAHB_ERROR_ACK
	Reset value																																0
0x020	SYSCFG_ VENCRAMCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VENCRAM_EN
	Reset value																																0
0x024	SYSCFG_ POTTAMPRSTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	POTTAMPERSETMASK
	Reset value																																0
0x028	SYSCFG_ NPUNICQOSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPUSS_ AWQOS [3:0]				CPUSS_ ARQOS [3:0]				NPU2_ AWQOS [3:0]				NPU2_ ARQOS [3:0]				NPU1_ AWQOS [3:0]				NPU1_ ARQOS [3:0]			
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	Reserved	Reserved																															
0x034	SYSCFG_ ICNEWRCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USB2_EARLY_WR_RSP_ENABLE	USB1_EARLY_WR_RSP_ENABLE	SDMMC2_EARLY_WR_RSP_ENABLE	SDMMC1_EARLY_WR_RSP_ENABLE
	Reset value																													0	0	0	0

Table 82. SYSCFG register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x038	SYSCFG_ICNCGCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		CPU_NOC_CG_DISABLE	CPU_NIC_CG_DISABLE	NPU_NOC_CG_DISABLE
	Reset value																														0	0	0
0x03C-0x040	Reserved	Reserved																															
0x044	SYSCFG_ VDDIO4CCCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CS	EN	RAPSRC [3:0]				RANSRC [3:0]			
	Reset value																							0	0	0	1	1	1	1	0	0	0
0x048	SYSCFG_ VDDIO4CCSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	READ	APSRC [3:0]				ANSRC [3:0]			
	Reset value																								0	0	0	0	0	0	0	0	0
0x04C	SYSCFG_ VDDIO5CCCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CS	EN	RAPSRC [3:0]				RANSRC [3:0]			
	Reset value																							0	0	0	1	1	1	1	0	0	0
0x050	SYSCFG_ VDDIO5CCSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	READ	APSRC [3:0]				ANSRC [3:0]			
	Reset value																								0	0	0	0	0	0	0	0	0
0x054	SYSCFG_ VDDIO2CCCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CS	EN	RAPSRC [3:0]				RANSRC [3:0]			
	Reset value																							0	0	0	1	1	1	1	0	0	0
0x058	SYSCFG_ VDDIO2CCSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	READ	APSRC [3:0]				ANSRC [3:0]			
	Reset value																								0	0	0	0	0	0	0	0	0
0x05C	SYSCFG_ VDDIO3CCCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CS	EN	RAPSRC [3:0]				RANSRC [3:0]			
	Reset value																							0	0	0	1	1	1	1	0	0	0
0x060	SYSCFG_ VDDIO3CCSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	READ	APSRC [3:0]				ANSRC [3:0]			
	Reset value																								0	0	0	0	0	0	0	0	0
0x064	SYSCFG_VDDCCCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CS	EN	RAPSRC [3:0]				RANSRC [3:0]			
	Reset value																							0	0	0	1	1	1	1	0	0	0
0x068	SYSCFG_VDDCCSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	READ	APSRC [3:0]				ANSRC [3:0]			
	Reset value																								0	0	0	0	0	0	0	0	0
0x06C	SYSCFG_CBR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CM55TCML	CM55CACHEL	Res.	BKPRAML	PVDL_LOCK	Res.	CM55L
	Reset value																										0	0		0	0		0
0x070	SYSCFG_ SEC_AIDCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		DMACID_SEC[2:0]		
	Reset value																														0	0	1

Table 82. SYSCFG register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x074	SYSCFG_ FMC_RETIMECR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDFBCLK_180	CFG_RETIME_TX	CFG_RETIME_RX
	Reset value																													0	0	0
0x078	SYSCFG_ NPU_ICNCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		INTERLEAVING_ACTIVE
	Reset value																																0
0x080-0x0FC	Reserved	Reserved																															
0x100	SYSCFG_BOOTSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOOT1	BOOT0
	Reset value																															0	0
0x104	SYSCFG_ AHBWP_ERROR_SR	PAHB_ERROR_ADDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x108 - 0x3FC	Reserved	Reserved																															
0x400	SYSCFG_ SMPSHDPCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMPSHDPSEL [3:0]			
	Reset value																													0	0	0	0
0x404 - 0x7FC	Reserved	Reserved																															
0x800	SYSCFG_ SECPRIV_AIDCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DMACID_ SECPRIV [2:0]		
	Reset value																														0	0	1
0x804 - 0xFEC	Reserved	Reserved																															
0xFF0	SYSCFG_DEVICEID	REV_ID[15:0]																Res.	Res.	Res.	Res.	Res.	DEV_ID[11:0]										
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X						X	X	X	X	X	X	X	X	X	X	X

Refer to *Section 2.3* for the register boundary addresses.

17 Peripherals interconnect matrix

17.1 Introduction

Several peripherals have direct connections between them. This allows autonomous communication and or synchronization between peripherals, saving CPU resources, thus power supply consumption. In addition, these hardware connections remove software latency, and improve the predictability of the system design.

Depending on peripherals, these interconnections can operate in various power modes: Run, Standby, Sleep, and Stop modes.

17.2 Connection summary

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

Table 83. Connectivity matrix

Resource	GFXMMU	STM	AHBSRAM1/2	BKPSRAM	FMC	XSPI1/2/3	FLEXRAM	AXISRAM1/2	AXISRAM3/4/5/6	VENCRAM/ AXISRAM8	BootROM	CACHEAXI/ AXISRAM7	ITCM	DTCM	AHBx/APBx
CPU M-AXI	X	X	X	X	X	X	X	X	X	X	X	X	X(1)	X(1)	X
CPU P-AHB	-	-	-	-	-	-	-	-	-	-	-	-	-	-	X(2)
AXI_AP	X	X	X	X	X	X	X	X	X	X	-	X	-	-	X
TRACE	-	-	-	-	X	X	X	X	-	X	-	-	-	-	-
NPU	-	-	-	-	X	X	X	X	X	-	-	X	-	X	-
GPU_M0/1	X	-	-	-	X	X	X	X	X	X	-	X	-	-	-
GPU_CL	-	-	-	-	X	X	X	X	X	X	-	X	-	-	-
ETH1	-	-	-	-	X	X	X	X	X	X	-	X	-	-	-
HPDMA1 AXI	-	-	X	X	X	X	X	X	X	X	-	X	X	X	X
HPDMA1 AHB	-	-	X	X	X	X	X	X	X	X	-	X	-	-	X
GPDMA1	-	-	X	X	X	X	X	X	X	X	-	X	-	-	X
SDMMC1/2	-	-	-	-	X	X	X	X	X	X	-	X	-	-	-
OTG1/2	-	-	-	-	X	X	X	X	X	X	-	X	-	-	-
DCMIPP	-	-	-	-	X	X	X	X	X	X	-	X	-	-	-
DMA2D	X	-	-	-	X	X	X	X	X	X	-	X	-	-	-
LTDC	X	-	-	-	X	X	X	X	X	X	-	X	-	-	-
VENC	-	-	-	-	X	X	X	X	X	-	-	X	-	-	-
GFXMMU	-	-	-	-	X	X	X	X	X	X	-	X	-	-	-

1. The CPU accesses internally to its DTCM and ITCM. The DAP uses the CPU internal to reach TCMs.
2. By default, the CPU uses P-AHB master port to reach the AHBx/APBx target.

18 High-performance direct memory access controller (HPDMA)

18.1 HPDMA introduction

The high-performance direct memory access (HPDMA) controller is a bus master and system peripheral.

The HPDMA is used to perform programmable data transfers between memory-mapped peripherals, and/or memories via linked-lists, upon the control of an off-loaded CPU.

18.2 HPDMA main features

- Single bidirectional 64-bit AXI master and single bidirectional 32-bit AHB master
- Memory-mapped data transfers from a source to a destination:
  - Peripheral-to-memory
  - Memory-to-peripheral
  - Memory-to-memory
  - Peripheral-to-peripheral

  Transfer arbitration based on a 4-grade programmed priority at channel level:
  - One high-priority traffic class, for time-sensitive channels (queue 3)
  - Three low-priority traffic classes, with a weighted round-robin allocation for non time-sensitive channels (queues 0, 1, 2)
- Per channel event generation, on any of the following events: transfer complete, half-transfer complete, data transfer error, user setting error, link transfer error, completed suspension, and trigger overrun
- Per channel interrupt generation, with separately programmed interrupt enable per event
- 16 concurrent HPDMA channels:
  - Per channel FIFO for queuing source and destination transfers (see *Section 18.3.1*)
  - Intra-channel HPDMA transfers chaining via programmable linked-list into memory, supporting two execution modes: run-to-completion and link step mode
- Per linked-list item within a channel:
  - Separately programmed source and destination transfers
  - Programmable data handling between source and destination: byte-based reordering, packing or unpacking, padding or truncation, sign extension and left/right realignment
  - Programmable number of data bytes to be transferred from the source, defining the block level
  - Linear source and destination addressing: either fixed or contiguously incremented addressing, programmed at block level, between successive burst transfers
  - 2D source and destination addressing: programmable signed address offsets between successive burst transfers (non-contiguous addressing within a block,

combined with programmable signed address offsets between successive blocks, at a second 2D/repeated block level, for a reduced set of channels (see *Section 18.3.1*)
  - Support for scatter-gather (multi-buffer transfers), data interleaving and deinterleaving via 2D addressing
  - Selection of programmable HPDMA request and trigger
  - Generation of programmable HPDMA half-transfer and transfer-complete event
  - Pointer to the next linked-list item and its data structure in memory, with automatic update of the HPDMA linked-list control registers
  - Channel abort and restart
- Debug:
  - Channel suspend and resume support
  - Channel status reporting, including FIFO level, and event flags
- TrustZone support:
  - Support for secure and nonsecure HPDMA transfers, independently at a first channel level, and independently at a source/destination and link sublevels
  - Secure and nonsecure interrupts reporting, resulting from any of the respectively secure and nonsecure channels
  - TrustZone-aware AHB slave port, protecting any HPDMA secure resource (register, bitfield) from a nonsecure access
- Privileged/unprivileged support:
  - Support for privileged and unprivileged HPDMA transfers, independently at channel level
  - Privileged-aware AHB slave port
- Channel isolation support:
  - Support for compartmented DMA transfers, independently at channel level, via compartment IDs (named CIDs)
  - CID-aware interrupts reporting
  - CID-aware AHB slave port, with integrated semaphores for a concurrent control, from any of the CPUs

18.3 HPDMA implementation

18.3.1 HPDMA channels

A given HPDMA channel x is implemented with the following features and intended use.

To make the best use of the HPDMA performance, the following table lists some general recommendations, allowing the user to select and allocate a channel, given its implemented FIFO size and the requested HPDMA transfer.

Table 84. Implementation of HPDMA1 channels

Channel x	Hardware parameters		Features
	dma_fifo_size[x]	dma_addressing[x]	
x = 0 to 11	3	0	Channel x (x = 0 to 11) is implemented with: – a FIFO of 16 bytes, 4 words, 2 double-words – fixed/contiguously incremented addressing These channels can be used for HPDMA transfers between an APB or AHB peripheral, an AHB/AXI SRAM, or CPU TCM.
x = 12 to 15	5	1	Channel x (x = 12 to 15) is implemented with: – a FIFO of 64 bytes, 8 double-words – 2D addressing These channels can be also used for HPDMA transfers, including AXI external memories.

18.3.2 HPDMA allowed AXI maximum burst length

For a data transfer, the user has to program:

– the source (and destination) burst length that a given channel x ideally performs, via SBL_1[5:0] (and respectively DBL_1[5:0]) field in HPDMA_CxTR1.

– the allocated master port, AXI or AHB, on which this source (or destination) data transfer must be performed, via SAP (or respectively DAP) field in HPDMA_CxTR1.

Caution: The maximum allowed AXI burst length is restricted to 16:

1. if SAP = 0 (AXI allocated port): the maximum allowed value of SBL_1[5:0] is 15.
2. if DAP = 0 (AXI allocated port): the maximum allowed value of DBL_1[5:0] is 15.

18.3.3 HPDMA in low-power modes

The HPDMA wake-up feature is implemented in the device low-power modes as per the table below.

Table 85. HPDMA1 in low-power modes

Feature	Low-power modes
Wake-up	HPDMA1 in Sleep mode

18.3.4 HPDMA requests

An HPDMA request from a peripheral can be assigned to a HPDMA channel x, via REQSEL[7:0] in HPDMA_CxTR2, provided that SWREQ = 0.

The HPDMA requests mapping is specified in the table below.

Table 86. Programmed HPDMA1 request

HPDMA_CxTR2.REQSEL[7:0]	Selected HPDMA request
0	jpeg_rx_dma
1	jpeg_tx_dma
2	xspi1_dma
3	xspi2_dma
4	xspi3_dma
5	fmc2_txrx_dma
6	fmc2_bch_dma
7	adc1_dma
8	adc2_dma
9	cryp_in_dma
10	cryp_out_dma
11	saes_out_dma
12	saes_in_dma
13	hash_in_dma
14	tim1_cc1_dma
15	tim1_cc2_dma
16	tim1_cc3_dma
17	tim1_cc4_dma
18	tim1_upd_dma
19	tim1_trg_dma
20	tim1_com_dma
21	tim2_cc1_dma
22	tim2_cc2_dma
23	tim2_cc3_dma
24	tim2_cc4_dma
25	tim2_upd_dma
26	tim2_trg_dma
27	tim3_cc1_dma
28	tim3_cc2_dma
29	tim3_cc3_dma
30	tim3_cc4_dma
31	tim3_upd_dma
32	tim3_trg_dma

Table 86. Programmed HPDMA1 request (continued)

HPDMA_CxTR2.REQSEL[7:0]	Selected HPDMA request
33	tim4_cc1_dma
34	tim4_cc2_dma
35	tim4_cc3_dma
36	tim4_cc4_dma
37	tim4_upd_dma
38	tim4_trg_dma
39	tim5_cc1_dma
40	tim5_cc2_dma
41	tim5_cc3_dma
42	tim5_cc4_dma
43	tim5_upd_dma
44	tim5_trg_dma
45	tim6_upd_dma
46	tim7_upd_dma
47	tim8_cc1_dma
48	tim8_cc2_dma
49	tim8_cc3_dma
50	tim8_cc4_dma
51	tim8_upd_dma
52	tim8_trg_dma
53	tim8_com_dma
54	-
55	-
56	tim15_cc1_dma
57	tim15_cc2_dma
58	tim15_upd_dma
59	tim15_trg_dma
60	tim15_com_dma
61	tim16_cc1_dma
62	tim16_upd_dma
63	tim16_com_dma
64	tim17_cc1_dma
65	tim17_upd_dma
66	tim17_com_dma
67	tim18_cc1_dma

Table 86. Programmed HPDMA1 request (continued)

HPDMA_CxTR2.REQSEL[7:0]	Selected HPDMA request
68	tim18_upd_dma
69	tim18_com_dma
70	lptim1_ic1_dma
71	lptim1_ic2_dma
72	lptim1_ue_dma
73	lptim2_ic1_dma
74	lptim2_ic2_dma
75	lptim2_ue_dma
76	lptim3_ic1_dma
77	lptim3_ic2_dma
78	lptim3_ue_dma
79	spi1_rx_dma
80	spi1_tx_dma
81	spi2_rx_dma
82	spi2_tx_dma
83	spi3_rx_dma
84	spi3_tx_dma
85	spi4_rx_dma
86	spi4_tx_dma
87	spi5_rx_dma
88	spi5_tx_dma
89	spi6_rx_dma
90	spi6_tx_dma
91	sai1_a_dma
92	sai1_b_dma
93	sai2_a_dma
94	sai2_b_dma
95	i2c1_rx_dma
96	i2c1_tx_dma
97	i2c2_rx_dma
98	i2c2_tx_dma
99	i2c3_rx_dma
100	i2c3_tx_dma
101	i2c4_rx_dma
102	i2c4_tx_dma

Table 86. Programmed HPDMA1 request (continued)

HPDMA_CxTR2.REQSEL[7:0]	Selected HPDMA request
103	i3c1_rx_dma
104	i3c1_tx_dma
105	i3c2_rx_dma
106	i3c2_tx_dma
107	usart1_rx_dma
108	usart1_tx_dma
109	usart2_rx_dma
110	usart2_tx_dma
111	usart3_rx_dma
112	usart3_tx_dma
113	uart4_rx_dma
114	uart4_tx_dma
115	uart5_rx_dma
116	uart5_tx_dma
117	usart6_rx_dma
118	usart6_tx_dma
119	uart7_rx_dma
120	uart7_tx_dma
121	uart8_rx_dma
122	uart8_tx_dma
123	uart9_rx_dma
124	uart9_tx_dma
125	usart10_rx_dma
126	usart10_tx_dma
127	lpuart1_rx_dma
128	lpuart1_tx_dma
129	spdifrx_cs_dma
130	spdifrx_dt_dma
131	adf1_flt0_dma
132	mdf1_flt0_dma
133	mdf1_flt1_dma
134	mdf1_flt2_dma
135	mdf1_flt3_dma
136	mdf1_flt4_dma
137	mdf1_flt5_dma

Table 86. Programmed HPDMA1 request (continued)

HPDMA_CxTR2.REQSEL[7:0]	Selected HPDMA request
138	ucpd_tx_dma
139	ucpd_rx_dma
140	cci_dma
141	i3c1_tc_dma
142	i3c1_rs_dma
143	i3c2_tc_dma
144	i3c2_rs_dma

18.3.5 HPDMA block requests

Some HPDMA requests must be programmed as a block request, and not as a burst request. Then BREQ in HPDMA_CxTR2 must be set for a correct HPDMA execution of the requested peripheral transfer at the hardware level.

Table 87. Programmed HPDMA1 request as a block request

HPDMA block requests
lptim1/2/3_ue

18.3.6 HPDMA channels with peripheral early termination

An HPDMA channel, if implemented with this feature, can support the early termination of the data transfer from the peripheral which does also support this feature.

Table 88. HPDMA1 channel with peripheral early termination

HPDMA channel x with peripheral early termination
x = 0, x = 1, and x = 15

This HPDMA support is activated when the channel x is programmed with HPDMA_CxTR2.PFREQ = 1. Then, the peripheral itself can initiate and request a data transfer completion, before the HPDMA has transferred the whole block (see *Section 18.4.14* for more details).

Table 89. Programmed HPDMA1 request with peripheral early termination

Programmed HPDMA channel x request with peripheral early termination
i3c1_rx_dma
i3c2_rx_dma
jpeg_tx_dma

18.3.7 HPDMA triggers

An HPDMA trigger can be assigned to an HPDMA channel x, via TRIGSEL[**6**:0] in HPDMA_CXTR2, provided that TRIGPOL[1:0] defines a rising or a falling edge of the selected trigger (TRIGPOL[1:0] = 01 or TRIGPOL[1:0] = 10).

Table 90. Programmed HPDMA1 trigger

HPDMA_CxTR2.TRIGSEL[6:0]	Selected HPDMA trigger
0	dcmipp_p1_frameend_evt
1	dcmipp_p1_lineend_evt
2	dcmipp_p1_hsync_evt
3	dcmipp_p1_vsync_evt
4	dcmipp_p1_frameend_evt
5	dcmipp_p1_lineend_evt
6	dcmipp_p1_hsync_evt
7	dcmipp_p1_vsync_evt
8	dcmipp_p2_frameend_evt
9	dcmipp_p2_lineend_evt
10	dcmipp_p2_hsync_evt
11	dcmipp_p2_vsync_evt
12	dma2d_ctc_flag
13	dma2d_tc_flag
14	dma2d_tw_flag
15	jpeg_eoc_flag
16	jpeg_ifnf_flag
17	jpeg_ift_flag
18	jpeg_ofne_flag
19	jpeg_oft_flag
20	lcd_li_flag
21	gpu2d1_gp_flag[0]
22	gpu2d1_gp_flag[1]
23	gpu2d1_gp_flag[2]
24	gpu2d1_gp_flag[3]
25	gfxtim1_0_gfxtim_evt[3]
26	gfxtim1_0_gfxtim_evt[2]
27	gfxtim1_0_gfxtim_evt[1]
28	gfxtim1_0_gfxtim_evt[0]
29	-
30	lptim1_ch1

Table 90. Programmed HPDMA1 trigger (continued)

HPDMA_CxTR2.TRIGSEL[6:0]	Selected HPDMA trigger
31	lptim1_ch2
32	lptim2_ch1
33	lptim2_ch2
34	lptim3_ch1
35	lptim3_ch2
36	lptim4_out
37	lptim5_out
38	-
39	rtc_wkup
40	lpuart1_it_r_wup_async
41	lpuart1_it_t_wup_async
42	spi6_it_or_spi6_ait_sync
43	-
44	tim1_trgo_cktim
45	tim1_trgo2_cktim
46	tim2_trgo_cktim
47	tim3_trgo_cktim
48	tim4_trgo_cktim
49	tim5_trgo_cktim
50	tim6_trgo_cktim
51	tim7_trgo_cktim
52	tim8_trgo_cktim
53	tim8_trgo2_cktim
54	-
55	-
56	-
57	tim12_trgo_cktim
58	tim15_trgo_cktim
59	-
60	hpdma1_ch0_tc
61	hpdma1_ch1_tc
62	hpdma1_ch2_tc
63	hpdma1_ch3_tc
64	hpdma1_ch4_tc
65	hpdma1_ch5_tc

Table 90. Programmed HPDMA1 trigger (continued)

HPDMA_CxTR2.TRIGSEL[6:0]	Selected HPDMA trigger
66	hpdma1_ch6_tc
67	hpdma1_ch7_tc
68	hpdma1_ch8_tc
69	hpdma1_ch9_tc
70	hpdma1_ch10_tc
71	hpdma1_ch11_tc
72	hpdma1_ch12_tc
73	hpdma1_ch13_tc
74	hpdma1_ch14_tc
75	hpdma1_ch15_tc
76	gpdma1_ch0_tc
77	gpdma1_ch1_tc
78	gpdma1_ch2_tc
79	gpdma1_ch3_tc
80	gpdma1_ch4_tc
81	gpdma1_ch5_tc
82	gpdma1_ch6_tc
83	gpdma1_ch7_tc
84	gpdma1_ch8_tc
85	gpdma1_ch9_tc
86	gpdma1_ch10_tc
87	gpdma1_ch11_tc
88	gpdma1_ch12_tc
89	gpdma1_ch13_tc
90	gpdma1_ch14_tc
91	gpdma1_ch15_tc
92	-
93	extit0_sync
94	extit1_sync
95	extit2_sync
96	extit3_sync
97	extit4_sync
98	extit5_sync
99	extit6_sync
100	extit7_sync

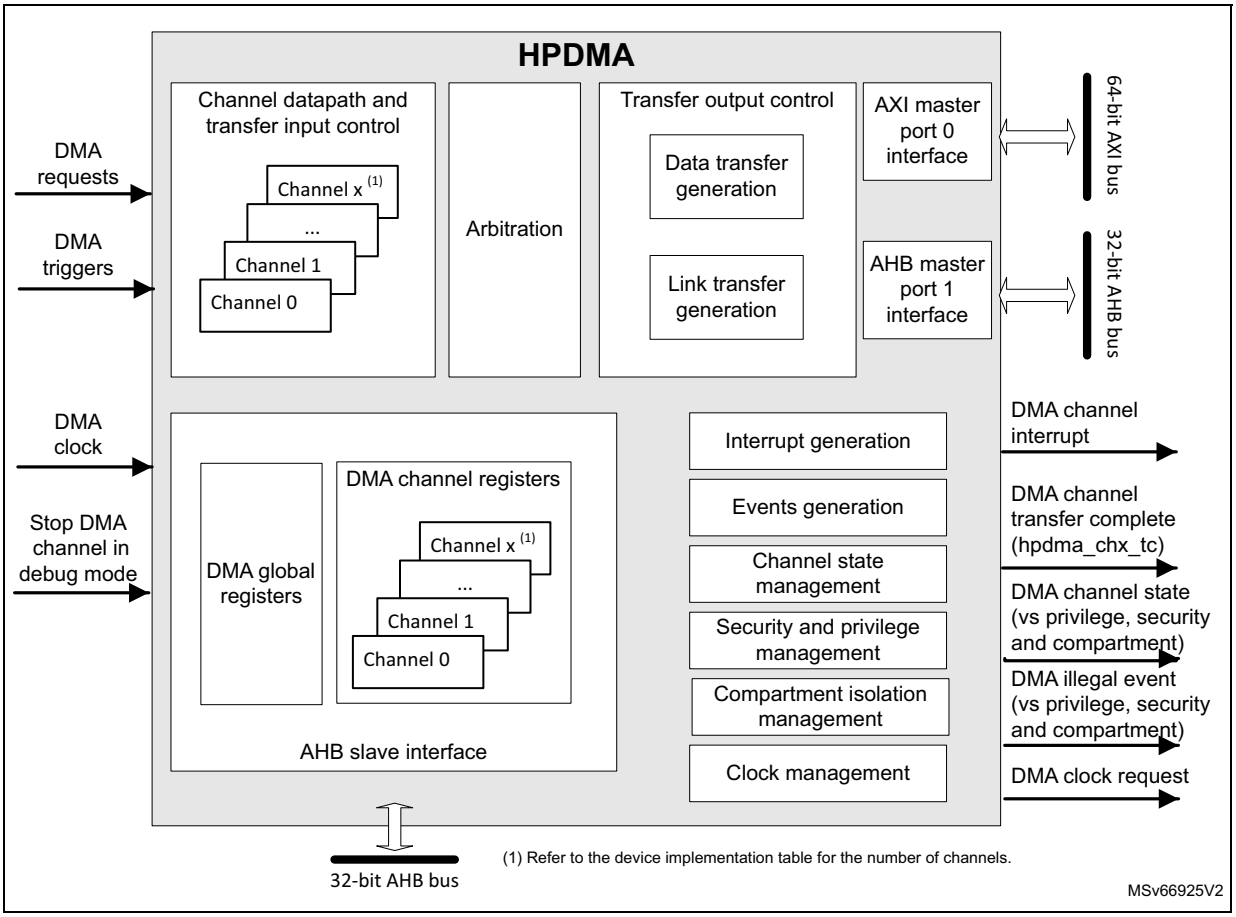
Table 90. Programmed HPDMA1 trigger (continued)

HPDMA_CxTR2.TRIGSEL[6:0]	Selected HPDMA trigger
101	extit8_sync
102	extit9_sync
103	extit10_sync
104	extit11_sync
105	extit12_sync
106	extit13_sync
107	extit14_sync
108	extit15_sync

18.4 HPDMA functional description

18.4.1 HPDMA block diagram

Figure 72. HPDMA block diagram



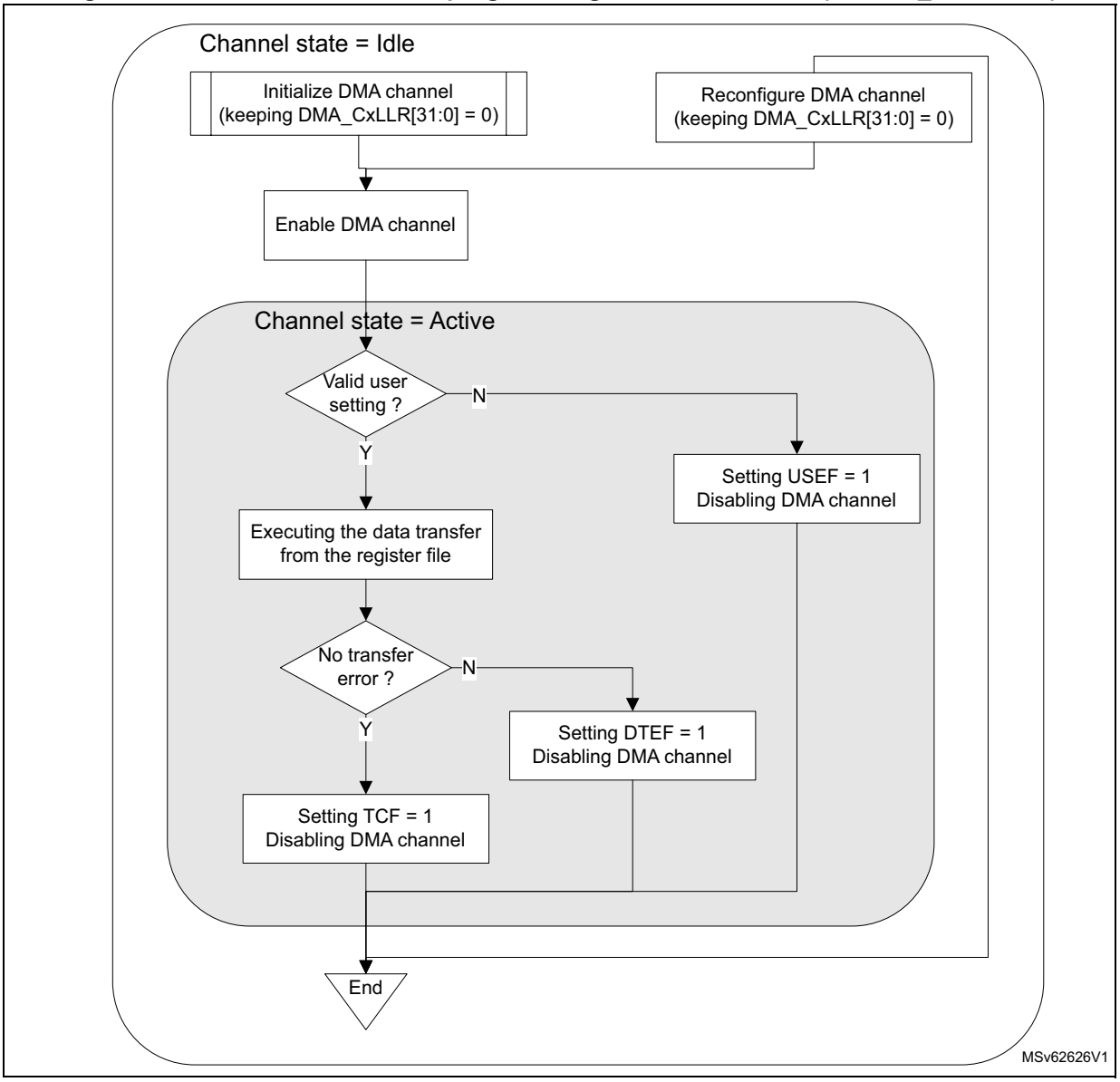
18.4.2 HPDMA channel state and direct programming without any linked-list

After an HPDMA reset, an HPDMA channel x is in idle state. When the software writes 1 into the HPDMA_CxCR.EN enable control bit, the channel takes into account the value of the different channel configuration registers (HPDMA_CxXXX), switches to the active/non-idle state, and starts to execute the corresponding requested data transfers.

After enabling/starting an HPDMA channel transfer by writing 1 into HPDMA_CxCR.EN, an HPDMA channel interrupt on a complete transfer notifies the software that the HPDMA channel is back in idle state (EN is then deasserted by hardware), and that the channel is ready to be reconfigured then enabled again.

The figure below illustrates this HPDMA direct programming without any linked-list (HPDMA_CxLLR = 0).

Figure 73. HPDMA channel direct programming without linked-list (HPDMA_CxLLR = 0)



18.4.3 HPDMA channel suspend and resume

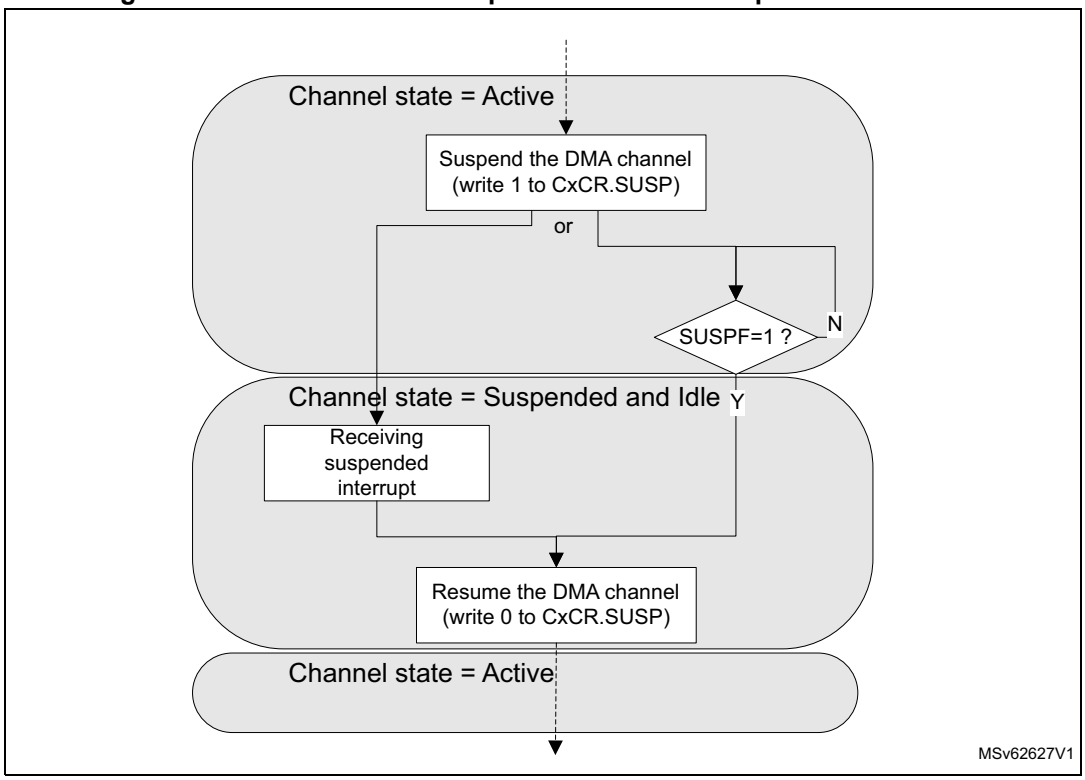
The software can suspend on its own a channel still active, with the following sequence:

1. The software writes 1 into the HPDMA_CxCR.SUSP bit.
2. The software polls the suspended flag HPDMA_CxSR.SUSPF until SUSPF = 1, or waits for an interrupt previously enabled by writing 1 to HPDMA_CxCR.SUSPIE. Wait for the channel to be effectively in suspended state means wait for the completion of any ongoing HPDMA transfer over its master ports. Then the software can observe, in a steady state, any read register or bitfield that is hardware modifiable.

Note: An ongoing HPDMA transfer can be a data transfer (a source/destination burst transfer,) or a link transfer for the internal update of the linked-list register file from the next linked-list item.

3. The software safely resumes the suspended channel by writing 0 to HPDMACxCR.SUSP.

Figure 74. HPDMA channel suspend and resume sequence



Note: A suspend and resume sequence does not impact the HPDMA_*CxCR.EN bit. Suspending a channel (transfer) does not suspend a started trigger detection.*

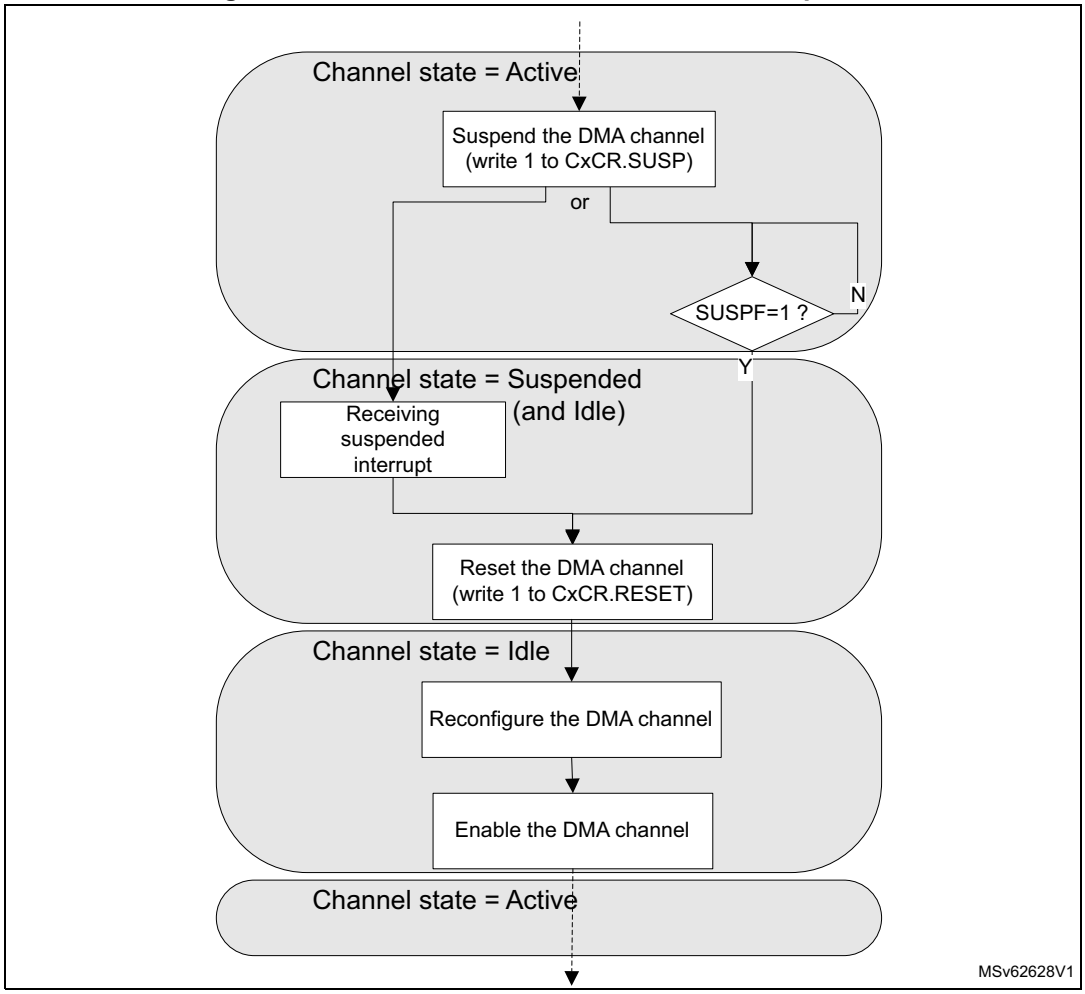
18.4.4 HPDMA channel abort and restart

Alternatively, like for aborting a continuous HPDMA transfer with a circular buffering or a double buffering, the software can abort, on its own, a still active channel with the following sequence:

1. The software writes 1 into the HPDMA_CxCR.SUSP bit.
2. The software polls suspended flag HPDMA_CxSR.SUSPF until SUSPF = 1, or waits for an interrupt previously enabled by writing 1 to HPDMA_CxCR.SUSPIE. Wait for the channel to be effectively in suspended state means wait for the completion of any ongoing HPDMA transfer over its master port.
3. The software resets the channel by writing 1 to HPDMA_CxCR.RESET. This causes the reset of the FIFO, the reset of the channel internal state, the reset of the HPDMA_CxCR.EN bit, and the reset of the HPDMA_CxCR.SUSP bit.
4. The software safely reconfigures the channel. The software must reprogram hardware-modified HPDMA_CxBR1, HPDMA_CxSAR, and HPDMA_CxDAR.

5. In order to restart the aborted then reprogrammed channel, the software enables it again by writing 1 to the HPDMA_CxCR.EN bit.

Figure 75. HPDMA channel abort and restart sequence



18.4.5 HPDMA linked-list data structure

Alternatively to the direct programming mode, a channel can be programmed by a list of transfers, known as a list of linked-list items (LLI). Each LLI is defined by its data structure.

For a channel x, the base address in memory of the data structure of a next LLI_{n+1} is the sum of the following:

- the link base address of the channel x (in HPDMA_CxLBAR)
- the link address offset LA[15:2] bitfield in HPDMA_CxLLR, that is the updated result from the data structure of the previous LLIn of the channel x

The data structure for each LLI can be specific.

A linked-list data structure is addressed following the value of UT1, UT2, UB1, USA, UDA, and ULL bits, plus UB2 and UT3 when present, in HPDMA_CxLLR.

In linked-list mode, each HPDMA linked-list register (HPDMA_CxTR1, HPDMA_CxTR2, HPDMA_CxBR1, HPDMA_CxSAR, HPDMA_CxDAR or HPDMA_CxLLR,

plus HPDMA_CxTR3 or HPDMA_CxBR2 when present) is conditionally and automatically updated from the next linked-list data structure in the memory, following the current value of HPDMA_CxLLR that was conditionally updated from the linked-list data structure of the previous LLI.

Caution: The user must program the pointer to the next linked-list data structure (HPDMA_CxLLR[15:0]) not to exceed the 64-Kbyte addressable space defined by the link base address register (HPDMA_CxLBAR). The complete linked-list data structure must be included in the 64-Kbyte addressable space.

Static linked-list data structure

For example, when the update bits (UT1, UT2, UB1, USA, UDA, and ULL, plus UB2 and UT3 when present) in HPDMA_CxLLR are all asserted, the linked-list data structure in the memory is maximal with:

- for channels without 2D addressing capability: six contiguous 32-bit locations, including HPDMA_CxTR1, HPDMA_CxTR2, HPDMA_CxBR1, HPDMA_CxSAR, HPDMA_CxDAR, and HPDMA_CxLLR (see *Figure 76*) and including the first linked-list register file (LLI_0), and the next LLIs (LLI_1, LLI_2,...) in the memory
- for channels with 2D addressing capability: eight contiguous 32-bit locations, including HPDMA_CxTR1, HPDMA_CxTR2, HPDMA_CxBR1, HPDMA_CxSAR, HPDMA_CxDAR, and HPDMA_CxLLR, plus HPDMA_CxTR3 and HPDMA_CxBR2 (see *Figure 77*), including the first linked-list register file (LLI_0), and the next LLIs (LLI_1, LLI_2,...) in the memory

Figure 76. Static linked-list data structure (all Uxx = 1) of a linear addressing channel x

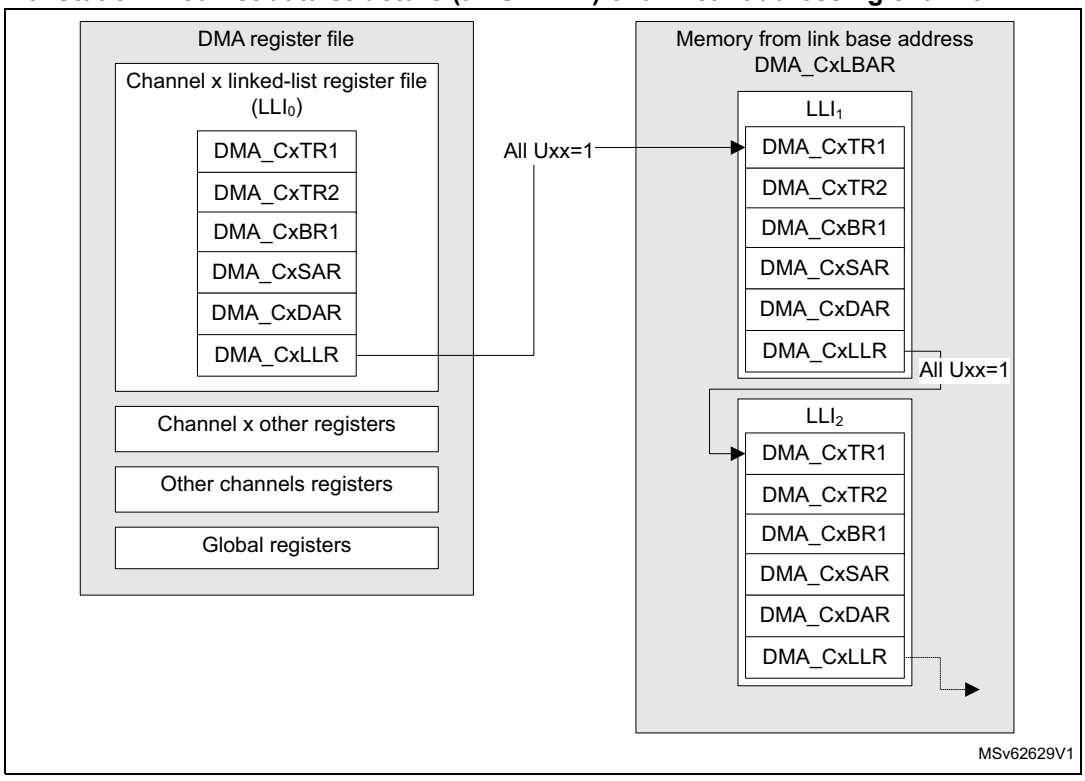
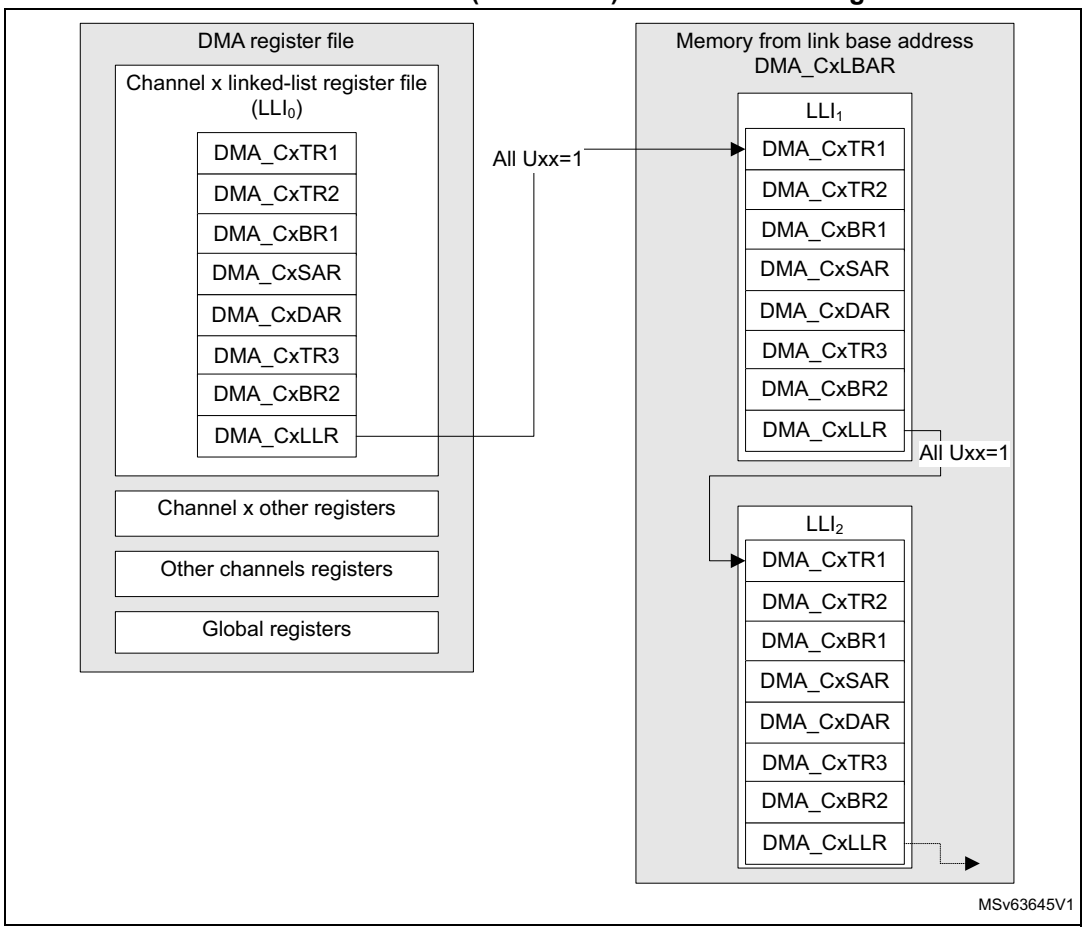


Figure 77. Static linked-list data structure (all Uxx = 1) of a 2D addressing channel x



Dynamic linked-list data structure

Alternatively, the memory organization for the full list of LLIs can be compacted with specific data structure for each LLI.

If UT1 = 0 and UT2 = 1, the link address offset of HPDMA_CxLLR points to the updated value of HPDMA_CxTR2, instead of HPDMA_CxTR1 which is not to be modified (see *Figure 78*).

Example: if UT1 = UB1 = USA = 0, and if UT3 = UB2 = 0 when channel x is with 2D addressing, and if UT2 = UDA = ULL = 1, the next LLI does not contain an (updated) value for HPDMA_CxTR1, nor HPDMA_CxBR1, nor HPDMA_CxSAR, nor HPDMA_CxTR3, nor HPDMA_CxBR2 when channel x is with 2D addressing. The next LLI contains an updated value for HPDMA_CxTR2, HPDMA_CxDAR, and HPDMA_CxLLR, as shown in *Figure 79*.

Figure 78. HPDMA dynamic linked-list data structure of linear addressing channel x

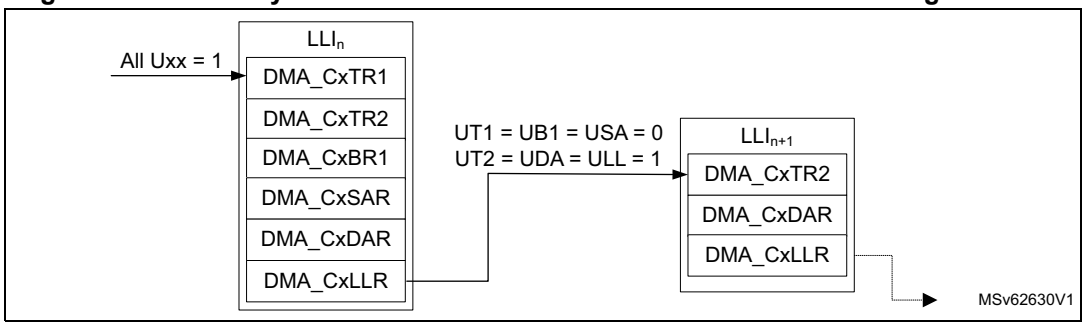
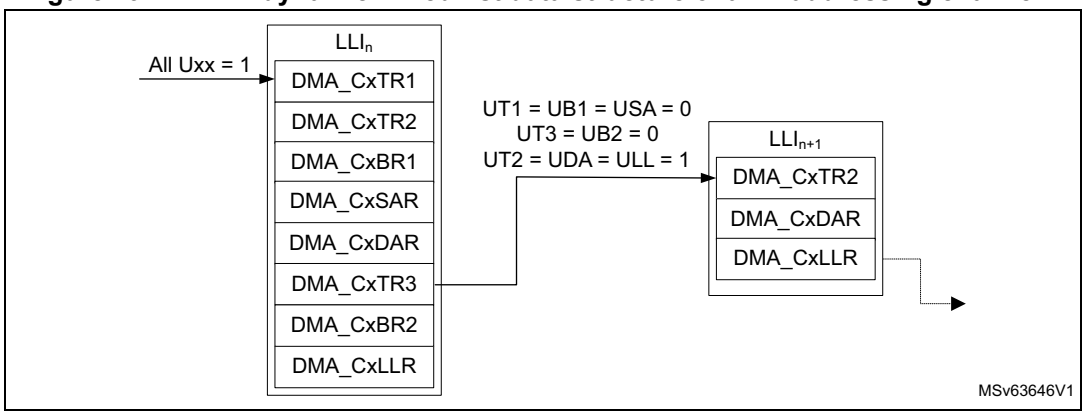


Figure 79. HPDMA dynamic linked-list data structure of a 2D addressing channel x



18.4.6 Linked-list item transfer execution

An LLI_n transfer is the sequence of:

1. a data transfer: the HPDMA executes the data transfer as described by the HPDMA internal register file (this data transfer can be void/null for LLI_0).
2. a conditional link transfer: the HPDMA automatically and conditionally updates its internal register file by the data structure of the next LLI_{n+1}, as defined by the HPDMA_CxLLR value of the LLI_n.

Note: The initial data transfer, as defined by the internal register file (LLI_0), can be null (HPDMA_CxBR1.BNDT[15:0] = 0 and HPDMA_CxTR2.PFREQ = 0), provided that UB1 is set in HPDMA_CxLLR (meaning there is a non-null data transfer described by the next LLI_1 in the memory to be executed).

Depending on the intended HPDMA use, an HPDMA channel x can be executed as described by the full linked-list (run-to-completion mode, HPDMA_CxCR.LSM = 0), or can be programmed for a single execution of a LLI (link step mode, HPDMA_CxCR.LSM = 1), as described in the next sections.

18.4.7 HPDMA channel state and linked-list programming in run-to-completion mode

When HPDMA_CxCR.LSM = 0 (in full-list execution mode, execution of the full sequence of LLIs, named run-to-completion mode), an HPDMA channel x is initially programmed, started

by writing 1 to HPDMA_CxCR.EN, and after completed at channel level.
The channel transfer is:

- configured with at least the following:
  - the first LLI_0, internal linked-list register file: HPDMA_CxTR1, HPDMA_CxTR2, HPDMA_CxBR1, HPDMA_CxSAR, HPDMA_CxDAR, and HPDMA_CxLLR, plus HPDMA_CxTR3 and HPDMA_CxBR2 when present
  - the last LLI_N described by the linked-list data structure in memory, as defined by HPDMA_CxLLR reflecting the before last LLI_{N-1}
- completed when HPDMA_CxLLR[31:0] = 0 and HPDMA_CxBR1.BRC[10:0] = 0, if BRC[10:0] is present and HPDMA_CxBR1.BNDT[15:0] = 0, at the end of the last LLI_{N-1} transfer

HPDMA_CxLLR[31:0] = 0 is the condition of a linked-list based channel completion, and means the following:

- The 16 low significant bits HPDMA_CxLLR.LA[15:0] of the next link address are null.
- All bits HPDMA_CxLLR.Uxx are null (UT1, UT2, UB1, USA, UDA, and ULL, plus UB2 and UT3 when present).

The channel may never be completed when HPDMA_CxLLR.LSM = 0:

- If the last LLI_N is recursive, pointing to itself as a next LLI, in one of the following:
  - HPDMA_CxLLR.ULL = 1 and HPDMA_CxLLR.LA[15:2] is updated with the same value.
  - HPDMA_CxLLR.ULL = 0
- If LLI_N points to a previous LLI

At the regular data transfer completion at a block level, HPDMA_CxBR1.BNDT[15:0] = 0 and HPDMA_CxBR1.BRC[10:0] = 0 (if present). Alternatively, a block transfer may be early completed by a peripheral (such as an I3C in Rx mode), and then BNDT[15:0] is not null (see *Section 18.4.14* for more details).

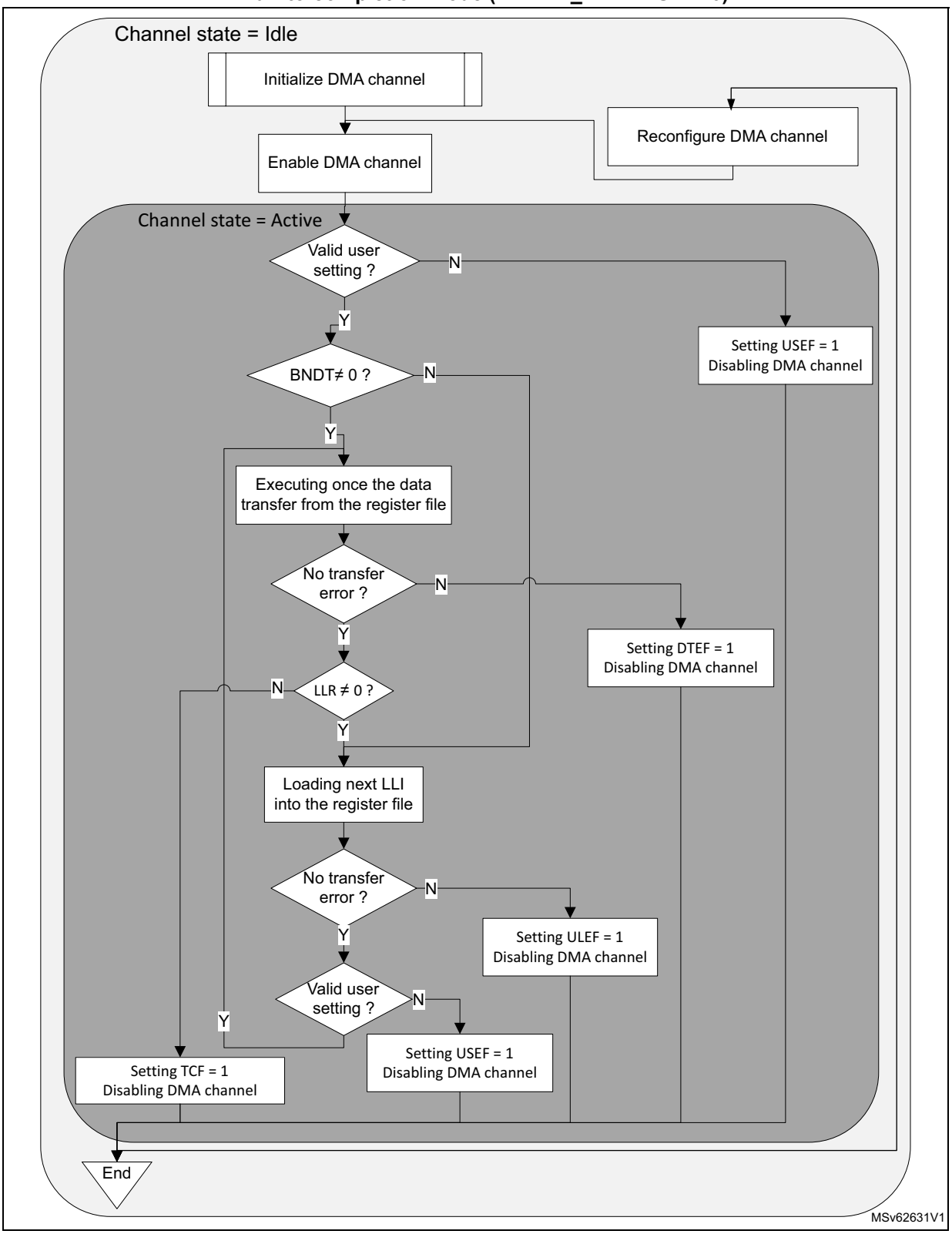
In the typical run-to-completion mode, the allocation of an HPDMA channel, including its fine programming, is done once during the HPDMA initialization. In order to have a reserved data communication link and HPDMA service during run-time, for continuously repeated transfers (from/to a peripheral respectively to/from memory or for memory-to-memory transfers). This reserved data communication link can consist of a channel, or the channel can be shared and a repeated transfer consists of a sequence of LLIs.

Figure 80 depicts the HPDMA channel execution and its registers programming in run-to-completion mode.

Note: Figure 80 is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in HPDMA_CxTR2. It can be raised at (each) block completion, at (each) 2D block completion, at (each) LLI completion, or only at channel completion. In run-to-completion mode, whatever is the value of TCEM[1:0], at the channel completion, the hardware always set TCEF = 1 and disables the channel.

In Figure 80, BNDT ≠ 0 is the typical condition for starting the first data transfer. This condition becomes (BNDT ≠ 0 and PFREQ = 1) if the peripheral requests a data transfer with early termination (see Section 18.3.6).

Figure 80. HPDMA channel execution and linked-list programming in run-to-completion mode (HPDMA_CxCR.LSM = 0)



Run-time inserting a LLI_n via an auxiliary channel, in run-to-completion mode

The start of the link transfer of the LLI_{n-1} (start of the LLI_n loading) can be conditioned by the occurrence of a trigger, when programming the following bitfields of HPDMA_CxTR2 in the data structure of the LLI_{n-1}:

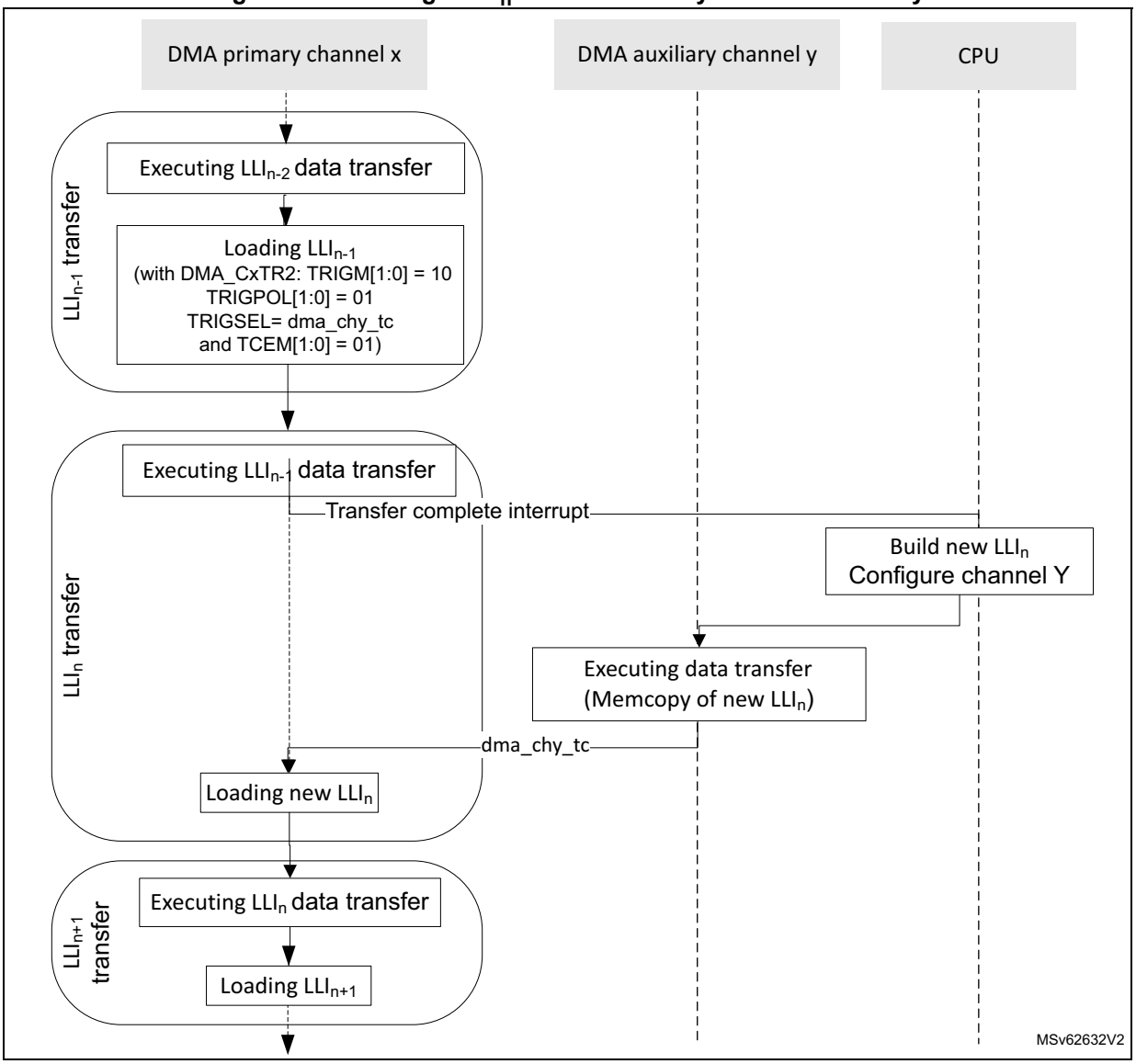
- TRIGM[1:0] = 10 (link transfer triggering mode)
- TRIGPOL[1:0] = 01 or 10 (rising or falling edge)
- TRIGSEL[6:0] (see *Section 18.3.7* for the trigger selection details)

Another auxiliary channel y can be used to store the channel x LLI_n in the memory, and to generate a transfer complete event hpdma_chy_tc. By selecting this event as the input trigger of the link transfer of the LLI_{n-1} of the channel x, the software can pause the primary channel x after its LLI_{n-1} data transfer, until it is indeed written the LLI_n.

Figure 81 depicts such a dynamic elaboration of a linked-list of a primary channel x, via another auxiliary channel y.

Caution: This use case is restricted to an application with an LLI_{n-1} data transfer that does not need a trigger. The triggering mode of this LLI_{n-1} is used to load the next LLI_n.

Figure 81. Inserting a LLI_n with an auxiliary HPDMA channel y



18.4.8 HPDMA channel state and linked-list programming in link step mode

When HPDMA_CxCR.LSM = 1 (in link step execution mode, single execution of one LLI), a channel transfer is executed and completed after each single execution of a LLI, including its (conditional) data transfer and its (conditional) link transfer.

An HPDMA channel transfer can be programmed at an LLI level, started by writing 1 into HPDMA_CxCR.EN, and after completed at LLI level:

- The current LLI_n transfer is described with the following:
  - HPDMA_CxTR1 defines the source/destination elementary single/burst transfers.
  - HPDMA_CxBR1 defines the number of bytes at a block level (BNDT[15:0]), and for channels with 2D addressing it defines also the number of blocks at a

2D/repeated block level (BRC[10:0]+1), and the incrementing/decrementing mode for address offsets.

  - HPDMA_CxTR2 defines the input control (request, trigger), and the output control (transfer-complete event) of the transfer.
  - HPDMA_CxSAR and HPDMA_CxDAR define the source/destination transfer start address.
  - HPDMA_CxTR3 for channels with 2D addressing defines the source/destination additional address offset between burst transfers.
  - HPDMA_CxBR2 for channels with 2D addressing defines the source/destination additional address offset between blocks at a 2D/repeated block level.
  - HPDMA_CxLLR defines the data structure, and the address offset of the next LLI_{n+1} in the memory.
- The current LLI_n transfer is completed after the single execution of the current LLI_n:
  - after the (conditional) data transfer completion (when HPDMA_CxBR1.BRC[10:0] = 0 if BRC[10:0] is present, and HPDMA_CxBR1.BNDT[15:0] = 0)
  - after the (conditional) update of the HPDMA link register file from the data structure of the next LLI_{n+1} in memory

Note: If an LLI is recursive (pointing to itself as a next LLI, either HPDMA_CxLLR.ULL = 1 and HPDMA_CxLLR.LA[15:2] is updated with the same value, or HPDMA_CxLLR.ULL = 0), a channel in link step mode is completed after each repeated single execution of this LLI.

At the regular data transfer completion at a block level, HPDMA_CxBR1.BNDT[15:0] = 0 and HPDMA_CxBR1.BRC[10:0] = 0 (if present). Alternatively, a block transfer can be early completed by a peripheral (such as an I3C in Rx mode), and then BNDT[15:0] is not null (see *Section 18.4.14* for more details).

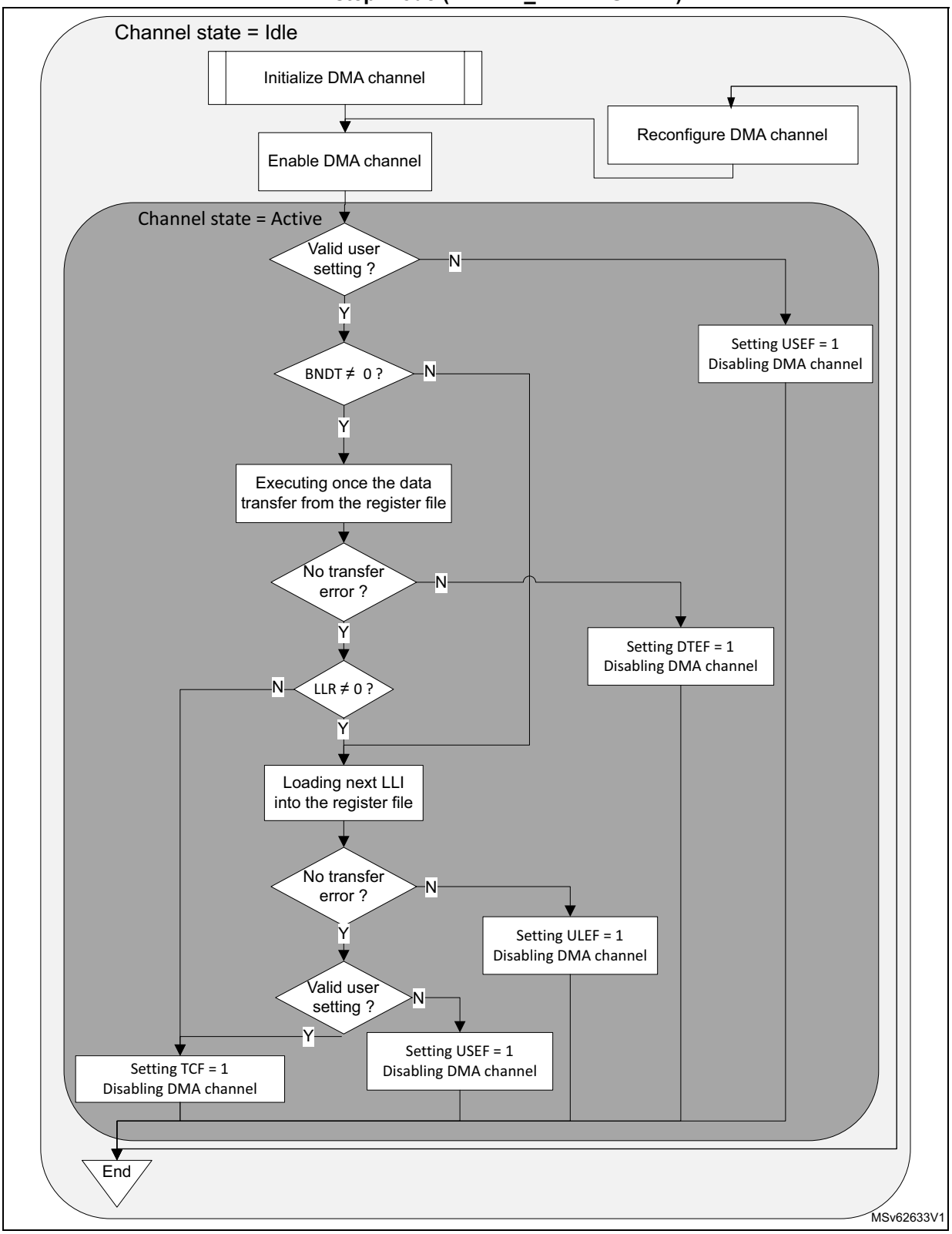
The link step mode can be used to elaborate dynamically LLIs in memory during run-time. The software can be facilitated by using a static data structure for any LLI_n (all update bits of HPDMA_CxLLR have a static value, LLI_n.LLR.LA = LLI_{n-1}.LLR.LA + constant).

Figure 82 depicts the HPDMA channel execution mode, and its programming in link step mode.

Note: Figure 82 is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in HPDMA_CxTR2. It can be raised at (each) block completion, at (each) 2D block completion, at (each) LLI completion, or only at the last LLI data transfer completion. In link step mode, the channel is disabled after each single execution of a LLI, and depending on the value of TCEM[1:0] a TCEF is raised or not.

In Figure 82, BNDT ≠ 0 is the typical condition for starting the first data transfer. This condition becomes BNDT ≠ 0 and PFREQ = 1 if the peripheral requests a data transfer with early termination (see Section 18.3.6).

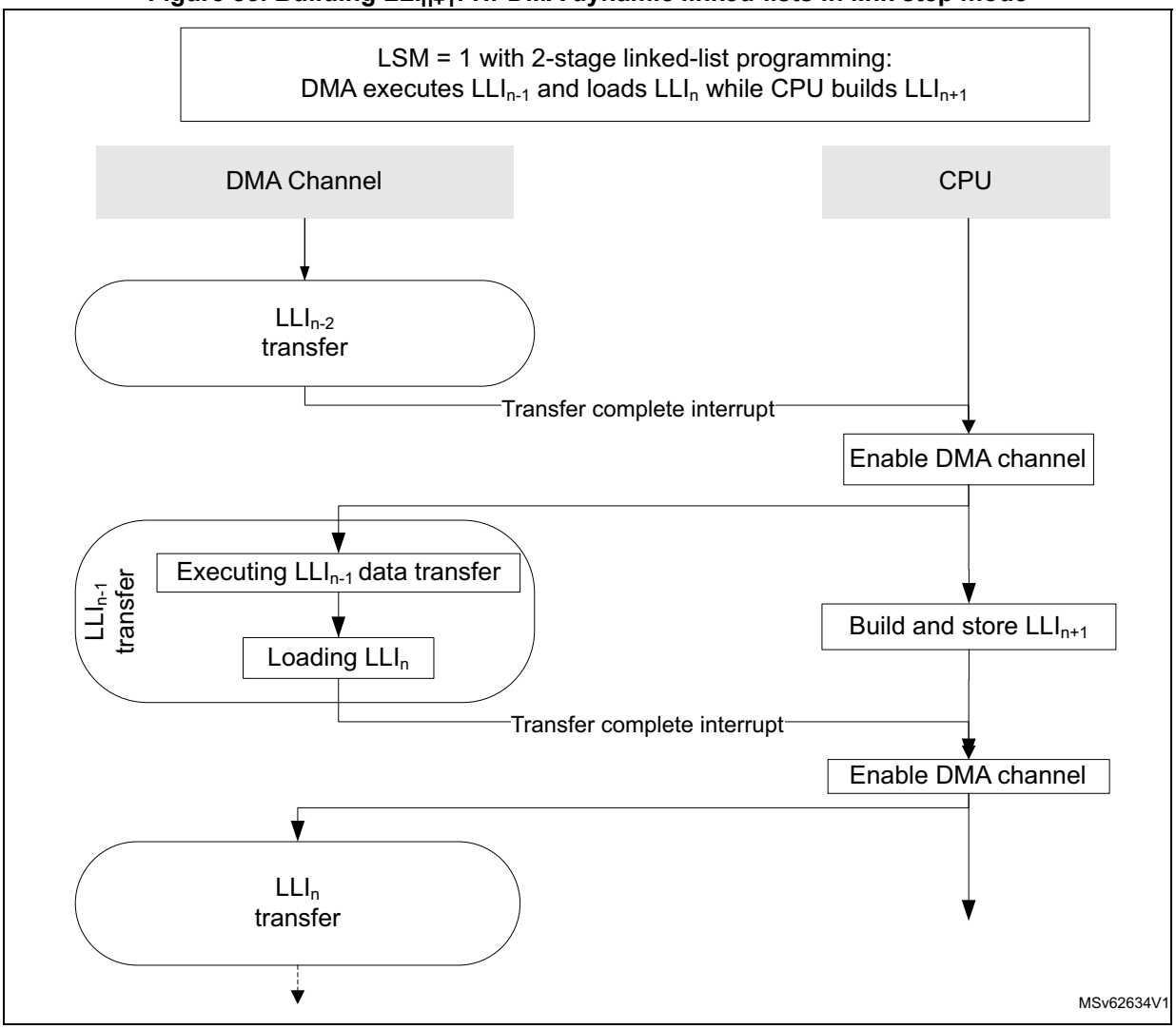
Figure 82. HPDMA channel execution and linked-list programming in link step mode (HPDMA_CxCR.LSM = 1)



Run-time adding a LLI_{n+1} in link step mode

During run-time, the software can defer the elaboration of the LLI_{n+1} (and next LLIs), until/after the HPDMA executed the transfer from the LLI_{n-1} and loaded the LLI_n from the memory, as shown in the figure below.

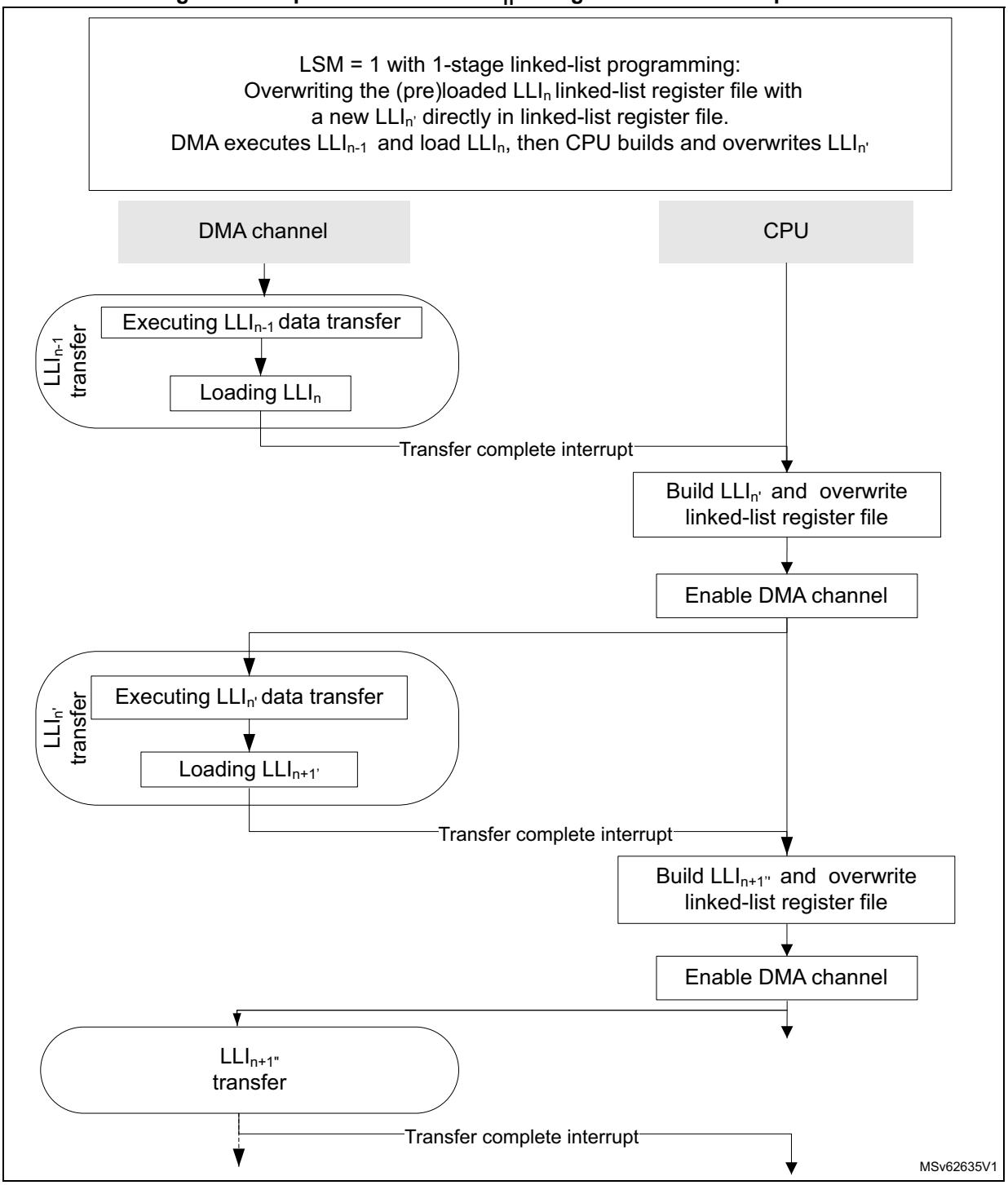
Figure 83. Building LLI_{n+1}: HPDMA dynamic linked-lists in link step mode



Run-time replacing an LLI_n with a new LLI_n', in link step mode (in linked-list register file)

In this link step mode, during run-time, the software can build and insert a new LLI_n', after the HPDMA executed the transfer from the LLI_{n-1}, and loaded a formerly elaborated LLI_n from the memory by overwriting directly the linked-list register file with the new LLI_n', as shown in *Figure 84*.

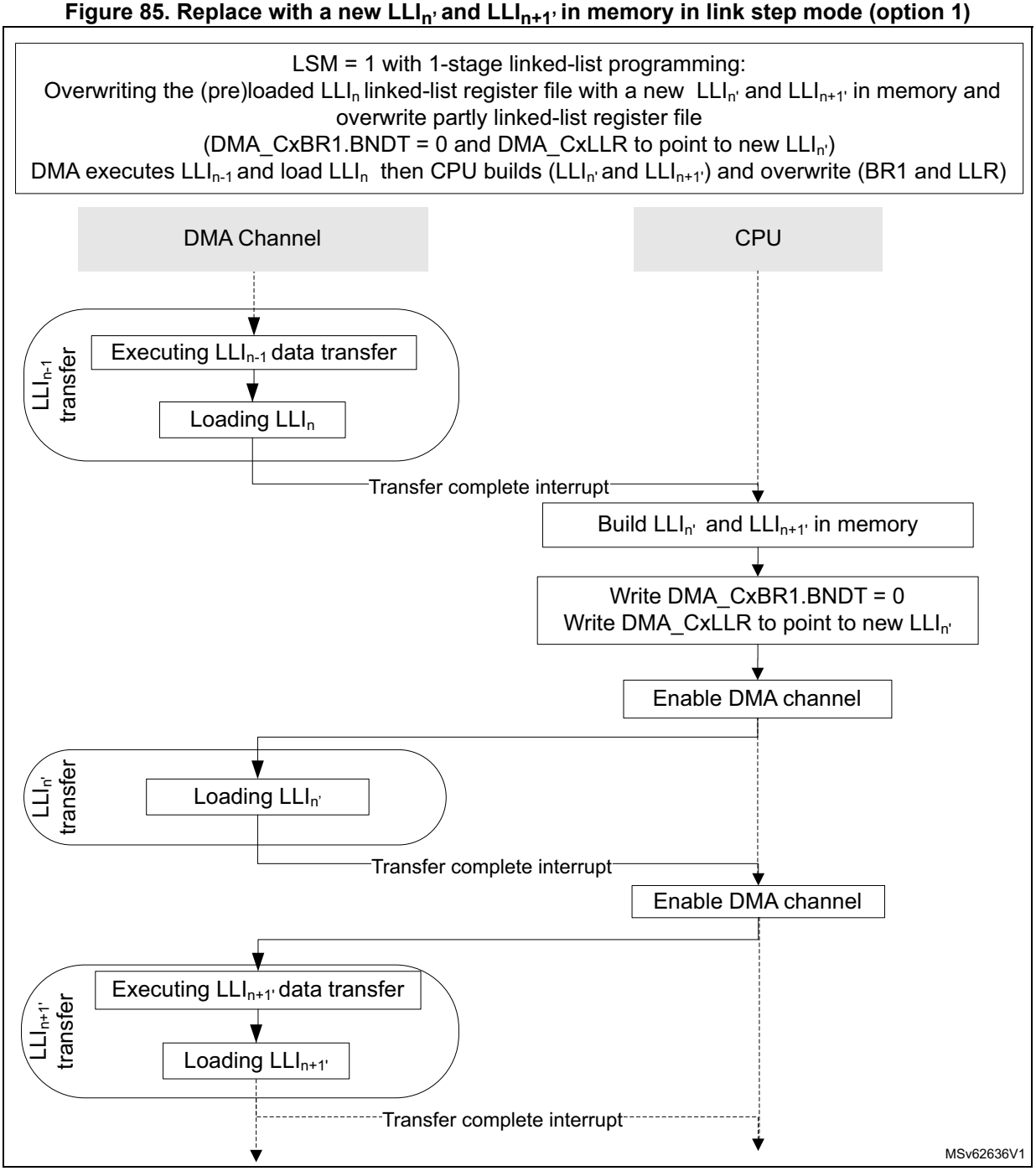
Figure 84. Replace with a new $LLI_{n'}$ in register file in link step mode



Run-time replacing an LLI_n with a new $LLI_{n'}$ in link step mode (in the memory)

The software can build and insert a new $LLI_{n'}$ and $LLI_{n+1'}$ in the memory, after HPDMA executed the transfer from the LLI_{n-1}, and loaded a formerly elaborated LLI_n from the memory, by overwriting partly the linked-list register file (HPDMA_CxBR1.BNDT[15:0] to be null, and HPDMA_CxLLR to point to new LLI_n) as shown in *Figure 85*.

Figure 85. Replace with a new $LLI_{n'}$ and $LLI_{n+1'}$ in memory in link step mode (option 1)



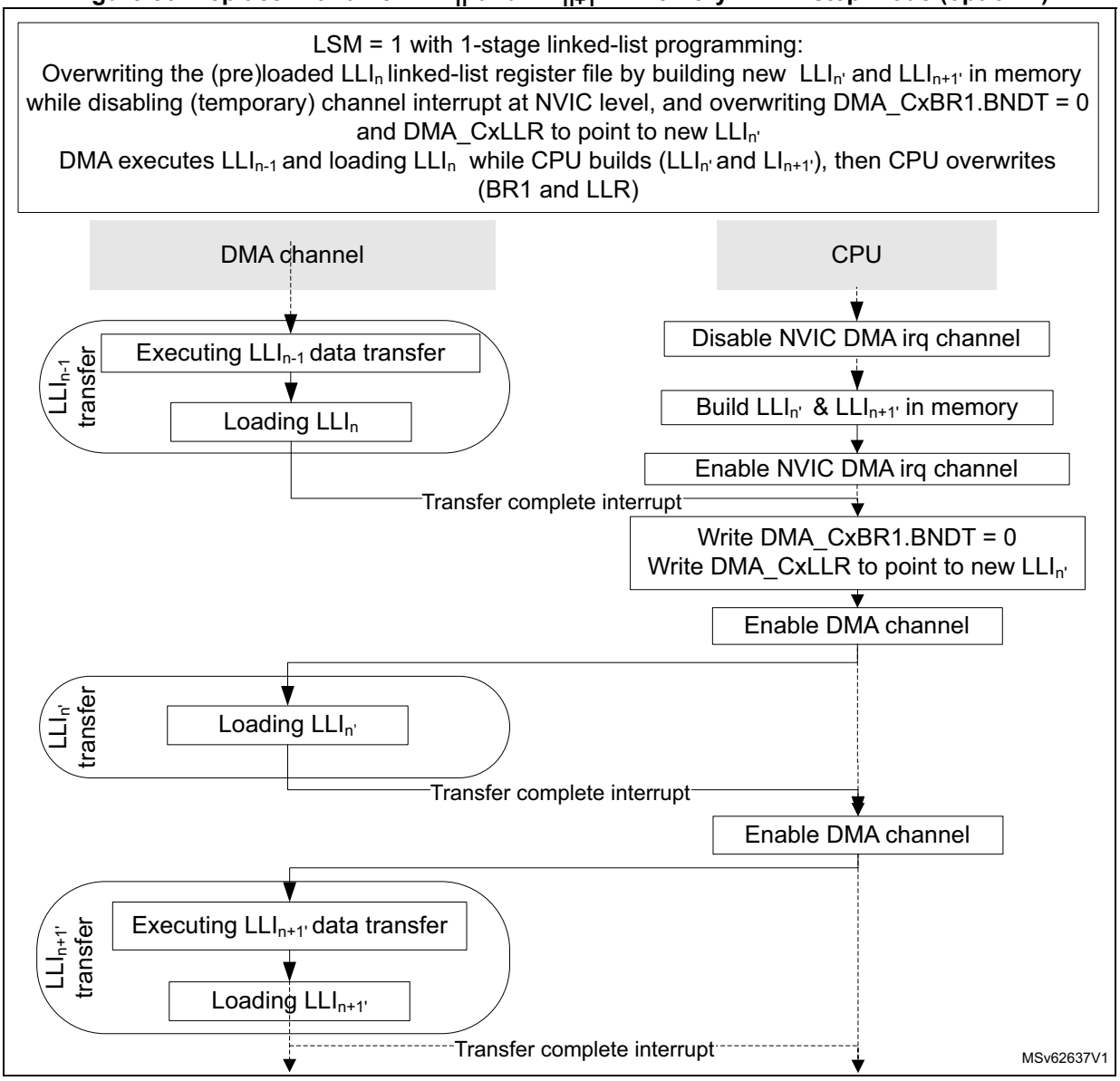
Run-time replacing a LLI_n with a new $LLI_{n'}$ in link step mode

Other software implementations exist. Meanwhile the HPDMA executes the transfer from the LLI_{n-1} and loads a formerly elaborated LLI_n from the memory (or even earlier), the software can do the following:

1. Disable the NVIC for not being interrupted by the interrupt handling.
2. Build a new $LLI_{n'}$ and a new $LLI_{n+1'}$.
3. Enable again the NVIC for the channel interrupt (transfer complete) notification.

The software in the interrupt handler for LLI_{n-1} is then restricted to overwrite HPDMA_CxBR1.BNDT[15:0] to be null and HPDMA_CxLLR to point to new $LLI_{n'}$, as shown in the figure below.

Figure 86. Replace with a new $LLI_{n'}$ and $LLI_{n+1'}$ in memory in link step mode (option 2)



18.4.9 HPDMA channel state and linked-list programming

The software can reconfigure a channel when the channel is disabled (HPDMA_CxCR.EN = 0), and update the execution mode (HPDMA_CxCR.LSM) to change from/to run-to-completion mode to/from link step mode.

In any execution mode, the software can:

- reprogram LLI_{n+1} in the memory to finally complete the channel by this LLI_{n+1} (clear the HPDMA_CxLLR of this LLI_{n+1}), before this LLI_{n+1} is loaded/used by the HPDMA channel
- abort and reconfigure the channel with an LSM update (see *Section 18.4.4*.)

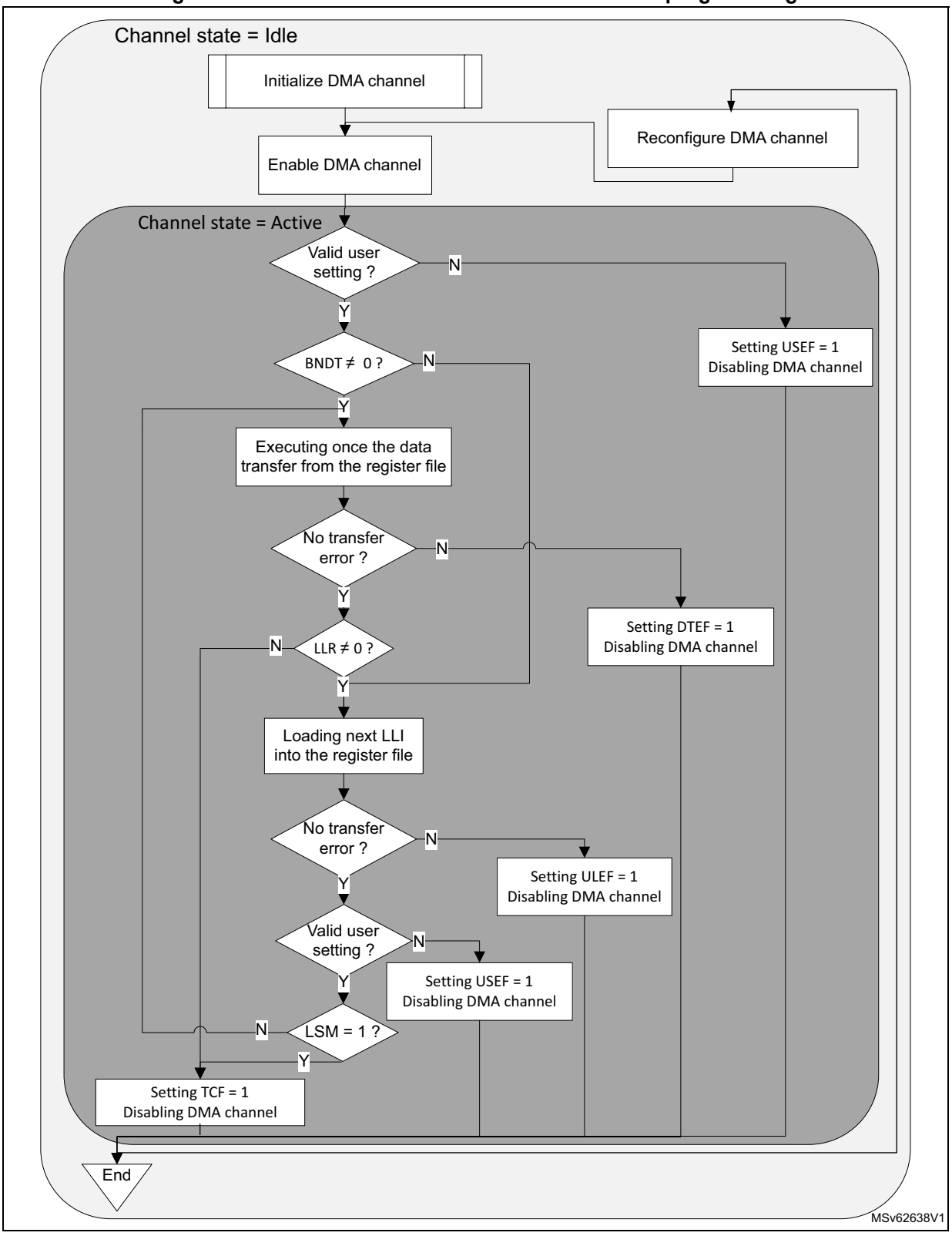
In link step mode, the software can clear LSM after each a single execution of any LLI, during LLI_{n-1}.

Figure 87 shows the overall and unified HPDMA linked-list programming, whatever is the execution mode.

Note: Figure 87 is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in HPDMA_CxTR2. It can be raised at (each) block completion, at (each) 2D block completion, at (each) LLI completion, or only at the last LLI data transfer completion. In run-to-completion mode, whatever is the value of TCEM[1:0], at the channel completion the hardware always set TCEF = 1 and disables the channel. In link step mode, the channel is disabled after each single execution of a LLI, and depending on the value of TCEM[1:0] a TCEF is raised or not.

In Figure 87, BNDT ≠ 0 is the typical condition for starting the first data transfer. This condition becomes BNDT ≠ 0 and PFREQ = 1 if the peripheral requests a data transfer with early termination (see Section 18.3.6).

Figure 87. HPDMA channel execution and linked-list programming



18.4.10 HPDMA FIFO-based transfers

There is a single transfer operation mode: the FIFO mode. There are FIFO-based transfers. Any channel x is implemented with a dedicated FIFO whose size is defined by dma_fifo_size[x] (see *Section 18.3.1* for more details).

HPDMA burst

A programmed transfer at the lowest level is an HPDMA burst.

An HPDMA burst is a burst of data received from the source, or a burst of data sent to the destination. A source (and destination) burst is programmed with a burst length by SBL_1[5:0] (respectively DBL_1[5:0]), and with a data width defined by SDW_LOG2[1:0] (respectively DDW_LOG2[1:0]) in HPDMA_CxTR1.

The addressing mode after each data (named beat) of an HPDMA burst is defined by SINC and DINC in HPDMA_CxTR1, for source and destination respectively: either a fixed addressing or an incremented addressing with contiguous data.

The start and next addresses of an HPDMA source/destination burst (defined by HPDMA_CxSAR and HPDMA_CxDAR) must be aligned with the respective data width.

The table below lists the main characteristics of an HPDMA burst.

Table 91. Programmed HPDMA source/destination burst

SAP/DAP (allocated port)	SDW_LOG2[1:0] DDW_LOG2[1:0]	Data width (bytes)	SINC/DINC	SBL_1[5:0] DBL_1[5:0]	Burst length (data/ beats)	Next data/ beat address	Next burst address	Burst address alignment
0: AXI 1: AHB	00	1	0 (fixed)	n = 0 to 63(1)(2)	n+1	+ 0	+ 0	1
	01	2						2
	10	4						4
0: AXI	11	8						8
0: AXI 1: AHB	00	1	1 (contiguously incremented)			+ 1	+ (n + 1)	1
	01	2				+ 2	+ 2 * (n + 1)	2
	10	4				+ 4	+ 4 * (n + 1)	4
0: AXI	11	8				+ 8	+ 8 * (n + 1)	8
0: AHB	11	forbidden user setting, causing USEF generation and none burst to be issued.						

1. When S/DBL_1[5:0] = 0, burst is of length 1. Then burst can be also named as single.
2. As highlighted in *Section 18.3.2*, the maximum allowed AXI burst length is 16. The user must set SBL_1[5:0] lower or equal to 15 if the source allocated port is AXI (if SAP = 0). The user must set DBL_1[5:0] lower or equal to 15 if the destination allocated port is AXI (if DAP = 0).

The next burst address in the above table is the next source/destination default address pointed by HPDMA_CxSAR or HPDMA_CxDAR, once the programmed source/destination burst is completed. This default value refers to the fixed/contiguously incremented address.

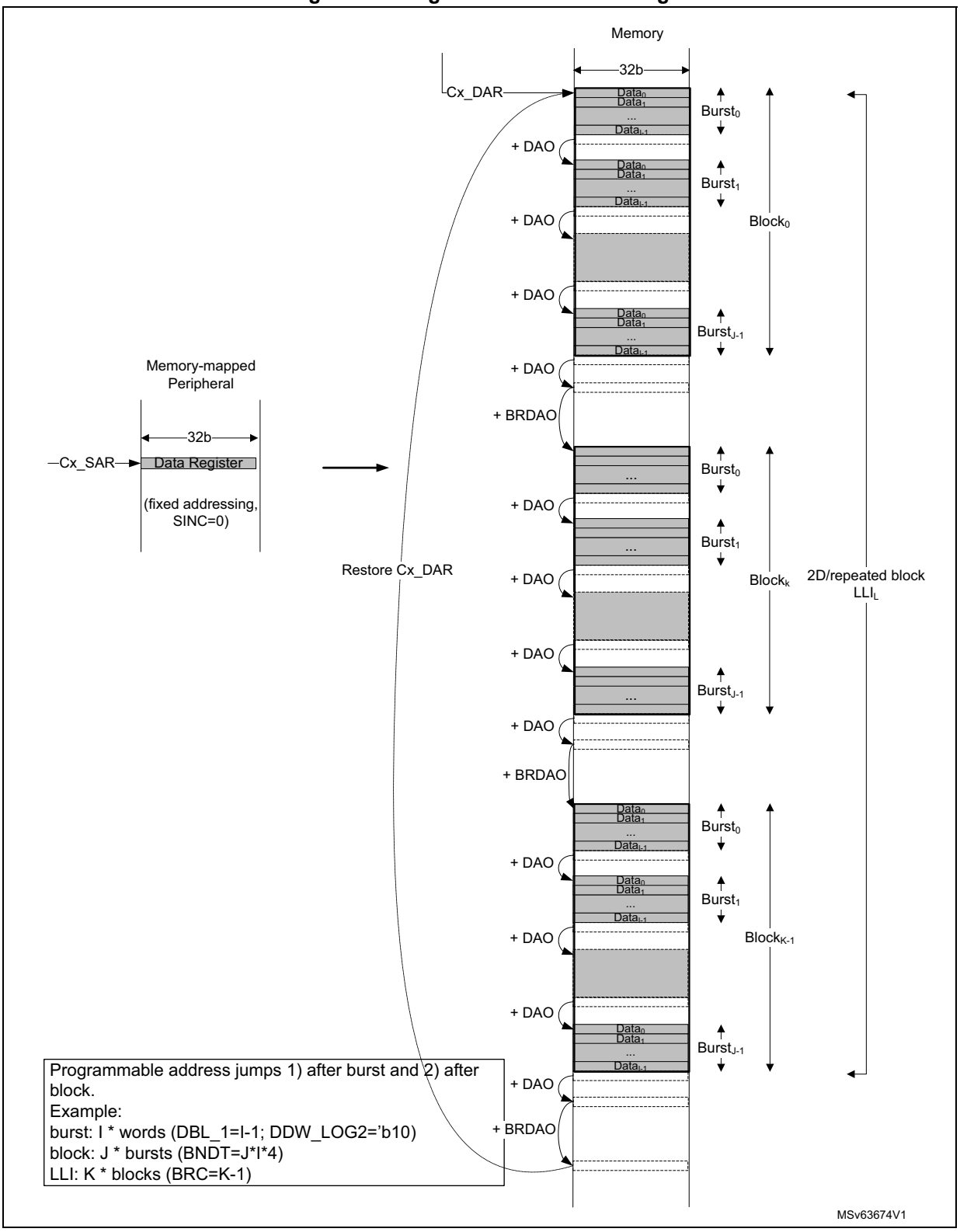
HPDMA burst with 2D addressing

When the channel has additional 2D addressing feature, this default value refers to the value without taking into account the two programmed incremented or decremented offsets. These two additional offsets (with a null default value) are applied:

- after each completed source/destination burst, as defined respectively by HPDMA_CxTR2.SAO[12:0]/DAO[12:0] and HPDMA_CxBR1.SDEC/DDEC
- after each completed block, as defined respectively by HPDMA_CxBR2.BRSAO[15:0]/BRDAO[15:0] and HPDMA_CxBR1.BRSDEC/BRDDEC)

Then, a 2D/repeated block can be addressed with a first programmed address jump after each completed burst, and with a second programmed address jump after each block, as depicted by *Figure 88* with a 2D destination buffer.

Figure 88. Programmed 2D addressing



HPDMA FIFO-based burst

In FIFO-mode, a transfer generally consists of two pipelined and separated burst transfers:

- one burst from the source to the FIFO over the allocated source master port, as defined by HPDMA_CxTR1.SAP
- one burst from the FIFO to the destination over the allocated destination master port, as defined by HPDMA_CxTR1.DAP

HPDMA source burst

The requested source burst transfer to the FIFO can be scheduled as early as possible over the allocated port, depending on the current FIFO level versus the programmed burst size (when the FIFO is ready to get one new burst from the source):

when FIFO level ≤ $2^{dma_fifo_size[x]}$ - (SBL_1[5:0]+1) * $2^{SDW_LOG2[1:0]}$

where:

- FIFO level is the current filling level of the FIFO, in bytes.
- $2^{dma_fifo_size[x]}$ is half the FIFO size of channel x, in bytes (see *Section 18.3.1* for the implementation details and dma_fifo_size[x] value).
- (SBL_1[5:0]+1) * $2^{SDW_LOG2[1:0]}$ is the size of the programmed source burst transfer, in bytes.

Based on the channel priority (HPDMA_CxCR.PRIO[1:0]), this ready FIFO-based source transfer is internally arbitrated versus the other requested and active channels.

HPDMA destination burst

The requested destination burst transfer from the FIFO can be scheduled as early as possible over the allocated port, depending on the current FIFO level versus the programmed burst size (when the FIFO is ready to push one new burst to the destination):

when FIFO level ≥ (DBL_1[5:0]+1) * $2^{DDW_LOG2[1:0]}$

where:

- FIFO level is the current filling level of the FIFO, in bytes.
- (DBL_1[5:0]+1) * $2^{DDW_LOG2[1:0]}$ is the size of the programmed destination burst transfer, in bytes.

Based on the channel priority, this ready FIFO-based destination transfer is internally arbitrated versus the other requested and active channels.

HPDMA burst versus source block size, 1- or 4-Kbyte address boundary and FIFO size

The programmed source/destination HPDMA burst is implemented with an AHB/AXI burst as is, unless one of the following conditions is met:

- When half of the FIFO size of the channel x is lower than the programmed source/destination burst size, the programmed source/destination HPDMA burst is implemented with a series of singles or bursts of a lower size. Each transfer is of a size that is lower than or equal than half of the FIFO size, without any user constraint.
- If the source block size (HPDMA_CxBR1.BNDT[15:0]) is not a multiple of the source burst size but is a multiple of the data width of the source burst (HPDMA_CxTR1.SDW_LOG2[1:0]), the HPDMA modifies and shortens bursts into

singles or bursts of lower length, in order to transfer exactly the source block size, without any user constraint.

- if the source/destination burst transfer have crossed the 1- or 4-Kbyte address boundary on, respectively, an AHB or AXI transfer, the HPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB/AXI protocol, without any user constraint.
- If the source/destination burst length exceeds 16 on a AHB transfer, or if the source/destination burst on an AXI transfer is both with fixed addressing and with a burst length which exceeds 16, the HPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB/AXI protocol, without any user constraint.

In any case, the HPDMA keeps ensuring source/destination data (and address) integrity without any user constraint. The current FIFO level (software readable in HPDMA_CxSR) is compared to and updated with the effective transfer size, and the HPDMA re-arbitrates between each AHB/AXI single or burst transfer, possibly modified.

Based on the channel priority, each single or burst of a lower burst size versus the programmed burst, is internally arbitrated versus the other requested and active channels.

Note: In linked-list mode, the HPDMA read transfers related to the update of the linked-list parameters from the memory to the internal HPDMA registers, are scheduled over the link allocated port, as programmed by HPDMA_CxCR.LAP.

HPDMA data handling: byte-based reordering, packing/unpacking, padding/truncation, sign extension, and left/right alignment

The data handling is controlled by HPDMA_CxTR1. The source/destination data width of the programmed burst is byte, half-word, word, or double-word, as per SDW_LOG2[1:0] and DDW_LOG2[1:0] (see *Table 92*).

The user can configure the data handling between transferred data from the source and transfer to the destination. More specifically, programmed data handling is orderly performed with:

1. Byte-based source reordering
   - If SBX = 1 and if the source data width is a word or a double-word (for AXI source bus, SAP = 0), the two bytes of the unaligned half-word at the middle of each source data word are exchanged.
2. Data-width conversion by packing, unpacking, padding, or truncation, if destination data width is different than the source data width, depending on PAM[1:0]:
   - If destination data width > source data width, the post SBX source data is either right-aligned and padded with 0s, or sign extended up to the destination data width, or is FIFO queued and packed up to the destination data width.
   - If destination data width < source data width, the post SBX data is either right-aligned and left-truncated down to the destination data width, or is FIFO queued and unpacked and streamed down to the destination data width.

3. Byte-based destination re-ordering:
   - If DBX = 1 and if the destination data width is not a byte, the two bytes are exchanged within the aligned post PAM[1:0] half-words.
   - If DHX = 1 and if the destination data width is neither a byte nor a half-word, the two aligned half-words are exchanged within the aligned post PAM[1:0] words.
   - If DWX = 1 and if the destination data width is a double-word and if the selected destination port (via DAP) is 64-bit capable, the two aligned words are exchanged within aligned (post PAM[1:0]) double-words).

Note: Left-alignment with 0s-padding can be achieved by programming both a right-alignment with a 0s-padding, and a destination byte-based re-ordering.

The table below lists the possible data handling from the source to the destination.

Table 92. Programmed data handling

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	DWX	Destination data stream(1)
00	Byte	$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$	x	00	Byte	xx	x	x	x	$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$
				01	Half-word	00 (RA, 0P)(3)(4)	0			$0B_3,0B_2,0B_1,0B_0$
							1			B_30,B_20,B_10,B_00
						01 (RA, SE)(3)(4)	0			SB_3,SB_2,SB_1,SB_0
							1			B_3S,B_2S,B_1S,B_0S
						1x (PACK)	0			$B_7B_6,B_5B_4,B_3B_2,B_1B_0$
							1			$B_6B_7,B_4B_5,B_2B_3,B_0B_1$
				10	Word	00 (RA, 0P)(3)(4)	0	0		$000B_1,000B_0$
							1			$00B_10,00B_00$
							0	1		$0B_100,0B_000$
							1			B_1000,B_0000
						01 (RA, SE)(3)(4)	0	0		$SSSB_1,SSSB_0$
							1			SSB_1S,SSB_0S
							0	1		SB_1SS,SB_0SS
							1			B_1SSS,B_0SSS
						1x (PACK)	0	0		$B_7B_6B_5B_4,B_3B_2B_1B_0$
							1			$B_6B_7B_4B_5,B_2B_3B_0B_1$
							0	1		$B_5B_4B_7B_6,B_1B_0B_3B_2$
							1			$B_4B_5B_6B_7,B_0B_1B_2B_3$

Table 92. Programmed data handling (continued)

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	DWX	Destination data stream(1)
00	Byte	$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$	x	11(5)	Double-word	00 (RA, 0P)(3)(4)	0	0	0	$0000000B_0$
							1			$000000B_00$
							0	1		$00000B_000$
							1			$0000B_0000$
							0	0	1	$000B_00000$
							1			$00B_000000$
							0	1		$0B_0000000$
							1			$B_00000000$
						01 (RA, SE)(3)(4)	0	0	0	$SSSSSSSB_0$
							1			$SSSSSSB_0S$
							0	1		$SSSSSB_0SS$
							1			$SSSSB_0SSS$
							0	0	1	$SSSB_0SSSS$
							1			SSB_0SSSSS
							0	1		$SB_0SSSSSS$
							1			$B_0SSSSSSS$
						1x (PACK)	0	0	0	$B_7B_6B_5B_4B_3B_2B_1B_0$
							1			$B_6B_7B_4B_5B_2B_3B_0B_1$
							0	1		$B_5B_4B_7B_6B_1B_0B_3B_2$
							1			$B_4B_5B_6B_7B_0B_1B_2B_3$
							0	0	1	$B_3B_2B_1B_0B_7B_6B_5B_4$
							1			$B_2B_3B_0B_1B_6B_7B_4B_5$
							0	1		$B_1B_0B_3B_2B_5B_4B_7B_6$
							1			$B_0B_1B_2B_3B_4B_5B_6B_7$

Table 92. Programmed data handling (continued)

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	DWX	Destination data stream(1)
01	Half-word	$B_7B_6,B_5B_4,$ B_3B_2,B_1B_0	x	00	Byte	00 (RA, LT)(3)	x	x	x	B_6,B_4,B_2,B_0
						01 (LA, RT)(3)				B_7,B_5,B_3,B_1
						1x (UNPACK)				$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$
				01	Half-word	xx	0			$B_7B_6,B_5B_4,B_3B_2,B_1B_0$
							1			$B_6B_7,B_4B_5,B_2B_3,B_0B_1$
				10	Word	00 (RA, 0P)(3)(4)	0	0		$00B_3B_2,00B_1B_0$
							1			B_3B_200,B_1B_000
							0	1		B_2B_300,B_0B_100
							1			$00B_2B_3,00B_0B_1$
						01 (RA, SE)(3)(4)	0	0		SSB_3B_2,SSB_1B_0
							1			B_3B_2SS,B_1B_0SS
							0	1		B_2B_3SS,B_0B_1SS
							1			SSB_2B_3,SSB_0B_1
						1x (PACK)	0	0		$B_7B_6B_5B_4,B_3B_2B_1B_0$
							1			$B_6B_7B_4B_5,B_2B_3B_0B_1$
							0	1		$B_5B_4B_7B_6,B_1B_0B_3B_2$
							1			$B_4B_5B_6B_7,B_0B_1B_2B_3$

Table 92. Programmed data handling (continued)

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	DWX	Destination data stream(1)
01	Half-word	B_7B_6,B_5B_4, B_3B_2,B_1B_0	x	11(5)	Double-word	00 (RA, 0P)(3)(4)	0	0	0	$000000B_1B_0$
							1			$000000B_0B_1$
							0	1		$0000B_1B_000$
							1			$0000B_1B_000$
							0	0	1	$00B_1B_00000$
							1			$00B_0B_10000$
							0	1		$B_1B_0000000$
							1			$B_0B_1000000$
						01 (RA, SE)(3)(4)	-	0	0	$SSSSSSB_1B_0$
							1			$SSSSSSB_0B_1$
							0	1		$SSSSB_1B_0SS$
							1			$SSSSB_1B_0SS$
							0	0	1	SSB_1B_0SSSS
							1			SSB_0B_1SSSS
							0	1		$B_1B_0SSSSSS$
							1			$B_0B_1SSSSSS$
						1x (PACK)	0	0	0	$B_7B_6B_5B_4B_3B_2B_1B_0$
							1			$B_6B_7B_4B_5B_2B_3B_0B_1$
							0	1		$B_5B_4B_7B_6B_1B_0B_3B_2$
							1			$B_4B_5B_6B_7B_0B_1B_2B_3$
							0	0	1	$B_3B_2B_1B_0B_7B_6B_5B_4$
							1			$B_2B_3B_0B_1B_6B_7B_4B_5$
							0	1		$B_1B_0B_3B_2B_5B_4B_7B_6$
							1			$B_0B_1B_2B_3B_4B_5B_6B_7$

Table 92. Programmed data handling (continued)

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	DWX	Destination data stream(1)
10	Word	$B_7B_6B_5B_4$, $B_3B_2B_1B_0$	0	00	Byte	00 (RA, LT)(3)	-	x	x	B_{12},B_8,B_4,B_0
						01 (LA, RT)(3)	-			B_{15},B_{11},B_7,B_3
						1x (UNPACK)	x			B_7,B_6,B_5,B_4,B_3,B_2,B_1, B_0
				01	Half-word	00 (RA, LT)(3)	0			B_5B_4,B_1B_0
							1			B_4B_5,B_0B_1
						01 (LA, RT)(3)	0			B_7B_6,B_3B_2
							1			B_6B_7,B_2B_3
						1x (UNPACK)	0			B_7B_6,B_5B_4,B_3B_2,B_1B_0
							1			B_6B_7,B_4B_5,B_2B_3,B_0B_1
				10	Word	xx	0	0		$B_7B_6B_5B_4$,$B_3B_2B_1B_0$
							1			$B_6B_7B_4B_5$,$B_2B_3B_0B_1$
							0	1		$B_5B_4B_7B_6$,$B_1B_0B_3B_2$
							1			$B_4B_5B_6B_7$,$B_0B_1B_2B_3$

Table 92. Programmed data handling (continued)

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	DWX	Destination data stream(1)
10	Word	$B_7B_6B_5B_4$, $B_3B_2B_1B_0$	0	11(5)	Double-word	1x (PACK)	0	0	0	$B_7B_6B_5B_4B_3B_2B_1B_0$
							1			$B_6B_7B_4B_5B_2B_3B_0B_1$
							0	1		$B_5B_4B_7B_6B_1B_0B_3B_2$
							1			$B_4B_5B_6B_7B_0B_1B_2B_3$
							0	0	1	$B_3B_2B_1B_0B_7B_6B_5B_4$
							1			$B_2B_3B_0B_1B_6B_7B_4B_5$
							0	1		$B_1B_0B_3B_2B_5B_4B_7B_6$
							1			$B_0B_1B_2B_3B_4B_5B_6B_7$
						00 (RA, 0P)(3)(4)	0	0	0	$0000B_3B_2B_1B_0$
							1			$0000B_2B_3B_0B_1$
							0	1		$0000B_1B_0B_3B_2$
							1			$0000B_0B_1B_2B_3$
							0	0	1	$B_3B_2B_1B_00000$
							1			$B_1B_0B_3B_20000$
							0	1		$B_3B_2B_1B_00000$
							1			$B_0B_1B_2B_30000$
						01 (RA, SE)(3)(4)	0	0	0	$SSSSB_3B_2B_1B_0$
							1			$SSSSB_2B_3B_0B_1$
							0	1		$SSSSB_1B_0B_3B_2$
							1			$SSSSB_0B_1B_2B_3$
							0	0	1	$B_3B_2B_1B_0SSSS$
							1			$B_1B_0B_3B_2SSSS$
							0	1		$B_3B_2B_1B_0SSSS$
							1			$B_0B_1B_2B_3SSSS$

Table 92. Programmed data handling (continued)

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	DWX	Destination data stream(1)
10	Word	$B_7B_6B_5B_4$, $B_3B_2B_1B_0$	1	00	Byte	00 (RA, LT)(3)	-	x	x	B_{12},B_8,B_4,B_0
						01 (LA, RT)(3)	-			B_{15},B_{11},B_7,B_3
						1x (UNPACK)	x			$B_7,B_5,B_6,B_4,B_3,B_1,B_2,B_0$
				01	Half-word	00 (RA, LT)(3)	0			B_6B_4,B_2B_0
							1			B_4B_6,B_0B_2
						01 (LA, RT)(3)	0			B_7B_5,B_3B_1
							1			B_5B_7,B_1B_3
						1x (UNPACK)	0			$B_7B_5,B_6B_4,B_3B_1,B_2B_0$
							1			$B_5B_7,B_4B_6,B_1B_3,B_0B_2$
				10	Word	xx	0	0		$B_7B_5B_6B_4,B_3B_1B_2B_0$
							1			$B_5B_7B_4B_6,B_1B_3B_0B_2$
							0	1		$B_6B_4B_7B_5,B_2B_0B_3B_1$
							1			$B_4B_6B_5B_7,B_0B_2B_1B_3$

Table 92. Programmed data handling (continued)

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	DWX	Destination data stream(1)
10	Word	$B_7B_6B_5B_4$, $B_3B_2B_1B_0$	1	11(5)	Double-word	00 (RA, 0P)(3)(4)	0	0	0	$0000B_3B_1B_2B_0$
							1			$0000B_1B_3B_0B_2$
							0	1		$0000B_2B_0B_3B_1$
							1			$0000B_0B_2B_1B_3$
							0	0	1	$B_3B_1B_2B_00000$
							1			$B_1B_3B_0B_20000$
							0	1		$B_2B_0B_3B_10000$
							1			$B_0B_2B_1B_30000$
						01 (RA, SE)(3)(4)	0	0	0	$SSSSB_3B_1B_2B_0$
							1			$SSSSB_1B_3B_0B_2$
							0	1		$SSSSB_2B_0B_3B_1$
							1			$SSSSB_0B_2B_1B_3$
							0	0	1	$B_3B_1B_2B_0SSSS$
							1			$B_1B_3B_0B_2SSSS$
							0	1		$B_2B_0B_3B_1SSSS$
							1			$B_0B_2B_1B_3SSSS$
						1x (PACK)	0	0	0	$B_7B_5B_6B_4B_3B_1B_2B_0$
							1			$B_5B_7B_4B_6B_1B_3B_0B_2$
							0	1		$B_6B_4B_7B_5B_2B_0B_3B_1$
							1			$B_4B_6B_5B_7B_0B_2B_1B_3$
							0	0	1	$B_3B_1B_2B_0B_7B_5B_6B_4$
							1			$B_1B_3B_0B_2B_5B_7B_4B_6$
							0	1		$B_2B_0B_3B_1B_6B_4B_7B_5$
							1			$B_0B_2B_1B_3B_4B_6B_5B_7$

Table 92. Programmed data handling (continued)

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	DWX	Destination data stream(1)
11(6)	Double-word	$B_7B_6B_5B_4B_3B_2$ B_1B_0	0	00	Byte	00 (RA, LT)(3)	-	x	x	B_{24},B_{16},B_8,B_0
						01 (LA, RT)(3)	-			B_{31},B_{23},B_{15},B_7
						1x (UNPACK)	x			$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$
				01	Half-word	00 (RA, LT)(3)	0			B_9B_8,B_1B_0
							1			B_8B_9,B_0B_1
						01 (LA, RT)(3)	0			$B_{15}B_{14},B_7B_6$
							1			$B_{14}B_{15},B_6B_7$
						1x (UNPACK)	0			$B_7B_6,B_5B_4,B_3B_2,B_1B_0$
							1			$B_6B_7,B_4B_5,B_2B_3,B_0B_1$
				10	Word	00 (RA, LT)(3)	0	0		$B_{11}B_{10}B_9B_8,$ $B_3B_2B_1B_0$
							1			$B_{10}B_{11}B_8B_9,$ $B_2B_3B_0B_1$
							0	1		$B_9B_8B_{11}B_{10},$ $B_1B_0B_3B_2$
							1			$B_8B_9B_{10}B_{11},$ $B_0B_1B_2B_3$
						01 (LA, RT)(3)	0	0		$B_{15}B_{14}B_{13}B_{12},$ $B_7B_6B_5B_4$
							1			$B_{14}B_{15}B_{12}B_{13},$ $B_6B_7B_4B_5$
							0	1		$B_{13}B_{12}B_{15}B_{14},$ $B_5B_4B_7B_6$
							1			$B_{12}B_{13}B_{14}B_{15},$ $B_4B_5B_6B_7$
						1x (UNPACK)	0	0		$B_7B_6B_5B_4,B_3B_2B_1B_0$
							1			$B_6B_7B_4B_5,B_2B_3B_0B_1$
							0	1		$B_5B_4B_7B_6,B_1B_0B_3B_2$
							1			$B_4B_5B_6B_7,B_0B_1B_2B_3$

Table 92. Programmed data handling (continued)

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	DWX	Destination data stream(1)
11(6)	Double-word	$B_7B_6B_5B_4B_3B_2B_1B_0$	0	11(5)	Double-word	xx	0	0	0	$B_7B_6B_5B_4B_3B_2B_1B_0$
							1			$B_6B_7B_4B_5B_2B_3B_0B_1$
							0	1		$B_5B_4B_7B_6B_1B_0B_3B_2$
							1			$B_4B_5B_6B_7B_0B_1B_2B_3$
							0	0	1	$B_3B_2B_1B_0B_7B_6B_5B_4$
							1			$B_2B_3B_0B_1B_6B_7B_4B_5$
							0	1		$B_1B_0B_3B_2B_5B_4B_7B_6$
							1			$B_0B_1B_2B_3B_4B_5B_6B_7$
			1	00	Byte	00 (RA, LT)(3)	x	x	x	B_{24},B_{16},B_8,B_0
						01 (LA, RT)(3)				B_{31},B_{23},B_{15},B_7
						1x (UNPACK)				$B_7,B_5,B_6,B_4,B_3,B_1,B_2,B_0$
				01	Half-word	00 (RA, LT)(3)	0			$B_{10}B_8,B_2B_0$
							1			B_8B_{10},B_0B_2
						01 (LA, RT)(3)	0			$B_{15}B_{13},B_7B_5$
							1			$B_{13}B_{15},B_5B_7$
						1x (UNPACK)	0			$B_7B_5,B_6B_4,B_3B_1,B_2B_0$
							1			$B_5B_7,B_4B_6,B_1B_3,B_0B_2$

Table 92. Programmed data handling (continued)

SDW_LOG2 [1:0]	Source data	Source data stream[1]	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0][2]	DBX	DHX	DWX	Destination data stream[1]
11[6]	Double-word	$B_7B_6B_5B_4B_3B_2B_1B_0$	1	10	Word	00 (RA, LT)[3]	0	0	x	$B_{11}B_9B_{10}B_8,B_3B_1B_2B_0$
							1			$B_9B_{11}B_8B_{10},B_1B_3B_0B_2$
							0	1		$B_{10}B_8B_{11}B_9,B_2B_0B_3B_1$
							1			$B_8B_{10}B_9B_{11},B_0B_2B_1B_3$
						01 (LA, RT)[3]	0	0		$B_{15}B_{13}B_{14}B_{12}$, $B_7B_5B_6B_4$
							1			$B_{13}B_{15}B_{12}B_{14}$, $B_5B_7B_4B_6$
							0	1		$B_{14}B_{12}B_{15}B_{13}$, $B_6B_4B_7B_5$
							1			$B_{12}B_{14}B_{13}B_{15}$, $B_4B_6B_5B_7$
						1x (UNPACK)	0	0		$B_7B_5B_6B_4,B_3B_1B_2B_0$
							1			$B_5B_7B_4B_6,B_1B_3B_0B_2$
							0	1		$B_6B_4B_7B_5,B_2B_0B_3B_1$
							1			$B_4B_6B_5B_7,B_0B_2B_1B_3$
				11[5]	Double-word	xx	0	0	0	$B_7B_5B_6B_4B_3B_1B_2B_0$
							1			$B_5B_7B_4B_6B_1B_3B_0B_2$
							0	1		$B_6B_4B_7B_5B_2B_0B_3B_1$
							1			$B_4B_6B_5B_7B_0B_2B_1B_3$
							0	0	1	$B_3B_1B_2B_0B_7B_5B_6B_4$
							1			$B_1B_3B_0B_2B_5B_7B_4B_6$
							0	1		$B_2B_0B_3B_1B_6B_4B_7B_5$
							1			$B_0B_2B_1B_3B_4B_6B_5B_7$

1. Data stream is timely ordered starting from the byte with the lowest index (B_0).
2. RA= right aligned, LA = left aligned, RT = right truncated, LT = left truncated, 0P = zero bit padding up to the destination data width, SE = sign bit extended up to the destination data width.
3. RA= right aligned. LA = left aligned. RT = right truncated. LT = left truncated.
4. 0P= zero-bit padding up to the destination data width. SE = sign bit extended up to the destination data width.
5. if DDW_LOG2[1:0] = 11 and if the selected destination port (via DAP) is 64-bit capable. If DDW_LOG2[1:0] = 11 and the selected destination port is not 64-bit capable, a user setting error (USEF) is reported.
6. if SDW_LOG2[1:0] = 11 and if the selected source port (via SAP) is 64-bit capable. If SDW_LOG2[1:0] = 11 and the selected source port is not 64-bit capable, a user setting error (USEF) is reported.

18.4.11 HPDMA transfer request and arbitration

HPDMA transfer request

As defined by HPDMA_CxTR2, a programmed HPDMA data transfer is requested with one of the following:

- a software request if the control bit SWREQ = 1: This is used typically by the CPU for a data transfer from a memory-mapped address to another memory mapped address (memory-to-memory, GPIO to/from memory)
- an input hardware request coming from a peripheral if SWREQ = 0: The selection of the HPDMA hardware peripheral request is driven by REQSEL[7:0] (see *Section 18.3.5*). The selected hardware request can be one of the following:
  - an hardware request from a peripheral configured in HPDMA mode (for a transfer from/to the peripheral data register respectively to/from the memory)
  - an hardware request from a peripheral for its control register update from the memory
  - an hardware request from a peripheral for a read of its status registers transferred to the memory

Caution: The user must not assign the same input hardware peripheral HPDMA request via HPDMA_CxTR.REQSEL[7:0] to two different channels, if at a given time this request is asserted by the peripheral, and each channel is ready to execute this requested data transfer. There is no user setting error reporting.

HPDMA transfer request for arbitration

A ready FIFO-based HPDMA source single/burst transfer (from the source address to the FIFO) to be scheduled over the allocated master port (HPDMA_CxTR1.SAP) is arbitrated based on the channel priority (HPDMA_CxCR.PRIO[1:0]) versus the other simultaneous requested HPDMA transfers to the same master port.

A ready FIFO-based HPDMA destination single/burst transfer (from the FIFO to the destination address) to be scheduled over the allocated master port (HPDMA_CxTR1.DAP) is arbitrated based on the channel priority (HPDMA_CxCR.PRIO[1:0]) versus the other simultaneous requested HPDMA transfers to the same master port.

An arbitrated HPDMA requested link transfer consists of one 32-bit read from the linked-list data structure in the memory to one of the linked-list registers (HPDMA_CxTR1, HPDMA_CxTR2, HPDMA_CxBR1, HPDMA_CxSAR, HPDMA_CxDAR, HPDMA_CxTR3, HPDMA_CxBR2 or HPDMA_CxLLR). Each 32-bit read from the memory is arbitrated with the same channel priority as for data transfers, in order to be scheduled over the allocated master port (HPDMA_CxCR.LAP).

Whatever the requested data transfer is programmed with a software request for a memory-to- memory transfer (HPDMA_CxTR2.SWREQ = 1), or with a hardware request (HPDMA_CxTR2.SWREQ = 0) for a memory-to-peripheral transfer or a peripheral-to-memory transfer and whatever is the hardware request type, re-arbitration occurs after each granted single/burst transfer.

When an hardware request is programmed from a destination peripheral (HPDMA_CxTR2.SWREQ = 0 and HPDMA_CxTR2.DREQ = 1), the first memory read of a (possibly 2D/repeated) block (the first ready FIFO-based source burst request), is gated by the occurrence of the corresponding and selected hardware request. This first read request

to memory is not taken into account earlier by the arbiter (not as soon as the block transfer is enabled and executable).

HPDMA arbitration

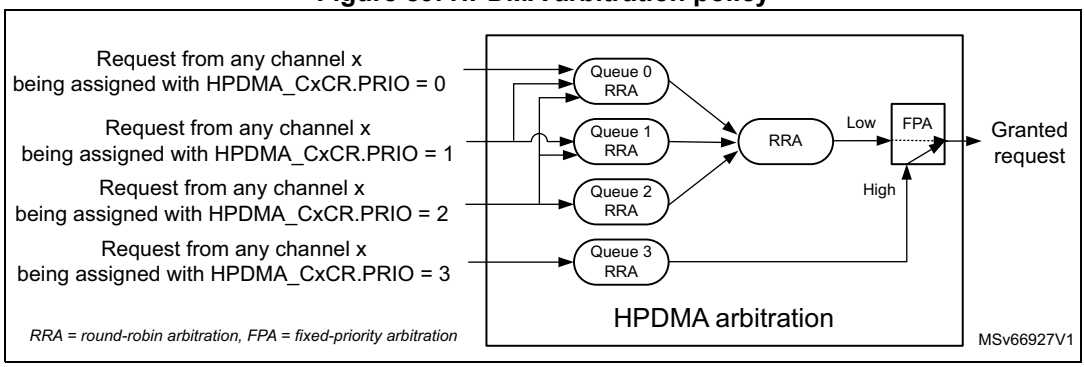
The HPDMA arbitration is directed from the 4-grade assigned channel priority (HPDMA_CxCR.PRIO[1:0]). The arbitration policy, as illustrated in *Figure 89*, is defined by:

- one high-priority traffic class (queue 3), dedicated to the assigned channels with priority 3, for time-sensitive channels

  This traffic class is granted via a fixed-priority arbitration against any other low-priority traffic class. Within this class, requested single/burst transfers are round-robin arbitrated.
- three low-priority traffic classes (queues 0, 1 or 2) for non time-sensitive channels with priority 0, 1 or 2

  Each requested single/burst transfer within this class is round-robin arbitrated, with a weight that is monotonically driven from the programmed priority:
  - Requests with priority 0 are allocated to the queue 0.
  - Requests with priority 1 are allocated and replicated to the queue 0 and queue 1.
  - Requests with priority 2 are allocated and replicated to the queue 0, queue 1, and queue 2.
  - Any queue 0, 1 or 2 equally grants any of its active input requests in a round-robin manner, provided there are simultaneous requests.
  - Additionally, there is a second stage for the low-traffic with a round-robin arbiter that fairly alternates between simultaneous selected requests from queue 0, queue 1 and queue 2.

Figure 89. HPDMA arbitration policy



HPDMA arbitration and bandwidth

With this arbitration policy, the following is guaranteed:

- equal maximum bandwidth between requests with same priority
- reserved bandwidth (noted as B_{Q3}) to the time-sensitive requests (with priority 3)
- residual weighted bandwidth between different low-priority requests (priority 0 versus priority 1 versus priority 2).

The two following examples highlight that the weighted round-robin arbitration is driven by the programmed priorities:

- **Example 1**: basic application with two non time-sensitive HPDMA requests: req0 and req1. There are the following programming possibilities:
  - If they are assigned with same priority, the allocated bandwidth by the arbiter to req0 (B_{req0}) is **equal** to the allocated bandwidth to req1(B_{req1}).

    $B_{req0} = B_{req1} = 1/2 * (1 - B_{Q3})$
  - If req0 is assigned to priority 0 and req1 to priority 1, the allocated bandwidth to req0 (B_{P0}) is **3 times less** than the allocated bandwidth to req1 (B_{P1}).

    $B_{req0} = B_{P0} = 1/2 * 1/2 * (1 - B_{Q3}) = 1/4 * (1 - B_{Q3})$

    $B_{req1} = B_{P1} = (1/2 + 1) * 1/2 * (1 - B_{Q3}) = 3/4 * (1 - B_{Q3})$
  - If req0 is assigned to priority 0 and req1 to priority 2, the allocated bandwidth to req0 (B_{P0}) is **5 times less** than the allocated bandwidth to req1 (B_{P2}).

    $B_{req0} = B_{P0} = 1/2 * 1/3 * (1 - B_{Q3}) = 1/6 * (1 - B_{Q3})$

    $B_{req1} = B_{P2} = (1/2 + 1 + 1) * 1/3 * (1 - B_{Q3}) = 5/6 * (1 - B_{Q3})$

  The above computed bandwidth calculation is based on a theoretical input request, always active for any HPDMA clock cycle. This computed bandwidth from the arbiter must be weighted by the frequency of the request given by the application, that cannot be always active and may be quite much variable from one HPDMA client (example I2C at 400 kHz) to another one (PWM at 1 kHz) than the above x3 and x5 ratios.
- **Example 2**: application where the user distributes a same non-null N number of HPDMA requests to every non time-sensitive priority 0, 1 and 2. The bandwidth calculation is then the following:
  - The allocated bandwidth to the set of requests of priority 0 (B_{P0}) is

    $B_{P0} = 1/3 * 1/3 * (1 - B_{Q3}) = 1/9 * (1 - B_{Q3})$
  - The allocated bandwidth to the set of requests of priority 1 (B_{P1}) is

    $B_{P1} = (1/3 + 1/2) * 1/3 * (1 - B_{Q3}) = 5/18 * (1 - B_{Q3})$
  - The allocated bandwidth to the set of requests of priority 2 (B_{P2}) is

    $B_{P2} = (1/3 + 1/2 + 1) * 1/3 * (1 - B_{Q3}) = 11/18 * (1 - B_{Q3})$
  - The allocated bandwidth to any request n ($B_{n)}$ among the N requests of that priority Pi (i = 0 to 2) is $B_n = 1/N * B_{Pi}$
  - The allocated bandwidth to any request n of priority i ($B_{n, Pi}$) is

    $B_{n, P0} = 1/N * 1/9 * (1 - B_{Q3})$

    $B_{n, P1} = 1/N * 5/18 * (1 - B_{Q3})$

    $B_{n, P2} = 1/N * 11/18 * (1 - B_{Q3})$

  In this example, when the master port bus bandwidth is not totally consumed by the time-sensitive queue 3, the residual bandwidth is such that 2.5 times less bandwidth is allocated to any request of priority 0 versus priority 1, and 5.5 times less bandwidth is allocated to any request of priority 0 versus priority 2.

More generally, assume that the following requests are present:

- I requests (I ≥ 0) assigned to priority 0
  If I > 0, these requests are noted from i = 0 to I-1.
- J requests (J ≥ 0) assigned to priority 1
  If J > 0, these requests are noted from j = 0 to J-1.
- K requests (K > 0) assigned to priority 2
  These requests are noted from k = 0 to K-1
- L requests (L ≥ 0) assigned to priority 3
  If L > 0, these requests are noted from l = 0 to L-1.

As B_{Q3} is the reserved bandwidth to time-sensitive requests, the bandwidth for each request L with priority 3 is:

- $B_l = B_{Q3} / L$ for L > 0 (else: $B_l = 0$)

The bandwidth for each non-time sensitive queue is:

- $B_{Q0} = 1/3 * (1 - B_{Q3})$
- $B_{Q1} = 1/3 * (1 - B_{Q3})$
- $B_{Q2} = 1/3 * (1 - B_{Q3})$

The bandwidth for the set of requests with priority 0 is:

- $B_{P0} = I / (I + J + K) * B_{Q0}$

The bandwidth for each request i with priority 0 is:

- $B_i = B_{P0} / I$ for L > 0 (else $B_{P0} = 0$)

The bandwidth for the set of requests with priority 1 and routed to queue 0 is:

- $B_{P1,Q0} = J / (I + J + K) * B_{Q0}$

The bandwidth for the set of requests with priority 1 and routed to queue 1 is:

- $B_{P1,Q1} = J / (J + K) * B_{Q1}$

The total bandwidth for the set of requests with priority 1 is:

- $B_{P1} = B_{P1,Q0} + B_{P1,Q1}$

The bandwidth for each request j with priority 1 is:

- $B_j = B_{P1} / J$ for J > 0 (else $B_j = 0$)

The bandwidth for the set of requests with priority 2 and routed to queue 0 is:

- $B_{P2,Q0} = K / (I + J + K) * B_{Q0}$

The bandwidth for the set of requests with priority 2 and routed to queue 1 is:

- $B_{P2,Q1} = K / (J + K) * B_{Q1}$

The bandwidth for the set of requests with priority 2 and routed to queue 2 is:

- $B_{P2,Q2} = B_{Q2}$

The total bandwidth for the set of requests with priority 2 is:

- $B_{P2} = B_{P2,Q0} + B_{P2,Q1} + B_{P2,Q2}$

The bandwidth for each request k with priority 2 is:

- $B_k = B_{P2} / K$ (K>0 in the general case)

Thus finally the maximum allocated residual bandwidths for any i, j, k non-time sensitive request are:

- in the general case (when there is at least one request k with a priority 2 (K > 0)):
  - $B_i = 1/I * 1/3 * I/(I + J + K) * (1 - B_{Q3})$
  - $B_j = 1/J * 1/3 *[J/(I + J + K) + J/(J + K)] * (1 - B_{Q3})$
  - $B_k = 1/K * 1/3 *[K/(I + J + K) + K/(J + K) + 1] * (1 - B_{Q3})$
- in the specific case (when there is no request k with a priority 2 (K = 0)):
  - $B_i = 1/I * 1/2 * I/(I + J) * (1 - B_{Q3})$
  - $B_j = 1/J * 1/2 *[J/(I + J) + 1] * (1 - B_{Q3})$

Consequently, the HPDMA arbiter can be used as a programmable weighted bandwidth limiter, for each queue and more generally for each request/channel. The different weights are monotonically resulting from the programmed channel priorities.

18.4.12 HPDMA triggered transfer

A programmed HPDMA transfer can be triggered by a rising/falling edge of a selected input trigger event, as defined by HPDMA_CxTR2.TRIGPOL[1:0] and HPDMA_CxTR2.TRIGSEL[6:0] (see *Section 18.3.7* for the trigger selection).

The triggered transfer, as defined by the trigger mode in HPDMA_CxTR2.TRIGM[1:0], can be at LLI data transfer level, to condition the first burst read of a block, the first burst read of a 2D/repeated block (for channel x = 12 to 15), or each programmed burst read/write. The trigger mode can also be programmed to condition the LLI link transfer (see TRIGM[1:0] in *HPDMA channel x transfer register 2 (HPDMA_CxTR2)* for more details).

Trigger hit memorization and trigger overrun flag generation

The HPDMA monitoring of a trigger for a channel x is started when the channel is enabled/loaded with a new active trigger configuration: rising or falling edge on a selected trigger (respectively TRIGPOL[1:0] = 01 or TRIGPOL[1:0] = 10).

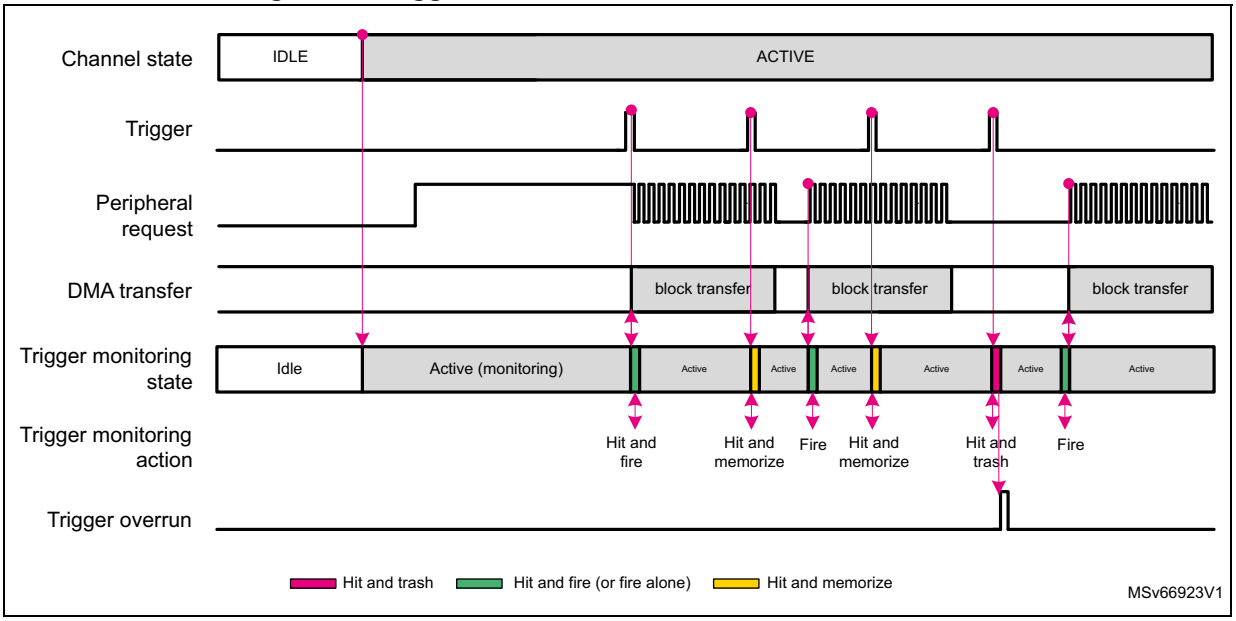
The monitoring of this trigger is kept active during the triggered and uncompleted (data or link) transfer. If a new trigger is detected, this hit is internally memorized to grant the next transfer, as long as the defined rising/falling edge and TRIGSEL[6:0] are not modified, and the channel is enabled.

Transferring a next $LLI_{n+1,}$ that updates HPDMA_CxTR2 with a new value for any of TRIGSEL[6:0] or TRIGPOL[1:0], resets the monitoring, trashing the possible memorized hit of the formerly defined LLI_n trigger.

Caution: After a first new trigger, hit_{n+1} is memorized. If another trigger hit_{n+2} is detected, and if the hit_n triggered transfer is still not completed, hit_{n+2} is lost and not memorized. A trigger overrun flag is reported (HPDMA_CxSR.TOF = 1) and an interrupt is generated if enabled (if HPDMA_CxCR.TOIE = 1). The channel is not automatically disabled by hardware due to a trigger overrun.

The figure below illustrates the trigger hit, memorization, and overrun in the configuration example with a block-level trigger mode and a rising edge trigger polarity.

Figure 90. Trigger hit, memorization and overrun waveform



Note: The user can assign the same input trigger event to different channels. This can be used to trigger different channels on a broadcast trigger event.

18.4.13 HPDMA circular buffering with linked-list programming

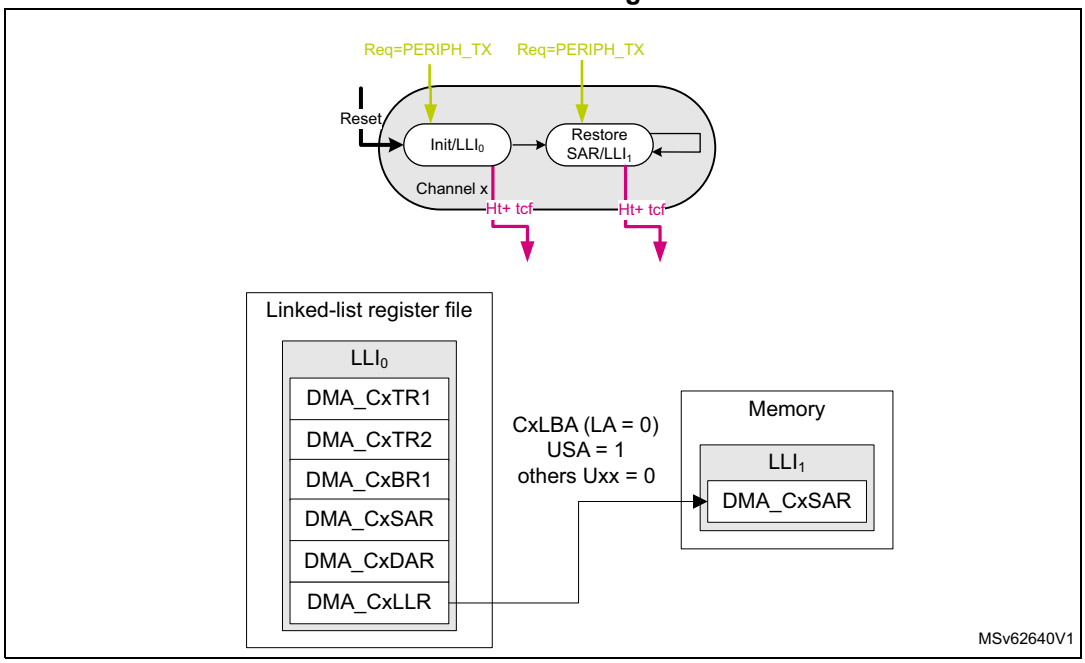
HPDMA circular buffering for memory-to-peripheral and peripheral-to-memory transfers, with a linear addressing channel

For a circular buffering, with a continuous memory-to-peripheral (or peripheral-to-memory) transfer, the software must set up a channel with half-transfer and complete-transfer event/interrupt generation (HPDMA_CxCR.HTIE = 1 and HPDMA_CxCR.TCIE = 1), in order to enable a concurrent buffer software processing.

LLI_0 is configured for the first block transfer with the linear addressing channel.
A continuously-executed LLI_1 is needed to restore the memory source (or destination) start address for the memory-to-peripheral transfer (respectively the peripheral-to-memory).
The HPDMA automatically reloads the initially programmed HPDMA_CxBR1.BNDT[15:0] when a block transfer is completed, and there is no need to restore HPDMA_CxBR1.

The figure below illustrates this programming with a linear addressing HPDMA channel and a source circular buffer.

Figure 91. HPDMA circular buffer programming: update of the memory start address with a linear addressing channel

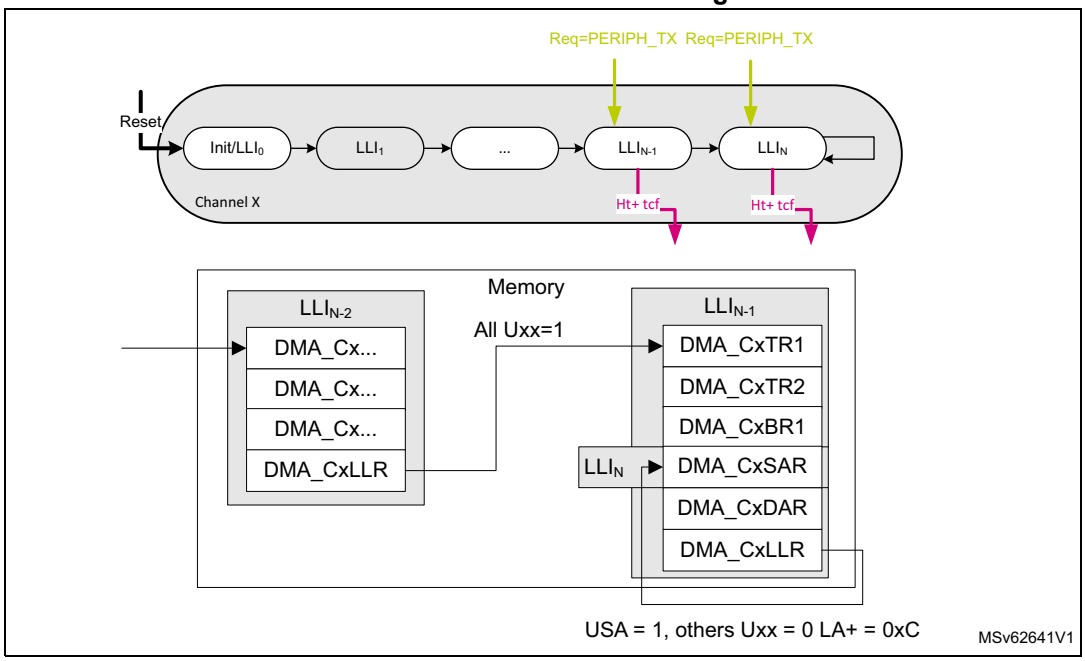


Note: With a 2D addressing channel, a single LLI can be used with HPDMA_CxBR1.BRC[10:0] = 1. The user can program a negative memory block address offset with HDMA_CxBR2 and HDMA_CxBR1, in order to jump back to the memory source or destination start address.

If the circular buffering must be executed after some other transfers over the shared HPDMA channel x, the before-last LLI_{N-1} in the memory is needed to configure the first block transfer. The last LLI_N restores the memory source (or destination) start address in memory-to-peripheral transfer (respectively in peripheral-to-memory).

The figure below illustrates this programming with a linear addressing shared HPDMA channel, and a source circular buffer.

Figure 92. Shared HPDMA channel with circular buffering: update of the memory start address with a linear addressing channel



18.4.14 HPDMA transfer in peripheral flow-control mode

A peripheral with the peripheral flow-control mode feature can decide to early terminate an HPDMA block transfer, provided that the allocated channel is implemented with this feature (see *Section 18.3.6*).

If the related HPDMA channel x is also programmed in peripheral flow-control mode (HPDMA_CxTR2.PFREQ = 1):

- The HPDMA block transfer starts as follows:
  - If HPDMA_CxBR1.BNDT[15:0] ≠ 0, the programmed value is internally taken into account by the HPDMA hardware.
  - If HPDMA_CxBR1.BNDT[15:0] = 0, the HPDMA hardware internally considers a 64-Kbyte value for the maximum source block size to be transferred.
- The HPDMA block transfer is completed as soon as the first occurrence of any of the following condition occurs:
  - when HPDMA_CxBR1.BNDT[15:0] = 0
  - when the peripheral early terminates the block. The complete transfer event is generated if programmed, depending on HPDMA_CxTR2. Then the software can read the current number of transferred bytes from the source (HPDMA_CxBR1.BNDT[15:0]), and/or read the current source or destination address of the buffer in memory (HPDMA_CxSAR[31:0] or HPDMA_CxDAR[31:0]).

In peripheral flow-control mode:

- a destination peripheral with an hardware requested transfer is not supported: memory-to-peripheral transfer is not supported.
- Data packing from a source peripheral is not supported.
- 2D/repeated block is not supported.
- HPDMA_CxBR1.BNDT[15:0] must be programmed as a multiple of the source (peripheral) burst size.

18.4.15 HPDMA secure/nonsecure channel

The HPDMA controller is compliant with the TrustZone hardware architecture at channel level, partitioning all its resources so that they exist in one of the secure and nonsecure worlds at any given time.

Any channel x is a secure or nonsecure hardware resource, as configured by HPDMA_SECCFGR.SECx.

When a channel x is configured in secure state by a secure and privileged agent, the following access control rules are applied:

- A nonsecure read access to a bitfield of this channel is forced to return 0, except for HPDMA_SECCFGR, HPDMA_PRIVCFGR, HPDMA_RCFGLOCKR, and HPDMA_CxSEMCR that are readable by a nonsecure agent.
- A nonsecure write access to a bitfield of this channel has no impact.

When a channel x is configured in secure state, a secure agent can configure separately as secure or nonsecure the HPDMA data transfer from the source (HPDMA_CxTR1.SSEC), and to the destination (HPDMA_CxTR1.DSEC).

When a channel x is configured in secure state and in linked-list mode, the loading of the next linked-list data structure from the HPDMA memory into its register file, is automatically performed with secure transfers via the HPDMA_CxCR.LAP allocated master port.

The HPDMA generates a secure bus that reflects HPDMA_SECCFGR, to keep the other peripherals informed of the secure/nonsecure state of each HPDMA channel x.

The HPDMA also generates a security illegal access pulse signal on an illegal nonsecure access to a secure HPDMA register. This signal is routed to the TrustZone interrupt controller.

When the secure software must switch a channel from a secure state to a nonsecure state, the secure software must abort the channel or wait until the secure channel is completed before switching. This is needed to dynamically re-allocate a channel to a next nonsecure transferas a nonsecure software is not allowed to do so, and must have HPDMA_CxCR.EN = 0 before the nonsecure software can reprogram HPDMA_CxCR for a next transfer. The secure software can reset not only the channel x (HPDMA_CxCR.RESET = 1), but also the full channel x register file to its reset value.

18.4.16 HPDMA privileged/unprivileged channel

Any channel x is a privileged or unprivileged hardware resource, as configured by a privileged agent via HPDMA_PRIVCFGR.PRIVx.

When a channel x is configured in a privileged state by a privileged agent, the following access control rules are applied:

- An unprivileged read access to a bitfield of this channel is forced to return 0, except for HPDMA_PRIVCFGR, HPDMA_SECCFGR, HPDMA_CxCIDCFGR and HPDMA_CxSEMCR that are readable by an unprivileged agent.
- An unprivileged write access to a bitfield of this channel has no impact.

When a channel is configured in a privileged (or unprivileged) state, the source and destination data transfers are privileged (respectively unprivileged) transfers over the AHB/AXI master port.

When a channel is configured in a privileged (or unprivileged) state and in linked-list mode, the loading of the next linked-list data structure from the HPDMA memory into its register file, is automatically performed with privileged (respectively unprivileged) transfers, via the HPDMA_CxCR.LAP allocated master port.

The HPDMA generates a privileged bus that reflects HPDMA_PRIVCFGR, to keep the other peripherals informed of the privileged/unprivileged state of each HPDMA channel x.

Additionally, the HPDMA generates a privileged illegal access pulse signal on an illegal unprivileged access to a privileged HPDMA register. This signal can be used or not, depending on the product (see the system security section for more details).

When the privileged software must switch a channel from a privileged state to an unprivileged state, the privileged software must abort the channel, or wait until the privileged channel is completed before switching. This is needed to dynamically re-allocate a channel to a next unprivileged transfer, as an unprivileged software is not allowed to do so, and must have HPDMA_CxCR.EN = 0 before the unprivileged software can reprogram the HPDMA_CxCR for a next transfer. The privileged software can reset not only the channel x (HPDMA_CxCR.RESET = 1), but also the full channel x register file to its reset value.

HPDMA compartmented channel

The HPDMA controller performs bus transfers over its master port, under the control of a resource isolation domain identification named compartment identification (CID), at a channel level. For more details about resource isolation and CID, refer to the system security section.

18.4.17 HPDMA error management

The HPDMA can manage and report to the user a transfer error, as follows, depending on the root cause.

Data transfer error

On a bus access (as an AHB/AXI single or a burst) to the source or the destination

- The source or destination target reports an AHB/AXI error.
- The programmed channel transfer is stopped (HPDMA_CxCR.EN cleared by the HPDMA hardware). The channel status register reports an idle state (HPDMA_CxSR.IDLEF = 1) and the data error (HPDMA_CxSR.DTEF = 1).

- After an HPDMA data transfer error, the user must perform a debug session, taking care of the product-defined memory mapping of the source and destination, including the protection attributes.
- After an HPDMA data transfer error, the user must issue a channel reset (set HPDMA_CxCR.RESET) to reset the hardware HPDMA channel data path and the FIFO content, before the user enables again the same channel for a next transfer.

Link transfer error

On a tentative update of a HPDMA channel register from the programmed LLI in the memory:

- The linked-list memory reports an AHB/AXI error.
- The programmed channel transfer is stopped (HPDMA_CxCR.EN cleared by the HPDMA hardware), the channel status register reports an idle state HPDMA_CxSR.IDLEF = 1), and the link error (HPDMA_CxSR.ULEF = 1).
- After an HPDMA link error, the user must perform a debug session, taking care of the product-defined memory mapping of the linked-list data structure (HPDMA_CxLBAR and HPDMA_CxLLR), including the protection attributes.
- After an HPDMA link error, the user must explicitly write the linked-list register file (HPDMA_CxTR1, HPDMA_CxTR2, HPDMA_CxBR1, HPDMA_CxSAR, HPDMA_CxDAR, and HPDMA_CxLLR, plus HPDMA_CxTR3 and HPDMA_CxBR2 when present), before the user enables again the same channel for a next transfer.

User setting error

On a tentative execution of an HPDMA transfer with an unauthorized user setting:

- The programmed channel transfer is disabled (HPDMA_CxCR.EN forced and cleared by the HPDMA hardware), preventing the next unauthorized programmed data transfer from being executed. The channel status register reports an idle state (HPDMA_CxSR.IDLEF = 1), and a user setting error (HPDMA_CxSR.USEF = 1).
- After an HPDMA user setting error, the user must perform a debug session, taking care of the HPDMA channel programming. A user setting error can be caused by one of the following:
  - a programmed null source block size without a programmed update of this value from the next LLI_1 (HPDMA_CxBR1.BNDT[15:0] = 0 and HPDMA_CxLLR.UB1 = 0)
  - a programmed non-null source block size being not a multiple of the programmed data width of a source burst transfer (HPDMA_CxBR1.BNDT versus HPDMA_CxTR1.SDW_LOG2)
  - when in packing/unpacking mode (if PAM[1] = 1), a programmed non-null source block size being not a multiple of the programmed data width of a destination burst transfer (HPDMA_CxBR1.BNDT versus HPDMA_CxTR1.DDW_LOG2)
  - a programmed unaligned source start address being not a multiple of the programmed data width of a source burst transfer (HPDMA_CxSAR versus HPDMA_CxTR1.SDW_LOG2)
  - for a channel with 2D addressing, a programmed unaligned source address offset being not a multiple of the programmed data width of a source burst transfer (HPDMA_CxTR3.SAO versus HPDMA_CxTR1.SDW_LOG2)

- for a channel with 2D addressing, a programmed unaligned block repeated source address offset being not a multiple of the programmed data width of a source burst transfer (HPDMA_CxBR2.BRSAO versus HPDMA_CxTR1.SDW_LOG2)
- a programmed unaligned destination start address, being not a multiple of the programmed data width of a destination burst transfer (HPDMA_CxDAR versus HPDMA_CxTR1.DDW_LOG2)
- for a channel with 2D addressing, a programmed unaligned destination address offset being not a multiple of the programmed data width of a destination burst transfer (HPDMA_CxTR3.DAO versus HPDMA_CxTR1.DDW_LOG2)
- for a channel with 2D addressing, a programmed unaligned block repeated destination address offset being not a multiple of the programmed data width of a destination burst transfer (HPDMA_CxBR2.BRDAO versus HPDMA_CxTR1.DDW_LOG2)
- a programmed source burst data width being superior than the data width of the allocated AXI/AHB source port (HPDMA_CxTR1.SDW_LOG2 and HPDMA_CxTR1.SAP vs the implemented AXI/AHB port data width).
- a programmed destination burst data width being superior than the data width of the allocated AXI/AHB destination port (HPDMA_CxTR1.DDW_LOG2 and HPDMA_CxTR1.DAP vs the implemented AXI/AHB port data width).
- a programmed linked-list item LLI_{n+1} with a null data transfer (HPDMA_CxLLR.UB1 = 1 and HPDMA_CxBR1. BNDT[15:0] = 0)

18.5 HPDMA in debug mode

When the device enters debug mode (core halted), any channel x can be individually either continued (default) or suspended, depending on the programmable control bit in the DBGMCU module.

Note: In debug mode, HPDMA_CxSR.SUSPF is not altered by a suspension from the programmable control bit in the DBGMCU module. In this case, HPDMA_CxSR.IDLEF can be checked to know the completion status of the channel suspension.

18.6 HPDMA in low-power modes

Table 93. Effect of low-power modes on HPDMA

Mode	Description
Sleep	No effect. HPDMA interrupts cause the device to exit Sleep mode.
Stop(1)	The content of HPDMA registers is kept when entering Stop mode.
Standby	The HPDMA is powered down, and must be reinitialized after exiting Standby mode.

1. Refer to *Section 18.3.3* to know which Stop mode is supported.

18.7 HPDMA interrupts

There is one HPDMA interrupt line for each channel, and separately for each CPU (if several ones in the devices).

Table 94. HPDMA interrupt requests

Interrupt acronym	Interrupt event	Interrupt enable	Event flag	Event clear method
HPDMA_CHx	Transfer complete	HPDMA_CxCR.TCIE	HPDMA_CxSR.TCF	Writes 1 to HPDMA_CxFCR.TCF
	Half transfer	HPDMA_CxCR.HTIE	HPDMA_CxSR.HTF	Writes 1 to HPDMA_CxFCR.HTF
	Data transfer error	HPDMA_CxCR.DTEIE	HPDMA_CxSR.DTEF	Writes 1 to HPDMA_CxFCR.DTEF
	Update link error	HPDMA_CxCR.ULEIE	HPDMA_CxSR.ULEF	Writes 1 to HPDMA_CxFCR.ULEF
	User setting error	HPDMA_CxCR.USEIE	HPDMA_CxSR.USEF	Writes 1 to HPDMA_CxFCR.USEF
	Suspended	HPDMA_CxCR.SUSPIE	HPDMA_CxSR.SUSPF	Writes 1 to HPDMA_CxFCR.SUSPF
	Trigger overrun	HPDMA_CxCR.TOFIE	HPDMA_CxSR.TOF	Writes 1 to HPDMA_CxFCR.TOF

An HPDMA channel x event can be:

- a transfer complete
- a half-transfer complete
- a transfer error, due to either:
  - a data transfer error
  - an update link error
  - a user setting error completed suspension
- a trigger overrun

Note: When a channel x transfer complete event occurs, the output signal hpdma_chx_tc is generated as a high pulse of one clock cycle.

An interrupt is generated following any xx event, provided that both:

- the corresponding interrupt event xx is enabled (HPDMA_CxCR.xxIE = 1)
- the corresponding event flag is cleared (HPDMA_CxSR.xxF = 0). This means that, after a previous same xx event occurrence, a software agent must have written 1 into the corresponding xx flag clear control bit (write 1 into HPDMA_CxFCR.xxF).

TCF (transfer complete) and HTF (half transfer) events generation is controlled by HPDMA_CxTR2.TCEM[1:0] as follows:

- A transfer-complete event is a block transfer complete, a 2D/repeated block transfer complete, or a LLI transfer complete including the upload of the next LLI if any, or the full linked-list completion, depending on the transfer complete event mode HPDMA_CxTR2.TCEM[1:0].
- A half-transfer event is a half-block transfer, or a half 2D/repeated block transfer, depending on the transfer complete event mode HPDMA_CxTR2.TCEM[1:0].

A half-block transfer occurs when half of the source block size bytes (rounded-up integer of HPDMA_CxBR1.BNDT[15:0] / 2) is transferred to the destination.

A half 2D/repeated block transfer occurs when half of the repeated blocks (rounded-up integer of (HPDMA_CxBR1.BRC[10:0] + 1) / 2) is transferred to the destination.

See *HPDMA channel x transfer register 2 (HPDMA_CxTR2)* for more details.

Note: The interrupt mode must be used (not the polling mode) to be notified on a half transfer when the write data transaction has been completed over the AXI destination allocated port (written at the destination memory-mapped address), and not just before when has been issued, at HPDMA level, this AXI burst transaction.

A transfer error rises in one of the following situations:

- during a single/burst data transfer from the source or to the destination (DTEF)
- during an update of an HPDMA channel register from the programmed LLI in memory (ULEF)
- during a tentative execution of an HPDMA channel with an unauthorized setting (USEF)

  The user must perform a debug session to correct the HPDMA channel programming versus the USEF root causes list (see *Section 18.4.17*).

A trigger overrun is described in *Trigger hit memorization and trigger overrun flag generation*.

18.8 HPDMA registers

The HPDMA registers must be accessed with an aligned 32-bit word data access.

18.8.1 HPDMA secure configuration register (HPDMA_SECCFGR)

Address offset: 0x000

Reset value: 0x0000 0000

A write access to this register must be secure and privileged. A read access may be secure or nonsecure, privileged or unprivileged and with any CID.

A write access is ignored at bit level if the corresponding channel x is locked (HPDMA_RCFGLOCKR.LOCKx = 1).

This register can mix information from different CIDs. If a channel x is configured as CID filtered (HPDMA_CxCIDCFGR.CFEN = 1), the SECx bit can be written only by an authorized CID (and secure and privileged) agent. If the debug domain feature is activated, an access to this register is granted to the debug domain CID, regardless of any CID filtering.

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1.

This register must be programmed at a bit level, at the initialization/closure of an HPDMA channel (when HPDMA_CxCR.EN = 0), to securely allocate individually any channel x to the secure or nonsecure world.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **SECx**: Secure state of channel x (x = 15 to 0)
0: Nonsecure
1: Secure

18.8.2 HPDMA privileged configuration register (HPDMA_PRIVCFGR)

Address offset: 0x004

Reset value: 0x0000 0000

A write access to this register must be privileged. A read access can be privileged or unprivileged, secure or nonsecure, and with any CID.

This register can mix secure and nonsecure information. If a channel x is configured as secure (HPDMA_SECCFGR.SECx = 1), the PRIVx bit can be written only by a secure (and privileged) agent.

This register can mix information from different CIDs. If a channel x is configured as CID filtered (HPDMA_CxCIDCFGR.CFEN = 1), the PRIVx bit can be written only by an authorized CID (and privileged) agent. If the debug domain feature is activated, an access to this register is granted to the debug domain CID, regardless of any CID filtering.

A write access is ignored at bit level if the corresponding channel x is locked (HPDMA_RCFGLOCKR.LOCKx = 1).

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1.

This register must be programmed at a bit level, at the initialization/closure of a HPDMA channel (HPDMA_CxCR.EN = 0), to individually allocate any channel x to the privileged or unprivileged world.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PRIVx**: Privileged state of channel x (x = 15 to 0)
0: Unprivileged
1: Privileged

18.8.3 HPDMA configuration lock register (HPDMA_RCFGLOCKR)

Address offset: 0x008

Reset value: 0x0000 0000

This register can be written by a software agent with secure privileged and trusted domain CID attributes in order to individually lock, for example at boot time, the secure privileged and CID attributes of any HPDMA channel/resource (to lock the setting of HPDMA_SECCFGR, HPDMA_PRIVCFGR, and HPDMA_CxCIDCFGR for any channel x at, for example at boot time).

A read access may be privileged or unprivileged, secure or nonsecure, and with any CID.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOCK15	LOCK14	LOCK13	LOCK12	LOCK11	LOCK10	LOCK9	LOCK8	LOCK7	LOCK6	LOCK5	LOCK4	LOCK3	LOCK2	LOCK1	LOCK0
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **LOCKx**: Lock of the configuration of HPDMA_SECCFGR.SECx, HPDMA_PRIVCFGR.PRIVx and HPDMA_CxCIDCFGR until a global HPDMA reset (x = 15 to 0)
This bit is cleared after reset and, once set, it cannot be reset until a global HPDMA reset.
0: Secure privilege and CID configuration of the channel x is writable.
1: Secure privilege and CID configuration of the channel x is not writable.

18.8.4 HPDMA nonsecure masked interrupt status register (HPDMA_MISR)

Address offset: 0x00C

Reset value: 0x0000 0000

This register is a read register.

This is a nonsecure register, containing the masked interrupt status bit MISx for each nonsecure channel x (channel x configured with HPDMA_SECCFGR.SECx = 0). It is a logical OR of all the flags of HPDMA_CxSR, each source flag being enabled by the corresponding interrupt enable bit of HPDMA_CxCR.

Every bit is deasserted by hardware when writing 1 to the corresponding flag clear bit in HPDMA_CxFCR.

If a channel x is in secure state (HPDMA_SECCFGR.SECx = 1), a read access to the masked interrupt status bit MISx of this channel x returns zero.

This register can mix privileged and unprivileged information, depending on the privileged state of each channel HPDMA_PRIVCFGR.PRIVx. A privileged software can read the full nonsecure interrupt status. An unprivileged software is restricted to read the status of unprivileged (and nonsecure) channels, other privileged bitfields returning zero.

This register can mix information from different CIDs. If a channel x is configured as CID filtered (HPDMA_CxCIDCFGR.CFEN = 1), the MISx bit can be read only by an authorized CID agent, else it is returned zero. If the debug domain feature is activated, an access to this register is granted to the debug domain CID, regardless of any CID filtering.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MIS15	MIS14	MIS13	MIS12	MIS11	MIS10	MIS9	MIS8	MIS7	MIS6	MIS5	MIS4	MIS3	MIS2	MIS1	MIS0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MISx**: Masked interrupt status of channel x (x = 15 to 0)
0: No interrupt occurred on channel x.
1: An interrupt occurred on channel x.

18.8.5 HPDMA secure masked interrupt status register (HPDMA_SMISR)

Address offset: 0x010

Reset value: 0x0000 0000

This is a secure read register, containing the masked interrupt status bit MISx for each secure channel x (HPDMA_SECCFGR.SECx = 1). It is a logical OR of all the HPDMA_CxSR flags, each source flag being enabled by the corresponding HPDMA_CxCR interrupt enable bit.

Every bit is deasserted by hardware when securely writing 1 to the corresponding HPDMA_CxFCR flag clear bit.

This register does not contain any information about a nonsecure channel.

This register can mix privileged and unprivileged information, depending on the privileged state of each channel HPDMA_PRIVCFGR.PRIVx. A privileged software can read the full secure interrupt status. An unprivileged software is restricted to read the status of unprivileged and secure channels, other privileged bitfields returning zero.

This register can mix information from different CIDs. If a channel x is configured as CID filtered (HPDMA_CxCIDCFGR.CFEN = 1), the MISx bit can be read only by an authorized

CID agent, else it is returned zero. If the debug domain feature is activated, an access to this register is granted to the debug domain CID, regardless of any CID filtering.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MIS15	MIS14	MIS13	MIS12	MIS11	MIS10	MIS9	MIS8	MIS7	MIS6	MIS5	MIS4	MIS3	MIS2	MIS1	MIS0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MISx**: Masked interrupt status of the secure channel x (x = 15 to 0)

0: No interrupt occurred on the secure channel x.

1: An interrupt occurred on the secure channel x.

18.8.6 HPDMA channel x linked-list base address register (HPDMA_CxLBAR)

Address offset: 0x050 + 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This register must be written by a privileged software. It is either privileged readable or not, depending on the privileged state of the channel x HPDMA_PRIVCFGR.PRIVx.

This register is either secure or nonsecure depending on the secure state of the channel x (HPDMA_SECCFGR.SECx).

This register is CID-filtered depending on the CID configuration of the channel x (HPDMA_CxCIDCFGR.CFEN = 1). If the debug domain feature is activated, an access to this register is granted to the debug domain CID, regardless of any CID filtering.

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1.

This channel-based register is the linked-list base address of the memory region, for a given channel x, from which the LLIs describing the programmed sequence of the HPDMA transfers, are conditionally and automatically updated.

This 64-Kbyte aligned channel x linked-list base address is offset by the 16-bit HPDMA_CxLLR register that defines the word-aligned address offset for each LLI.

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

Bits 31:16 **LBA[31:16]**: Linked-list base address of HPDMA channel x

Bits 15:0 Reserved, must be kept at reset value.

18.8.7 HPDMA channel x CID register (HPDMA_CxCIDCFGR)

Address offset: 0x054+ 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This channel-based register can be written by the secure, privileged, and trusted domain CID, in order to individually configure the CID allocation of any HPDMA channel x. If the debug domain feature is activated, a read/write access to this register is granted to the debug domain CID, regardless of any CID filtering.

This register must be written when the resource configuration of the channel x is unlocked (HPDMA_RCFGLOCKR.LOCKx = 0) and when the channel is disabled (HPDMA_CxCR.EN = 0).

A read access may be privileged or unprivileged, secure or nonsecure, and with any CID.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEM_WLIST_CID6	SEM_WLIST_CID5	SEM_WLIST_CID4	SEM_WLIST_CID3	SEM_WLIST_CID2	SEM_WLIST_CID1	SEM_WLIST_CID0
									rw	rw	rw	rw	rw	rw	rw

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

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **SEM_WLIST_CID6**: White-listed CID6 in the CID allocation pool (for when the channel x in semaphore mode)
0: CID6 black-listed in the semaphore-based CID allocation pool
1: CID6 white-listed in the semaphore-based CID allocation pool

Bit 21 **SEM_WLIST_CID5**: White-listed CID5 in the CID allocation pool (for when the channel x in semaphore mode)
0: CID5 black-listed in the semaphore-based CID allocation pool
1: CID5 white-listed in the semaphore-based CID allocation pool

Bit 20 **SEM_WLIST_CID4**: White-listed CID4 in the CID allocation pool (for when the channel x in semaphore mode)
0: CID4 black-listed in the semaphore-based CID allocation pool
1: CID4 white-listed in the semaphore-based CID allocation pool

Bit 19 **SEM_WLIST_CID3**: White-listed CID3 in the CID allocation pool (for when the channel x in semaphore mode)
0: CID3 black-listed in the semaphore-based CID allocation pool
1: CID3 white-listed in the semaphore-based CID allocation pool

Bit 18 **SEM_WLIST_CID2**: White-listed CID2 in the CID allocation pool (for when the channel x in semaphore mode)
0: CID2 black-listed in the semaphore-based CID allocation pool
1: CID2 white-listed in the semaphore-based CID allocation pool

Bit 17 **SEM_WLIST_CID1**: White-listed CID1 in the CID allocation pool (for when the channel x in semaphore mode)
0: CID1 black-listed in the semaphore-based CID allocation pool
1: CID1 white-listed in the semaphore-based CID allocation pool

Bit 16 **SEM_WLIST_CID0**: White-listed CID0 in the CID allocation pool (for when the channel x in semaphore mode)
0: CID0 black-listed in the semaphore-based CID allocation pool
1: CID0 white-listed in the semaphore-based CID allocation pool

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:4 **SCID[2:0]**: Allocation of a static/single CID to the channel x (for when the channel x CID configuration is not in semaphore mode)
000: CID0 allocated to the channel x
001: CID1 allocated to the channel x
010: CID2 allocated to the channel x
011: CID3 allocated to the channel x
100: CID4 allocated to the channel x
101: CID5 allocated to the channel x
110: CID6 allocated to the channel x
111: Reserved (write ignored)

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **SEM_EN**: Semaphore mode enable (for the CID allocation policy to the channel x)
0: Semaphore mode disabled. CID allocation policy to the channel x is defined by SCID[1:0].
1: Semaphore mode enabled. CID allocation policy to the channel x is defined by the white-listed allocation pool SEM_WLIST_CIDx and HPDMA_CxSEMCR.SEM_MUTEX.
Note: If SEM_EN = 1 and if a trusted domain or debug domain CID agent clears this bit, then the HPDMA hardware automatically clears the HPDMA_CxSEMCR.SEM_MUTEX.

Bit 0 **CFEN**: CID filtering enable of the channel x
0: CID filtering disabled for when accessing a channel x register/bitfield
1: CID filtering enabled for when accessing a channel x register/bitfield
Note: If CFEN = 1 and if a trusted domain or debug domain CID agent clears this bit, then the HPDMA hardware automatically clears the HPDMA_CxSEMCR.SEM_MUTEX.

18.8.8 HPDMA channel x semaphore control register (HPDMA_CxSEMCR)

Address offset: 0x058+ 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

When the secure, privileged and trusted domain has set up the channel x in semaphore mode (HPDMA_CxCIDCFGR.SEM_EN = 1), this register is used during run-time by an

authorized white-listed CID agent, in order to take and after possibly release the control of the channel x.

A read access may be privileged or unprivileged, secure or nonsecure, and with any CID.

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

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 **SEM_CCID[2:0]**: Current CID allocated to the channel x (in semaphore mode)

This read-only bitfield is internally updated when a white-listed CID took the control of the channel x, in semaphore mode. If SEM_MUTEX = 0, this same CID also released it.

000: CID0 is the last white-listed CID that took the control of the channel x.

001: CID1 is the last white-listed CID that took the control of the channel x.

010: CID2 is the last white-listed CID that took the control of the channel x.

011: CID3 is the last white-listed CID that took the control of the channel x.

100: CID4 is the last white-listed CID that took the control of the channel x.

101: CID5 is the last white-listed CID that took the control of the channel x.

110: CID6 is the last white-listed CID that took the control of the channel x.

111: Reserved

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **SEM_MUTEX**: Mutual exclusion semaphore for the CID allocation of the channel x (in semaphore mode)

If the channel x is in secure state (HPDMA_SECCFGR.SECx = 1), this bit can only be written by a secure agent. If the channel x is in privileged state (HPDMA_PRIVCFGR.PRIVx = 1), this bit can only be written by a privileged agent.

If the channel x is CID-filtered (HPDMA_CxCIDCFGR.CFEN = 1) and in semaphore mode (HPDMA_CIDCFGR.SEM_EN = 1), this bit can only be written either by an authorized (white-listed) CID agent to take the control, or by the same (white-listed) CID agent to release the control.

Condition: write

0: Release the control of the channel x (in semaphore mode) to any white-listed CID.

1: Take the control of the channel x (in semaphore mode), from one of the white-listed CID pool.

Condition: read

0: Channel x CID-free (not currently under the control of any white-listed CID)

1: Channel x CID-allocated (currently taken and under the control of one white-listed CID)

Note: This bit must be written when HPDMA_CxCR.EN = 0. This bit is read-only when HPDMA_CxCR.EN = 1.

When SEM_EN or CFEN bit is cleared in HPDMA_CxSEMCR, the HPDMA hardware automatically clears this SEM_MUTEX bit.

18.8.9 HPDMA channel x flag clear register (HPDMA_CxFCR)

Address offset: 0x05C+ 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This is a write register, secure or nonsecure depending on the secure state of channel x (HPDMA_SECCFGR.SECx) and privileged or unprivileged, depending on the privileged state of the channel x (HPDMA_PRIVCFGR.PRIVx).

This register is CID-filtered depending on the CID configuration of the channel x (HPDMA_CxCIDCFGR.CFEN = 1). If the debug domain feature is activated, an access to this register is granted to the debug domain CID, regardless of any CID filtering.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	w	w	w	w	w	w	w								

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **TOF**: Trigger overrun flag clear
0: No effect
1: Corresponding TOF flag cleared

Bit 13 **SUSPF**: Completed suspension flag clear
0: No effect
1: Corresponding SUSPF flag cleared

Bit 12 **USEF**: User setting error flag clear
0: No effect
1: Corresponding USEF flag cleared

Bit 11 **ULEF**: Update link transfer error flag clear
0: No effect
1: Corresponding ULEF flag cleared

Bit 10 **DTEF**: Data transfer error flag clear
0: No effect
1: Corresponding DTEF flag cleared

Bit 9 **HTF**: Half-transfer flag clear
0: No effect
1: Corresponding HTF flag cleared

Bit 8 **TCF**: Transfer complete flag clear
0: No effect
1: Corresponding TCF flag cleared

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

18.8.10 HPDMA channel x status register (HPDMA_CxSR)

Address offset: 0x060 + 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0001

This is a read register, reporting the channel status.

This register is secure or nonsecure, depending on the secure state of channel x (HPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of the channel (HPDMA_PRIVCFGR.PRIVx).

This register is CID-filtered depending on the CID configuration of the channel x (HPDMA_CxCIDCFGR.CFEN = 1). If the debug domain feature is activated, an access to this register is granted to the debug domain CID, regardless of any CID filtering.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	FIFOL[8:0]								
							r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDLEF
	r	r	r	r	r	r	r								r

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:16 **FIFOL[8:0]**: Monitored FIFO level

Number of available write beats in the FIFO, in units of the programmed destination data width (see *Section 18.8.12: HPDMA channel x transfer register 1 (HPDMA_CxTR1)* DDW_LOG2[1:0], in units of bytes, half-words, words or double-words).

Note: After having suspended an active transfer, the user may need to read FIFOL[8:0], additionally to HPDMA_CxBR1.BDNT[15:0] and HPDMA_CxBR1.BRC[10:0], to know how many data have been transferred to the destination. Before reading, the user may wait for the transfer to be suspended (HPDMA_CxSR.SUSPF = 1).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TOF**: Trigger overrun flag

0: No trigger overrun event

1: A trigger overrun event occurred.

Bit 13 **SUSPF**: Completed suspension flag

0: No completed suspension event

1: A completed suspension event occurred.

Bit 12 **USEF**: User setting error flag

0: No user setting error event

1: A user setting error event occurred.

Bit 11 **ULEF**: Update link transfer error flag

0: No update link transfer error event

1: A master bus error event occurred while updating a linked-list register from memory.

Bit 10 **DTEF**: Data transfer error flag

0: No data transfer error event

1: A master bus error event occurred on a data transfer.

Bit 9 **HTF**: Half-transfer flag

0: No half-transfer event

1: A half-transfer event occurred.

A half-transfer event is either a half-block transfer or a half 2D/repeated block transfer, depending on the transfer complete event mode (HPDMA_CxTR2.TCEM[1:0]).

A half-block transfer occurs when half of the bytes of the source block size (rounded up integer of HPDMA_CxBR1.BNDT[15:0]/2) has been transferred to the destination.

A half 2D/repeated block transfer occurs when half of the repeated blocks (rounded up integer of (HPDMA_CxBR1.BRC[10:0]+1)/2)) has been transferred to the destination.

Bit 8 **TCF**: Transfer complete flag

0: No transfer complete event

1: A transfer complete event occurred.

A transfer complete event is either a block transfer complete, a 2D/repeated block transfer complete, a LLI transfer complete including the upload of the next LLI if any, or the full linked-list completion, depending on the transfer complete event mode (HPDMA_CxTR2.TCEM[1:0]).

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **IDLEF**: Idle flag

0: Channel not in idle state

1: Channel in idle state

This idle flag is deasserted by hardware when the channel is enabled (HPDMA_CxCR.EN = 1) with a valid channel configuration (no USEF to be immediately reported).

This idle flag is asserted after hard reset or by hardware when the channel is back in idle state (in suspended or disabled state).

18.8.11 HPDMA channel x control register (HPDMA_CxCR)

Address offset: 0x064 + 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (HPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of the channel x (HPDMA_PRIVCFGR.PRIVx).

This register is CID-filtered depending on the CID configuration of the channel x (HPDMA_CxCIDCFGR.CFEN = 1). If the debug domain feature is activated, an access to this register is granted to the debug domain CID, regardless of any CID filtering.

This register is used to control a channel (activate, suspend, abort or disable it).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIO[1:0]		Res.	Res.	Res.	Res.	LAP	LSM
								rw	rw					rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TOIE	SUSPIE	USEIE	ULEIE	DTEIE	HTIE	TCIE	Res.	Res.	Res.	Res.	Res.	SUSP	RESET	EN
	rw	rw	rw	rw	rw	rw	rw						rw	w	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:22 **PRIO[1:0]:** Priority level of the channel x HPDMA transfer versus others
00: Low priority, low weight
01: Low priority, mid weight
10: Low priority, high weight
11: High priority
Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bits 21:18 Reserved, must be kept at reset value.

Bit 17 **LAP**: Linked-list allocated port
This bit is used to allocate the master port for the update of the HPDMA linked-list registers from the memory.
0: Port 0 (AXI) allocated
1: Port 1 (AHB) allocated
Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bit 16 **LSM**: Link step mode
0: Channel executed for the full linked-list and completed at the end of the last LLI (HPDMA_CxLLR = 0). The 16 low-significant bits of the link address are null (LA[15:0] = 0) and all the update bits are null (UT1 = UB1 = UT2 = USA = UDA = ULL = 0 and UT3 = UB2 = 0 if present). Then HPDMA_CxBR1.BNDT[15:0] = 0 and HPDMA_CxBR1.BRC[10:0] = 0 if present.
1: Channel executed once for the current LLI
First the (possible 1D/repeated) block transfer is executed as defined by the current internal register file until HPDMA_CxBR1.BNDT[15:0] = 0 and HPDMA_CxBR1.BRC[10:0] = 0, if present. Secondly the next linked-list data structure is conditionally uploaded from memory as defined by HPDMA_CxLLR. Then channel execution is completed.
Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TOIE**: Trigger overrun interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 13 **SUSPIE**: cCmpleted suspension interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 12 **USEIE**: User setting error interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 11 **ULEIE**: Update link transfer error interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 10 **DTEIE**: Data transfer error interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 9 **HTIE**: Half-transfer complete interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 8 **TCIE**: Transfer complete interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **SUSP:** Suspend

Writing 1 to RESET in this register causes the hardware to deassert this SUSP bit, whatever is written into this SUSP. Else:

Software must write 1 in order to suspend an active channel (a channel with an ongoing HPDMA transfer over its master ports).

The software must write 0 in order to resume a suspended channel, following the programming sequence detailed in *Figure 74*.

0: Write: resume channel, read: channel not suspended

1: Write: suspend channel, read: channel suspended

Bit 1 **RESET:** Reset

This bit is write only. Writing 0 has no impact. Writing 1 implies the reset of the following: the FIFO, the channel internal state, SUSP and EN bits (whatever is written receptively in bit 2 and bit 0).

The reset is effective when the channel is in steady state, meaning one of the following:

- active channel in suspended state (HPDMA_CxSR.SUSPF = 1 and HPDMA_CxSR.IDLEF = HPDMA_CxCR.EN = 1)
- channel in disabled state (HPDMA_CxSR.IDLEF = 1 and HPDMA_CxCR.EN = 0).

After writing a RESET, to continue using this channel, the user must explicitly reconfigure the channel including the hardware-modified configuration registers (HPDMA_CxBR1, HPDMA_CxSAR and HPDMA_CxDAR) before enabling again the channel (see the programming sequence in *Figure 75*).

0: No channel reset

1: Channel reset

Bit 0 **EN:** Enable

Writing 1 to RESET in this register causes the hardware to deassert this EN bit, whatever is written into this bit 0. Else:

This bit is deasserted by hardware when there is a transfer error (master bus error or user setting error) or when there is a channel transfer complete (channel ready to be configured, for example if LSM = 1 at the end of a single execution of the LLI).

Else, this bit can be asserted by software.

Writing 0 into this EN bit is ignored.

0: Write: ignored, read: channel disabled

1: Write: enable channel, read: channel enabled

18.8.12 HPDMA channel x transfer register 1 (HPDMA_CxTR1)

Address offset: 0x090 + 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (HPDMA_SECCFGR.SECx) except for secure DSEC and SSEC, privileged or unprivileged, depending on the privileged state of the channel x in HPDMA_PRIVCFGR.PRIVx.

This register is CID-filtered depending on the CID configuration of the channel x (HPDMA_CxCIDCFGR.CFEN = 1). If the debug domain feature is activated, an access to this register is granted to the debug domain CID, regardless of any CID filtering.

This register controls the transfer of a channel x.

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1.

This register must be written when the channel is completed. Then the hardware has de-asserted HPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI, or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by HPDMA from the memory if HPDMA_CxLLR.UT1 = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DSEC	DAP	Res.	DWX	DHX	DBX	DBL_1[5:0]						DINC	Res.	DDW_LOG2[1:0]	
rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SSEC	SAP	SBX	PAM[1:0]		Res.	SBL_1[5:0]						SINC	Res.	SDW_LOG2[1:0]	
rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw		rw	rw

Bit 31 **DSEC**: Security attribute of the HPDMA transfer to the destination

If HPDMA_SECCFGR.SECx = 1 and the access is secure:

0: HPDMA transfer nonsecure

1: HPDMA transfer secure

This is a secure register bit. This bit can only be read by a secure software. This bit must be written by a secure software when HPDMA_SECCFGR.SECx = 1. A secure write is ignored when HPDMA_SECCFGR.SECx = 0.

When HPDMA_SECCFGR.SECx is deasserted, this DSEC bit is also deasserted by hardware (on a secure reconfiguration of the channel as nonsecure), and the HPDMA transfer to the destination is nonsecure.

Bit 30 **DAP**: Destination allocated port

This bit is used to allocate the master port for the destination transfer

0: Port 0 (AXI) allocated

1: Port 1 (AHB) allocated

Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bit 29 Reserved, must be kept at reset value.

Bit 28 **DWX**: Destination word exchange

If the destination data size is not a double-word, this bit is ignored.

If the destination data size is a double-word and if destination bus is AXI (DAP = 0):

0: No word-based exchanged within double-word

1: The two consecutive (post PAM) words are exchanged in each destination double-word.

Bit 27 **DHX**: Destination half-word exchange

If the destination data size is shorter than a word, this bit is ignored.

If the destination data size is a word or double-word and if destination bus is AXI (DAP = 0):

0: No half-word-based exchanged within word

1: The two consecutive (post PAM) half-words are exchanged in each destination word.

Bit 26 **DBX**: Destination byte exchange

If the destination data size is a byte, this bit is ignored.

If the destination data size is not a byte:

0: No byte-based exchange within half-word

1: The two consecutive (post PAM) bytes are exchanged in each destination half-word.

Bits 25:20 **DBL_1[5:0]**: Destination burst length minus 1, between 0 and 63

The burst length unit is one data named beat within a burst. If DBL_1[5:0] = 0, the burst can be named as single. Each data/beat has a width defined by the destination data width DDW_LOG2[1:0].

Caution: As highlighted in *Section 18.3.2*, the maximum allowed AXI burst length is 16. The user must set DBL_1[5:0] lower or equal to 15 if the destination allocated port is AXI (if DAP = 0).

Note: If a burst transfer crossed a 1- or 4-Kbyte address boundary on respectively an AHB or AXI transfer, the HPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB/AXI protocol.

If the burst length exceeds 16 on an AHB transfer, or if the burst on an AXI transfer is both with fixed addressing (DINC = 0) and with a burst length which exceeds 16, the HPDMA modifies and shortens the programmed burst into bursts of lower length, to be compliant with the AHB or AXI protocol.

If a burst transfer is of length greater than the FIFO size of the channel x, the HPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the FIFO size. Transfer performance is lower, with HPDMA re-arbitration between effective and lower singles/bursts, but the data integrity is guaranteed.

Bit 19 **DINC**: Destination incrementing burst

0: Fixed burst

1: Contiguously incremented burst

The destination address, pointed by HPDMA_CxDAR, is kept constant after a burst beat/single transfer, or is incremented by the offset value corresponding to a contiguous data after a burst beat/single transfer.

Bit 18 Reserved, must be kept at reset value.

Bits 17:16 **DDW_LOG2[1:0]**: Binary logarithm of the destination data width of a burst, in bytes

00: Byte

01: Half-word (2 bytes)

10: Word (4 bytes)

11: Double-word (8 bytes)

Note: A destination burst data width must be less or equal to the implemented data width of the allocated AXI/AHB destination port via the DAP bit (it is recommended to be equal for best performance). Otherwise a user setting error is reported and no transfer is issued.

A destination burst transfer must have an aligned destination i) address (HPDMA_CxDAR[2:0]), ii) address offset HPDMA_CxTR3.DAO[2:0] if present, with its destination data width (DDW_LOG2[2:0]). Otherwise a user setting error is reported and no transfer is issued.

When configured in packing mode (PAM[1] = 1 and destination data width different from source data width), a source block size must be a multiple of the destination data width (see HPDMA_CxBR1.BNDT[2:0] versus DDW_LOG2[1:0]). Else a user setting error is reported and none transfer is issued.

Bit 15 **SSEC**: Security attribute of the HPDMA transfer from the source

If HPDMA_SECCFGR.SECx = 1 and the access is secure:

0: HPDMA transfer nonsecure

1: HPDMA transfer secure

This is a secure register bit. This bit can only be read by a secure software. This bit must be written by a secure software when HPDMA_SECCFGR.SECx = 1. A secure write is ignored when HPDMA_SECCFGR.SECx = 0.

When HPDMA_SECCFGR.SECx is deasserted, this SSEC bit is also deasserted by hardware (on a secure reconfiguration of the channel as nonsecure), and the HPDMA transfer from the source is nonsecure.

Bit 14 **SAP**: Source allocated port

This bit is used to allocate the master port for the source transfer

0: Port 0 (AXI) allocated

1: Port 1 (AHB) allocated

Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bit 13 **SBX**: Source byte exchange within the unaligned half-word of each source word

If the source data width is shorter than a word, this bit is ignored.

If the source data width is a word or a double-word, and if source bus is AXI (SAP = 0):

0: No byte-based exchange within the unaligned half-word of each source word

1: The two consecutive bytes within the unaligned half-word of each source word are exchanged.

Bits 12:11 **PAM[1:0]**: Padding/alignment mode

If DDW_LOG2[1:0] = SDW_LOG2[1:0]: if the data width of a burst destination transfer is equal to the data width of a burst source transfer, these bits are ignored.

Else, in the following enumerated values, the condition PAM_1 is when destination data width is higher than source data width, and the condition PAM_2 is when source data width is higher than destination data width.

Condition: PAM_1

00: Source data is transferred as right aligned, padded with 0s up to the destination data width

01: Source data is transferred as right aligned, sign extended up to the destination data width

10-11: Successive source data are FIFO queued and packed at the destination data width, in a left (LSB) to right (MSB) order (named little endian), before a destination transfer

Condition: PAM_2

00: Source data is transferred as right aligned, left-truncated down to the destination data width

01: Source data is transferred as left-aligned, right-truncated down to the destination data width

10-11: Source data is FIFO queued and unpacked at the destination data width, to be transferred in a left (LSB) to right (MSB) order (named little endian) to the destination

Note: If the transfer from the source peripheral is configured with peripheral flow-control mode (SWREQ = 0 and PFREQ = 1 and DREQ = 0), and if the destination data width > the source data width, packing is not supported.

Bit 10 Reserved, must be kept at reset value.

Bits 9:4 **SBL_1[5:0]**: Source burst length minus 1, between 0 and 63

The burst length unit is one data named beat within a burst. If SBL_1[5:0] = 0, the burst can be named as single. Each data/beat has a width defined by the destination data width SDW_LOG2[1:0].

Caution: As highlighted in *Section 18.3.2*, the maximum allowed AXI burst length is 16. The user must set SBL_1[5:0] lower or equal to 15 if the source allocated port is AXI (if SAP = 0).

Note: If a burst transfer crossed a 1- or 4-Kbyte address boundary on respectively an AHB or an AXI transfer, the HPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB protocol.

If the burst length exceeds 16 on a AHB transfer, or if the burst on a AXI transfer is both with fixed addressing (SINC = 0) and with a burst length which exceeds 16, the HPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the FIFO size. Transfer performance is lower, with HPDMA re-arbitration between effective and lower singles/bursts, but the data integrity is guaranteed.

Bit 3 **SINC**: Source incrementing burst

0: Fixed burst

1: Contiguously incremented burst

The source address, pointed by HPDMA_CxSAR, is kept constant after a burst beat/single transfer or is incremented by the offset value corresponding to a contiguous data after a burst beat/single transfer.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **SDW_LOG2[1:0]**: Binary logarithm of the source data width of a burst in bytes
00: Byte
01: Half-word (2 bytes)
10: Word (4 bytes)
11: Double-word (8 bytes)

Note: A source burst data width must be less or equal to the implemented data width of the allocated AXI/AHB source port via the SAP bit (it is recommended to be equal for best performance). Otherwise a user setting error is reported and no transfer is issued.

A source block size must be a multiple of the source data width (HPDMA_CxBR1.BNDT[2:0] versus SDW_LOG2[1:0]). Otherwise, a user setting error is reported and no transfer is issued.

A source burst transfer must have an aligned source i) address (HPDMA_CxSAR[2:0]), ii) address offset HPDMA_CxTR3.SAO[2:0] if present, with its source data width (SDW_LOG2[1:0]). Otherwise a user setting error is reported and no transfer is issued.

18.8.13 HPDMA channel x transfer register 2 (HPDMA_CxTR2)

Address offset: 0x094 + 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (HPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (HPDMA_PRIVCFGR.PRIVx).

This register is CID-filtered depending on the CID configuration of the channel x (HPDMA_CxCIDCFGR.CFEN = 1). If the debug domain feature is activated, a write access to this register is granted to the debug domain CID, regardless of any CID filtering.

This register controls the transfer of a channel x.

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (the hardware deasserted HPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI, or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by HPDMA from the memory, if HPDMA_CxLLR.UT2 = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TCEM[1:0]		Res.	Res.	Res.	Res.	TRIGPOL[1:0]		Res.	TRIGSEL[6:0]						
rw	rw					rw	rw		rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRIGM[1:0]		Res.	PFREQ	BREQ	DREQ	SWREQ	Res.	REQSEL[7:0]							
rw	rw		rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 **TCEM[1:0]**: Transfer complete event mode

These bits define the transfer granularity for the transfer complete and half-transfer complete events generation.

00: At block level (when HPDMA_CxBR1.BNDT[15:0] = 0): The complete (and the half) transfer event is generated at the (respectively half of the) end of a block.

Note: If the initial LLI_0 data transfer is null/void (directly programmed by the internal register file with HPDMA_CxBR1.BNDT[15:0] = 0), then neither the complete transfer event nor the half-transfer event is generated.

01: if the channel is not 2D addressing capable: same as 00; if the channel is 2D addressing capable: at 2D/repeated block level, meaning when HPDMA_CxBR1.BRC[10:0] = 0 and HPDMA_CxBR1.BNDT[15:0] = 0), the complete (and the half) transfer event is generated at the end (respectively half of the end) of the 2D/repeated block.

Note: If the initial LLI_0 data transfer is null/void (directly programmed by the internal register file with HPDMA_CxBR1.BNDT[15:0] = 0), then neither the complete transfer event nor the half-transfer event is generated.

10: At LLI level: The complete transfer event is generated at the end of the LLI transfer, including the update of the LLI if any. The half-transfer event is generated at the half of the LLI data transfer (the LLI data transfer being a block transfer or a 2D/repeated block transfer for a 2D addressing capable channel), if any data transfer.

Note: If the initial LLI_0 data transfer is null/void (directly programmed by the internal register file with HPDMA_CxBR1.BNDT[15:0] = 0), then the half-transfer event is not generated, and the transfer complete event is generated when is completed the loading of the LLI_1.

11: At channel level: The complete transfer event is generated at the end of the last LLI transfer. The half-transfer event is generated at the half of the data transfer of the last LLI. The last LLI updates the link address HPDMA_CxLLR.LA[15:2] to zero and clears all the HPDMA_CxLLR update bits (UT1, UT2, UB1, USA, UDA and ULL, plus UT3 and UB2 if present). If the channel transfer is continuous/infinite, no event is generated.

Bits 29:26 Reserved, must be kept at reset value.

Bits 25:24 **TRIGPOL[1:0]**: Trigger event polarity

These bits define the polarity of the selected trigger event input defined by TRIGSEL[6:0].

00: No trigger (masked trigger event)

01: Trigger on the rising edge

10: Trigger on the falling edge

11: Same as 00

Bit 23 Reserved, must be kept at reset value.

Bits 22:16 **TRIGSEL[6:0]**: Trigger event input selection

These bits select the trigger event input of the HPDMA transfer (as per *Section 18.3.7*), with an active trigger event if TRIGPOL[1:0] ≠ 00.

Bits 15:14 **TRIGM[1:0]**: Trigger mode

These bits define the transfer granularity for its conditioning by the trigger.

If the channel x is enabled (HPDMA_CxCR.EN asserted) with TRIGPOL[1:0] = 00 or 11, these TRIGM[1:0] bits are ignored.

Else, an HPDMA transfer is conditioned by at least one trigger hit:

00: at block level: the first burst read of each block transfer is conditioned by one hit trigger. This is also valid for a 2D addressing capable channel: for each block even if a 2D/repeated block is configured with HPDMA_CxBR1.BRC[10:0] ≠ 0.

01: if the channel is not 2D addressing capable: same as 00; if the channel is 2D addressing capable: at 2D/repeated block level, meaning the first burst read of a 2D/repeated block transfer is conditioned by one hit trigger.

10: at link level: a LLI link transfer is conditioned by one hit trigger. The LLI data transfer (if any) is not conditioned.

11: at programmed burst level: If SWREQ = 1, each programmed burst read is conditioned by one hit trigger. If SWREQ = 0, each programmed burst that is requested by the selected peripheral, is conditioned by one hit trigger.

– If the peripheral is programmed as a source (DREQ = 0) of the LLI data transfer, each programmed burst read is conditioned.

– If the peripheral is programmed as a destination (DREQ = 1) of the LLI data transfer, each programmed burst write is conditioned. The first memory burst read of a (possibly 2D/repeated) block, also named as the first ready FIFO-based source burst, is gated by the occurrence of both the hardware request and the first trigger hit.

The HPDMA monitoring of a trigger for channel x is started when the channel is enabled/loaded with a new active trigger configuration: rising or falling edge on a selected trigger (TRIGPOL[1:0] = 01 or respectively TRIGPOL[1:0] = 10).

The monitoring of this trigger is kept active during the triggered and uncompleted (data or link) transfer; and if a new trigger is detected then, this hit is internally memorized to grant the next transfer, as long as the defined rising or falling edge is not modified, and the TRIGSEL[6:0] is not modified, and the channel is enabled.

Transferring a next LLI_{n+1} that updates the HPDMA_CxTR2 with a new value for any of TRIGSEL[6:0] or TRIGPOL[1:0], resets the monitoring, trashing the memorized hit of the formerly defined LLI_n trigger.

After a first new trigger hit_{n+1} is memorized, if another second trigger hit_{n+2} is detected and if the hit_n triggered transfer is still not completed, hit_{n+2} is lost and not memorized. A trigger overrun flag is reported (HPDMA_CxSR.TOF = 1), and an interrupt is generated if enabled (HPDMA_CxCR.TOIE = 1). The channel is not automatically disabled by hardware due to a trigger overrun.

Note: When the source block size is not a multiple of the source burst size and is a multiple of the source data width, then the last programmed source burst is not completed and is internally shorten to match the block size. In this case, if TRIGM[1:0] = 11 and (SWREQ = 1 or (SWREQ = 0 and DREQ = 0)), the shortened burst transfer (by singles or/and by bursts of lower length) is conditioned once by the trigger.

When the programmed destination burst is internally shortened by singles or/and by bursts of lower length (versus FIFO size, versus block size, 1/4-Kbyte boundary address crossing maximum burst length versus AHB/AXI protocol): if the trigger is conditioning the programmed destination burst (if TRIGM[1:0] = 11 and SWREQ = 0 and DREQ = 1), this shortened destination burst transfer is conditioned once by the trigger.

Bit 13 Reserved, must be kept at reset value.

Bit 12 **PFREQ**: Hardware request in peripheral flow control mode

Important: If a given channel x is not implemented with this feature, this bit is reserved and PFREQ is not present (see Section 18.3.6 for the list of the implemented channels with this feature.

If the channel x is activated (HPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer), this bit is ignored. Else:

0: The selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol in HPDMA control mode. The HPDMA is programmed with HPDMA_CxCBR1.BNDT[15:0] and this is internally used by the hardware for the block transfer completion.

1: The selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol in peripheral control mode. The HPDMA block transfer can be early completed by the peripheral itself (see *Section 18.3.6* for more details).

Note: In peripheral flow control mode, there are the following restrictions:

- no 2D/repeated block support (HPDMA_CxBR1.BRC[10:0] must be set to 0 if present)

- the peripheral must be set as the source of the transfer (DREQ = 0).

- data packing to a wider destination width is not supported (if destination width > source data width, HPDMA_CxTR1.PAM[1] must be set to 0).

- HPDMA_CxBR1.BNDT[15:0] must be set as a multiple of the source (peripheral) burst size.

Bit 11 **BREQ**: Block hardware request

If the channel x is activated (HPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer), this bit is ignored. Else:

0: The selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol at a burst level.

1: The selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol at a block level (see *Section 18.3.5*).

Bit 10 **DREQ**: Destination hardware request

This bit is ignored if channel x is activated (HPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer). Else:

0: Selected hardware request driven by a source peripheral (request signal taken into account by the HPDMA transfer scheduler over the source/read port)

1: Selected hardware request driven by a destination peripheral (request signal taken into account by the HPDMA transfer scheduler over the destination/write port)

Note: If the channel x is activated (HPDMA_CxCR.EN is asserted) with SWREQ = 0 and PFREQ = 1 (peripheral hardware request with peripheral flow-control mode), any software assertion to this DREQ bit is ignored: in peripheral flow-control mode, only a peripheral-to-memory transfer is supported.

Bit 9 **SWREQ**: Software request

This bit is internally taken into account when HPDMA_CxCR.EN is asserted.

0: No software request. The selected hardware request REQSEL[7:0] is taken into account.

1: Software request for a memory-to-memory transfer. The default selected hardware request as per REQSEL[7:0] is ignored.

Bit 8 Reserved, must be kept at reset value.

Bits 7:0 **REQSEL[7:0]**: Hardware request selection

These bits are ignored if channel x is activated (HPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer). Else, the selected hardware request is internally taken into account as per *Section 18.3.4*.

Caution: The user must not assign a same input hardware request (same REQSEL[7:0] value) to different active HPDMA channels (HPDMA_CxCR.EN = 1 and HPDMA_CxTR2.SWREQ = 0 for these channels). The HPDMA is not intended to hardware support the case of simultaneous enabled channels incorrectly configured with a same hardware peripheral request signal, and there is no user setting error reporting.

18.8.14 HPDMA channel x block register 1 (HPDMA_CxBR1)

Address offset: 0x098 + 0x80 * x (x = 0 to 11)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (HPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (HPDMA_PRIVCFGR.PRIVx).

This register is CID-filtered depending on the CID configuration of the channel x (HPDMA_CxCIDCFGR.CFEN = 1). If the debug domain feature is activated, a write access to this register is granted to the debug domain CID, regardless of any CID filtering.

This register controls the transfer of a channel x at a block level.

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1.

This register must be written when channel x is completed (then the hardware has de-asserted HPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI, or full linked-list.

In linked-list mode, during the link transfer:

- if HPDMA_CxLLR.UB1 = 1, this register is automatically updated by the HPDMA from the next LLI in memory.
- If HPDMA_CxLLR.UB1 = 0 and if there is at least one linked-list register to be updated from the next LLI in memory, this register is automatically and internally restored with the programmed value for the bitfield BNDT[15:0].
- If all the update bits HPDMA_CxLLR.Uxx are null and if HPDMA_CxLLR.LA[15:0] ≠ 0, the current LLI is the last one and is continuously executed: this register is automatically and internally restored with the programmed value for BNDT[15:0] after each execution of this final LLI
- If HPDMA_CxLLR = 0, this register and BNDT[15:0] are kept as null, channel x is completed.

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **BNDT[15:0]**: Block number of data bytes to transfer from the source

Block size transferred from the source. When the channel is enabled, this bitfield becomes read-only and is decremented, indicating the remaining number of data items in the current source block to be transferred. BNDT[15:0] is programmed in number of bytes, maximum source block size is 64 Kbytes -1.

Once the last data transfer is completed (BNDT[15:0] = 0):

- if HPDMA_CxLLR.UB1 = 1, this bitfield is updated with the LLI in the memory.
- if HPDMA_CxLLR.UB1 = 0 and if there is at least one non null Uxx update bit, this bitfield is internally restored to the programmed value.
- if all HPDMA_CxLLR.Uxx = 0 and if HPDMA_CxLLR.LA[15:0] = 0, this bitfield is internally restored to the programmed value (infinite/continuous last LLI).
- if HPDMA_CxLLR = 0, this bitfield is kept as zero following the last LLI data transfer.

Note: A non-null source block size must be a multiple of the source data width (BNDT versus HPDMA_CxTR1.SDW_LOG2). Else a user setting error is reported and no transfer is issued.

When configured in packing mode (HPDMA_CxTR1.PAM[1] = 1 and destination data width different from source data width), a non-null source block size must be a multiple of the destination data width (BNDT versus HPDMA_CxTR1.DDW_LOG2). Else a user setting error is reported and no transfer is issued.

18.8.15 HPDMA channel x alternate block register 1 (HPDMA_CxBR1)

Address offset: 0x098 + 0x80 * x (x = 12 to 15)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (HPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (HPDMA_PRIVCFGR.PRIVx).

This register is CID-filtered depending on the CID configuration of the channel x (HPDMA_CxCIDCFGR.CFEN = 1). If the debug domain feature is activated, a write access to this register is granted to the debug domain CID, regardless of any CID filtering.

This register controls the transfer of a channel x at a block level.

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1.

This register must be written when channel x is completed (then the hardware has de-asserted HPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI, or full linked-list.

In linked-list mode, during the link transfer:

- if HPDMA_CxLLR.UB1 = 1, this register is automatically updated by the HPDMA from the next LLI in memory.
- If HPDMA_CxLLR.UB1 = 0 and if there is at least one linked-list register to be updated from the next LLI in memory, this register is automatically and internally restored with the programmed value for the bitfields BNDT[15:0] and BRC[10:0].
- If all the update bits HPDMA_CxLLR.Uxx are null and if HPDMA_CxLLR.LA[15:0] ≠ 0, the current LLI is the last one and is continuously executed: this register is automatically and internally restored with the programmed value for the bitfields BNDT[15:0] and BRC[10:0] after each execution of this final LLI

- If HPDMA_CxLLR = 0, BNDT[15:0] and BRC[10:0] are kept as null, channel x is completed.

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

Bit 31 **BRDDEC**: Block repeat destination address decrement

0: At the end of a block transfer, the HPDMA_CxDAR register is updated by adding the programmed offset HPDMA_CxBR2.BRDAO to the current HPDMA_CxDAR value (current destination address)

1: At the end of a block transfer, the HPDMA_CxDAR register is updated by subtracting the programmed offset HPDMA_CxBR2.BRDAO from the current HPDMA_CxDAR value (current destination address)

Note: On top of this increment/decrement (depending on BRDDEC), HPDMA_CxDAR is in the same time also updated by the increment/decrement (depending on DDEC) of the HPDMA_CxTR3.DAO value, as it is usually done at the end of each programmed burst transfer.

Bit 30 **BRSDEC**: Block repeat source address decrement

0: At the end of a block transfer, the HPDMA_CxSAR register is updated by adding the programmed offset HPDMA_CxBR2.BRSAO to the current HPDMA_CxSAR value (current source address)

1: At the end of a block transfer, the HPDMA_CxSAR register is updated by subtracting the programmed offset HPDMA_CxBR2.BRSAO from the current HPDMA_CxSAR value (current source address)

Note: On top of this increment/decrement (depending on BRSDEC), HPDMA_CxSAR is in the same time also updated by the increment/decrement (depending on SDEC) of the HPDMA_CxTR3.SAO value, as it is done after any programmed burst transfer.

Bit 29 **DDEC**: destination address decrement

0: At the end of a programmed burst transfer to the destination, the HPDMA_CxDAR register is updated by adding the programmed offset HPDMA_CxTR3.DAO to the current HPDMA_CxDAR value (current destination address)

1: At the end of a programmed burst transfer to the destination, the HPDMA_CxDAR register is updated by subtracting the programmed offset HPDMA_CxTR3.DAO to the current HPDMA_CxDAR value (current destination address)

Bit 28 **SDEC**: source address decrement

0: At the end of a programmed burst transfer from the source, the HPDMA_CxSAR register is updated by adding the programmed offset HPDMA_CxTR3.SAO to the current HPDMA_CxSAR value (current source address)

1: At the end of a programmed burst transfer from the source, the HPDMA_CxSAR register is updated by subtracting the programmed offset HPDMA_CxTR3.SAO to the current HPDMA_CxSAR value (current source address)

Bit 27 Reserved, must be kept at reset value.

Bits 26:16 **BRC[10:0]**: Block repeat counter

This bitfield contains the number of repetitions of the current block (0 to 2047).

When the channel is enabled, this bitfield becomes read-only. After decrements, this bitfield indicates the remaining number of blocks, excluding the current one. This counter is hardware decremented for each completed block transfer.

Once the last block transfer is completed (BRC[10:0] = BNDT[15:0] = 0):

– If HPDMA_CxLLR.UB1 = 1, all HPDMA_CxBR1 bitfields are updated by the next LLI in the memory.

– If HPDMA_CxLLR.UB1 = 0 and if there is at least one not null Uxx update bit, this bitfield is internally restored to the programmed value.

– if all HPDMA_CxLLR.Uxx = 0 and if HPDMA_CxLLR.LA[15:0] ≠ 0, this bitfield is internally restored to the programmed value (infinite/continuous last LLI).

– if HPDMA_CxLLR = 0, this bitfield is kept as zero following the last LLI and data transfer.

Bits 15:0 **BNDT[15:0]**: Block number of data bytes to transfer from the source

Block size transferred from the source. When the channel is enabled, this bitfield becomes read-only and is decremented, indicating the remaining number of data items in the current source block to be transferred.

BNDT[15:0] is programmed in number of bytes, maximum source block size is 64 Kbytes -1.

Once the last data transfer is completed (BNDT[15:0] = 0):

- if HPDMA_CxLLR.UB1 = 1, this bitfield is updated with the LLI in the memory.

- if HPDMA_CxLLR.UB1 = 0 and if there is at least one not null Uxx update bit, this bitfield is internally restored to the programmed value.

- if all HPDMA_CxLLR.Uxx = 0 and if HPDMA_CxLLR.LA[15:0] ≠ 0, this bitfield is internally restored to the programmed value (infinite/continuous last LLI).

- if HPDMA_CxLLR = 0, this bitfield is kept as zero following the last LLI data transfer.

Note: A non-null source block size must be a multiple of the source data width (BNDT versus HPDMA_CxTR1.SDW_LOG2). Else a user setting error is reported and no transfer is issued.

When configured in packing mode (HPDMA_CxTR1.PAM[1] = 1 and destination data width different from source data width), a non-null source block size must be a multiple of the destination data width (BNDT versus HPDMA_CxTR1.DDW_LOG2). Else a user setting error is reported and no transfer is issued.

18.8.16 HPDMA channel x source address register (HPDMA_CxSAR)

Address offset: 0x09C + 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (HPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (HPDMA_PRIVCFGR.PRIVx).

This register is CID-filtered depending on the CID configuration of the channel x (HPDMA_CxCIDCFGR.CFEN = 1). If the debug domain feature is activated, an access to this register is granted to the debug domain CID, regardless of any CID filtering.

This register configures the source start address of a transfer.

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1, and continuously updated by hardware, in order to reflect the address of the next burst transfer from the source.

This register must be written when the channel is completed (then the hardware has deasserted HPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI, or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the HPDMA from the memory if HPDMA_CxLLR.USA = 1.

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

Bits 31:0 **SA[31:0]**: Source address

This bitfield is the pointer to the address from which the next data is read.

During the channel activity, depending on the source addressing mode (HPDMA_CxTR1.SINC), this bitfield is kept fixed or incremented by the data width (HPDMA_CxTR1.SDW_LOG2[1:0]) after each burst source data, reflecting the next address from which data is read.

During the channel activity, this address is updated after each completed source burst, consequently to:

– the programmed source burst; either in fixed addressing mode or in contiguous-data incremented mode. If contiguously incremented (HPDMA_CxTR1.SINC = 1), then the additional address offset value is the programmed burst size, as defined by HPDMA_CxTR1.SBL_1[5:0] and HPDMA_CxTR1.SDW_LOG2[1:0]

– the additional source incremented/decremented offset value as programmed by HPDMA_CxBR1.SDEC and HPDMA_CxTR3.SAO[12:0]

– once/if completed source block transfer, for a channel x with 2D addressing capability (x = 12 to 15). additional block repeat source incremented/decremented offset value as programmed by HPDMA_CxBR1.BRSDEC and HPDMA_CxBR2.BRSAO[15:0]

In linked-list mode, after a LLI data transfer is completed, this register is automatically updated by HPDMA from the memory, provided the LLI is set with HPDMA_CxLLR.USA = 1.

Note: A source address must be aligned with the programmed data width of a source burst (SA versus HPDMA_CxTR1.SDW_LOG2). Else, a user setting error is reported and no transfer is issued.

When the source block size is not a multiple of the source burst size and is a multiple of the source data width, the last programmed source burst is not completed and is internally shorten to match the block size. In this case, the additional HPDMA_CxTR3.SAO[12:0] is not applied.

18.8.17 HPDMA channel x destination address register (HPDMA_CxDAR)

Address offset: 0x0A0 + 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (HPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (HPDMA_PRIVCFGR.PRIVx).

This register is CID-filtered depending on the CID configuration of the channel x (HPDMA_CxCIDCFGR.CFEN = 1). If the debug domain feature is activated, an access to this register is granted to the debug domain CID, regardless of any CID filtering.

This register configures the destination start address of a transfer.

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1, and continuously updated by hardware, in order to reflect the address of the next burst transfer to the destination.

This register must be written when the channel is completed (then the hardware has deasserted HPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI, or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by HPDMA from the memory if HPDMA_CxLLR.UDA = 1.

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

Bits 31:0 **DA[31:0]**: destination address

This bitfield is the pointer to the address from which the next data is written.

During the channel activity, depending on the destination addressing mode (HPDMA_CxTR1.DINC), this bitfield is kept fixed or incremented by the data width (HPDMA_CxTR1.DDW_LOG2[1:0]) after each burst destination data, reflecting the next address from which data is written.

During the channel activity, this address is updated after each completed destination burst, consequently to:

– the programmed destination burst; either in fixed addressing mode or in contiguous-data incremented mode. If contiguously incremented (HPDMA_CxTR1.DINC = 1), then the additional address offset value is the programmed burst size, as defined by HPDMA_CxTR1.DBL_1[5:0] and HPDMA_CxTR1.DDW_LOG2[1:0]

– the additional destination incremented/decremented offset value as programmed by HPDMA_CxBR1.DDEC and HPDMA_CxTR3.DAO[12:0]

– once/if completed destination block transfer, for a channel x with 2D addressing capability , the additional block repeat destination incremented/decremented offset value as programmed by HPDMA_CxBR1.BRDDEC and HPDMA_CxBR2.BRDAO[15:0]

In linked-list mode, after a LLI data transfer is completed, this register is automatically updated by the HPDMA from the memory, provided the LLI is set with HPDMA_CxLLR.UDA = 1.

Note: A destination address must be aligned with the programmed data width of a destination burst (DA versus HPDMA_CxTR1.DDW_LOG2). Else, a user setting error is reported and no transfer is issued.

18.8.18 HPDMA channel x transfer register 3 (HPDMA_CxTR3)

Address offset: 0x0A4 + 0x80 * x (x = 12 to 15)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (HPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (HPDMA_PRIVCFGR.PRIVx).

This register is CID-filtered depending on the CID configuration of the channel x (HPDMA_CxCIDCFGR.CFEN = 1). If the debug domain feature is activated, a write access to this register is granted to the debug domain CID, regardless of any CID filtering.

This register controls the transfer of a channel x.

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted HPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI, or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the HPDMA from the memory if HPDMA_CxLLR.UT3 = 1.

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

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:16 **DAO[12:0]**: Destination address offset increment

The destination address, pointed by HPDMA_CxDAR, is incremented or decremented (depending on HPDMA_CxBR1.DDEC) by this offset DAO[12:0] for each programmed destination burst. This offset is not including and is added to the programmed burst size when the completed burst is addressed in incremented mode (HPDMA_CxTR1.DINC = 1).

Note: A destination address offset must be aligned with the programmed data width of a destination burst (DAO versus HPDMA_CxTR1.DDW_LOG2). Else, a user setting error is reported and no transfer is issued.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:0 **SAO[12:0]**: Source address offset increment

The source address, pointed by HPDMA_CxSAR, is incremented or decremented (depending on HPDMA_CxBR1.SDEC) by this offset SAO[12:0] for each programmed source burst. This offset is not including and is added to the programmed burst size when the completed burst is addressed in incremented mode (HPDMA_CxTR1.SINC = 1).

Note: A source address offset must be aligned with the programmed data width of a source burst (SAO versus HPDMA_CxTR1.SDW_LOG2). Else a user setting error is reported and none transfer is issued.

When the source block size is not a multiple of the destination burst size and is a multiple of the source data width, then the last programmed source burst is not completed and is internally shorten to match the block size. In this case, the additional HPDMA_CxTR3.SAO[12:0] is not applied.

18.8.19 HPDMA channel x block register 2 (HPDMA_CxBR2)

Address offset: 0x0A8 + 0x80 * x (x = 12 to 15)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (HPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (HPDMA_PRIVCFGR.PRIVx).

This register is CID-filtered depending on the CID configuration of the channel x (HPDMA_CxCIDCFGR.CFEN = 1). If the debug domain feature is activated, a write access to this register is granted to the debug domain CID, regardless of any CID filtering.

This register controls the transfer of a channel x at a 2D/repeated block level.

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted HPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI, or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the HPDMA from the memory if HPDMA_CxLLR.UB2 = 1.

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

Bits 31:16 **BRDAO[15:0]**: Block repeated destination address offset

For a channel with 2D addressing capability, this bitfield is used to update (by addition or subtraction depending on HPDMA_CxBR1.BRDDEC) the current destination address (HPDMA_CxDAR) at the end of a block transfer.

Note: A block repeated destination address offset must be aligned with the programmed data width of a destination burst (BRDAO versus HPDMA_CxTR1.DDW_LOG2). Else a user setting error is reported and no transfer is issued.
BRDAO[15:0] must be set to 0 in peripheral flow-control mode (if HPDMA_CxTR2.PFREQ = 1).

Bits 15:0 **BRSAO[15:0]**: Block repeated source address offset

For a channel with 2D addressing capability, this bitfield is used to update (by addition or subtraction depending on HPDMA_CxBR1.BRSDEC) the current source address (HPDMA_CxSAR) at the end of a block transfer.

Note: A block repeated source address offset must be aligned with the programmed data width of a source burst (BRSAO versus HPDMA_CxTR1.SDW_LOG2). Else a user setting error is reported and no transfer is issued.
BRSAO[15:0] must be set to 0 in peripheral flow-control mode (if HPDMA_CxTR2.PFREQ = 1).

18.8.20 HPDMA channel x linked-list address register (HPDMA_CxLLR)

Address offset: 0x0CC + 0x80 * x (x = 0 to 11)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (HPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (HPDMA_PRIVCFGR.PRIVx).

This register is CID-filtered depending on the CID configuration of the channel x (HPDMA_CxCIDCFGR.CFEN = 1). If the debug domain feature is activated, a write access to this register is granted to the debug domain CID, regardless of any CID filtering.

This register configures the data structure of the next LLI in the memory and its address pointer. A channel transfer is completed when this register is null.

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted HPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI, or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the HPDMA from the memory if HPDMA_CxLLR.ULL = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UT1	UT2	UB1	USA	UDA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ULL
rw	rw	rw	rw	rw											rw

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

Bit 31 **UT1**: Update HPDMA_CxTR1 from memory
This bit controls the update of HPDMA_CxTR1 from the memory during the link transfer.
0: No HPDMA_CxTR1 update
1: HPDMA_CxTR1 update

Bit 30 **UT2**: Update HPDMA_CxTR2 from memory
This bit controls the update of HPDMA_CxTR2 from the memory during the link transfer.
0: No HPDMA_CxTR2 update
1: HPDMA_CxTR2 update

Bit 29 **UB1**: Update HPDMA_CxBR1 from memory
This bit controls the update of HPDMA_CxBR1 from the memory during the link transfer.
If UB1 = 0 and if HPDMA_CxLLR ≠ 0, the linked-list is not completed.
HPDMA_CxBR1.BNDT[15:0] is then restored to the programmed value after data transfer is completed and before the link transfer.
0: No HPDMA_CxBR1 update from memory (HPDMA_CxBR1.BNDT[15:0] restored if any link transfer)
1: HPDMA_CxBR1 update

Bit 28 **USA**: Update HPDMA_CxSAR from memory
This bit controls the update of HPDMA_CxSAR from the memory during the link transfer.
0: No HPDMA_CxSAR update
1: HPDMA_CxSAR update

Bit 27 **UDA**: Update HPDMA_CxDAR register from memory
This bit is used to control the update of HPDMA_CxDAR from the memory during the link transfer.
0: No HPDMA_CxDAR update
1: HPDMA_CxDAR update

Bits 26:17 Reserved, must be kept at reset value.

Bit 16 **ULL**: Update HPDMA_CxLLR register from memory
This bit is used to control the update of HPDMA_CxLLR from the memory during the link transfer.
0: No HPDMA_CxLLR update
1: HPDMA_CxLLR update

Bits 15:2 **LA[15:2]**: Pointer (16-bit low-significant address) to the next linked-list data structure
If UT1 = UT2 = UB1 = USA = UDA = ULL = 0 and if LA[15:2] = 0, the current LLI is the last one. The channel transfer is completed without any update of the linked-list HPDMA register file.
Else, this bitfield is the pointer to the memory address offset from which the next linked-list data structure is automatically fetched from, once the data transfer is completed, in order to conditionally update the linked-list HPDMA internal register file (HPDMA_CxTR1, HPDMA_CxTR2, HPDMA_CxBR1, HPDMA_CxSAR, HPDMA_CxDAR, and HPDMA_CxLLR).

Caution: the user must program this pointer not to exceed the 64-Kbyte addressable space defined by the link base address register HPDMA_CxLBAR. The complete linked-list data structure must be included in the 64-Kbyte addressable space.

Note: The user must program the pointer to be 32-bit aligned. The two low-significant bits are write ignored.

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

18.8.21 HPDMA channel x alternate linked-list address register (HPDMA_CxLLR)

Address offset: 0x0CC + 0x80 * x (x = 12 to 15)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (HPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (HPDMA_PRIVCFGR.PRIVx).

This register is CID-filtered depending on the CID configuration of the channel x (HPDMA_CxCIDCFGR.CFEN = 1). If the debug domain feature is activated, a write access to this register is granted to the debug domain CID, regardless of any CID filtering.

This register configures the data structure of the next LLI in the memory and its address pointer. A channel transfer is completed when this register is null.

This register must be written when HPDMA_CxCR.EN = 0.

This register is read-only when HPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted HPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI, or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the HPDMA from the memory if HPDMA_CxLLR.ULL = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UT1	UT2	UB1	USA	UDA	UT3	UB2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ULL
rw	rw	rw	rw	rw	rw	rw									rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LA[15:2]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bit 31 **UT1**: Update HPDMA_CxTR1 from memory

This bit controls the update of HPDMA_CxTR1 from the memory during the link transfer.
0: No HPDMA_CxTR1 update
1: HPDMA_CxTR1 update

Bit 30 **UT2**: Update HPDMA_CxTR2 from memory

This bit controls the update of HPDMA_CxTR2 from the memory during the link transfer.
0: No HPDMA_CxTR2 update
1: HPDMA_CxTR2 update

Bit 29 **UB1**: Update HPDMA_CxBR1 from memory

This bit controls the update of HPDMA_CxBR1 from the memory during the link transfer.
If UB1 = 0 and if HPDMA_CxLLR ≠ 0, the linked-list is not completed.
HPDMA_CxBR1.BNDT[15:0] is then restored to the programmed value after data transfer is completed and before the link transfer.
0: No HPDMA_CxBR1 update from memory (HPDMA_CxBR1.BNDT[15:0] restored if any link transfer)
1: HPDMA_CxBR1 update

Bit 28 **USA**: Update HPDMA_CxSAR from memory
This bit controls the update of HPDMA_CxSAR from the memory during the link transfer.
0: No HPDMA_CxSAR update
1: HPDMA_CxSAR update

Bit 27 **UDA**: Update HPDMA_CxDAR register from memory
This bit is used to control the update of HPDMA_CxDAR from the memory during the link transfer.
0:No HPDMA_CxDAR update
1: HPDMA_CxDAR update

Bit 26 **UT3**: Update HPDMA_CxTR3 from memory
This bit controls the update of HPDMA_CxTR3 from the memory during the link transfer.
0: No HPDMA_CxTR3 update
1: HPDMA_CxTR3 update

Bit 25 **UB2**: Update HPDMA_CxBR2 from memory
This bit controls the update of HPDMA_CxBR2 from the memory during the link transfer.
0: No HPDMA_CxBR2 update
1: HPDMA_CxBR2 update

Bits 24:17 Reserved, must be kept at reset value.

Bit 16 **ULL**: Update HPDMA_CxLLR register from memory
This bit is used to control the update of HPDMA_CxLLR from the memory during the link transfer.
0: No HPDMA_CxLLR update
1: HPDMA_CxLLR update

Bits 15:2 **LA[15:2]**: Pointer (16-bit low-significant address) to the next linked-list data structure
If UT1 = UT2 = UB1 = USA = UDA = ULL= 0 and if LA[15:2] = 0, the current LLI is the last one. The channel transfer is completed without any update of the linked-list HPDMA register file.
Else, this bitfield is the pointer to the memory address offset from which the next linked-list data structure is automatically fetched from, once the data transfer is completed, in order to conditionally update the linked-list HPDMA internal register file (HPDMA_CxTR1, HPDMA_CxTR2, HPDMA_CxBR1, HPDMA_CxSAR, HPDMA_CxDAR and HPDMA_CxLLR).

Caution: the user must program this pointer not to exceed the 64-Kbyte addressable space defined by the link base address register HPDMA_CxLBAR. The complete linked-list data structure must be included in the 64-Kbyte addressable space.

Note: The user must program the pointer to be 32-bit aligned. The two low-significant bits are write ignored.

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

18.8.22 HPDMA register map

Table 95. HPDMA register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	HPDMA_ SECCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	HPDMA_ PRIVCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	HPDMA_ RCFGLOCKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCK15	LOCK14	LOCK13	LOCK12	LOCK11	LOCK10	LOCK9	LOCK8	LOCK7	LOCK6	LOCK5	LOCK4	LOCK3	LOCK2	LOCK1	LOCK0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	HPDMA_MISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MIS15	MIS14	MIS13	MIS12	MIS11	MIS10	MIS9	MIS8	MIS7	MIS6	MIS5	MIS4	MIS3	MIS2	MIS1	MIS0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	HPDMA_SMISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MIS15	MIS14	MIS13	MIS12	MIS11	MIS10	MIS9	MIS8	MIS7	MIS6	MIS5	MIS4	MIS3	MIS2	MIS1	MIS0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014 - 0x04C	Reserved	Reserved																															
0x050 + 0x80 * x (x = 0 to 15)	HPDMA_CxLBAR	LBA[31:16]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																
0x054 + 0x80 * x (x = 0 to 15)	HPDMA_ CxCIDCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEM_WLIST_CID6	SEM_WLIST_CID5	SEM_WLIST_CID4	SEM_WLIST_CID3	SEM_WLIST_CID2	SEM_WLIST_CID1	SEM_WLIST_CID0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SCID[2:0]			Res.	Res.	SEM_EN	CFEN
	Reset value										0	0	0	0	0	0	0										0	0	0			0	0
0x058 + 0x80 * x (x = 0 to 15)	HPDMA_ CxSEMCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	SEM_CCID [2:0]			Res.	Res.	Res.	SEM_MUTEX
	Reset value																										0	0	0				0
0x05C + 0x080 * x (x=0 to 15)	HPDMA_CxFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																		0	0	0	0	0	0	0								
0x060 + 0x080 * x (x=0 to 15)	HPDMA_CxSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FIFOL[8:0]									Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDLEF
	Reset value								0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0								1
0x064 + 0x080 * x (x=0 to 15)	HPDMA_CxCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIO[1:0]		Res.	Res.	Res.	Res.	LAP	LSM	Res.	TOIE	SUSPIE	USEIE	ULEIE	DTEIE	HTIE	TCIE	Res.	Res.	Res.	Res.	Res.	SUSP	RESET	EN
	Reset value									0	0					0	0		0	0	0	0	0	0	0						0	0	0
0x090 + 0x080 * x (x=0 to 15)	HPDMA_CxTR1	DSEC	DAP	Res.	DWX	DHX	DBX	DBL_1[5:0]						DINC	Res.	DDW_LOG2[1:0]		SSEC	SAP	SBX	PAM[1:0]		Res.	SBL_1[5:0]						SINC	Res.	SDW_LOG2[1:0]	
	Reset value	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0

Table 95. HPDMA register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x098 + 0x080 * x (x=0 to 11)	HPDMA_CxBR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BNDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x098+ 0x080 * x (x=12 to 15)	HPDMA_CxBR1	BRDDEC	BRSDEC	DDEC	SDEC	Res.	BRC[10:0]											BNDT[15:0]															
	Reset value	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x09C + 0x080 * x (x=0 to 15)	HPDMA_CxSAR	SA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A0 + 0x080 * x (x=0 to 15)	HPDMA_CxDAR	DA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A4 + 0x080 * x (x=12 to 15)	HPDMA_CxTR3	Res.		Res.	DAO[12:0]													Res.		Res.	SAO[12:0]												
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A8 + 0x080 * x (x=12 to 15)	HPDMA_CxBR2	BRDAO[15:0]																BRSAO[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0CC + 0x080 * x (x=0 to 11)	HPDMA_CxLLR	UT1	UT2	UB1	USA	UDA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ULL	LA[15:2]														Res.	Res.
	Reset value	0	0	0	0	0											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x0CC + 0x080 * x (x=12 to 15)	HPDMA_CxLLR	UT1	UT2	UB1	USA	UDA	UT3	UB2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ULL	LA[15:2]														Res.	Res.
	Reset value	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

19 General purpose direct memory access controller (GPDMA)

19.1 GPDMA introduction

The general purpose direct memory access (GPDMA) controller is a bus master and system peripheral.

The GPDMA is used to perform programmable data transfers between memory-mapped peripherals and/or memories via linked-lists, upon the control of an off-loaded CPU.

19.2 GPDMA main features

- Dual bidirectional AHB master
- Memory-mapped data transfers from a source to a destination:
  - Peripheral-to-memory
  - Memory-to-peripheral
  - Memory-to-memory
  - Peripheral-to-peripheral

  Transfer arbitration based on a 4-grade programmed priority at channel level:
  - One high-priority traffic class, for time-sensitive channels (queue 3)
  - Three low-priority traffic classes, with a weighted round-robin allocation for non time-sensitive channels (queues 0, 1, 2)
- Per channel event generation, on any of the following events: transfer complete, half-transfer complete, data transfer error, user setting error, link transfer error, completed suspension, and trigger overrun
- Per channel interrupt generation, with separately programmed interrupt enable per event
- 16 concurrent GPDMA channels:
  - Per channel FIFO for queuing source and destination transfers (see *Section 19.3.1*)
  - Intra-channel GPDMA transfers chaining via programmable linked-list into memory, supporting two execution modes: run-to-completion and link step mode
  - Intra-channel and inter-channel GPDMA transfers chaining via programmable GPDMA input triggers connection to GPDMA task completion events
- Per linked-list item within a channel:
  - Separately programmed source and destination transfers
  - Programmable data handling between source and destination: byte-based reordering, packing or unpacking, padding or truncation, sign extension and left/right realignment
  - Programmable number of data bytes to be transferred from the source, defining the block level
  - Linear source and destination addressing: either fixed or contiguously incremented addressing, programmed at a block level, between successive burst transfers

  - 2D source and destination addressing: programmable signed address offsets between successive burst transfers (non-contiguous addressing within a block, combined with programmable signed address offsets between successive blocks, at a second 2D/repeated block level, for a reduced set of channels (see *Section 19.3.1*)
  - Support for scatter-gather (multi-buffer transfers), data interleaving and deinterleaving via 2D addressing
  - Programmable GPDMA request and trigger selection
  - Programmable GPDMA half-transfer and transfer complete events generation
  - Pointer to the next linked-list item and its data structure in memory, with automatic update of the GPDMA linked-list control registers
  - Channel abort and restart
- Debug:
  - Channel suspend and resume support
  - Channel status reporting, including FIFO level, and event flags
- TrustZone support:
  - Support for secure and nonsecure GPDMA transfers, independently at a first channel level, and independently at a source/destination and link sublevels
  - Secure and nonsecure interrupts reporting, resulting from any of the respectively secure and nonsecure channels
  - TrustZone-aware AHB slave port, protecting any GPDMA secure resource (register, register field) from a nonsecure access
- Privileged/unprivileged support:
  - Support for privileged and unprivileged GPDMA transfers, independently at a channel level
  - Privileged-aware AHB slave port

19.3 GPDMA implementation

19.3.1 GPDMA channels

A given GPDMA channel x is implemented with the following features and intended usage. To make the best use of the GPDMA performances, the table below lists some general recommendations, allowing the user to select and allocate a channel, given its implemented FIFO size and the requested GPDMA transfer.

Table 96. GPDMA1 channel implementation

Channel x	Hardware parameters		Features
	dma_fifo_size[x]	dma_addressing[x]	
x = 0 to 11	2	0	Channel x (x = 0 to 11) is implemented with: – a FIFO of 8 bytes, 2 words – fixed/contiguously incremented addressing These channels must be typically allocated for GPDMA transfers between an APB or AHB peripheral and SRAM.
x = 12 to 15	4	1	Channel x (x = 12 to 15) is implemented with: – a FIFO of 32 bytes, 8 words – 2D addressing These channels may be also used for GPDMA transfers, between a demanding AHB peripheral and SRAM, or for transfers from/to external memories.

19.3.2 GPDMA in low-power modes

The GPDMA wake-up feature is implemented in the device low-power modes as per the table below.

Table 97. GPDMA1 wake-up in low-power modes

Feature	Low-power modes
Wake-up	GPDMA1 in Sleep mode

19.3.3 GPDMA requests

A GPDMA request from a peripheral can be assigned to a GPDMA channel x, via REQSEL[7:0] in GPDMA_CxTR2, provided that SWREQ = 0.

The GPDMA requests mapping is specified in the table below.

Table 98. Programmed GPDMA1 request

GPDMA_CxTR2.REQSEL[7:0]	Selected GPDMA request
0	jpeg_rx_dma
1	jpeg_tx_dma
2	xspi1_dma
3	xspi2_dma
4	xspi3_dma
5	fmc_txrx_dma
6	fmc_bch_dma
7	adc1_dma
8	adc2_dma

Table 98. Programmed GPDMA1 request (continued)

GPDMA_CxTR2.REQSEL[7:0]	Selected GPDMA request
9	cryp_in_dma
10	cryp_out_dma
11	saes_out_dma
12	saes_in_dma
13	hash_in_dma
14	tim1_cc1_dma
15	tim1_cc2_dma
16	tim1_cc3_dma
17	tim1_cc4_dma
18	tim1_upd_dma
19	tim1_trg_dma
20	tim1_com_dma
21	tim2_cc1_dma
22	tim2_cc2_dma
23	tim2_cc3_dma
24	tim2_cc4_dma
25	tim2_upd_dma
26	tim2_trg_dma
27	tim3_cc1_dma
28	tim3_cc2_dma
29	tim3_cc3_dma
30	tim3_cc4_dma
31	tim3_upd_dma
32	tim3_trg_dma
33	tim4_cc1_dma
34	tim4_cc2_dma
35	tim4_cc3_dma
36	tim4_cc4_dma
37	tim4_upd_dma
38	tim4_trg_dma
39	tim5_cc1_dma
40	tim5_cc2_dma
41	tim5_cc3_dma
42	tim5_cc4_dma
43	tim5_upd_dma

Table 98. Programmed GPDMA1 request (continued)

GPDMA_CxTR2.REQSEL[7:0]	Selected GPDMA request
44	tim5_trg_dma
45	tim6_upd_dma
46	tim7_upd_dma
47	tim8_cc1_dma
48	tim8_cc2_dma
49	tim8_cc3_dma
50	tim8_cc4_dma
51	tim8_upd_dma
52	tim8_trg_dma
53	tim8_com_dma
54	-
55	-
56	tim15_cc1_dma
57	tim15_cc2_dma
58	tim15_upd_dma
59	tim15_trg_dma
60	tim15_com_dma
61	tim16_cc1_dma
62	tim16_upd_dma
63	tim16_com_dma
64	tim17_cc1_dma
65	tim17_upd_dma
66	tim17_com_dma
67	tim18_cc1_dma
68	tim18_upd_dma
69	tim18_com_dma
70	lptim1_ic1_dma
71	lptim1_ic2_dma
72	lptim1_ue_dma
73	lptim2_ic1_dma
74	lptim2_ic2_dma
75	lptim2_ue_dma
76	lptim3_ic1_dma
77	lptim3_ic2_dma
78	lptim3_ue_dma

Table 98. Programmed GPDMA1 request (continued)

GPDMA_CxTR2.REQSEL[7:0]	Selected GPDMA request
79	spi1_rx_dma
80	spi1_tx_dma
81	spi2_rx_dma
82	spi2_tx_dma
83	spi3_rx_dma
84	spi3_tx_dma
85	spi4_rx_dma
86	spi4_tx_dma
87	spi5_rx_dma
88	spi5_tx_dma
89	spi6_rx_dma
90	spi6_tx_dma
91	sai1_a_dma
92	sai1_b_dma
93	sai2_a_dma
94	sai2_b_dma
95	i2c1_rx_dma
96	i2c1_tx_dma
97	i2c2_rx_dma
98	i2c2_tx_dma
99	i2c3_rx_dma
100	i2c3_tx_dma
101	i2c4_rx_dma
102	i2c4_tx_dma
103	i3c1_rx_dma
104	i3c1_tx_dma
105	i3c2_rx_dma
106	i3c2_tx_dma
107	usart1_rx_dma
108	usart1_tx_dma
109	usart2_rx_dma
110	usart2_tx_dma
111	usart3_rx_dma
112	usart3_tx_dma
113	uart4_rx_dma

Table 98. Programmed GPDMA1 request (continued)

GPDMA_CxTR2.REQSEL[7:0]	Selected GPDMA request
114	uart4_tx_dma
115	uart5_rx_dma
116	uart5_tx_dma
117	usart6_rx_dma
118	usart6_tx_dma
119	uart7_rx_dma
120	uart7_tx_dma
121	uart8_rx_dma
122	uart8_tx_dma
123	uart9_rx_dma
124	uart9_tx_dma
125	usart10_rx_dma
126	usart10_tx_dma
127	lpuart1_rx_dma
128	lpuart1_tx_dma
129	spdifrx_cs_dma
130	spdifrx_dt_dma
131	adf1_flt0_dma
132	mdf1_flt0_dma
133	mdf1_flt1_dma
134	mdf1_flt2_dma
135	mdf1_flt3_dma
136	mdf1_flt4_dma
137	mdf1_flt5_dma
138	ucpd1_tx_dma
139	ucpd1_rx_dma
140	dcmi_dma or pssi_dma(1)
141	i3c1_tc_dma
142	i3c1_rs_dma
143	i3c2_tc_dma
144	i3c2_rs_dma

1. Depends on which exclusive function is used.

19.3.4 GPDMA block requests

Some GPDMA requests must be programmed as a block request, and not as a burst request. Then BREQ in GPDMA_CxTR2 must be set for a correct GPDMA execution of the requested peripheral transfer at the hardware level.

Table 99. Programmed GPDMA1 request as a block request

GPDMA block requests
lptim1_ue_dma
lptim2_ue_dma
lptim3_ue_dma

19.3.5 GPDMA channels with peripheral early termination

A GPDMA channel, if implemented with this feature, can support the early termination of the data transfer from the peripheral which does also support this feature.

Table 100. GPDMA1 channel with peripheral early termination

GPDMA channel x with peripheral early termination
x = 0, x = 1, and x = 15

This GPDMA support is activated when the channel x is programmed with GPDMA_CxTR2.PFREQ = 1. Then, the peripheral itself can initiate and request a data transfer completion, before the GPDMA has transferred the whole block (see *Section 19.4.14* for more details).

Table 101. Programmed GPDMA1 request with peripheral early termination

Programmed GPDMA channel x request with peripheral early termination
i3c1_rx_dma
i3c2_rx_dma
jpeg_tx_dma

19.3.6 GPDMA triggers

A GPDMA trigger can be assigned to a GPDMA channel x, via TRIGSEL[6:0] in GPDMA_CxTR2, provided that TRIGPOL[1:0] defines a rising or a falling edge of the selected trigger (TRIGPOL[1:0] = 01 or TRIGPOL[1:0] = 10).

Table 102. Programmed GPDMA1 trigger

GPDMA_CxTR2.TRIGSEL[7:0]	Selected GPDMA trigger
0	dcmipp_p1_frameend_evt
1	dcmipp_p1_lineend_evt
2	dcmipp_p1_hsync_evt
3	dcmipp_p1_vsync_evt

Table 102. Programmed GPDMA1 trigger (continued)

GPDMA_CxTR2.TRIGSEL[7:0]	Selected GPDMA trigger
4	dcmipp_p1_frameend_evt
5	dcmipp_p1_lineend_evt
6	dcmipp_p1_hsync_evt
7	dcmipp_p1_vsync_evt
8	dcmipp_p2_frameend_evt
9	dcmipp_p2_lineend_evt
10	dcmipp_p2_hsync_evt
11	dcmipp_p2_vsync_evt
12	dma2d_ctc_flag
13	dma2d_tc_flag
14	dma2d_tw_flag
15	jpeg_eoc_flag
16	jpeg_ifnf_flag
17	jpeg_ift_flag
18	jpeg_ofne_flag
19	jpeg_oft_flag
20	lcd_li_flag
21	gpu2d1_gp_flag[0]
22	gpu2d1_gp_flag[1]
23	gpu2d1_gp_flag[2]
24	gpu2d1_gp_flag[3]
25	gfxtim1_0_gfxtim_evt[3]
26	gfxtim1_0_gfxtim_evt[2]
27	gfxtim1_0_gfxtim_evt[1]
28	gfxtim1_0_gfxtim_evt[0]
29	-
30	lptim1_ch1
31	lptim1_ch2
32	lptim2_ch1
33	lptim2_ch2
34	lptim3_ch1
35	lptim3_ch2
36	lptim4_out
37	lptim5_out
38	-

Table 102. Programmed GPDMA1 trigger (continued)

GPDMA_CxTR2.TRIGSEL[7:0]	Selected GPDMA trigger
39	rtc_wkup
40	lpuart1_it_r_wup_async
41	lpuart1_it_t_wup_async
42	spi6_it_or_spi6_ait_sync
43	-
44	tim1_trgo_cktim
45	tim1_trgo2_cktim
46	tim2_trgo_cktim
47	tim3_trgo_cktim
48	tim4_trgo_cktim
49	tim5_trgo_cktim
50	tim6_trgo_cktim
51	tim7_trgo_cktim
52	tim8_trgo_cktim
53	tim8_trgo2_cktim
54	-
55	-
56	-
57	tim12_trgo_cktim
58	tim15_trgo_cktim
59	-
60	hpdma1_ch0_tc
61	hpdma1_ch1_tc
62	hpdma1_ch2_tc
63	hpdma1_ch3_tc
64	hpdma1_ch4_tc
65	hpdma1_ch5_tc
66	hpdma1_ch6_tc
67	hpdma1_ch7_tc
68	hpdma1_ch8_tc
69	hpdma1_ch9_tc
70	hpdma1_ch10_tc
71	hpdma1_ch11_tc
72	hpdma1_ch12_tc
73	hpdma1_ch13_tc

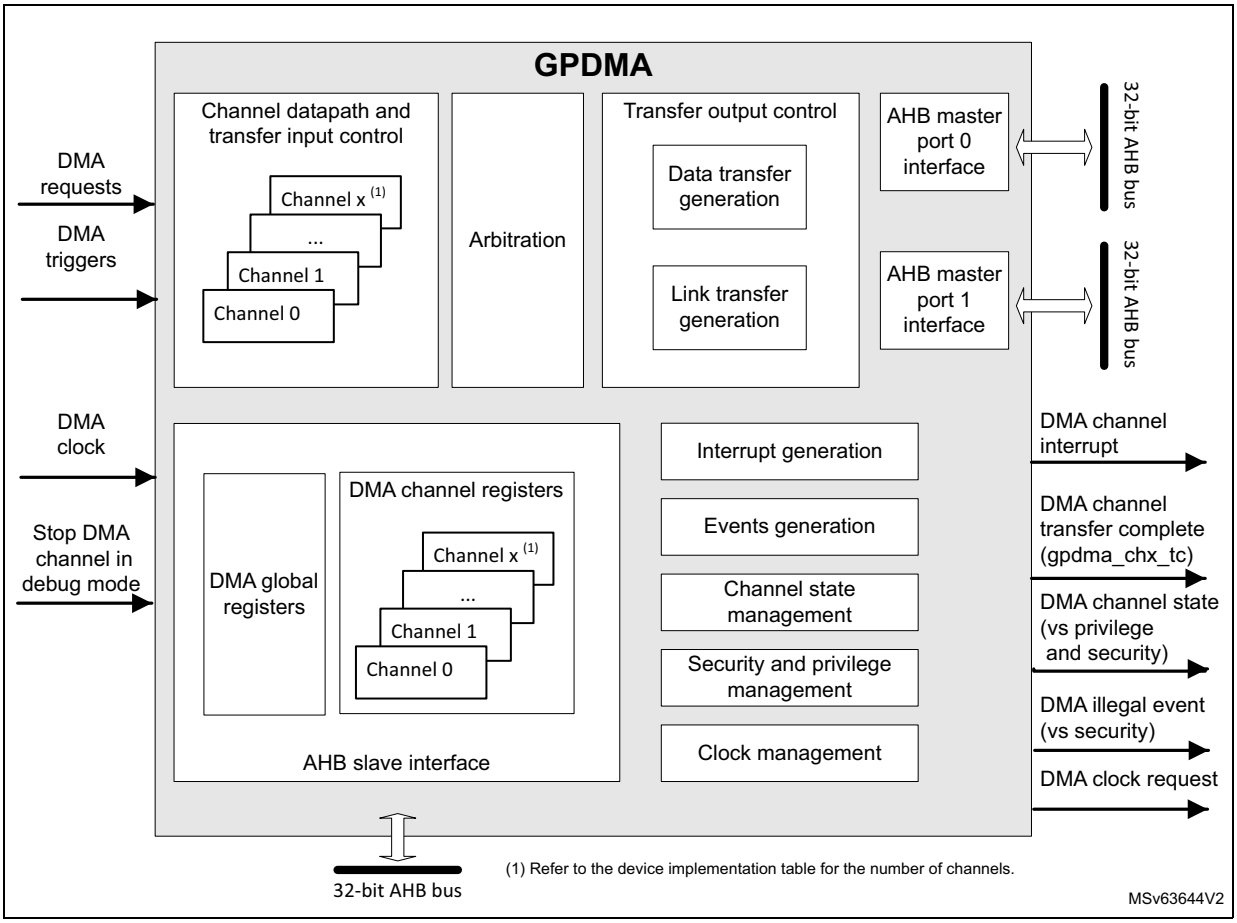
Table 102. Programmed GPDMA1 trigger (continued)

GPDMA_CxTR2.TRIGSEL[7:0]	Selected GPDMA trigger
74	hpdma1_ch14_tc
75	hpdma1_ch15_tc
76	gpdma1_ch0_tc
77	gpdma1_ch1_tc
78	gpdma1_ch2_tc
79	gpdma1_ch3_tc
80	gpdma1_ch4_tc
81	gpdma1_ch5_tc
82	gpdma1_ch6_tc
83	gpdma1_ch7_tc
84	gpdma1_ch8_tc
85	gpdma1_ch9_tc
86	gpdma1_ch10_tc
87	gpdma1_ch11_tc
88	gpdma1_ch12_tc
89	gpdma1_ch13_tc
90	gpdma1_ch14_tc
91	gpdma1_ch15_tc
92	-
93	extit0_sync
94	extit1_sync
95	extit2_sync
96	extit3_sync
97	extit4_sync
98	extit5_sync
99	extit6_sync
100	extit7_sync
101	extit8_sync
102	extit9_sync
103	extit10_sync
104	extit11_sync
105	extit12_sync
106	extit13_sync
107	extit14_sync
108	extit15_sync

19.4 GPDMA functional description

19.4.1 GPDMA block diagram

Figure 93. GPDMA block diagram



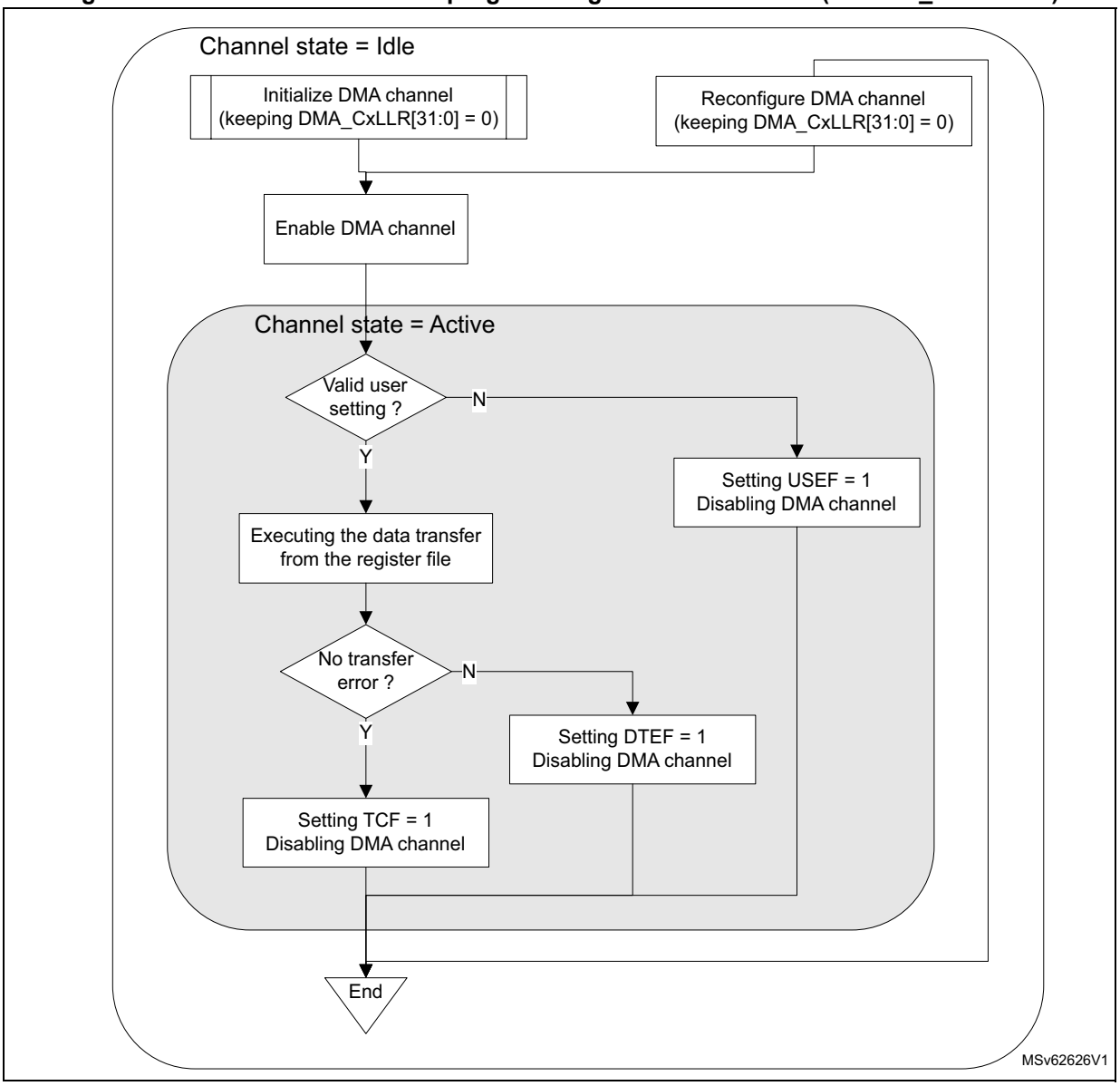
19.4.2 GPDMA channel state and direct programming without any linked-list

After a GPDMA reset, a GPDMA channel x is in idle state. When the software writes 1 in GPDMA_CxCR.EN, the channel takes into account the value of the different channel configuration registers (GPDMA_CxXXX), switches to the active/non-idle state and starts to execute the corresponding requested data transfers.

After enabling/starting a GPDMA channel transfer by writing 1 in GPDMA_CxCR.EN, a GPDMA channel interrupt on a complete transfer notifies the software that the GPDMA channel is back in idle state (EN is then deasserted by hardware) and that the channel is ready to be reconfigured then enabled again.

Figure 94 illustrates this GPDMA direct programming without any linked-list (GPDMA_CxLLR = 0).

Figure 94. GPDMA channel direct programming without linked-list (GPDMA_CxLLR = 0)



19.4.3 GPDMA channel suspend and resume

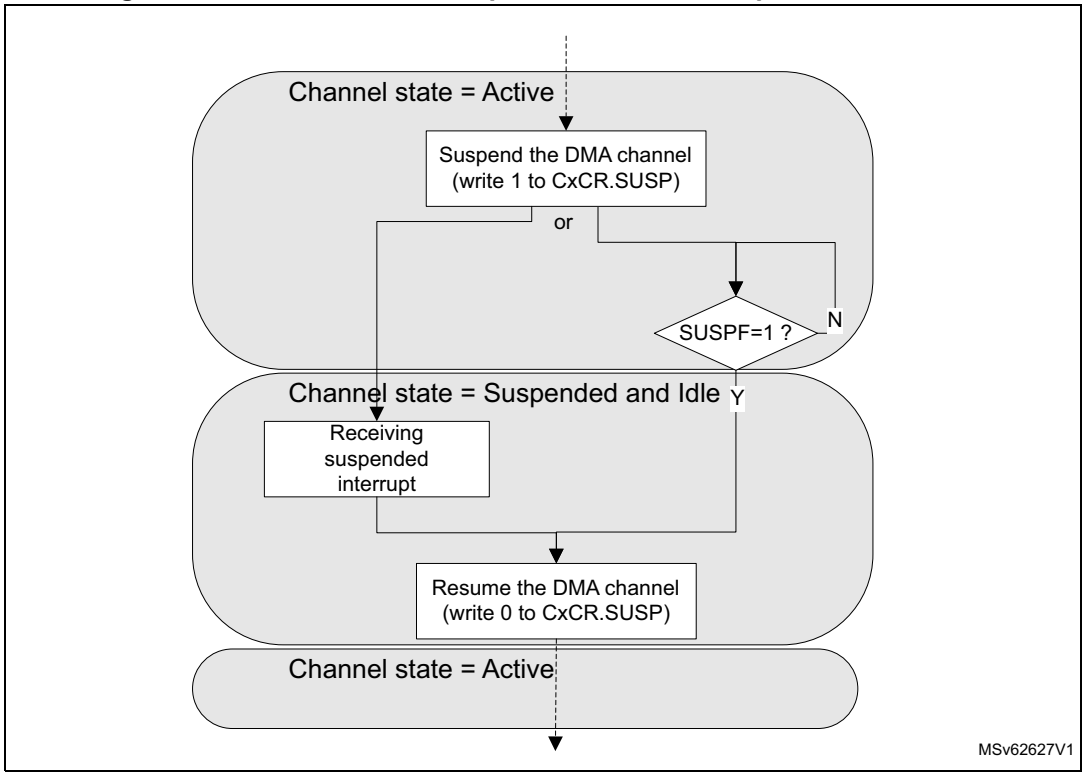
The software can suspend on its own a channel still active, with the following sequence:

1. The software writes 1 into the GPDMA_CxCR.SUSP bit.
2. The software polls the suspended flag GPDMA_CxSR.SUSPF until SUSPF = 1, or waits for an interrupt previously enabled by writing 1 to GPDMA_CxCR.SUSPIE. Wait for the channel to be effectively in suspended state means wait for the completion of any ongoing GPDMA transfer over its master ports. Then the software can observe, in a steady state, any read register or register field that is hardware modifiable.

Note: An ongoing GPDMA transfer can be a data transfer (a source/destination burst transfer) or a link transfer for the internal update of the linked-list register file from the next linked-list item.

3. The software safely resumes the suspended channel by writing 0 to GPDMA_CxCR.SUSP.

Figure 95. GPDMA channel suspend and resume sequence



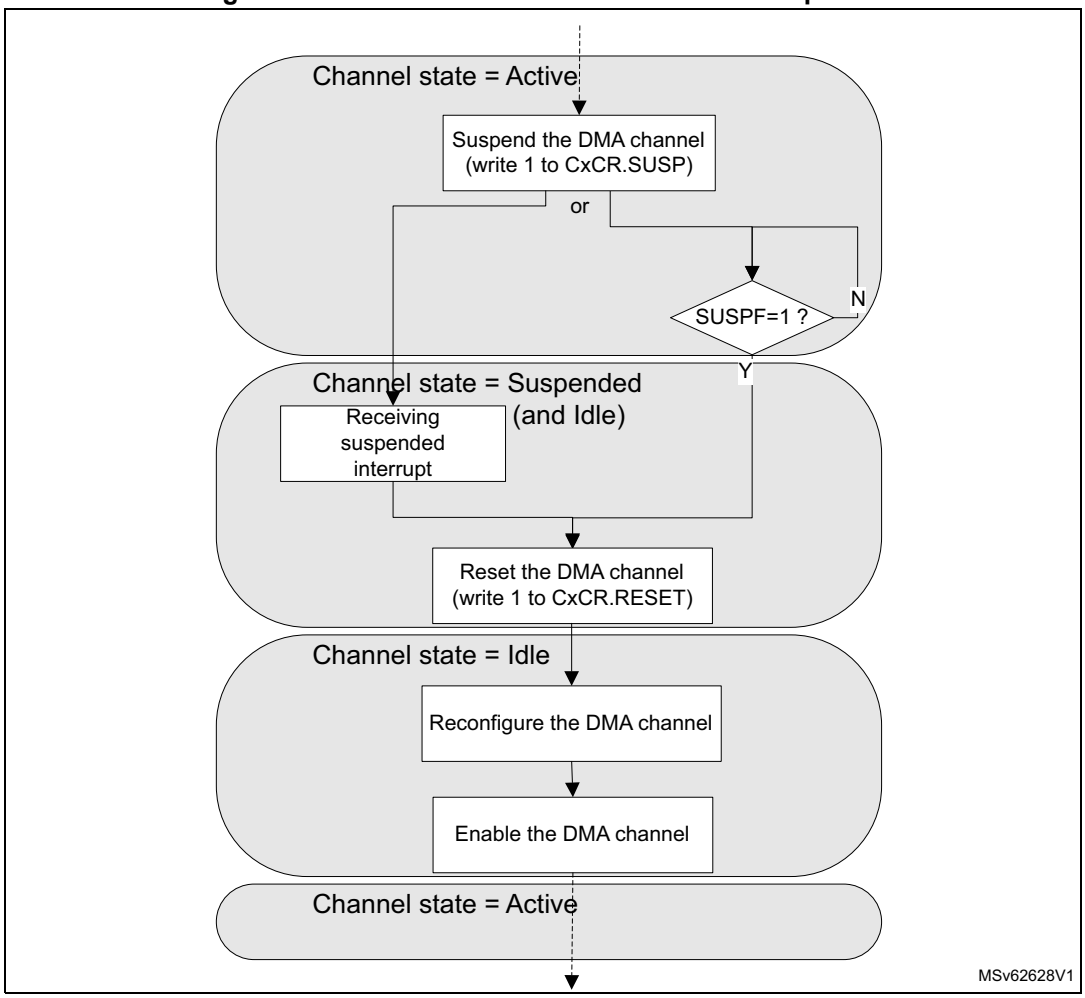
Note: A suspend and resume sequence does not impact the GPDMA_CxCR.EN bit. Suspending a channel (transfer) does not suspend a started trigger detection.

19.4.4 GPDMA channel abort and restart

Alternatively, like for aborting a continuous GPDMA transfer with a circular buffering or a double buffering, the software can abort, on its own, a still active channel with the following sequence:

1. The software writes 1 into the GPDMA_CxCR.SUSP bit.
2. The software polls suspended flag GPDMA_CxSR.SUSPF until SUSPF = 1, or waits for an interrupt previously enabled by writing 1 to GPDMA_CxCR.SUSPIE. Wait for the channel to be effectively in suspended state means wait for the completion of any ongoing GPDMA transfer over its master port.
3. The software resets the channel by writing 1 to GPDMA_CxCR.RESET. This causes the reset of the FIFO, the reset of the channel internal state, the reset of the GPDMA_CxCR.EN bit, and the reset of the GPDMA_CxCR.SUSP bit.
4. The software safely reconfigures the channel. The software must reprogram the hardware-modified GPDMA_CxBR1, GPDMA_CxSAR, and GPDMA_CxDAR registers.
5. In order to restart the aborted then reprogrammed channel, the software enables it again by writing 1 to the GPDMA_CxCR.EN bit.

Figure 96. GPDMA channel abort and restart sequence



19.4.5 GPDMA linked-list data structure

Alternatively to the direct programming mode, a channel can be programmed by a list of transfers, known as a list of linked-list items (LLI). Each LLI is defined by its data structure.

The base address in memory of the data structure of a next LLI_{n+1} of a channel x is the sum of the following:

- the link base address of the channel x (in GPDMA_CxLBAR)
- the link address offset (LA[15:2] field in GPDMA_CxLLR). The linked-list register GPDMA_CxLLR is the updated result from the data structure of the previous LLIn of the channel x.

The data structure for each LLI may be specific.

A linked-list data structure is addressed following the value of the following UT1, UT2, UB1, USA, UDA, UT3 and UB2 if present, and ULL bits in GPDMA_CxLLR.

In linked-list mode, each GPDMA linked-list register (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR, GPDMA_CxTR3 and GPDMA_CxBR2 if present, and GPDMA_CxLLR) is conditionally and automatically updated from the next linked-list data structure in the memory, following the current value of the

GPDMA_CxLLR register that was conditionally updated from the linked-list data structure of the previous LLI.

Caution: The user must program the pointer to the next linked-list data structure (GPDMA_CxLLR[15:0]) not to exceed the 64-Kbyte addressable space defined by the link base address register (GPDMA_CxLBAR). The complete linked-list data structure must be included in the 64-Kbyte addressable space.

Static linked-list data structure

For example, when the update bits (UT1, UT2, UB1, USA, UDA, plus UB2 and UT3 if present, and ULL) in GPDMA_CxLLR are all asserted, the linked-list data structure in the memory is maximal with:

- channel x (x = 0 to 11) contiguous 32-bit locations, including GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR and GPDMA_CxLLR (see *Figure 97*) and including the first linked-list register file (LLI_0) and the next LLIs (such as LLI_1, LLI_2) in the memory
- channel x (x = 12 to 15), contiguous 32-bit locations, including GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR, GPDMA_CxTR3, GPDMA_CxBR2 and GPDMA_CxLLR, (see *Figure 98*), and including the first linked-list register file (LLI_0) and the next LLIs (such as LLI_1, LLI_2) in the memory

Figure 97. Static linked-list data structure (all Uxx = 1) of a linear addressing channel x

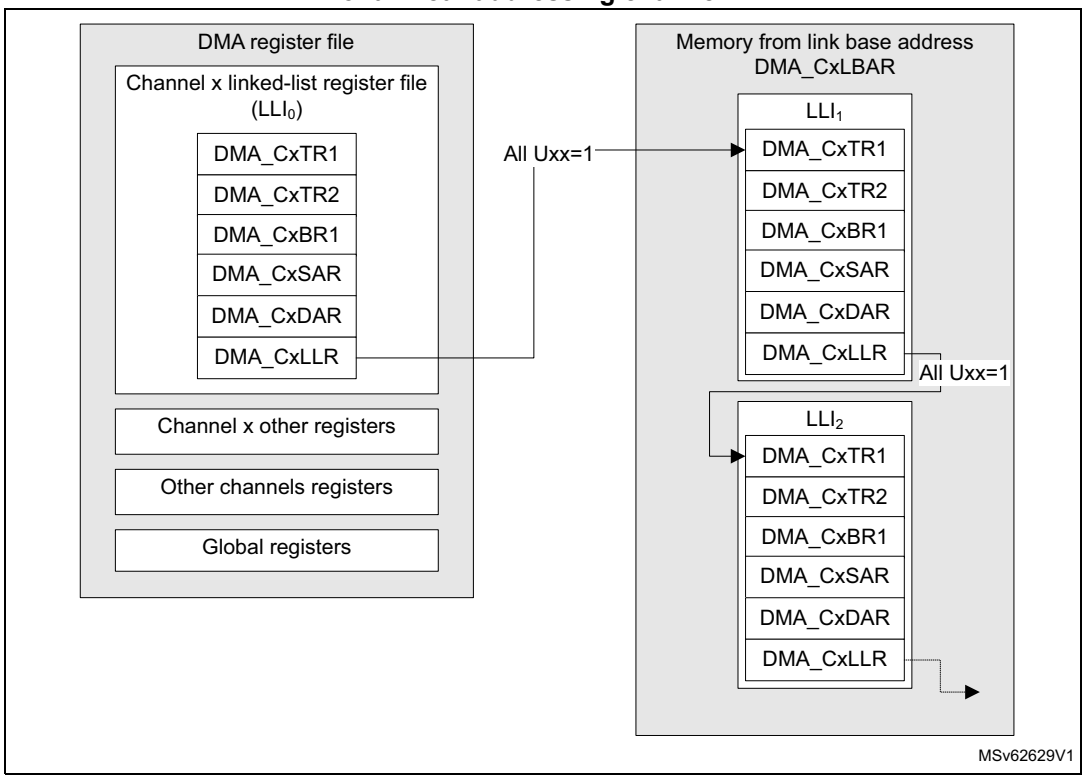
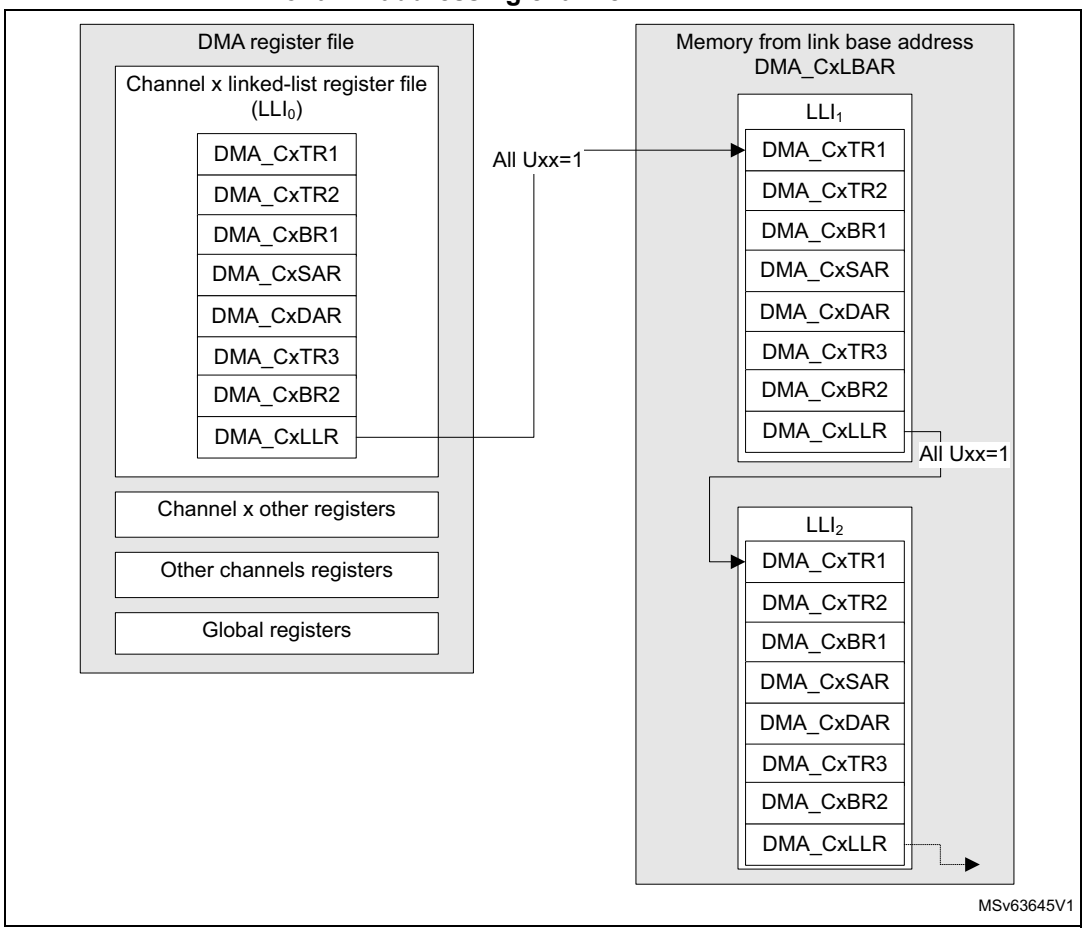


Figure 98. Static linked-list data structure (all Uxx = 1) of a 2D addressing channel x



Dynamic linked-list data structure

Alternatively, the memory organization for the full list of LLIs can be compacted with specific data structure for each LLI.

If UT1 = 0 and UT2 = 1, the link address offset of the register GPDMA_CxLLR is pointing to the updated value of the GPDMA_CxTR2 instead of the GPDMA_CxTR1 which is not to be modified (see *Figure 99*).

Example: if UT1 = UB1 = USA = 0 and if UT3 = UB2 = 0, when channel x is with 2D addressing, and if UT2 = UDA = ULL = 1, the next LLI does not contain an (updated) value for GPDMA_CxTR1, nor GPDMA_CxBR1, nor GPDMA_CxSAR, nor GPDMA_CxTR3, nor GPDMA_CxBR2 when channel x is with 2D addressing. The next LLI contains an updated value for GPDMA_CxTR2, GPDMA_CxDAR, and GPDMA_CxLLR, as shown in *Figure 100*.

Figure 99. GPDMA dynamic linked-list data structure of a linear addressing channel x

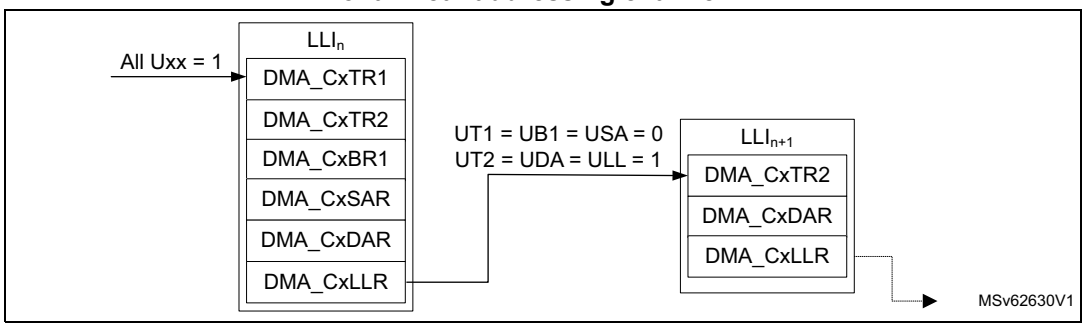
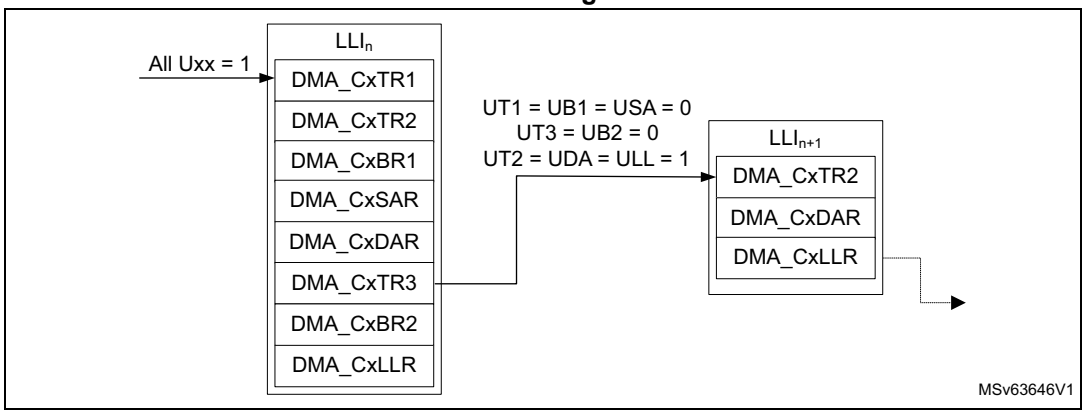


Figure 100. GPDMA dynamic linked-list data structure of a 2D addressing channel x



19.4.6 Linked-list item transfer execution

A LLI_n transfer is the sequence of:

1. a data transfer: GPDMA executes the data transfer as described by the GPDMA internal register file (this data transfer can be void/null for LLI_0)
2. a conditional link transfer: GPDMA automatically and conditionally updates its internal register file by the data structure of the next LLI_{n+1}, as defined by the GPDMA_CxLLR value of the LLI_n.

Note: The initial data transfer as defined by the internal register file (LLI_0) can be null (GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxTR2.PFREQ = 0) provided that the conditional update bit UB1 in GPDMA_CxLLR is set (meaning there is a non-null data transfer described by the next LLI_1 in the memory to be executed).

Depending on the intended GPDMA usage, a GPDMA channel x can be executed as described by the full linked-list (run-to-completion mode, GPDMA_CxCR.LSM = 0) or a GPDMA channel x can be programmed for a single execution of a LLI (link step mode, GPDMA_CxCR.LSM = 1), as described in the next sections.

19.4.7 GPDMA channel state and linked-list programming in run-to-completion mode

When GPDMA_CxCR.LSM = 0 (in full list execution mode, execution of the full sequence of LLIs, named run-to-completion mode), a GPDMA channel x is initially programmed, started

by writing 1 to GPDMA_CxCR.EN, and after completed at channel level. The channel transfer is:

- configured with at least the following:
  - the first LLI_0, internal linked-list register file: GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR, and GPDMA_CxLLR, plus GPDMA_CxTR3 and GPDMA_CxBR2
  - the last LLI_N, described by the linked-list data structure in memory, as defined by the GPDMA_CxLLR reflecting the before last LLI_{N-1}
- completed when GPDMA_CxLLR[31:0] = 0, GPDMA_CxBR1.BRC[10:0] = 0, and GPDMA_CxBR1.BNDT[15:0] = 0, at the end of the last LLI_{N-1} transfer

GPDMA_CxLLR[31:0] = 0 is the condition of a linked-list based channel completion and means the following:

- The 16 low significant bits GPDMA_CxLLR.LA[15:0] of the next link address are null.
- All the update bits GPDMA_CxLLR.Uxx are null (UT1, UT2, UB1, USA, UDA and ULL, plus UB2 and UT3).

The channel may never be completed when GPDMA_CxLLR.LSM = 0:

- If the last LLI_N is recursive, pointing to itself as a next LLI:
  - either GPDMA_CxLLR.ULL = 1 and GPDMA_CxLLR.LA[15:2] is updated with the same value
  - or GPDMA_CxLLR.ULL = 0
- If LLI_N is pointing to a previous LLI

In the regular data transfer completion at a block level, GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxBR1.BRC[10:0] = 0 (if present). Alternatively, a block transfer may be early completed by a peripheral (such as an I3C in Rx mode), and then BNDT[15:0] is not null (see *Section 19.4.14* for more details).

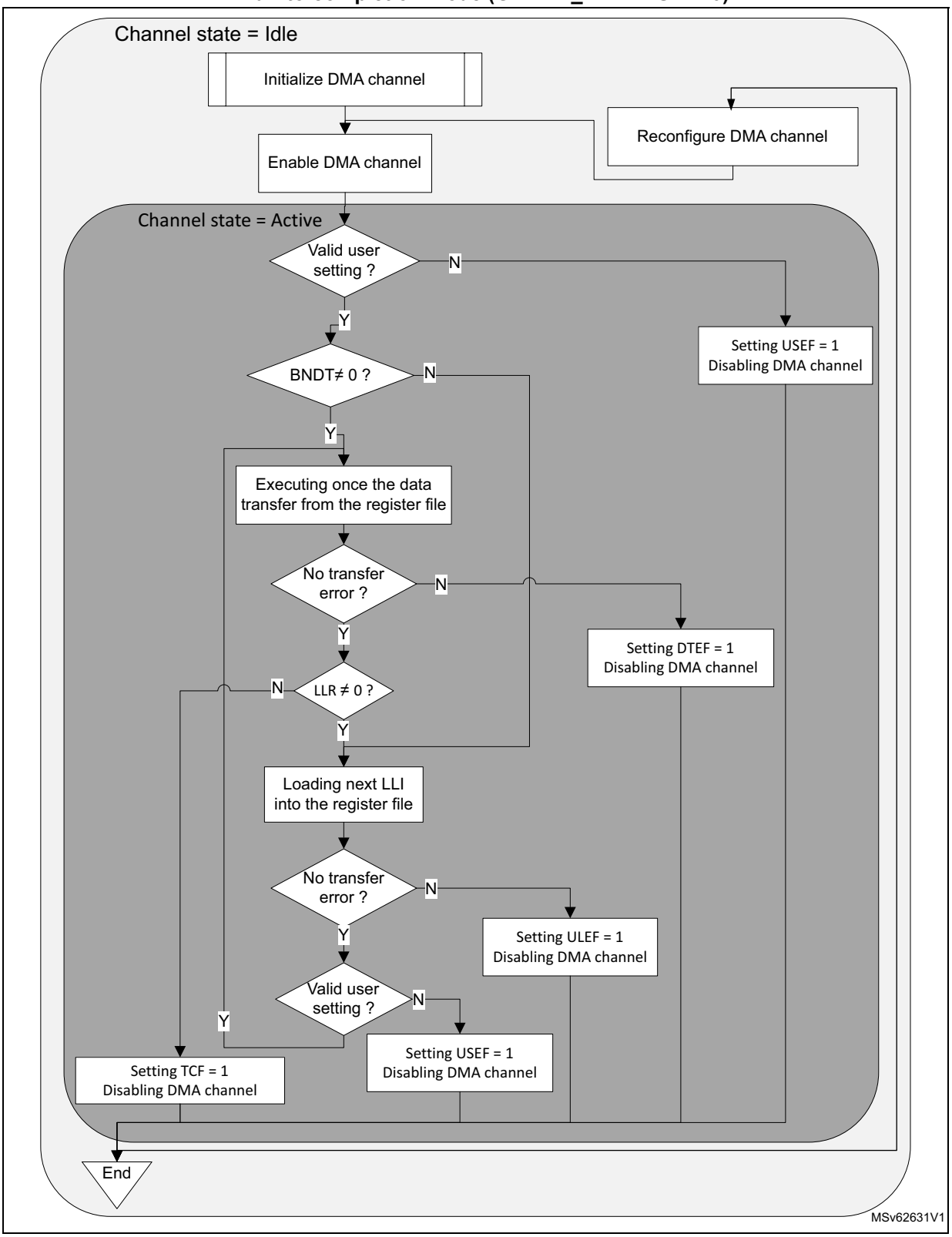
In the typical run-to-completion mode, the allocation of a GPDMA channel, including its fine programming, is done once during the GPDMA initialization. In order to have a reserved data communication link and GPDMA service during run-time, for continuously repeated transfers (from/to a peripheral respectively to/from memory or for memory-to-memory transfers). This reserved data communication link can consist of a channel, or the channel can be shared and a repeated transfer consists of a sequence of LLIs.

Figure 101 depicts the GPDMA channel execution and its registers programming in run-to-completion mode.

Note: Figure 101 is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in GPDMA_CxTR2. It can be raised at (each) block completion, at (each) 2D block completion, at (each) LLI completion, or only at channel completion. In run-to-completion mode, whatever is the value of TCEM[1:0], at the channel completion, the hardware always set TCEF = 1 and disables the channel.

In Figure 101, BNDT ≠ 0 is the typical condition for starting the first data transfer in this figure. This condition becomes (BNDT ≠ 0 and PFREQ = 1) if the peripheral requests a data transfer with early termination (see Section 19.3.5).

Figure 101. GPDMA channel execution and linked-list programming in run-to-completion mode (GPDMA_CxCR.LSM = 0)



Run-time inserting a LLI_n via an auxiliary channel, in run-to-completion mode

The start of the link transfer of the LLI_{n-1} (start of the LLI_n loading) can be conditioned by the occurrence of a trigger, when programming the following fields of the GPDMA_CxTR2 in the data structure of the LLI_{n-1}:

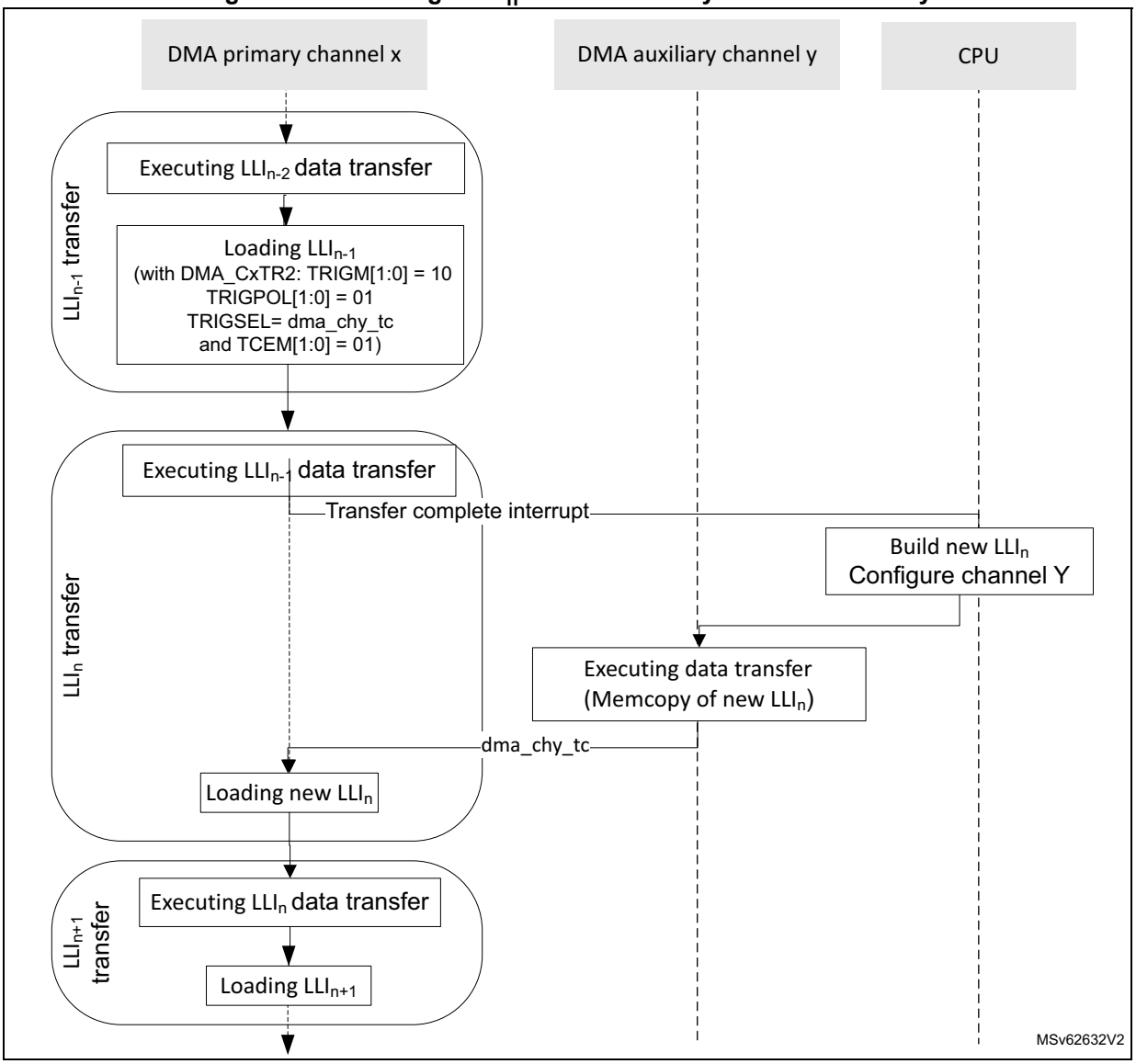
- TRIGM[1:0] = 10 (link transfer triggering mode)
- TRIGPOL[1:0] = 01 or 10 (rising or falling edge)
- TRIGSEL[6:0] (see *Section 19.3.6* for the trigger selection details)

Another auxiliary channel y can be used to store the channel x LLI_n in the memory and to generate a transfer complete event gpdma_chy_tc. By selecting this event as the input trigger of the link transfer of the LLI_{n-1} of the channel x, the software can pause the primary channel x after its LLI_{n-1} data transfer, until it is indeed written the LLI_n.

Figure 102 depicts such a dynamic elaboration of a linked-list of a primary channel x, via another auxiliary channel y.

Caution: This use case is restricted to an application with a LLI_{n-1} data transfer that does not need a trigger. The triggering mode of this LLI_{n-1} is used to load the next LLI_n.

Figure 102. Inserting a LLI_n with an auxiliary GPDMA channel y



19.4.8 GPDMA channel state and linked-list programming in link step mode

When GPDMA_CxCR.LSM = 1 (in link step execution mode, single execution of one LLI), a channel transfer is executed and completed after each single execution of a LLI, including its (conditional) data transfer and its (conditional) link transfer.

A GPDMA channel transfer can be programmed at LLI level, started by writing 1 into GPDMA_CxCR.EN, and after completed at LLI level:

- The current LLI_n transfer is described with:
  - GPDMA_CxTR1 defines the source/destination elementary single/burst transfers.
  - GPDMA_CxBR1 defines the number of bytes at a block level (BNDT[15:0]) and, for channel x (x = 12 to 15), the number of blocks at a 2D/repeated block level (BRC[10:0]+1) and the incrementing/decrementing mode for address offsets.

  - GPDMA_CxTR2 defines the input control (request, trigger) and the output control (transfer complete event) of the transfer.
  - GPDMA_CxSAR/GPDMA_CxDAR define the source/destination transfer start address.
  - GPDMA_CxTR3 for channel x (x = 12 to 15) defines the source/destination additional address offset between burst transfers.
  - GPDMA_CxBR2 for channel x (x = 12 to 15) defines the source/destination additional address offset between blocks at a 2D/repeated block level.
  - GPDMA_CxLLR defines the data structure and the address offset of the next LLI_{n+1} in the memory.
- The current LLI_n transfer is completed after the single execution of the current LLI_n:
  - after the (conditional) data transfer completion (when GPDMA_CxBR1.BRC[10:0] = 0, and GPDMA_CxBR1.BNDT[15:0] = 0
  - after the (conditional) update of the GPDMA link register file from the data structure of the next LLI_{n+1} in memory

Note: If a LLI is recursive (pointing to itself as a next LLI, either GPDMA_CxLLR.ULL = 1 and GPDMA_CxLLR.LA[15:2] is updated with the same value, or GPDMA_CxLLR.ULL = 0), a channel in link step mode is completed after each repeated single execution of this LLI.

In the regular data transfer completion at a block level, GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxBR1.BRC[10:0] = 0. Alternatively, a block transfer may be early completed by a peripheral (such as an I3C in Rx mode), and then BNDT[15:0] is not null (see *Section 19.4.14* for more details).

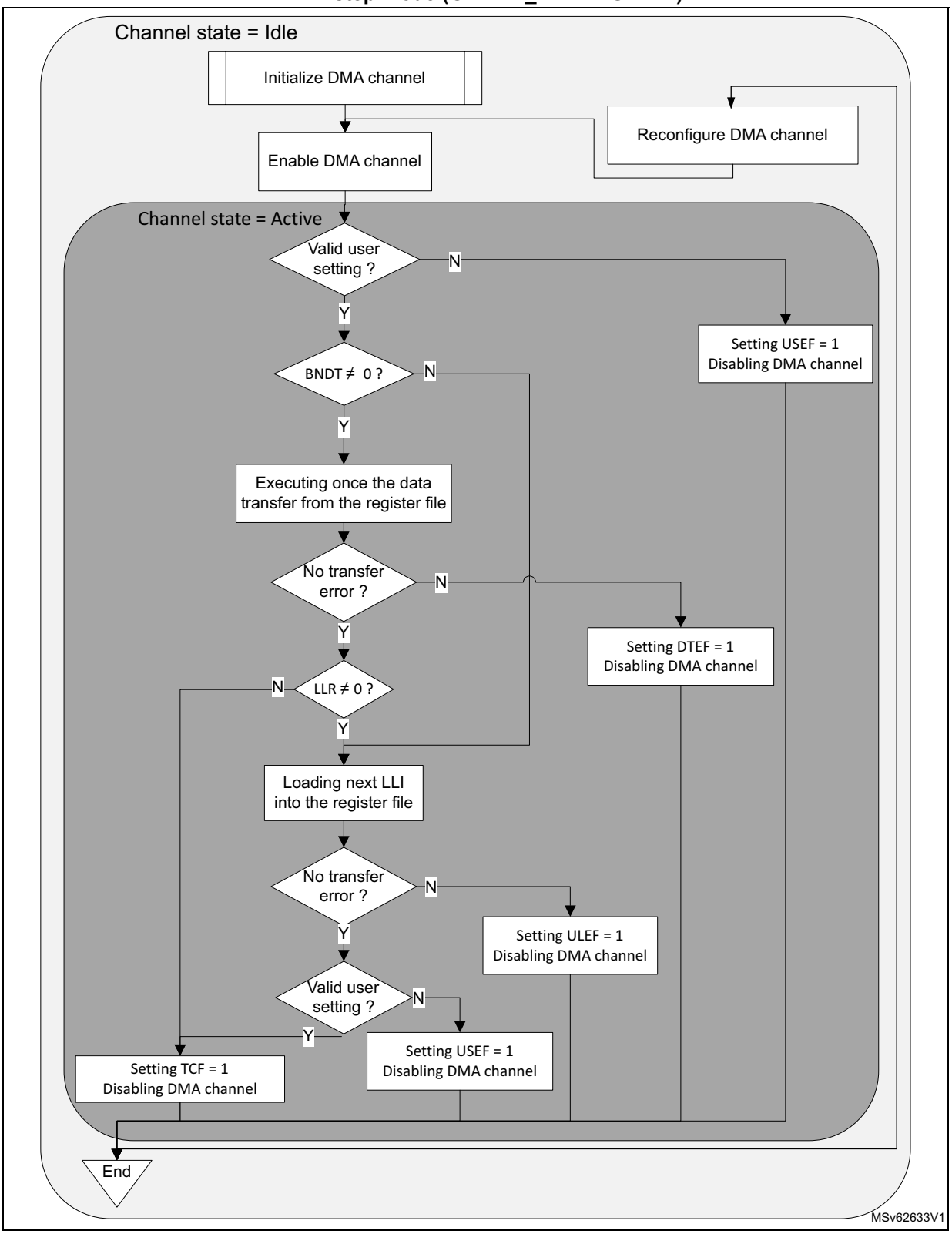
The link step mode can be used to elaborate dynamically LLIs in memory during run-time. The software can be facilitated by using a static data structure for any LLI_n (all update bits of GPDMA_CxLLR have a static value, LLI_n.LLR.LA = LLI_{n-1}.LLR.LA + constant).

Figure 103 depicts the GPDMA channel execution mode, and its programming in link step mode.

Note: Figure 103 is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in GPDMA_CxTR2. It can be raised at (each) block completion, at (each) 2D block completion, at (each) LLI completion, or only at the last LLI data transfer completion. In link step mode, the channel is disabled after each single execution of a LLI, and depending on the value of TCEM[1:0] a TCEF is raised or not.

In Figure 103, BNDT ≠ 0 is the typical condition for starting the first data transfer. This condition becomes (BNDT ≠ 0 and PFREQ = 1) if the peripheral requests a data transfer with early termination (see Section 19.3.5).

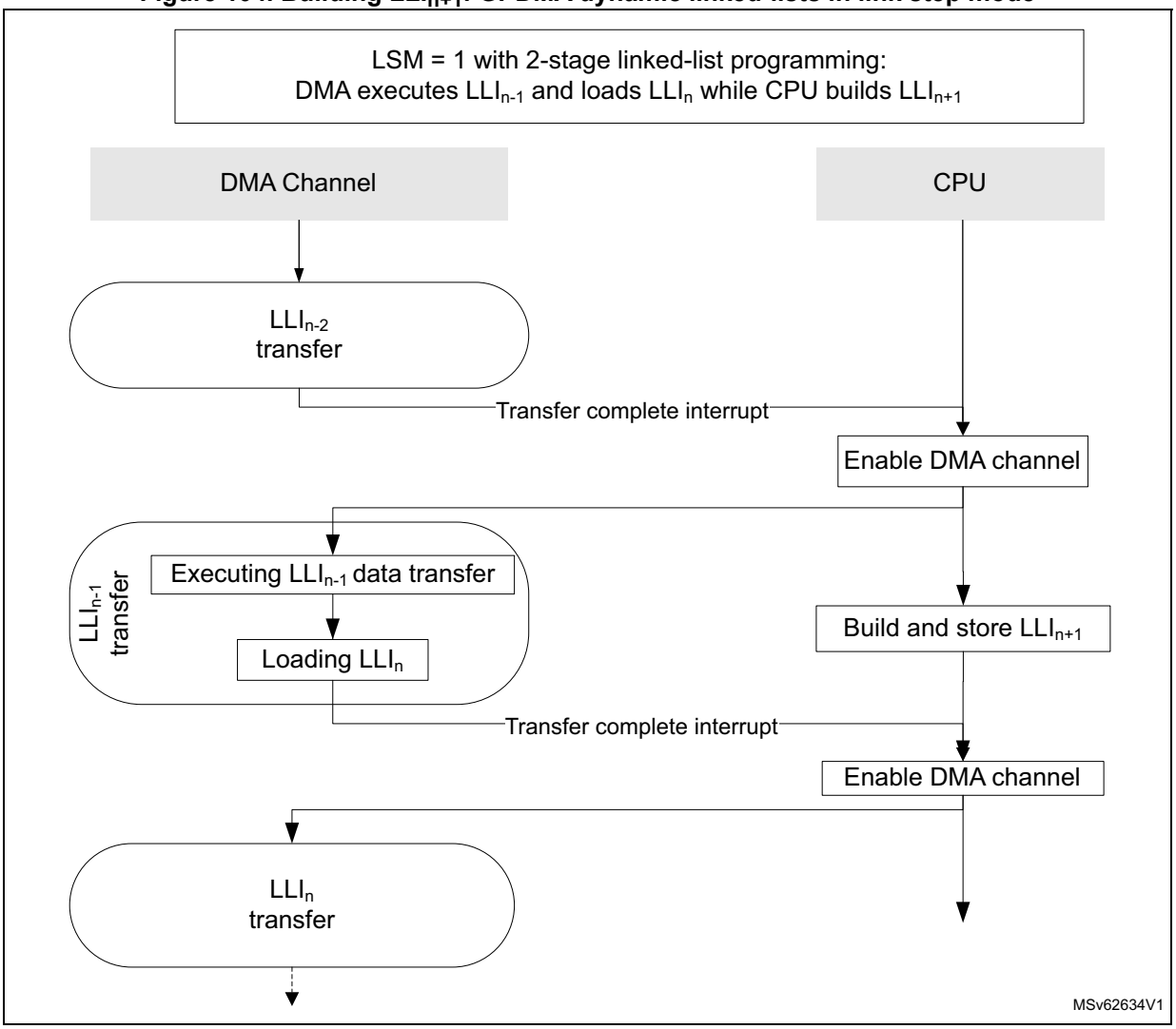
Figure 103. GPDMA channel execution and linked-list programming in link step mode (GPDMA_CxCR.LSM = 1)



Run-time adding a LLI_{n+1} in link step mode

During run-time, the software can defer the elaboration of the LLI_{n+1} (and next LLIs), until/after GPDMA executed the transfer from the LLI_{n-1} and loaded the LLI_n from the memory, as shown in *Figure 104*.

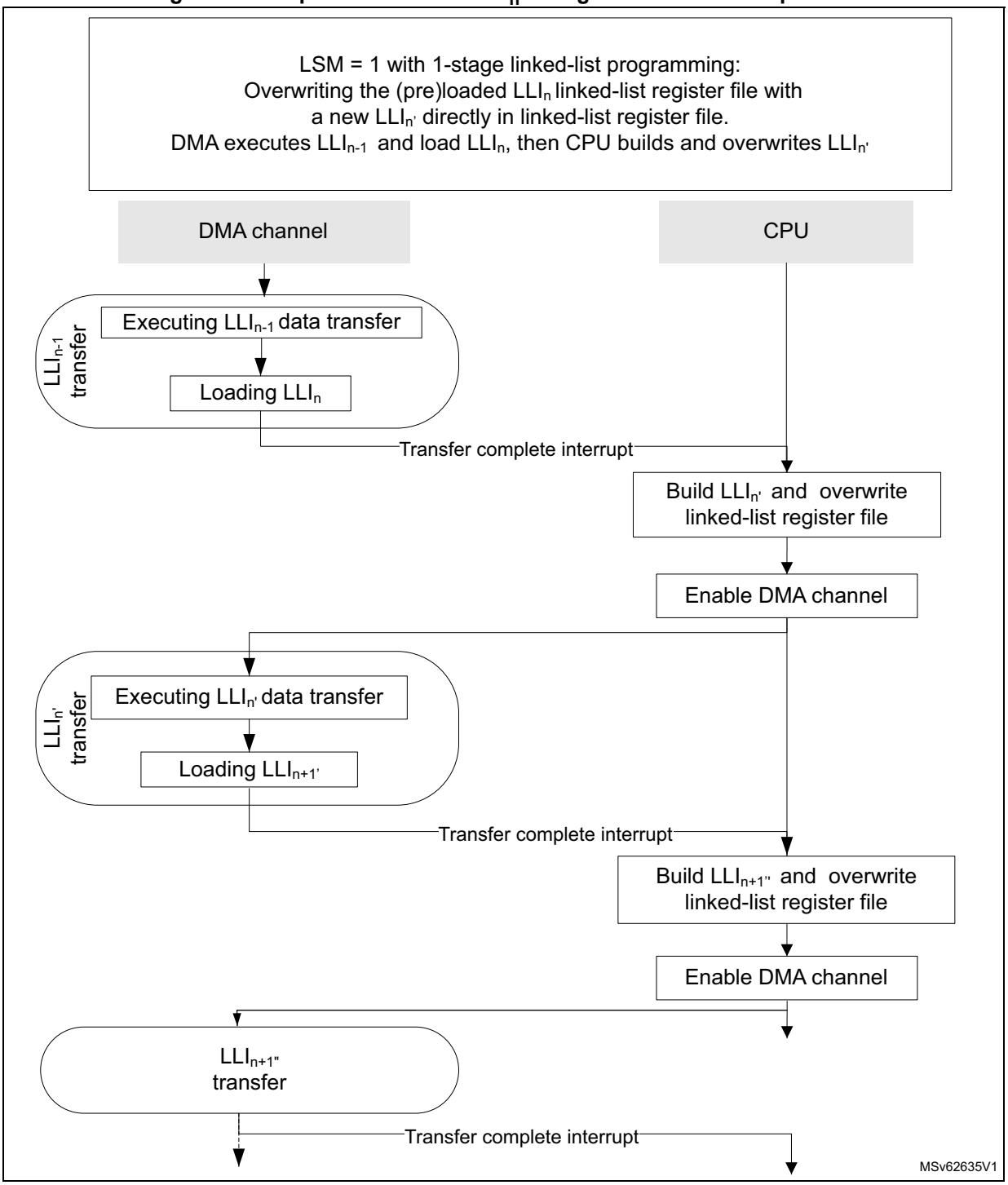
Figure 104. Building LLI_{n+1}: GPDMA dynamic linked-lists in link step mode



Run-time replacing a LLI_n with a new LLI_n', in link step mode (in linked-list register file)

In this link step mode, during run-time, the software can build and insert a new LLI_n', after GPDMA executed the transfer from the LLI_{n-1} and loaded a formerly elaborated LLI_n from the memory by overwriting directly the linked-list register file with the new LLI_n', as shown in *Figure 105*.

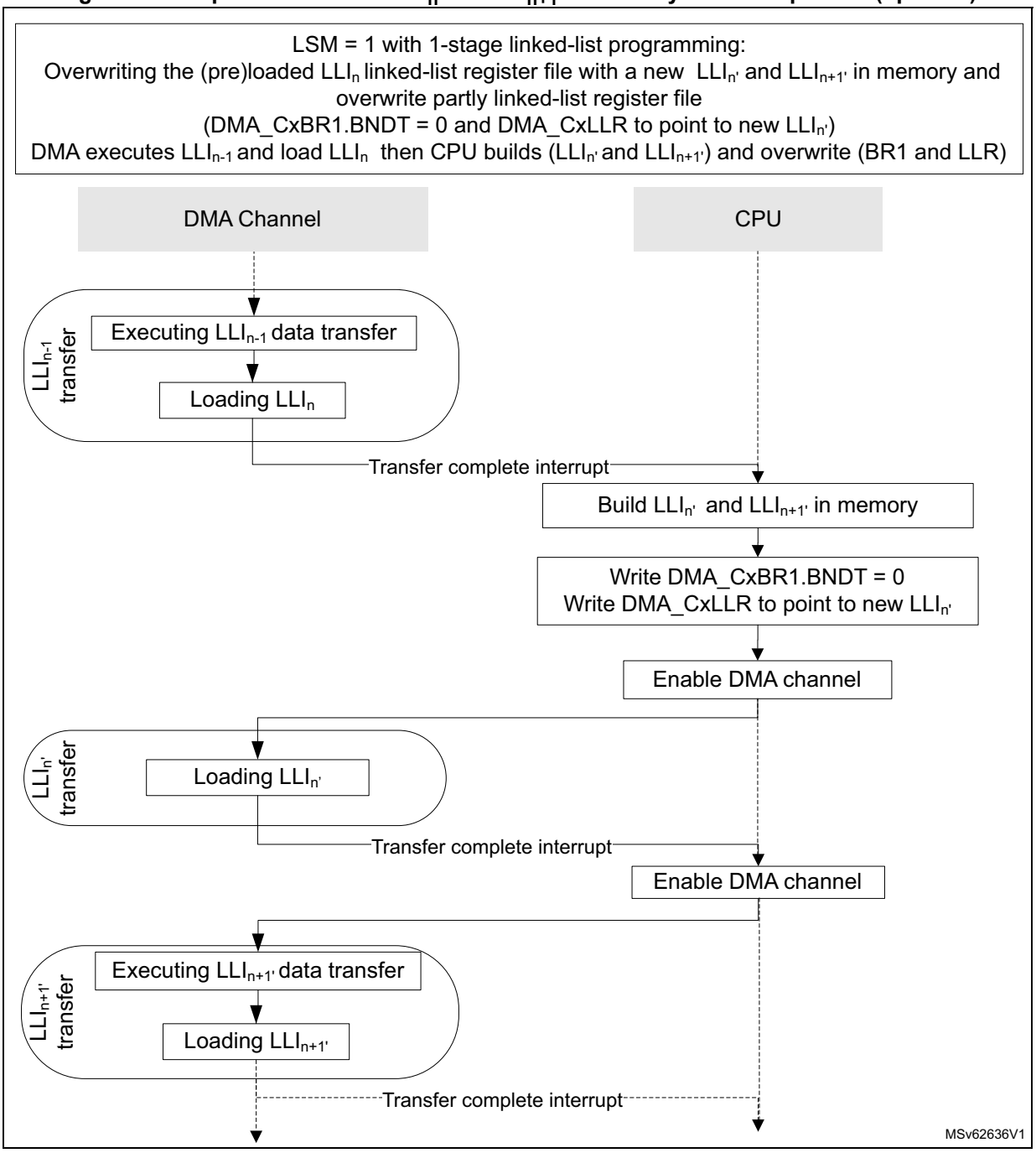
Figure 105. Replace with a new $LLI_{n'}$ in register file in link step mode



Run-time replacing a LLI_n with a new $LLI_{n'}$ in link step mode (in the memory)

The software can build and insert a new $LLI_{n'}$ and $LLI_{n+1'}$ in the memory, after GPDMA executed the transfer from the LLI_{n-1} and loaded a formerly elaborated LLI_n from the memory, by overwriting partly the linked-list register file (GPDMA_CxBR1.BNDT[15:0] to be null and GPDMA_CxLLR to point to new $LLI_{n')}$ as shown in *Figure 106*.

Figure 106. Replace with a new $LLI_{n'}$ and $LLI_{n+1'}$ in memory in link step mode (option 1)



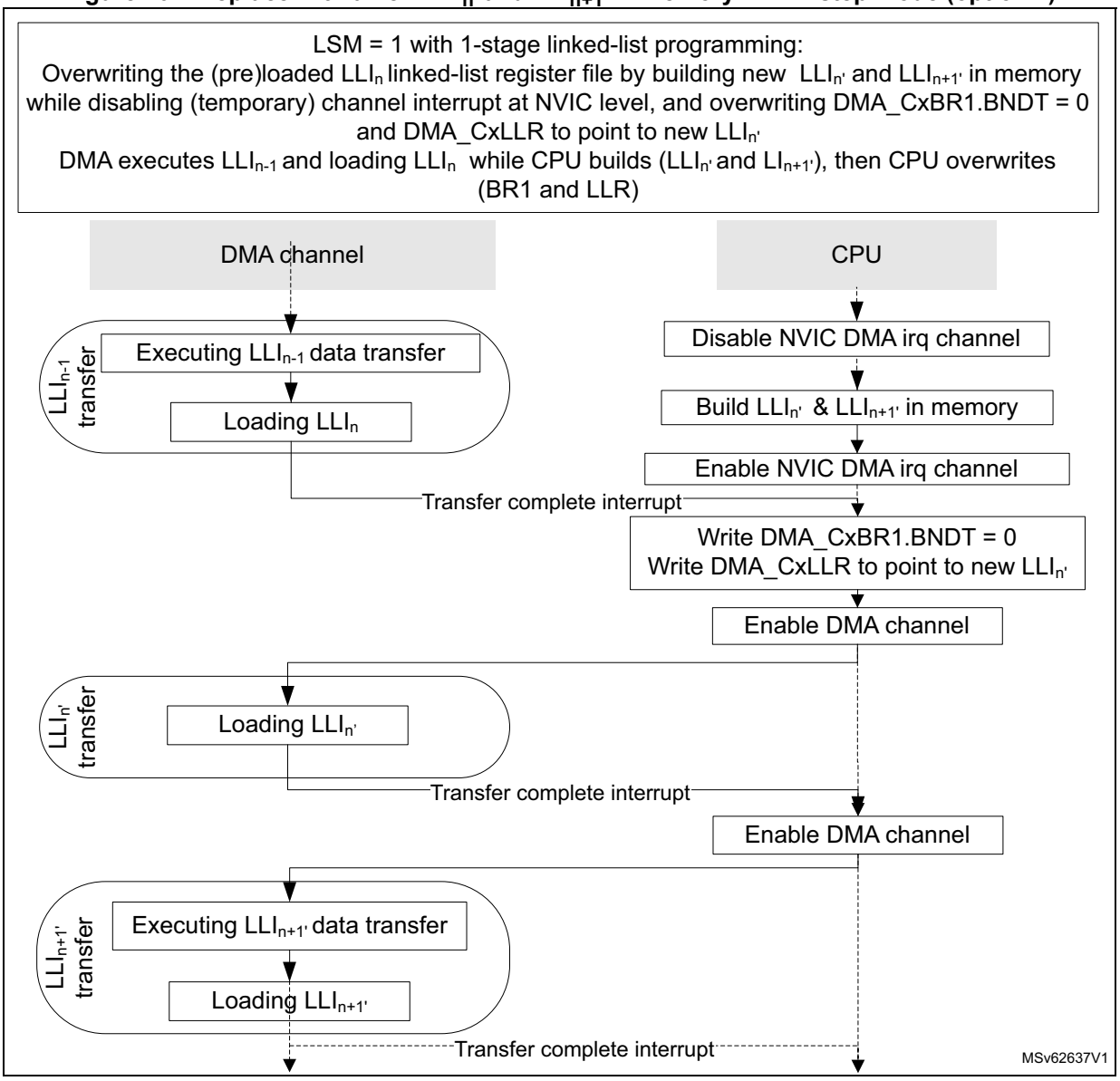
Run-time replacing a LLI_n with a new $LLI_{n'}$ in link step mode

Other software implementations exist. Meanwhile GPDMA executes the transfer from the LLI_{n-1} and loads a formerly elaborated LLI_n from the memory (or even earlier), the software can do the following:

1. Disable the NVIC for not being interrupted by the interrupt handling.
2. Build a new $LLI_{n'}$ and a new $LLI_{n+1'}$.
3. Enable again the NVIC for the channel interrupt (transfer complete) notification.

The software in the interrupt handler for LLI_{n-1} is then restricted to overwrite GPDMA_CxBR1.BNDT[15:0] to be null and GPDMA_CxLLR to point to new $LLI_{n'}$, as shown in *Figure 107*.

Figure 107. Replace with a new $LLI_{n'}$ and $LLI_{n+1'}$ in memory in link step mode (option 2)



19.4.9 GPDMA channel state and linked-list programming

The software can reconfigure a channel when the channel is disabled (GPDMA_CxCR.EN = 0) and update the execution mode (GPDMA_CxCR.LSM) to change from/to run-to-completion mode to/from link step mode.

In any execution mode, the software can:

- reprogram LLI_{n+1} in the memory to finally complete the channel by this LLI_{n+1} (clear the GPDMA_CxLLR of this LLI_{n+1}), before this LLI_{n+1} is loaded/used by the GPDMA channel
- abort and reconfigure the channel with a LSM update (see *Section 19.4.4*.)

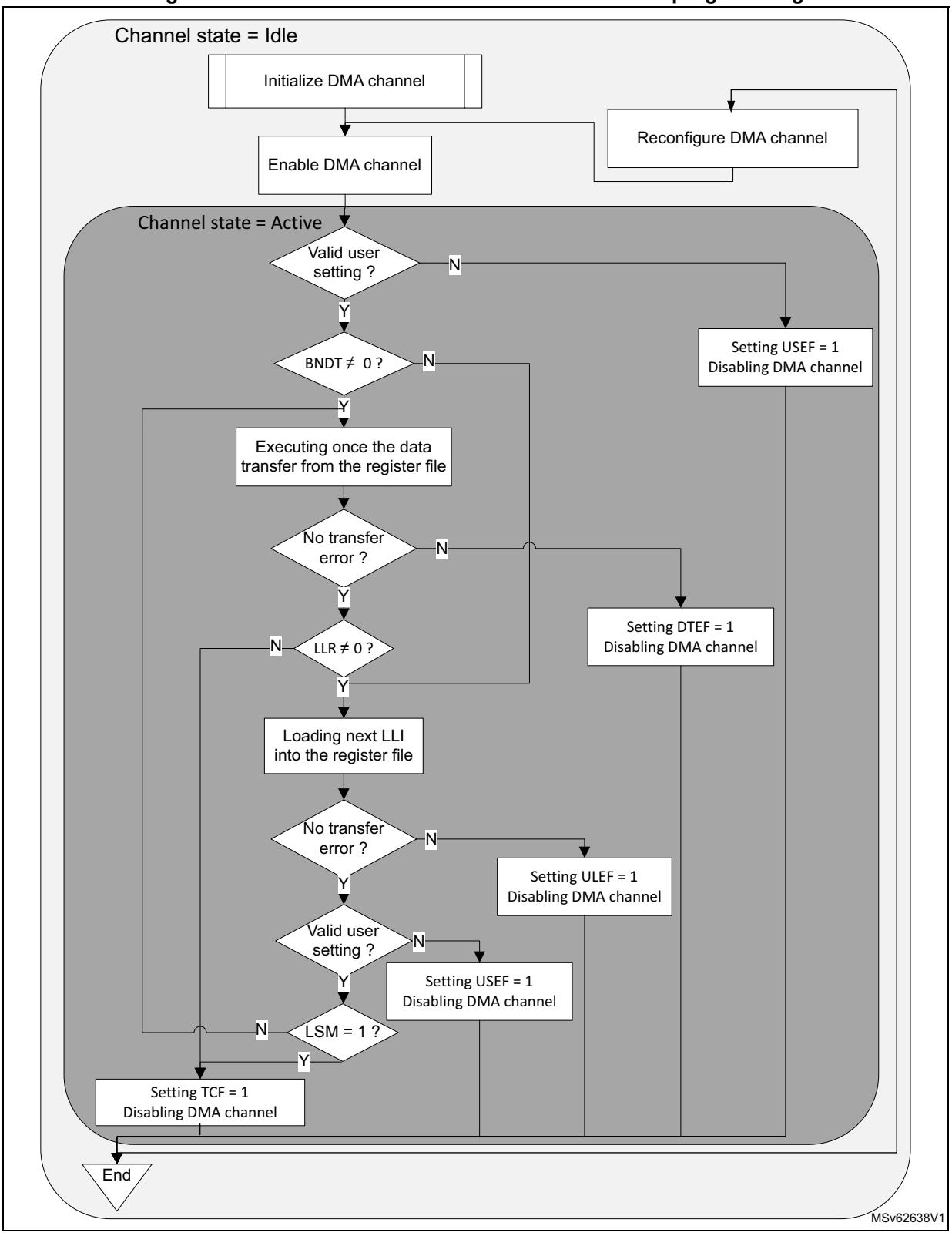
In link step mode, the software can clear LSM after each a single execution of any LLI, during LLI_{n-1}.

Figure 108 shows the overall and unified GPDMA linked-list programming, whatever is the execution mode.

Note: Figure 108 is not intended to illustrate how often a TCEF can be raised, depending on the programmed value of TCEM[1:0] in GPDMA_CxTR2. It can be raised at (each) block completion, at (each) 2D block completion, at (each) LLI completion, or only at the last LLI data transfer completion. In run-to-completion mode, whatever is the value of TCEM[1:0], at the channel completion the hardware always set TCEF = 1 and disables the channel. In link step mode, the channel is disabled after each single execution of a LLI, and depending on the value of TCEM[1:0] a TCEF is raised or not.

In Figure 108, BNDT ≠ 0 is the typical condition for starting the first data transfer. This condition becomes (BNDT ≠ 0 and PFREQ = 1) if the peripheral requests a data transfer with early termination (see Section 19.3.5).

Figure 108. GPDMA channel execution and linked-list programming



19.4.10 GPDMA FIFO-based transfers

There is a single transfer operation mode: the FIFO mode. There are FIFO-based transfers. Any channel x is implemented with a dedicated FIFO whose size is defined by dma_fifo_size[x] (see *Section 19.3.1* for more details).

GPDMA burst

A programmed transfer at the lowest level is a GPDMA burst.

A GPDMA burst is a burst of data received from the source, or a burst of data sent to the destination. A source (and destination) burst is programmed with a burst length by the field SBL_1[5:0] (respectively DBL_1[5:0]), and with a data width defined by the field SDW_LOG2[1:0] (respectively DDW_LOG2[1:0]) in the GPDMA_CxTR1 register.

The addressing mode after each data (named beat) of a GPDMA burst is defined by SINC and DINC in GPDMA_CxTR1, for source and destination respectively: either a fixed addressing or an incremented addressing with contiguous data.

The start and next addresses of a GPDMA source/destination burst (defined by GPDMA_CxSAR and GPDMA_CxDAR) must be aligned with the respective data width.

The table below lists the main characteristics of a GPDMA burst.

Table 103. Programmed GPDMA source/destination burst

SDW_LOG2[1:0] DDW_LOG2[1:0]	Data width (bytes)	SINC/DINC	SBL_1[5:0] DBL_1[5:0]	Burst length (data/beats)	Next data/ beat address	Next burst address	Burst address alignment
00	1	0 (fixed)	n = 0 to 63(1)	n+1	+ 0	+ 0	1
01	2						2
10	4						4
00	1	1 (contiguously incremented)			+ 1	+ (n + 1)	1
01	2				+ 2	+ 2 * (n + 1)	2
10	4				+ 4	+ 4 * (n + 1)	4
11	forbidden user setting, causing USEF generation and none burst to be issued.						

1. When S/DBL_1[5:0] = 0, burst is of length 1. Then burst can be also named as single.

The next burst address in the above table is the next source/destination default address pointed by GPDMA_CxSAR or GPDMA_CxDAR, once the programmed source/destination burst is completed. This default value refers to the fixed/contiguously incremented address.

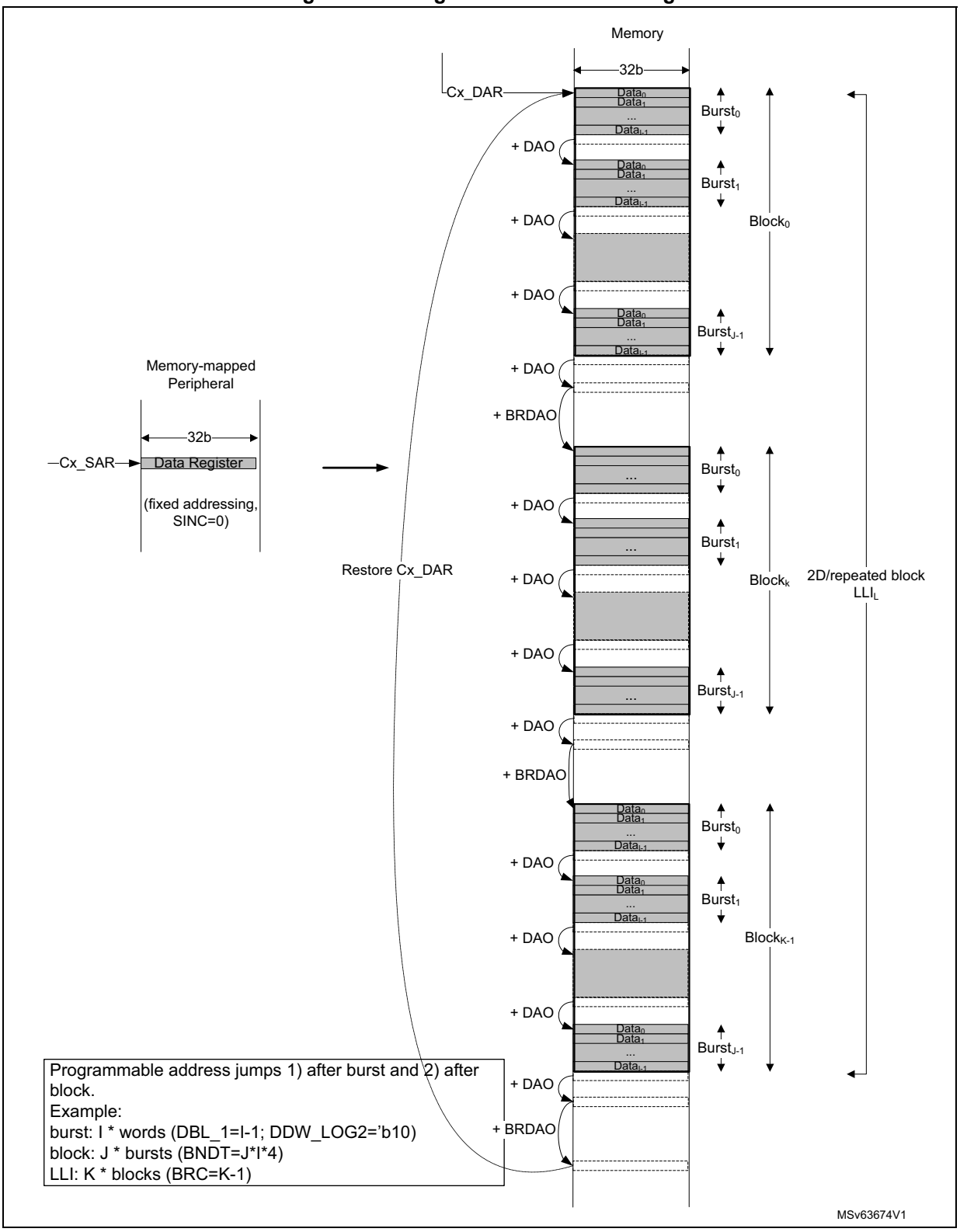
GPDMA burst with 2D addressing (channel x = 12 to 15)

When the channel has additional 2D addressing feature, this default value refers to the value without taking into account the two programmed incremented or decremented offsets. These two additional offsets (with a null default value) are applied:

- after each completed source/destination burst, as defined respectively by GPDMA_CxTR2.SAO[12:0]/DAO[12:0] and GPDMA_CxBR1.SDEC/DDEC
- after each completed block, as defined respectively by GPDMA_CxBR2.BRSAO[15:0]/BRDAO[15:0] and GPDMA_CxBR1.BRSDEC/BRDDEC)

Then, a 2D/repeated block can be addressed with a first programmed address jump after each completed burst, and with a second programmed address jump after each block, as depicted by *Figure 109* with a 2D destination buffer.

Figure 109. Programmed 2D addressing



GPDMA FIFO-based burst

In FIFO-mode, a transfer generally consists of two pipelined and separated burst transfers:

- one burst from the source to the FIFO over the allocated source master port, as defined by GPDMA_CxTR1.SAP
- one burst from the FIFO to the destination over the allocated destination master port, as defined by GPDMA_CxTR1.DAP

GPDMA source burst

The requested source burst transfer to the FIFO can be scheduled as early as possible over the allocated port, depending on the current FIFO level versus the programmed burst size (when the FIFO is ready to get one new burst from the source):

when FIFO level ≤ $2^{dma_fifo_size[x]}$ - (SBL_1[5:0]+1) * $2^{SDW_LOG2[1:0]}$

where:

- FIFO level is the current filling level of the FIFO, in bytes.
- $2^{dma_fifo_size[x]}$ is the half the FIFO size of channel x, in bytes (see *Section 19.3.1* for the implementation details and dma_fifo_size[x] value).
- (SBL_1[5:0]+1) * $2^{SDW_LOG2[1:0]}$ is the size of the programmed source burst transfer, in bytes.

Based on the channel priority (GPDMA_CxCR.PRIO[1:0]), this ready FIFO-based source transfer is internally arbitrated versus the other requested and active channels.

GPDMA destination burst

The requested destination burst transfer from the FIFO can be scheduled as early as possible over the allocated port, depending on the current FIFO level versus the programmed burst size (when the FIFO is ready to push one new burst to the destination):

when FIFO level ≥ (DBL_1[5:0]+1) * $2^{DDW_LOG2[1:0]}$

where:

- FIFO level is the current filling level of the FIFO, in bytes.
- (DBL_1[5:0]+1) * $2^{DDW_LOG2[1:0]}$ is the size of the programmed destination burst transfer, in bytes.

Based on the channel priority, this ready FIFO-based destination transfer is internally arbitrated versus the other requested and active channels.

GPDMA burst vs source block size, 1-Kbyte address boundary and FIFO size

The programmed source/destination GPDMA burst is implemented with an AHB burst as is, unless one of the following conditions is met:

- When half of the FIFO size of the channel x is lower than the programmed source/destination burst size, the programmed source/destination GPDMA burst is implemented with a series of singles or bursts of a lower size. Each transfer is of a size that is lower than or equal to half of the FIFO size, without any user constraint.
- If the source block size (GPDMA_CxBR1.BNDT[15:0]) is not a multiple of the source burst size but is a multiple of the data width of the source burst (GPDMA_CxTR1.SDW_LOG2[1:0]), the GPDMA modifies and shortens bursts into singles or bursts of lower length, in order to transfer exactly the source block size, without any user constraint.

- if the source/destination burst transfer have crossed the 1-Kbyte address boundary on a AHB transfer, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB protocol, without any user constraint.
- If the source/destination burst length exceeds 16 on a AHB transfer, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB protocol, without any user constraint.

In any case, the GPDMA keeps ensuring source/destination data (and address) integrity without any user constraint. The current FIFO level (software readable in GPDMA_CxSR) is compared to and updated with the effective transfer size, and the GPDMA re-arbitrates between each AHB single or burst transfer, possibly modified.

Based on the channel priority, each single or burst of a lower burst size versus the programmed burst, is internally arbitrated versus the other requested and active channels.

Note: In linked-list mode, the GPDMA read transfers related to the update of the linked-list parameters from the memory to the internal GPDMA registers, are scheduled over the link allocated port, as programmed by GPDMA_CxCR.LAP.

GPDMA data handling: byte-based reordering, packing/unpacking, padding/truncation, sign extension and left/right alignment

The data handling is controlled by GPDMA_CxTR1. The source/destination data width of the programmed burst is byte, half-word or word, as per the SDW_LOG2[1:0] and DDW_LOG2[1:0] fields (see *Table 104*).

The user can configure the data handling between transferred data from the source and transfer to the destination. More specifically, programmed data handling is orderly performed with:

1. Byte-based source reordering
   - If SBX = 1 and if source data width is a word, the two bytes of the unaligned half-word at the middle of each source data word are exchanged.
2. Data width conversion by packing, unpacking, padding or truncation, if destination data width is different than the source data width, depending on PAM[1:0]:
   - If destination data width > source data width, the post SBX source data is either right-aligned and padded with 0 s, or sign extended up to the destination data width, or is FIFO queued and packed up to the destination data width.
   - If destination data width < source data width, the post SBX data is either right-aligned and left-truncated down to the destination data width, or is FIFO queued and unpacked and streamed down to the destination data width.
3. Byte-based destination re-ordering:
   - If DBX = 1 and if the destination data width is not a byte, the two bytes are exchanged within the aligned post PAM[1:0] half-words.
   - If DHX = 1 and if the destination data width is neither a byte nor a half-word, the two aligned half-words are exchanged within the aligned post PAM[1:0] words.

Note: Left-alignment with 0s-padding can be achieved by programming both a right-alignment with a 0s-padding and a destination byte-based re-ordering.

The table below lists the possible data handling from the source to the destination.

Table 104. Programmed data handling

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	Destination data stream(1)
00	Byte	$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$	x	00	Byte	xx	x	x	$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$
				01	Half-word	00 (RA, 0P)	0		$0B_3,0B_2,0B_1,0B_0$
							1		B_30,B_20,B_10,B_00
						01 (RA, SE)	0		SB_3,SB_2,SB_1,SB_0
							1		B_3S,B_2S,B_1S,B_0S
						1x (PACK)	0		$B_7B_6,B_5B_4,B_3B_2,B_1B_0$
							1		$B_6B_7,B_4B_5,B_2B_3,B_0B_1$
				10	Word	00 (RA, 0P)	0	0	$000B_1,000B_0$
							1		$00B_10,00B_00$
							0	1	$0B_100,0B_000$
							1		B_1000,B_0000
						01 (RA, SE)	0	0	$SSSB_1,SSSB_0$
							1		SSB_1S,SSB_0S
							0	1	SB_1SS,SB_0SS
							1		B_1SSS,B_0SSS
						1x (PACK)	0	0	$B_7B_6B_5B_4,B_3B_2B_1B_0$
							1		$B_6B_7B_4B_5,B_2B_3B_0B_1$
							0	1	$B_5B_4B_7B_6,B_1B_0B_3B_2$
							1		$B_4B_5B_6B_7,B_0B_1B_2B_3$
01	Half-word	$B_7B_6,B_5B_4,B_3B_2,B_1B_0$		00	Byte	00 (RA, LT)	x	x	B_6,B_4,B_2,B_0
						01 (LA, RT)			B_7,B_5,B_3,B_1
						1x (UNPACK)			$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$

Table 104. Programmed data handling (continued)

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	Destination data stream(1)
01	Half-word	$B_7B_6,B_5B_4,B_3B_2,B_1B_0$	x	01	Half-word	xx	0	x	$B_7B_6,B_5B_4,B_3B_2,B_1B_0$
							1		$B_6B_7,B_4B_5,B_2B_3,B_0B_1$
				10	Word	00 (RA, 0P)	0	0	$00B_3B_2,00B_1B_0$
							1		$00B_2B_3,00B_0B_1$
							0	1	B_3B_200,B_1B_000
							1		B_2B_300,B_0B_100
						01 (RA, SE)	0	0	SSB_3B_2,SSB_1B_0
							1		SSB_2B_3,SSB_0B_1
							0	1	B_3B_2SS,B_1B_0SS
							1		B_2B_3SS,B_0B_1SS
						1x (PACK)	0	0	$B_7B_6B_5B_4,B_3B_2B_1B_0$
							1		$B_6B_7B_4B_5,B_2B_3B_0B_1$
							0	1	$B_5B_4B_7B_6,B_1B_0B_3B_2$
							1		$B_4B_5B_6B_7,B_0B_1B_2B_3$
10	Word	$B_7B_6B_5B_4,B_3B_2B_1B_0$	0	00	Byte	00 (RA, LT)	x	x	B_{12},B_8,B_4,B_0
						01 (LA, RT)			B_{15},B_{11},B_7,B_3
						10 (UNPACK)			$B_7,B_6,B_5,B_4,B_3,B_2,B_1,B_0$
				01	Half-word	00 (RA, LT)	0		B_5B_4,B_1B_0
							1		B_4B_5,B_0B_1
						01 (LA, RT)	0		B_7B_6,B_3B_2
							1		B_6B_7,B_2B_3
						1x (UNPACK)	0		$B_7B_6,B_5B_4,B_3B_2,B_1B_0$
							1		$B_6B_7,B_4B_5,B_2B_3,B_0B_1$

Table 104. Programmed data handling (continued)

SDW_LOG2 [1:0]	Source data	Source data stream(1)	SBX	DDW_LOG2 [1:0]	Destination data	PAM[1:0](2)	DBX	DHX	Destination data stream(1)
10	Word	$B_7B_6B_5B_4$, $B_3B_2B_1B_0$	0	10	Word	xx	0	0	$B_7B_6B_5B_4,B_3B_2B_1B_0$
							1		$B_6B_7B_4B_5,B_2B_3B_0B_1$
							0	1	$B_5B_4B_7B_6,B_1B_0B_3B_2$
							1		$B_4B_5B_6B_7,B_0B_1B_2B_3$
			1	00	Byte	00 (RA, LT)	x	x	B_{12},B_8,B_4,B_0
						01 (LA, RT)			B_{15},B_{11},B_7,B_3
						1x (UNPACK)			$B_7,B_5,B_6,B_4,B_3,B_1,B_2,B_0$
				01	Half-word	00 (RA, LT)	0		B_6B_4,B_2B_0
							1		B_4B_6,B_0B_2
						01 (LA, RT)	0		B_7B_5,B_3B_1
							1		B_5B_7,B_1B_3
						1x (UNPACK)	0		$B_7B_5,B_6B_4,B_3B_1,B_2B_0$
							1		$B_5B_7,B_4B_6,B_1B_3,B_0B_2$
				10	Word	xx	0	0	$B_7B_5B_6B_4,B_3B_1B_2B_0$
							1		$B_5B_7B_4B_6,B_1B_3B_0B_2$
							0	1	$B_6B_4B_7B_5,B_2B_0B_3B_1$
							1		$B_4B_6B_5B_7,B_0B_2B_1B_3$

1. Data stream is timely ordered starting from the byte with the lowest index (B_0).
2. RA= right aligned, LA = left aligned, RT = right truncated, LT = left truncated, 0P = zero bit padding up to the destination data width, SE = sign bit extended up to the destination data width.

19.4.11 GPDMA transfer request and arbitration

GPDMA transfer request

As defined by GPDMA_CxTR2, a programmed GPDMA data transfer is requested with one of the following:

- a software request if the control bit SWREQ = 1: This is used typically by the CPU for a data transfer from a memory-mapped address to another memory mapped address (memory-to-memory, GPIO to/from memory)
- an input hardware request coming from a peripheral if SWREQ = 0: The selection of the GPDMA hardware peripheral request is driven by the REQSEL[7:0] field (see *Section 19.3.3*). The selected hardware request can be one of the following:
  - an hardware request from a peripheral configured in GPDMA mode (for a transfer from/to the peripheral data register respectively to/from the memory)
  - an hardware request from a peripheral for its control registers update from the memory
  - an hardware request from a peripheral for a read of its status registers transferred to the memory

Caution: The user must not assign a same input hardware peripheral GPDMA request via GPDMA_CxTR.REQSEL[7:0] to two different channels, if at a given time this request is asserted by the peripheral and each channel is ready to execute this requested data transfer. There is no user setting error reporting.

GPDMA transfer request for arbitration

A ready FIFO-based GPDMA source single/burst transfer (from the source address to the FIFO) to be scheduled over the allocated master port (GPDMA_CxTR1.SAP) is arbitrated based on the channel priority (GPDMA_CxCR.PRIO[1:0]) versus the other simultaneous requested GPDMA transfers to the same master port.

A ready FIFO-based GPDMA destination single/burst transfer (from the FIFO to the destination address) to be scheduled over the allocated master port (GPDMA_CxTR1.DAP) is arbitrated based on the channel priority (GPDMA_CxCR.PRIO[1:0]) versus the other simultaneous requested GPDMA transfers to the same master port.

An arbitrated GPDMA requested link transfer consists of one 32-bit read from the linked-list data structure in memory to one of the linked-list registers (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR or GPDMA_CxLLR, plus GPDMA_CxTR3, GPDMA_CxBR2). Each 32-bit read from memory is arbitrated with the same channel priority as for data transfers, in order to be scheduled over the allocated master port (GPDMA_CxCR.LAP).

Whatever the requested data transfer is programmed with a software request for a memory-to- memory transfer (GPDMA_CxTR2.SWREQ = 1), or with a hardware request (GPDMA_CxTR2.SWREQ = 0) for a memory-to-peripheral transfer or a peripheral-to-memory transfer and whatever is the hardware request type, re-arbitration occurs after each granted single/burst transfer.

When an hardware request is programmed from a destination peripheral (GPDMA_CxTR2.SWREQ = 0 and GPDMA_CxTR2.DREQ = 1), the first memory read of a (possibly 2D/repeated) block (the first ready FIFO-based source burst request), is gated by the occurrence of the corresponding and selected hardware request. This first read request to memory is not taken into account earlier by the arbiter (not as soon as the block transfer is enabled and executable).

GPDMA arbitration

The GPDMA arbitration is directed from the 4-grade assigned channel priority (GPDMA_CxCR.PRIO[1:0]). The arbitration policy, as illustrated in *Figure 110*, is defined by:

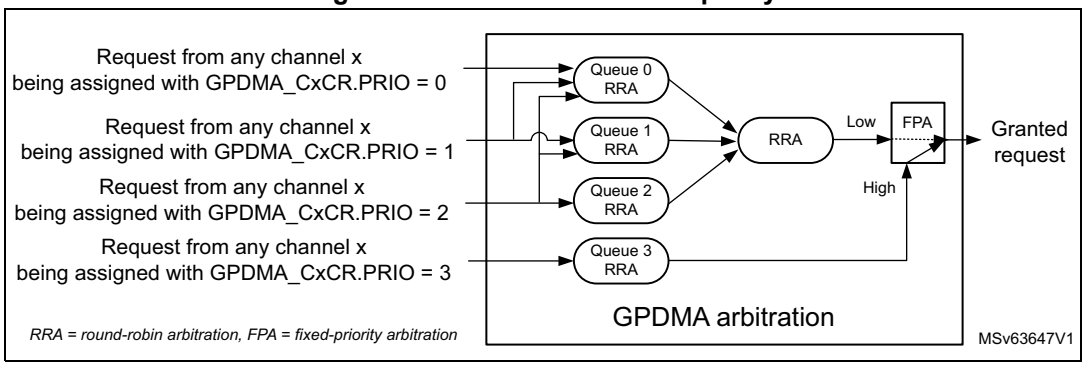
- one high-priority traffic class (queue 3), dedicated to the assigned channels with priority 3, for time-sensitive channels

  This traffic class is granted via a fixed-priority arbitration against any other low-priority traffic class. Within this class, requested single/burst transfers are round-robin arbitrated.
- three low-priority traffic classes (queues 0, 1, or 2) for non time-sensitive channels with priority 0, 1, or 2

  Each requested single/burst transfer within this class is round-robin arbitrated, with a weight that is monotonically driven from the programmed priority:
  - Requests with priority 0 are allocated to the queue 0.
  - Requests with priority 1 are allocated and replicated to the queue 0 and queue 1.

– Requests with priority 2 are allocated and replicated to the queue 0, queue 1, and queue 2.
– Any queue 0, 1, or 2 equally grants any of its active input requests in a round-robin manner, provided there are simultaneous requests.
– Additionally, there is a second stage for the low-traffic with a round-robin arbiter that fairly alternates between simultaneous selected requests from queue 0, queue 1, and queue 2.

Figure 110. GPDMA arbitration policy



GPDMA arbitration and bandwidth

With this arbitration policy, the following is guaranteed:

- Equal maximum bandwidth between requests with same priority
- Reserved bandwidth (noted as B_{Q3}) to the time-sensitive requests (with priority 3)
- Residual weighted bandwidth between different low-priority requests (priority 0 versus priority 1 versus priority 2).

The two following examples highlight that the weighted round-robin arbitration is driven by the programmed priorities:

- **Example 1**: basic application with two non time-sensitive GPDMA requests: req0 and req1. There are the following programming possibilities:
  – If they are assigned with same priority, the allocated bandwidth by the arbiter to req0 (B_{req0}) is **equal** to the allocated bandwidth to req1(B_{req1}).
  $B_{req0} = B_{req1} = 1/2 * (1 - B_{Q3})$
  – If req0 is assigned to priority 0 and req1 to priority 1, the allocated bandwidth to req0 (B_{P0}) is **3 times less** than the allocated bandwidth to req1 (B_{P1}).
  $B_{req0} = B_{P0} = 1/2 * 1/2 * (1 - B_{Q3}) = 1/4 * (1 - B_{Q3})$
  $B_{req1} = B_{P1} = (1/2 + 1) * 1/2 * (1 - B_{Q3}) = 3/4 * (1 - B_{Q3})$
  – If req0 is assigned to priority 0 and req1 to priority 2, the allocated bandwidth to req0 (B_{P0}) is **5 times less** than the allocated bandwidth to req1 (B_{P2}).
  $B_{req0} = B_{P0} = 1/2 * 1/3 * (1 - B_{Q3}) = 1/6 * (1 - B_{Q3})$
  $B_{req1} = B_{P2} = (1/2 + 1 + 1) * 1/3 * (1 - B_{Q3}) = 5/6 * (1 - B_{Q3})$
  The above computed bandwidth calculation is based on a theoretical input request, always active for any GPDMA clock cycle. This computed bandwidth from the arbiter must be weighted by the frequency of the request given by the application, that cannot be always active and may be quite much variable from one GPDMA client (example I2C at 400 kHz) to another one (PWM at 1 kHz) than the above x3 and x5 ratios.

- **Example 2**: application where the user distributes a same non-null N number of GPDMA requests to every non time-sensitive priority 0, 1 and 2. The bandwidth calculation is then the following:
  - The allocated bandwidth to the set of requests of priority 0 (B_{P0}) is
    $B_{P0} = 1/3 * 1/3 * (1 - B_{Q3}) = 1/9 * (1 - B_{Q3})$
  - The allocated bandwidth to the set of requests of priority 1(B_{P1}) is
    $B_{P1} = (1/3 + 1/2) * 1/3 * (1 - B_{Q3}) = 5/18 * (1 - B_{Q3})$
  - The allocated bandwidth to the set of requests of priority 2(B_{P2}) is
    $B_{P2} = (1/3 + 1/2 + 1) * 1/3 * (1 - B_{Q3}) = 11/18 * (1 - B_{Q3})$
  - The allocated bandwidth to any request n ($B_{n)}$ among the N requests of that priority Pi (i = 0 to 2) is $B_n = 1/N * B_{Pi}$
  - The allocated bandwidth to any request n of priority 0i ($B_{n, Pi}$) is
    $B_{n, P0} = 1/N *1/9 * (1 - B_{Q3})$
    $B_{n, P1} = 1/N *5/18 * (1 - B_{Q3})$
    $B_{n, P2} = 1/N *11/18 * (1 - B_{Q3})$

  In this example, when the master port bus bandwidth is not totally consumed by the time-sensitive queue 3, the residual bandwidth is such that 2.5 times less bandwidth is allocated to any request of priority 0 versus priority 1, and 5.5 times less bandwidth is allocated to any request of priority 0 versus priority 2.

More generally, assume that the following requests are present:

- I requests (I ≥ 0) assigned to priority 0
  If I > 0, these requests are noted from i = 0 to I-1.
- J requests (J ≥ 0) assigned to priority 1
  If J > 0, these requests are noted from j = 0 to J-1.
- K requests (K > 0) assigned to priority 2
  These requests are noted from k = 0 to K-1
- L requests (L ≥ 0) assigned to priority 3
  If L > 0, these requests are noted from l = 0 to L-1.

As B_{Q3} is the reserved bandwidth to time-sensitive requests, the bandwidth for each request L with priority 3 is:

- $B_l = B_{Q3} / L$ for L > 0 (else: $B_l = 0$)

The bandwidth for each non-time sensitive queue is:

- $B_{Q0} = 1/3 * (1 - B_{Q3})$
- $B_{Q1} = 1/3 * (1 - B_{Q3})$
- $B_{Q2} = 1/3 * (1 - B_{Q3})$

The bandwidth for the set of requests with priority 0 is:

- $B_{P0} = I / (I + J + K) * B_{Q0}$

The bandwidth for each request i with priority 0 is:

- $B_i = B_{P0} / I$ for L > 0 (else $B_{P0} = 0$)

The bandwidth for the set of requests with priority 1 and routed to queue 0 is:

- $B_{P1,Q0} = J / (I + J + K) * B_{Q0}$

The bandwidth for the set of requests with priority 1 and routed to queue 1 is:

- $B_{P1,Q1} = J / (J + K) * B_{Q1}$

The total bandwidth for the set of requests with priority 1 is:

- $B_{P1} = B_{P1,Q0} + B_{P1,Q1}$

The bandwidth for each request j with priority 1 is:

- $B_j = B_{P1} / J$ for $J > 0$ (else $B_j = 0$)

The bandwidth for the set of requests with priority 2 and routed to queue 0 is:

- $B_{P2,Q0} = K / (I + J + K) * B_{Q0}$

The bandwidth for the set of requests with priority 2 and routed to queue 1 is:

- $B_{P2,Q1} = K / (J + K) * B_{Q1}$

The bandwidth for the set of requests with priority 2 and routed to queue 2 is:

- $B_{P2,Q2} = B_{Q2}$

The total bandwidth for the set of requests with priority 2 is:

- $B_{P2} = B_{P2,Q0} + B_{P2,Q1} + B_{P2,Q2}$

The bandwidth for each request k with priority 2 is:

- $B_k = B_{P2} / K$ ($K>0$ in the general case)

Thus finally the maximum allocated residual bandwidths for any i, j, k non-time sensitive request are:

- in the general case (when there is at least one request k with a priority 2 ($K > 0$)):
  - $B_i = 1/I * 1/3 * I/(I + J + K) * (1 - B_{Q3})$
  - $B_j = 1/J * 1/3 *[J/(I + J + K) + J/(J + K)] * (1 - B_{Q3})$
  - $B_k = 1/K * 1/3 *[K/(I + J + K) + K/(J + K) + 1] * (1 - B_{Q3})$
- in the specific case (when there is no request k with a priority 2 ($K = 0$)):
  - $B_i = 1/I * 1/2 * I/(I + J) * (1 - B_{Q3})$
  - $B_j = 1/J * 1/2 *[J/(I + J) + 1] * (1 - B_{Q3})$

Consequently, the GPDMA arbiter can be used as a programmable weighted bandwidth limiter, for each queue and more generally for each request/channel. The different weights are monotonically resulting from the programmed channel priorities.

19.4.12 GPDMA triggered transfer

A programmed GPDMA transfer can be triggered by a rising/falling edge of a selected input trigger event, as defined by GPDMA_CxTR2.TRIGPOL[1:0] and GPDMA_CxTR2.TRIGSEL[6:0] (see *Section 19.3.6* for the trigger selection).

The triggered transfer, as defined by the trigger mode in GPDMA_CxTR2.TRIGM[1:0], can be at LLI data transfer level, to condition the first burst read of a block, the first burst read of a 2D/repeated block for channel x (x = 12 to 15), or each programmed single read. The trigger mode can also be programmed to condition the LLI link transfer (see TRIGM[1:0] in GPDMA_CxTR2 for more details).

Trigger hit memorization and trigger overrun flag generation

The GPDMA monitoring of a trigger for a channel x is started when the channel is enabled/loaded with a new active trigger configuration: rising or falling edge on a selected trigger (respectively TRIGPOL[1:0] = 01 or TRIGPOL[1:0] = 10).

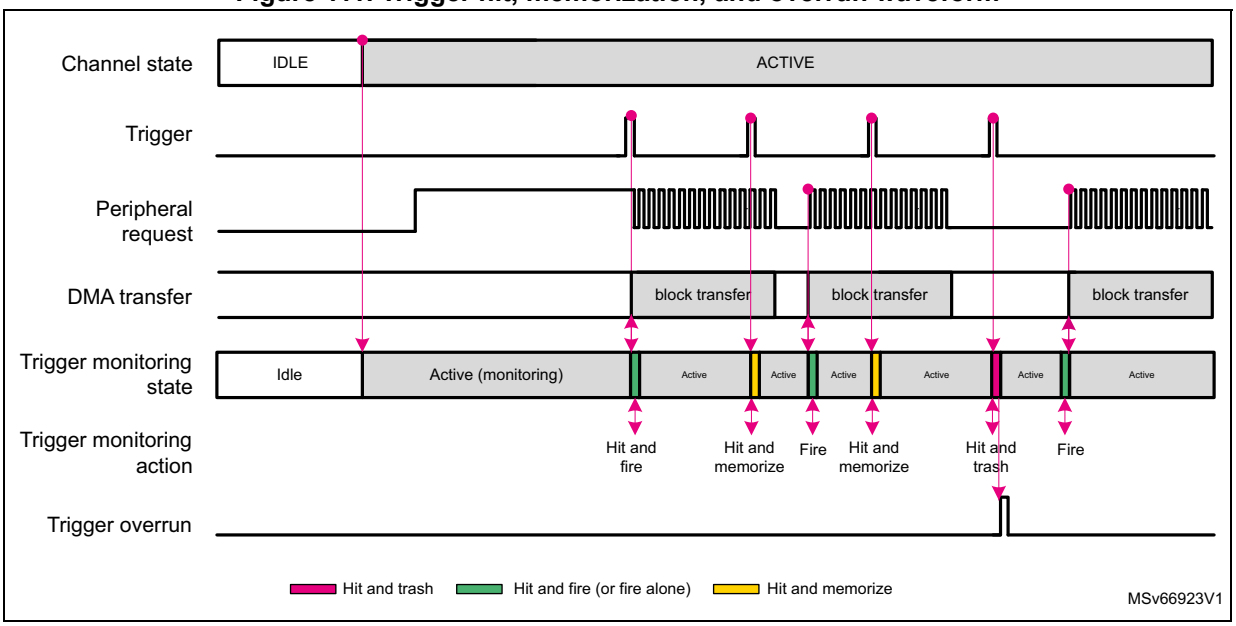
The monitoring of this trigger is kept active during the triggered and uncompleted (data or link) transfer. If a new trigger is detected, this hit is internally memorized to grant the next transfer, as long as the defined rising/falling edge and TRIGSEL[6:0] are not modified, and the channel is enabled.

Transferring a next $LLI_{n+1,}$ that updates the GPDMA_CxTR2 with a new value for any of TRIGSEL[6:0] or TRIGPOL[1:0], resets the monitoring, trashing the possible memorized hit of the formerly defined LLI_n trigger.

Caution: After a first new trigger hit_{n+1} is memorized, if another trigger hit_{n+2} is detected and if the hit_n triggered transfer is still not completed, hit_{n+2} is lost and not memorized. A trigger overrun flag is reported (GPDMA_CxSR.TOF = 1) and an interrupt is generated if enabled (if GPDMA_CxCR.TOIE = 1). The channel is not automatically disabled by hardware due to a trigger overrun.

Figure 111 illustrates the trigger hit, memorization and overrun in the configuration example with a block-level trigger mode and a rising edge trigger polarity.

Figure 111. Trigger hit, memorization, and overrun waveform



Note: The user can assign the same input trigger event to different channels. This can be used to trigger different channels on a broadcast trigger event.

19.4.13 GPDMA circular buffering with linked-list programming

GPDMA circular buffering for memory-to-peripheral and peripheral-to-memory transfers, with a linear addressing channel

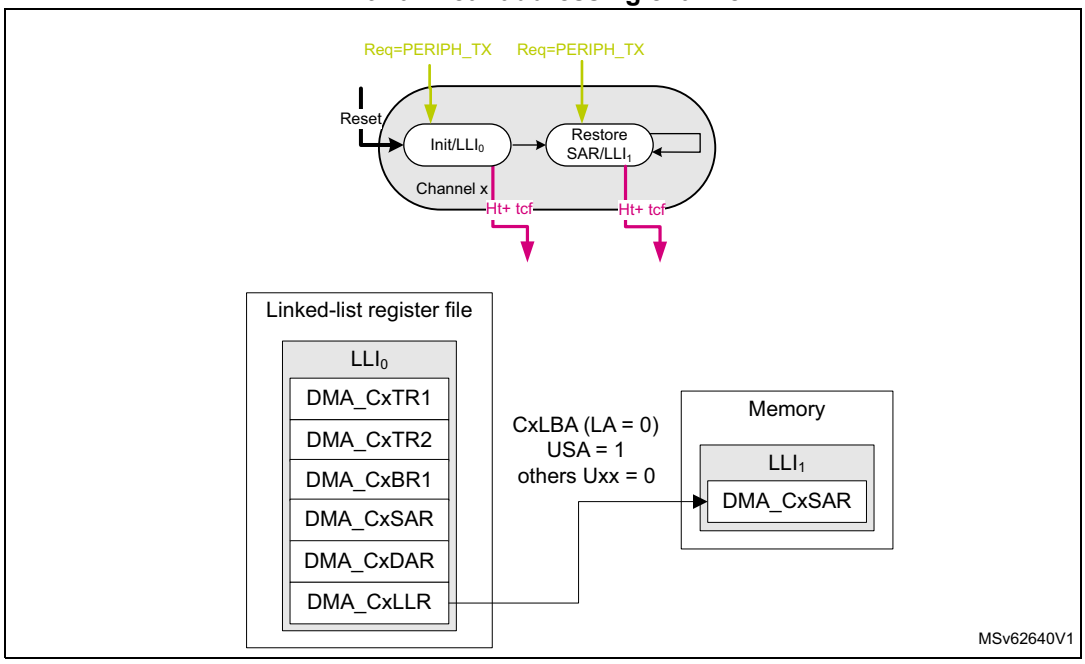
For a circular buffering, with a continuous memory-to-peripheral (or peripheral-to-memory) transfer, the software must set up a channel with half-transfer and complete transfer

events/interrupts generation (GPDMA_CxCR.HTIE = 1 and GPDMA_CxCR.TCIE = 1), in order to enable a concurrent buffer software processing.

LLI_0 is configured for the first block transfer with the linear addressing channel. A continuously-executed LLI_1 is needed to restore the memory source (or destination) start address, for the memory-to-peripheral transfer (respectively the peripheral-to-memory transfer). GPDMA automatically reloads the initially programmed GPDMA_CxBR1.BNDT[15:0] when a block transfer is completed, and there is no need to restore GPDMA_CxBR1.

Figure 112 illustrates this programming with a linear addressing GPDMA channel and a source circular buffer.

Figure 112. GPDMA circular buffer programming: update of the memory start address with a linear addressing channel

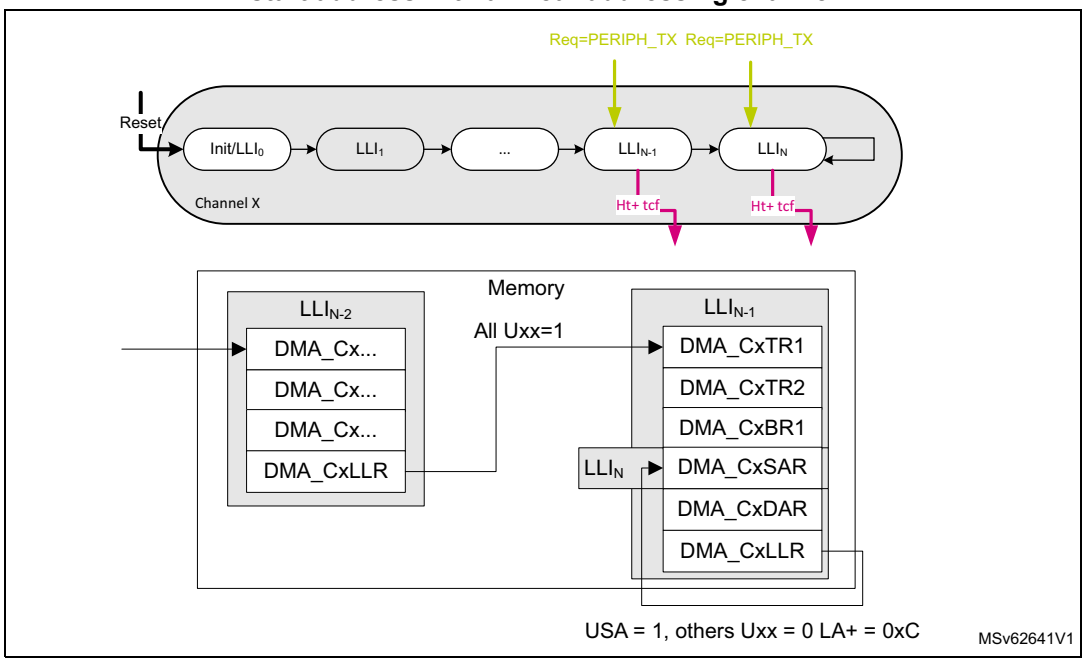


Note: With a 2D addressing channel, the user may use a single LLI with GPDMA_CxBR1.BRC[10:0] = 1, and program a negative memory block address offset with GDMA_CxBR2 and GDMA_CxBR1, in order to jump back to the memory source or the destination start address.

If circular buffering must be executed after some other transfers over the shared GPDMA channel x, the before-last LLI_{N-1} in memory is needed to configure the first block transfer. And the last LLI_N restores the memory source (or destination) start address in memory-to-peripheral transfer (respectively in peripheral-to-memory transfer).

Figure 113 illustrates this programming with a linear addressing shared GPDMA channel, and a source circular buffer.

Figure 113. Shared GPDMA channel with circular buffering: update of the memory start address with a linear addressing channel



19.4.14 GPDMA transfer in peripheral flow-control mode

A peripheral with the peripheral flow-control mode feature can decide to early terminate a GPDMA block transfer, provided that the allocated channel is implemented with this feature (see *Section 19.3.5*).

If the related GPDMA channel x is also programmed in peripheral flow-control mode (GPDMA_CxTR2.PFREQ = 1):

- The GPDMA block transfer starts as follows:
  - If GPDMA_CxBR1.BNDT[15:0] ≠ 0, the programmed value is internally taken into account by the GPDMA hardware.
  - If GPDMA_CxBR1.BNDT[15:0] = 0, the GPDMA hardware internally considers a 64-Kbyte value for the maximum source block size to be transferred.
- The GPDMA block transfer is completed as soon as the first occurrence of one of the following condition occurs:
  - when GPDMA_CxBR1.BNDT[15:0] = 0
  - when the peripheral early terminates the block. The complete transfer event is generated if programmed, depending on GPDMA_CxTR2 (see *GPDMA channel x transfer register 2 (GPDMA_CxTR2)*). Then the software can read the current number of transferred bytes from the source (GPDMA_CxBR1.BNDT[15:0]), and/or read the current source or destination address of the buffer in memory (GPDMA_CxSAR[31:0] or GPDMA_CxDAR[31:0]).

In peripheral flow-control mode:

- a destination peripheral with a hardware requested transfer is not supported: memory-to-peripheral transfer is not supported.
- Data packing from a source peripheral is not supported.
- 2D/repeated block is not supported.
- GPDMA_CxBR1.BNDT[15:0] must be programmed as a multiple of the source (peripheral) burst size.

19.4.15 GPDMA secure/nonsecure channel

The GPDMA controller is compliant with the TrustZone hardware architecture at channel level, partitioning all its resources so that they exist in one of the secure and nonsecure worlds at any given time.

Any channel x is a secure or a nonsecure hardware resource, as configured by GPDMA_SECCFGR.SECx.

When a channel x is configured in secure state by a secure and privileged agent, the following access control rules are applied:

- A nonsecure read access to a register field of this channel is forced to return 0, except for GPDMA_SECCFGR, GPDMA_PRIVCFGR, GPDMA_RCFGLOCKR that are readable by a nonsecure agent.
- A nonsecure write access to a register field of this channel has no impact.

When a channel x is configured in secure state, a secure agent can configure separately as secure or nonsecure the GPDMA data transfer from the source (GPDMA_CxTR1.SSEC) and the GPDMA data transfer to the destination (GPDMA_CxTR1.DSEC).

When a channel x is configured in secure state and in linked-list mode, the loading of the next linked-list data structure from the GPDMA memory into its register file, is automatically performed with secure transfers via the GPDMA_CxCR.LAP allocated master port.

The GPDMA generates a secure bus that reflects GPDMA_SECCFGR, to keep the other peripherals informed of the secure/nonsecure state of each GPDMA channel x.

The GPDMA also generates a security illegal access pulse signal on an illegal nonsecure access to a secure GPDMA register. This signal is routed to the TrustZone interrupt controller.

When the secure software must switch a channel from a secure state to a nonsecure state, the secure software must abort the channel or wait until the secure channel is completed before switching. This is needed to dynamically re-allocate a channel to a next nonsecure transfer as a nonsecure software is not allowed to do so and must have GPDMA_CxCR.EN = 0 before the nonsecure software can reprogram the GPDMA_CxCR for a next transfer. The secure software may reset not only the channel x (GPDMA_CxCR.RESET = 1) but also the full channel x register file to its reset value.

19.4.16 GPDMA privileged/unprivileged channel

Any channel x is a privileged or unprivileged hardware resource, as configured by a privileged agent via GPDMA_PRIVCFGR.PRIVx.

When a channel x is configured in a privileged state by a privileged agent, the following access control rules are applied:

- An unprivileged read access to a register field of this channel is forced to return 0, except for GPDMA_PRIVCFGR, GPDMA_SECCFGR, GPDMA_RCFGLOCKR that are readable by an unprivileged agent.
- An unprivileged write access to a register field of this channel has no impact.

When a channel is configured in a privileged (or unprivileged) state, the source and destination data transfers are privileged (respectively unprivileged) transfers over the AHB master port.

When a channel is configured in a privileged (or unprivileged) state and in linked-list mode, the loading of the next linked-list data structure from the GPDMA memory into its register file, is automatically performed with privileged (respectively unprivileged) transfers, via the GPDMA_CxCR.LAP allocated master port.

The GPDMA generates a privileged bus that reflects GPDMA_PRIVCFGR, to keep the other peripherals informed of the privileged/unprivileged state of each GPDMA channel x.

When the privileged software must switch a channel from a privileged state to an unprivileged state, the privileged software must abort the channel or wait until that the privileged channel is completed before switching. This is needed to dynamically re-allocate a channel to a next unprivileged transfer as an unprivileged software is not allowed to do so, and must have GPDMA_CxCR.EN = 0 before the unprivileged software can reprogram the GPDMA_CxCR for a next transfer. The privileged software may reset not only the channel x (GPDMA_CxCR.RESET = 1) but also the full channel x register file to its reset value.

19.4.17 GPDMA error management

The GPDMA is able to manage and report to the user a transfer error, as follows, depending on the root cause.

Data transfer error

On a bus access (as a AHB single or a burst) to the source or the destination:

- The source or destination target reports an AHB error.
- The programmed channel transfer is stopped (GPDMA_CxCR.EN cleared by the GPDMA hardware). The channel status register reports an idle state (GPDMA_CxSR.IDLEF = 1) and the data error (GPDMA_CxSR.DTEF = 1).
- After a GPDMA data transfer error, the user must perform a debug session, taking care of the product-defined memory mapping of the source and destination, including the protection attributes.
- After a GPDMA data transfer error, the user must issue a channel reset (set GPDMA_CxCR.RESET) to reset the hardware GPDMA channel data path and the content of the FIFO, before the user enables again the same channel for a next transfer.

Link transfer error

On a tentative update of a GPDMA channel register from the programmed LLI in the memory:

- The linked-list memory reports an AHB error.
- The programmed channel transfer is stopped (GPDMA_CxCR.EN cleared by the GPDMA hardware), the channel status register reports an idle state (GPDMA_CxSR.IDLEF = 1) and the link error (GPDMA_CxSR.ULEF = 1).
- After a GPDMA link error, the user must perform a debug session, taking care of the product-defined memory mapping of the linked-list data structure (GPDMA_CxLBAR and GPDMA_CxLLR), including the protection attributes.
- After a GPDMA link error, the user must explicitly write the linked-list register file (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR and GPDMA_CxLLR, plus GPDMA_CxTR3 and GPDMA_CxBR2), before the user enables again the same channel for a next transfer.

User setting error

On a tentative execution of a GPDMA transfer with an unauthorized user setting:

- The programmed channel transfer is disabled (GPDMA_CxCR.EN forced and cleared by the GPDMA hardware) preventing the next unauthorized programmed data transfer from being executed. The channel status register reports an idle state (GPDMA_CxSR.IDLEF = 1) and a user setting error (GPDMA_CxSR.USEF = 1).
- After a GPDMA user setting error, the user must perform a debug session, taking care of the GPDMA channel programming. A user setting error can be caused by one of the following:
  - a programmed null source block size without a programmed update of this value from the next LLI_1 (GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxLLR.UB1 = 0)
  - a programmed non-null source block size being not a multiple of the programmed data width of a source burst transfer (GPDMA_CxBR1.BNDT[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0])
  - when in packing/unpacking mode (if PAM[1] = 1), a programmed non-null source block size being not a multiple of the programmed data width of a destination burst transfer (GPDMA_CxBR1.BNDT[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0])
  - a programmed unaligned source start address, being not a multiple of the programmed data width of a source burst transfer (GPDMA_CxSAR[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0])
  - for channel x (x = 12 to 15): a programmed unaligned source address offset being not a multiple of the programmed data width of a source burst transfer (GPDMA_CxTR3.SAO[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0])
  - for channel x (x = 12 to 15): a programmed unaligned block repeated source address offset being not a multiple of the programmed data width of a source burst transfer (GPDMA_CxBR2.BRSAO[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0])
  - a programmed unaligned destination start address, being not a multiple of the programmed data width of a destination burst transfer (GPDMA_CxDAR[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0])
  - for channel x (x = 12 to 15): a programmed unaligned destination address offset being not a multiple of the programmed data width of a destination burst transfer (GPDMA_CxTR3.DAO[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0])

- for channel x (x = 12 to 15): a programmed unaligned block repeated destination address offset being not a multiple of the programmed data width of a destination burst transfer (GPDMA_CxBR2.BRDAO[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0])
- a programmed double-word source data width (GPDMA_CxTR1.SDW_LOG2[1:0] = 11)
- a programmed double-word destination data width (GPDMA_CxTR1.DDW_LOG2[1:0] = 11)
- a programmed linked-list item LLI_{n+1} with a null data transfer (GPDMA_CxLLR.UB1 = 1 and GPDMA_CxBR1. BNDT = 0)

19.5 GPDMA in debug mode

When the microcontroller enters debug mode (core halted), any channel x can be individually either continued (default) or suspended, depending on the programmable control bit in the DBGMCU module.

Note: In debug mode, GPDMA_CxSR.SUSPF is not altered by a suspension from the programmable control bit in the DBGMCU module. In this case, GPDMA_CxSR.IDLEF can be checked to know the completion status of the channel suspension.

19.6 GPDMA in low-power modes

Table 105. Effect of low-power modes on GPDMA

Mode	Description
Sleep	No effect. GPDMA interrupts cause the device to exit Sleep mode.
Stop(1)	The content of the GPDMA registers is kept when entering Stop mode.
Standby	The GPDMA is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 19.3.2* to know if any Stop mode is supported.

19.7 GPDMA interrupts

There is one GPDMA interrupt line for each channel, and separately for each CPU (if several ones in the devices).

Table 106. GPDMA interrupt requests

Interrupt acronym	Interrupt event	Interrupt enable	Event flag	Event clear method
GPDMA_CHx	Transfer complete	GPDMA_CxCR.TCIE	GPDMA_CxSR.TCF	Write 1 to GPDMA_CxFCR.TCF
	Half transfer	GPDMA_CxCR.HTIE	GPDMA_CxSR.HTF	Write 1 to GPDMA_CxFCR.HTF
	Data transfer error	GPDMA_CxCR.DTEIE	GPDMA_CxSR.DTEF	Write 1 to GPDMA_CxFCR.DTEF
	Update link error	GPDMA_CxCR.ULEIE	GPDMA_CxSR.ULEF	Write 1 to GPDMA_CxFCR.ULEF
	User setting error	GPDMA_CxCR.USEIE	GPDMA_CxSR.USEF	Write 1 to GPDMA_CxFCR.USEF
	Suspended	GPDMA_CxCR.SUSPIE	GPDMA_CxSR.SUSPF	Write 1 to GPDMA_CxFCR.SUSPF
	Trigger overrun	GPDMA_CxCR.TOFIE	GPDMA_CxSR.TOF	Write 1 to GPDMA_CxFCR.TOF

A GPDMA channel x event may be:

- a transfer complete
- a half-transfer complete
- a transfer error, due to either:
  - a data transfer error
  - an update link error
  - a user setting error completed suspension
- a trigger overrun

Note: When a channel x transfer complete event occurs, the output signal gpdma_chx_tc is generated as a high pulse of one clock cycle.

An interrupt is generated following any xx event, provided that both:

- the corresponding interrupt event xx is enabled (GPDMA_CxCR.xxIE = 1)
- the corresponding event flag is cleared (GPDMA_CxSR.xxF = 0). This means that, after a previous same xx event occurrence, a software agent must have written 1 into the corresponding xx flag clear control bit (write 1 into GPDMA_CxFCR.xxF).

TCF (transfer complete) and HTF (half transfer) events generation is controlled by GPDMA_CxTR2.TCEM[1:0] as follows:

- A transfer complete event is a block transfer complete, a 2D/repeated block transfer complete, or a LLI transfer complete including the upload of the next LLI if any, or the full linked-list completion, depending on the transfer complete event mode GPDMA_CxTR2.TCEM[1:0].

- A half-transfer event is a half-block transfer or a half 2D/repeated block transfer, depending on the transfer complete event mode GPDMA_CxTR2.TCEM[1:0].

  A half-block transfer occurs when half of the source block size bytes (rounded-up integer of GPDMA_CxBR1.BNDT[15:0] / 2) is transferred to the destination.

  A half 2D/repeated block transfer occurs when half of the repeated blocks (rounded-up integer of (GPDMA_CxBR1.BRC[10:0] + 1) / 2) is transferred to the destination.

See *GPDMA channel x transfer register 2 (GPDMA_CxTR2)* for more details.

A transfer error rises in one of the following situations:

- during a single/burst data transfer from the source or to the destination (DTEF)
- during an update of a GPDMA channel register from the programmed LLI in memory (ULEF)
- during a tentative execution of a GPDMA channel with an unauthorized setting (USEF)

  The user must perform a debug session to correct the GPDMA channel programming versus the USEF root causes list (see *Section 19.4.17*).

A trigger overrun is described in *Trigger hit memorization and trigger overrun flag generation*.

19.8 GPDMA registers

The GPDMA registers must be accessed with an aligned 32-bit word data access.

19.8.1 GPDMA secure configuration register (GPDMA_SECCFGR)

Address offset: 0x00

Reset value: 0x0000 0000

A write access to this register must be secure and privileged. A read access is secure or nonsecure, privileged or unprivileged.

A write access is ignored at bit level if the corresponding channel x is locked (GPDMA_RCFGLOCKR.LOCKx = 1).

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be programmed at a bit level, at the initialization/closure of a GPDMA channel (when GPDMA_CxCR.EN = 0), to securely allocate individually any channel x to the secure or nonsecure world.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **SECx**: secure state of channel x (x = 15 to 0)
0: Nonsecure
1: Secure

19.8.2 GPDMA privileged configuration register (GPDMA_PRIVCFGR)

Address offset: 0x04

Reset value: 0x0000 0000

A write access to this register must be privileged. A read access can be privileged or unprivileged, secure or nonsecure.

This register can mix secure and nonsecure information. If a channel x is configured as secure (GPDMA_SECCFGR.SECx = 1), the PRIVx bit can be written only by a secure (and privileged) agent.

A write access is ignored at bit level if the corresponding channel x is locked (GPDMA_RCFGLOCKR.LOCKx = 1).

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN =1.

This register must be programmed at a bit level, at the initialization/closure of a GPDMA channel (GPDMA_CxCR.EN = 0), to individually allocate any channel x to the privileged or unprivileged world.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PRIVx**: privileged state of channel x (x = 15 to 0)
0: unprivileged
1: privileged

19.8.3 GPDMA configuration lock register (GPDMA_RCFGLOCKR)

Address offset: 0x08

Reset value: 0x0000 0000

This register can be written by a software agent with secure privileged attributes in order to individually lock, for example at boot time, the secure privileged attributes of any GPDMA

channel/resource (to lock the setting of GPDMA_CxSECCFGR, GPDMA_CxPRIVCFGR for any channel x, for example at boot time).

A read access may be privileged or unprivileged, secure or nonsecure.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LOCK15	LOCK14	LOCK13	LOCK12	LOCK11	LOCK10	LOCK9	LOCK8	LOCK7	LOCK6	LOCK5	LOCK4	LOCK3	LOCK2	LOCK1	LOCK0
rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs	rs

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **LOCKx**: lock the configuration of GPDMA_SECCFGR.SECx, GPDMA_PRIVCFGR.PRIVx until a global GPDMA reset (x = 15 to 0)

This bit is cleared after reset and, once set, it cannot be reset until a global GPDMA reset.

0: secure privilege configuration of the channel x is writable.

1: secure privilege configuration of the channel x is not writable.

19.8.4 GPDMA nonsecure masked interrupt status register (GPDMA_MISR)

Address offset: 0x0C

Reset value: 0x0000 0000

This register is a read register.

This is a nonsecure register, containing the masked interrupt status bit MISx for each nonsecure channel x (channel x configured with GPDMA_SECCFGR.SECx = 0). It is a logical OR of all the flags of GPDMA_CxSR, each source flag being enabled by the corresponding interrupt enable bit of GPDMA_CxCR.

Every bit is deasserted by hardware when writing 1 to the corresponding flag clear bit in GPDMA_CxFCR.

If a channel x is in secure state (GPDMA_SECCFGR.SECx = 1), a read access to the masked interrupt status bit MISx of this channel x returns zero.

This register may mix privileged and unprivileged information, depending on the privileged state of each channel GPDMA_PRIVCFGR.PRIVx. A privileged software can read the full nonsecure interrupt status. An unprivileged software is restricted to read the status of unprivileged (and nonsecure) channels, other privileged bit fields returning zero.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MIS15	MIS14	MIS13	MIS12	MIS11	MIS10	MIS9	MIS8	MIS7	MIS6	MIS5	MIS4	MIS3	MIS2	MIS1	MIS0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MISx**: masked interrupt status of channel x (x = 15 to 0)
0: no interrupt occurred on channel x
1: an interrupt occurred on channel x

19.8.5 GPDMA secure masked interrupt status register (GPDMA_SMISR)

Address offset: 0x10

Reset value: 0x0000 0000

This is a secure read register, containing the masked interrupt status bit MISx for each secure channel x (GPDMA_SECCFGR.SECx = 1). It is a logical OR of all the GPDMA_CxSR flags, each source flag being enabled by the corresponding GPDMA_CxCR interrupt enable bit.

Every bit is deasserted by hardware when securely writing 1 to the corresponding GPDMA_CxFCR flag clear bit.

This register does not contain any information about a nonsecure channel.

This register can mix privileged and unprivileged information, depending on the privileged state of each channel GPDMA_PRIVCFGR.PRIVx. A privileged software can read the full secure interrupt status. An unprivileged software is restricted to read the status of unprivileged and secure channels, other privileged bit fields returning zero.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MIS15	MIS14	MIS13	MIS12	MIS11	MIS10	MIS9	MIS8	MIS7	MIS6	MIS5	MIS4	MIS3	MIS2	MIS1	MIS0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MISx**: masked interrupt status of the secure channel x (x = 15 to 0)
0: no interrupt occurred on the secure channel x
1: an interrupt occurred on the secure channel x

19.8.6 GPDMA channel x linked-list base address register (GPDMA_CxLBAR)

Address offset: 0x50 + 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This register must be written by a privileged software. It is either privileged readable or not, depending on the privileged state of the channel x GPDMA_PRIVCFGR.PRIVx.

This register is either secure or nonsecure depending on the secure state of the channel x (GPDMA_SECCFGR.SECx).

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This channel-based register is the linked-list base address of the memory region, for a given channel x, from which the LLIs describing the programmed sequence of the GPDMA transfers, are conditionally and automatically updated.

This 64-Kbyte aligned channel x linked-list base address is offset by the 16-bit GPDMA_CxLLR register that defines the word-aligned address offset for each LLI.

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

Bits 31:16 **LBA[31:16]**: linked-list base address of GPDMA channel x

Bits 15:0 Reserved, must be kept at reset value.

19.8.7 GPDMA channel x flag clear register (GPDMA_CxFCR)

Address offset: 0x5C+ 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This is a write register, secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx) and privileged or unprivileged, depending on the privileged state of the channel x (GPDMA_PRIVCFGR.PRIVx).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	w	w	w	w	w	w	w								

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **TOF**: trigger overrun flag clear
0: no effect
1: corresponding TOF flag cleared

Bit 13 **SUSPF**: completed suspension flag clear
0: no effect
1: corresponding SUSPF flag cleared

Bit 12 **USEF**: user setting error flag clear
0: no effect
1: corresponding USEF flag cleared

Bit 11 **ULEF**: update link transfer error flag clear
0: no effect
1: corresponding ULEF flag cleared

Bit 10 **DTEF**: data transfer error flag clear
0: no effect
1: corresponding DTEF flag cleared

Bit 9 **HTF**: half-transfer flag clear
0: no effect
1: corresponding HTF flag cleared

Bit 8 **TCF**: transfer complete flag clear
0: no effect
1: corresponding TCF flag cleared

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

19.8.8 GPDMA channel x status register (GPDMA_CxSR)

Address offset: 0x60 + 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0001

This is a read register, reporting the channel status.

This register is secure or nonsecure, depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of the channel (GPDMA_PRIVCFGR.PRIVx).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FIFOL[7:0]							
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDLEF
	r	r	r	r	r	r	r								r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **FIFOL[7:0]**: monitored FIFO level
Number of available write beats in the FIFO, in units of the programmed destination data width (see GPDMA_CxTR1.DDW_LOG2[1:0], in units of bytes, half-words, or words).

Note: After having suspended an active transfer, the user may need to read FIFOL[7:0], additionally to GPDMA_CxBR1.BDNT[15:0] and GPDMA_CxBR1.BRC[10:0], to know how many data have been transferred to the destination. Before reading, the user may wait for the transfer to be suspended (GPDMA_CxSR.SUSPF = 1).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TOF**: trigger overrun flag
0: no trigger overrun event
1: a trigger overrun event occurred

Bit 13 **SUSPF**: completed suspension flag
0: no completed suspension event
1: a completed suspension event occurred

Bit 12 **USEF**: user setting error flag
0: no user setting error event
1: a user setting error event occurred

Bit 11 **ULEF**: update link transfer error flag
0: no update link transfer error event
1: a master bus error event occurred while updating a linked-list register from memory

Bit 10 **DTEF**: data transfer error flag
0: no data transfer error event
1: a master bus error event occurred on a data transfer

Bit 9 **HTF**: half-transfer flag
0: no half-transfer event
1: a half-transfer event occurred
A half-transfer event is either a half-block transfer or a half 2D/repeated block transfer, depending on the transfer complete event mode (GPDMA_CxTR2.TCEM[1:0]).
A half-block transfer occurs when half of the bytes of the source block size (rounded up integer of GPDMA_CxBR1.BNDT[15:0]/2) has been transferred to the destination.
A half 2D/repeated block transfer occurs when half of the repeated blocks (rounded up integer of (GPDMA_CxBR1.BRC[10:0] + 1) / 2)) has been transferred to the destination.

Bit 8 **TCF**: transfer complete flag
0: no transfer complete event
1: a transfer complete event occurred
A transfer complete event is either a block transfer complete, a 2D/repeated block transfer complete, or a LLI transfer complete including the upload of the next LLI if any, or the full linked-list completion, depending on the transfer complete event mode (GPDMA_CxTR2.TCEM[1:0]).

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **IDLEF**: idle flag
0: channel not in idle state
1: channel in idle state
This idle flag is deasserted by hardware when the channel is enabled (GPDMA_CxCR.EN = 1) with a valid channel configuration (no USEF to be immediately reported).
This idle flag is asserted after hard reset or by hardware when the channel is back in idle state (in suspended or disabled state).

19.8.9 GPDMA channel x control register (GPDMA_CxCR)

Address offset: 0x64 + 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of the channel x (GPDMA_PRIVCFGR.PRIVx).

This register is used to control a channel (activate, suspend, abort or disable it).

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TOIE	SUSPIE	USEIE	ULEIE	DTEIE	HTIE	TCIE	Res.	Res.	Res.	Res.	Res.	SUSP	RESET	EN
	rw	rw	rw	rw	rw	rw	rw						rw	w	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:22 **PRIO[1:0]:** priority level of the channel x GPDMA transfer versus others
00: low priority, low weight
01: low priority, mid weight
10: low priority, high weight
11: high priority
Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bits 21:18 Reserved, must be kept at reset value.

Bit 17 **LAP**: linked-list allocated port
This bit is used to allocate the master port for the update of the GPDMA linked-list registers from the memory.
0: port 0 (AHB) allocated
1: port 1 (AHB) allocated
Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bit 16 **LSM**: Link step mode
0: channel executed for the full linked-list and completed at the end of the last LLI (GPDMA_CxLLR = 0). The 16 low-significant bits of the link address are null (LA[15:0] = 0) and all the update bits are null (UT1 = UB1 = UT2 = USA = UDA = ULL = 0 and UT3 = UB2 = 0). Then GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxBR1.BRC[10:0] = 0.
1: channel executed **once** for the current LLI
First the (possible 1D/repeated) block transfer is executed as defined by the current internal register file until GPDMA_CxBR1.BNDT[15:0] = 0 and GPDMA_CxBR1.BRC[10:0] = 0. Secondly the next linked-list data structure is conditionally uploaded from memory as defined by GPDMA_CxLLR. Then channel execution is completed.
Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TOIE**: trigger overrun interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 13 **SUSPIE**: completed suspension interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 12 **USEIE**: user setting error interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 11 **ULEIE**: update link transfer error interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 10 **DTEIE**: data transfer error interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 9 **HTIE**: half-transfer complete interrupt enable
0: interrupt disabled
1: interrupt enabled

Bit 8 **TCIE**: transfer complete interrupt enable
0: interrupt disabled
1: interrupt enabled

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **SUSP:** suspend
Writing 1 into the field RESET (bit 1) causes the hardware to de-assert this bit, whatever is written into this bit 2. Else:
Software must write 1 in order to suspend an active channel (channel with an ongoing GPDMA transfer over its master ports).
The software must write 0 in order to resume a suspended channel, following the programming sequence detailed in *Figure 95*.
0: write: resume channel, read: channel not suspended
1: write: suspend channel, read: channel suspended.

Bit 1 **RESET:** reset
This bit is write only. Writing 0 has no impact. Writing 1 implies the reset of the following: the FIFO, the channel internal state, SUSP and EN bits (whatever is written receptively in bit 2 and bit 0).
The reset is effective when the channel is in steady state, meaning one of the following:
- active channel in suspended state (GPDMA_CxSR.SUSPF = 1 and GPDMA_CxSR.IDLEF = GPDMA_CxCR.EN = 1)
- channel in disabled state (GPDMA_CxSR.IDLEF = 1 and GPDMA_CxCR.EN = 0).
After writing a RESET, to continue using this channel, the user must explicitly reconfigure the channel including the hardware-modified configuration registers (GPDMA_CxBR1, GPDMA_CxSAR, and GPDMA_CxDAR) before enabling again the channel (see the programming sequence in *Figure 96*).
0: no channel reset
1: channel reset

Bit 0 **EN:** enable
Writing 1 into the field RESET (bit 1) causes the hardware to de-assert this bit, whatever is written into this bit 0. Else:
this bit is deasserted by hardware when there is a transfer error (master bus error or user setting error) or when there is a channel transfer complete (channel ready to be configured, for example if LSM = 1 at the end of a single execution of the LLI).
Else, this bit can be asserted by software.
Writing 0 into this EN bit is ignored.
0: write: ignored, read: channel disabled
1: write: enable channel, read: channel enabled

19.8.10 GPDMA channel x transfer register 1 (GPDMA_CxTR1)

Address offset: 0x90 + 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx) except for secure DSEC and SSEC, privileged or unprivileged, depending on the privileged state of the channel x in GPDMA_PRIVCFGR.PRIVx.

This register controls the transfer of a channel x.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed. Then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by GPDMA from the memory if GPDMA_CxLLR.UT1 = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DSEC	DAP	Res.	Res.	DHX	DBX	DBL_1[5:0]						DINC	Res.	DDW_LOG2[1:0]	
rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SSEC	SAP	SBX	PAM[1:0]		Res.	SBL_1[5:0]						SINC	Res.	SDW_LOG2[1:0]	
rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw		rw	rw

Bit 31 **DSEC**: security attribute of the GPDMA transfer to the destination

If GPDMA_SECCFGR.SECx = 1 and the access is secure:

0: GPDMA transfer nonsecure

1: GPDMA transfer secure

This is a secure register bit. This bit can only be read by a secure software. This bit must be written by a secure software when GPDMA_SECCFGR.SECx = 1. A secure write is ignored when GPDMA_SECCFGR.SECx = 0.

When GPDMA_SECCFGR.SECx is deasserted, this DSEC bit is also deasserted by hardware (on a secure reconfiguration of the channel as nonsecure), and the GPDMA transfer to the destination is nonsecure.

Bit 30 **DAP**: destination allocated port

This bit is used to allocate the master port for the destination transfer

0: port 0 (AHB) allocated

1: port 1 (AHB) allocated

Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bits 29:28 Reserved, must be kept at reset value.

Bit 27 **DHX**: destination half-word exchange

If the destination data size is shorter than a word, this bit is ignored.

If the destination data size is a word:

0: no halfword-based exchanged within word

1: the two consecutive (post PAM) half-words are exchanged in each destination word.

Bit 26 **DBX**: destination byte exchange

If the destination data size is a byte, this bit is ignored.

If the destination data size is not a byte:

0:no byte-based exchange within half-word

1: the two consecutive (post PAM) bytes are exchanged in each destination half-word.

Bits 25:20 **DBL_1[5:0]**: destination burst length minus 1, between 0 and 63

The burst length unit is one data named beat within a burst. If DBL_1[5:0] = 0, the burst can be named as single. Each data/beat has a width defined by the destination data width DDW_LOG2[1:0].

Note: If a burst transfer crossed a 1-Kbyte address boundary on a AHB transfer, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB protocol.

If a burst transfer is of length greater than the FIFO size of the channel x, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the FIFO size. Transfer performance is lower, with GPDMA re-arbitration between effective and lower singles/bursts, but the data integrity is guaranteed.

Bit 19 **DINC**: destination incrementing burst

0: fixed burst

1: contiguously incremented burst

The destination address, pointed by GPDMA_CxDAR, is kept constant after a burst beat/single transfer, or is incremented by the offset value corresponding to a contiguous data after a burst beat/single transfer.

Bit 18 Reserved, must be kept at reset value.

Bits 17:16 **DDW_LOG2[1:0]**: binary logarithm of the destination data width of a burst, in bytes

00: byte

01: half-word (2 bytes)

10: word (4 bytes)

11: user setting error reported and no transfer issued

Note: A destination burst transfer must have an aligned address with its data width (start address GPDMA_CxDAR[2:0] and if present address offset GPDMA_CxTR3.DAO[2:0], versus DDW_LOG2[1:0]). Otherwise a user setting error is reported and no transfer is issued.

When configured in packing mode (PAM[1] = 1 and destination data width different from source data width), a source block size must be a multiple of the destination data width (see GPDMA_CxBR1.BNDT[2:0] versus DDW_LOG2[1:0]). Else a user setting error is reported and none transfer is issued.

Setting a 8-byte data width causes a user setting error to be reported and none transfer is issued.

Bit 15 **SSEC**: security attribute of the GPDMA transfer from the source

If GPDMA_SECCFGR.SECx = 1 and the access is secure:

0: GPDMA transfer nonsecure

1: GPDMA transfer secure

This is a secure register bit. This bit can only be read by a secure software. This bit must be written by a secure software when GPDMA_SECCFGR.SECx = 1. A secure write is ignored when GPDMA_SECCFGR.SECx = 0.

When GPDMA_SECCFGR.SECx is deasserted, this SSEC bit is also deasserted by hardware (on a secure reconfiguration of the channel as nonsecure), and the GPDMA transfer from the source is nonsecure.

Bit 14 **SAP**: source allocated port

This bit is used to allocate the master port for the source transfer

0: port 0 (AHB) allocated

1: port 1 (AHB) allocated

Note: This bit must be written when EN = 0. This bit is read-only when EN = 1.

Bit 13 **SBX**: source byte exchange within the unaligned half-word of each source word
If the source data width is shorter than a word, this bit is ignored.
If the source data width is a word:
0:no byte-based exchange within the unaligned half-word of each source word
1: the two consecutive bytes within the unaligned half-word of each source word are exchanged.

Bits 12:11 **PAM[1:0]**: padding/alignment mode
If DDW_LOG2[1:0] = SDW_LOG2[1:0]: if the data width of a burst destination transfer is equal to the data width of a burst source transfer, these bits are ignored.
Else, in the following enumerated values, the condition PAM_1 is when destination data width is higher than source data width, and the condition PAM_2 is when source data width is higher than destination data width.
Condition: PAM_1
00: source data is transferred as right aligned, padded with 0s up to the destination data width
01: source data is transferred as right aligned, sign extended up to the destination data width
10-11: successive source data are FIFO queued and packed at the destination data width, in a left (LSB) to right (MSB) order (named little endian), before a destination transfer
Condition: PAM_2
00: source data is transferred as right aligned, left-truncated down to the destination data width
01: source data is transferred as left-aligned, right-truncated down to the destination data width
10-11: source data is FIFO queued and unpacked at the destination data width, to be transferred in a left (LSB) to right (MSB) order (named little endian) to the destination

Note: If the transfer from the source peripheral is configured with peripheral flow-control mode (SWREQ = 0 and PFREQ = 1 and DREQ = 0), and if the destination data width > the source data width, packing is not supported.

Bit 10 Reserved, must be kept at reset value.

Bits 9:4 **SBL_1[5:0]**: source burst length minus 1, between 0 and 63
The burst length unit is one data named beat within a burst. If SBL_1[5:0] = 0, the burst can be named as single. Each data/beat has a width defined by the destination data width SDW_LOG2[1:0].

Note: If a burst transfer crossed a 1-Kbyte address boundary on a AHB transfer, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the AHB protocol.

If a burst transfer is of length greater than the FIFO size of the channel x, the GPDMA modifies and shortens the programmed burst into singles or bursts of lower length, to be compliant with the FIFO size. Transfer performance is lower, with GPDMA re-arbitration between effective and lower singles/bursts, but the data integrity is guaranteed.

Bit 3 **SINC**: source incrementing burst
0: fixed burst
1: contiguously incremented burst
The source address, pointed by GPDMA_CxSAR, is kept constant after a burst beat/single transfer or is incremented by the offset value corresponding to a contiguous data after a burst beat/single transfer.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **SDW_LOG2[1:0]**: binary logarithm of the source data width of a burst in bytes
00: byte
01: half-word (2 bytes)
10: word (4 bytes)
11: user setting error reported and no transfer issued

Note: A source block size must be a multiple of the source data width (GPDMA_CxBR1.BNDT[2:0] versus SDW_LOG2[1:0]). Otherwise, a user setting error is reported and no transfer is issued.

A source burst transfer must have an aligned address with its data width (start address GPDMA_CxSAR[2:0] versus SDW_LOG2[1:0]). Otherwise, a user setting error is reported and none transfer is issued.

Setting a 8-byte data width causes a user setting error to be reported and no transfer is issued.

19.8.11 GPDMA channel x transfer register 2 (GPDMA_CxTR2)

Address offset: 0x94 + 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x. This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (the hardware deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by GPDMA from the memory, if GPDMA_CxLLR.UT2 = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TCEM[1:0]		Res.	Res.	Res.	Res.	TRIGPOL[1:0]		Res.	TRIGSEL[6:0]						
rw	rw					rw	rw		rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRIGM[1:0]		Res.	PFREQ	BREQ	DREQ	SWREQ	Res.	REQSEL[7:0]							
rw	rw		rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 **TCEM[1:0]**: transfer complete event mode

These bits define the transfer granularity for the transfer complete and half-transfer complete events generation.

00: at block level (when GPDMA_CxBR1.BNDT[15:0] = 0): the complete (and the half) transfer event is generated at the (respectively half of the) end of a block.

Note: If the initial LLI_0 data transfer is null/void (directly programmed by the internal register file with GPDMA_CxBR1.BNDT[15:0] = 0), then neither the complete transfer event nor the half-transfer event is generated.

01: channel x (x = 0 to 11), same as 00, channel x (x =12 to 15), at 2D/repeated block level (when GPDMA_CxBR1.BRC[10:0] = 0 and GPDMA_CxBR1.BNDT[15:0] = 0). The complete (and the half) transfer event is generated at the end (respectively half of the end) of the 2D/repeated block.

Note: If the initial LLI_0 data transfer is null/void (directly programmed by the internal register file with GPDMA_CxBR1.BNDT[15:0] = 0), then neither the complete transfer event nor the half-transfer event is generated.

10: at LLI level: the complete transfer event is generated at the end of the LLI transfer, including the update of the LLI if any. The half-transfer event is generated at the half of the LLI data transfer The LLI data transfer is a block transfer or a 2D/repeated block transfer for channel x (x =12 to 15), if any data transfer.

Note: If the initial LLI_0 data transfer is null/void (directly programmed by the internal register file with GPDMA_CxBR1.BNDT[15:0] = 0), then the half-transfer event is not generated, and the transfer complete event is generated when is completed the loading of the LLI_1.

11: at channel level: the complete transfer event is generated at the end of the last LLI transfer. The half-transfer event is generated at the half of the data transfer of the last LLI. The last LLI updates the link address GPDMA_CxLLR.LA[15:2] to zero and clears all the GPDMA_CxLLR update bits (UT1, UT2, UB1, USA, UDA and ULL, plus UT3 and UB2). If the channel transfer is continuous/infinite, no event is generated.

Bits 29:26 Reserved, must be kept at reset value.

Bits 25:24 **TRIGPOL[1:0]**: trigger event polarity

These bits define the polarity of the selected trigger event input defined by TRIGSEL[6:0].

00: no trigger (masked trigger event)

01: trigger on the rising edge

10: trigger on the falling edge

11: same as 00

Bit 23 Reserved, must be kept at reset value.

Bits 22:16 **TRIGSEL[6:0]**: trigger event input selection

These bits select the trigger event input of the GPDMA transfer (as per *Section 19.3.6*), with an active trigger event if TRIGPOL[1:0] ≠ 00.

Bits 15:14 **TRIGM[1:0]**: trigger mode

These bits define the transfer granularity for its conditioning by the trigger.

If the channel x is enabled (GPDMA_CxCR.EN asserted) with TRIGPOL[1:0] = 00 or 11, these TRIGM[1:0] bits are ignored.

Else, a GPDMA transfer is conditioned by one trigger hit:

00: at block level: the first burst read of each block transfer is conditioned by one hit trigger (channel x (x = 12 to 15): same, each block also if a 2D/repeated block is configured with GPDMA_CxBR1.BRC[10:0] ≠ 0).

01: channel x (x = 0 to 11): same as 00; channel x (x = 12 to 15): at 2D/repeated block level: the first burst read of the 2D/repeated block transfer is conditioned by one hit trigger.

10: at link level: a LLI link transfer is conditioned by one hit trigger. LLI data transfer is not conditioned.

11: at programmed burst level: If SWREQ = 1, each programmed burst read is conditioned by one hit trigger. If SWREQ = 0, each programmed burst that is requested by the selected peripheral, is conditioned by one hit trigger:

– If the peripheral is the source (DREQ = 0) of the LLI data transfer, each programmed burst read is conditioned.

– If the peripheral is the destination (DREQ = 1) of the LLI data transfer, each programmed burst write is conditioned. The first memory burst read of a (possibly 2D/repeated) block, also named as the first ready FIFO-based source burst, is gated by the occurrence of both the hardware request and one trigger hit.

As shown in *Figure 111: Trigger hit, memorization, and overrun waveform*, the GPDMA monitoring of a trigger for channel x is started when the channel is enabled/loaded with a new active trigger configuration: rising or falling edge on a selected trigger (TRIGPOL[1:0] = 01 or respectively TRIGPOL[1:0] = 10). The monitoring of this trigger is kept active during the triggered and uncompleted (data or link) transfer; and if a new trigger is detected then, this hit is internally memorized to grant the next transfer, as long as the defined rising or falling edge is not modified, and the TRIGSEL[6:0] is not modified, and the channel is enabled. After a first new trigger hit_{n+1} is memorized, if another second trigger hit_{n+2} is detected and if the hit_n triggered transfer is still not completed, hit_{n+2} is lost and not memorized. A trigger overrun flag is reported (GPDMA_CxSR.TOF = 1), and an interrupt is generated if enabled (GPDMA_CxCR.TOIE = 1). The channel is not automatically disabled by hardware due to a trigger overrun.

Transferring a next LLI_{n+1} that updates the GPDMA_CxTR2 with a new value for any of TRIGSEL[6:0] or TRIGPOL[1:0], resets the monitoring, trashing the memorized hit of the formerly defined LLI_n trigger.

Note: When the source block size is not a multiple of the source burst size and is a multiple of the source data width, then the last programmed source burst is not completed and is internally shorten to match the block size. In this case, if TRIGM[1:0] = 11 and (SWREQ = 1 or (SWREQ = 0 and DREQ = 0)), the shortened burst transfer (by singles or/and by bursts of lower length) is conditioned once by the trigger.

When the programmed destination burst is internally shortened by singles or/and by bursts of lower length (versus FIFO size, versus block size, 1-Kbyte boundary address crossing): if the trigger is conditioning the programmed destination burst (if TRIGM[1:0] = 11 and SWREQ = 0 and DREQ = 1), this shortened destination burst transfer is conditioned once by the trigger.

Bit 13 Reserved, must be kept at reset value.

Bit 12 **PFREQ**: Hardware request in peripheral flow control mode

Important: If a given channel x is not implemented with this feature, this bit is reserved and PFREQ is not present (see Section 19.3.1 for the list of the implemented channels with this feature).

If the channel x is activated (GPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer), this bit is ignored. Else:

0: the selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol in GPDMA control mode. The GPDMA is programmed with GPDMA_CxTR1.BNDT[15:0] and this is internally used by the hardware for the block transfer completion.

1: the selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol in peripheral control mode. The GPDMA block transfer can be early completed by the peripheral itself (see *Section 19.3.5* for more details).

Note: In peripheral flow control mode, there are the following restrictions:

- no 2D/repeated block support (GPDMA_CxBR1.BRC[10:0] must be set to 0)

- the peripheral must be set as the source of the transfer (DREQ = 0).

- data packing to a wider destination width is not supported (if destination width > source data width, GPDMA_CxTR1.PAM[1] must be set to 0).

- GPDMA_CxBR1.BNDT[15:0] must be programmed as a multiple of the source (peripheral) burst size.

Bit 11 **BREQ**: Block hardware request

If the channel x is activated (GPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer), this bit is ignored. Else:

0: the selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol at a burst level.

1: the selected hardware request is driven by a peripheral with a hardware request/acknowledge protocol at a block level (see *Section 19.3.3*).

Bit 10 **DREQ**: destination hardware request

This bit is ignored if channel x is activated (GPDMA_CxCR.EN asserted) with SWREQ = 1 (software request for a memory-to-memory transfer). Else:

0: selected hardware request driven by a source peripheral (request signal taken into account by the GPDMA transfer scheduler over the source/read port)

1: selected hardware request driven by a destination peripheral (request signal taken into account by the GPDMA transfer scheduler over the destination/write port)

Note: If the channel x is activated (GPDMA_CxCR.EN is asserted) with SWREQ = 0 and PFREQ = 1 (peripheral hardware request with peripheral flow-control mode), any software assertion to this DREQ bit is ignored: in peripheral flow-control mode, only a peripheral-to-memory transfer is supported.

Bit 9 **SWREQ**: software request

This bit is internally taken into account when GPDMA_CxCR.EN is asserted.

0: no software request. The selected hardware request REQSEL[7:0] is taken into account.

1: software request for a memory-to-memory transfer. The default selected hardware request as per REQSEL[7:0] is ignored.

Bit 8 Reserved, must be kept at reset value.

Bits 7:0 **REQSEL[7:0]**: GPDMA hardware request selection

These bits are ignored if channel x is activated (GPDMA_CxCR.EN asserted) with SWREQ = 1(software request for a memory-to-memory transfer). Else, the selected hardware request is internally taken into account as per *Section 19.3.3*.

Caution: The user must not assign a same input hardware request (same REQSEL[7:0] value) to different active GPDMA channels (GPDMA_CxCR.EN = 1 and GPDMA_CxTR2.SWREQ = 0 for these channels). GPDMA is not intended to hardware support the case of simultaneous enabled channels incorrectly configured with a same hardware peripheral request signal, and there is no user setting error reporting.

19.8.12 GPDMA channel x block register 1 (GPDMA_CxBR1)

Address offset: 0x98 + 0x80 * x (x = 0 to 11)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x at a block level.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when channel x is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, or LLI or full linked-list.

In linked-list mode, during the link transfer:

- if GPDMA_CxLLR.UB1 = 1, this register is automatically updated by the GPDMA from the next LLI in memory.
- If GPDMA_CxLLR.UB1 = 0 and if there is at least one linked-list register to be updated from the next LLI in memory, this register is automatically and internally restored with the programmed value for the field BNDT[15:0].
- If all the update bits GPDMA_CxLLR.Uxx are null and if GPDMA_CxLLR.LA[15:0] ≠ 0, the current LLI is the last one and is continuously executed: this register is automatically and internally restored with the programmed value for BNDT[15:0] after each execution of this final LLI
- If GPDMA_CxLLR = 0, this register and BNDT[15:0] are kept as null, channel x is completed.

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **BNDT[15:0]**: block number of data bytes to transfer from the source

Block size transferred from the source. When the channel is enabled, this field becomes read-only and is decremented, indicating the remaining number of data items in the current source block to be transferred. BNDT[15:0] is programmed in number of bytes, maximum source block size is 64 Kbytes -1.

Once the last data transfer is completed (BNDT[15:0] = 0):

- if GPDMA_CxLLR.UB1 = 1, this field is updated with the LLI in the memory.
- if GPDMA_CxLLR.UB1 = 0 and if there is at least one non null Uxx update bit, this field is internally restored to the programmed value.
- if all GPDMA_CxLLR.Uxx = 0 and if GPDMA_CxLLR.LA[15:0] = 0, this field is internally restored to the programmed value (infinite/continuous last LLI).
- if GPDMA_CxLLR = 0, this field is kept as zero following the last LLI data transfer.

Note: A non-null source block size must be a multiple of the source data width (BNDT[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued.

When configured in packing mode (GPDMA_CxTR1.PAM[1] = 1 and destination data width different from source data width), a non-null source block size must be a multiple of the destination data width (BNDT[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued.

19.8.13 GPDMA channel x alternate block register 1 (GPDMA_CxBR1)

Address offset: 0x98 + 0x80 * x (x = 12 to 15)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x at a block level.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when channel x is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, or LLI or full linked-list.

In linked-list mode, during the link transfer:

- if GPDMA_CxLLR.UB1 = 1, this register is automatically updated by the GPDMA from the next LLI in memory.
- If GPDMA_CxLLR.UB1 = 0 and if there is at least one linked-list register to be updated from the next LLI in memory, this register is automatically and internally restored with the programmed value for the fields BNDT[15:0] and BRC[10:0].
- If all the update bits GPDMA_CxLLR.Uxx are null and if GPDMA_CxLLR.LA[15:0] ≠ 0, the current LLI is the last one and is continuously executed: this register is

automatically and internally restored with the programmed value for the fields BNDT[15:0] and BRC[10:0] after each execution of this final LLI

- If GPDMA_CxLLR = 0, BNDT[15:0] and BRC[10:0] are kept as null, channel x is completed.

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

Bit 31 **BRDDEC**: Block repeat destination address decrement

0: at the end of a block transfer, the GPDMA_CxDAR register is updated by adding the programmed offset GPDMA_CxBR2.BRDAO to the current GPDMA_CxDAR value (current destination address)

1: at the end of a block transfer, the GPDMA_CxDAR register is updated by subtracting the programmed offset GPDMA_CxBR2.BRDAO from the current GPDMA_CxDAR value (current destination address)

Note: On top of this increment/decrement (depending on BRDDEC), GPDMA_CxDAR is in the same time also updated by the increment/decrement (depending on DDEC) of the GPDMA_CxTR3.DAO value, as it is usually done at the end of each programmed burst transfer.

Bit 30 **BRSDEC**: Block repeat source address decrement

0: at the end of a block transfer, the GPDMA_CxSAR register is updated by adding the programmed offset GPDMA_CxBR2.BRSAO to the current GPDMA_CxSAR value (current source address)

1: at the end of a block transfer, the GPDMA_CxSAR register is updated by subtracting the programmed offset GPDMA_CxBR2.BRSAO from the current GPDMA_CxSAR value (current source address)

Note: On top of this increment/decrement (depending on BRSDEC), GPDMA_CxSAR is in the same time also updated by the increment/decrement (depending on SDEC) of the GPDMA_CxTR3.SAO value, as it is done after any programmed burst transfer.

Bit 29 **DDEC**: destination address decrement

0: At the end of a programmed burst transfer to the destination, the GPDMA_CxDAR register is updated by adding the programmed offset GPDMA_CxTR3.DAO to the current GPDMA_CxDAR value (current destination address)

1: At the end of a programmed burst transfer to the destination, the GPDMA_CxDAR register is updated by subtracting the programmed offset GPDMA_CxTR3.DAO to the current GPDMA_CxDAR value (current destination address)

Bit 28 **SDEC**: source address decrement

0: At the end of a programmed burst transfer from the source, the GPDMA_CxSAR register is updated by adding the programmed offset GPDMA_CxTR3.SAO to the current GPDMA_CxSAR value (current source address)

1: At the end of a programmed burst transfer from the source, the GPDMA_CxSAR register is updated by subtracting the programmed offset GPDMA_CxTR3.SAO to the current GPDMA_CxSAR value (current source address)

Bit 27 Reserved, must be kept at reset value.

Bits 26:16 **BRC[10:0]**: Block repeat counter

This field contains the number of repetitions of the current block (0 to 2047).

When the channel is enabled, this field becomes read-only. After decrements, this field indicates the remaining number of blocks, excluding the current one. This counter is hardware decremented for each completed block transfer.

Once the last block transfer is completed (BRC[10:0] = BNDT[15:0] = 0):

– If GPDMA_CxLLR.UB1 = 1, all GPDMA_CxBR1 fields are updated by the next LLI in the memory.

– If GPDMA_CxLLR.UB1 = 0 and if there is at least one not null Uxx update bit, this field is internally restored to the programmed value.

– if all GPDMA_CxLLR.Uxx = 0 and if GPDMA_CxLLR.LA[15:0] ≠ 0, this field is internally restored to the programmed value (infinite/continuous last LLI).

– if GPDMA_CxLLR = 0, this field is kept as zero following the last LLI and data transfer.

Bits 15:0 **BNDT[15:0]**: block number of data bytes to transfer from the source

Block size transferred from the source. When the channel is enabled, this field becomes read-only and is decremented, indicating the remaining number of data items in the current source block to be transferred. BNDT[15:0] is programmed in number of bytes, maximum source block size is 64 Kbytes -1.

Once the last data transfer is completed (BNDT[15:0] = 0):

- if GPDMA_CxLLR.UB1 = 1, this field is updated with the LLI in the memory.

- if GPDMA_CxLLR.UB1 = 0 and if there is at least one not null Uxx update bit, this field is internally restored to the programmed value.

- if all GPDMA_CxLLR.Uxx = 0 and if GPDMA_CxLLR.LA[15:0] ≠ 0, this field is internally restored to the programmed value (infinite/continuous last LLI).

- if GPDMA_CxLLR = 0, this field is kept as zero following the last LLI data transfer.

Note: A non-null source block size must be a multiple of the source data width (BNDT[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued.

When configured in packing mode (GPDMA_CxTR1.PAM[1] = 1 and destination data width different from source data width), a non-null source block size must be a multiple of the destination data width (BNDT[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued.

19.8.14 GPDMA channel x source address register (GPDMA_CxSAR)

Address offset: 0x9C + 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register configures the source start address of a transfer.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1, and continuously updated by hardware, in order to reflect the address of the next burst transfer from the source.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the GPDMA from the memory if GPDMA_CxLLR.USA = 1.

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

Bits 31:0 **SA[31:0]**: source address

This field is the pointer to the address from which the next data is read.

During the channel activity, depending on the source addressing mode (GPDMA_CxTR1.SINC), this field is kept fixed or incremented by the data width (GPDMA_CxTR1.SDW_LOG2[1:0]) after each burst source data, reflecting the next address from which data is read.

During the channel activity, this address is updated after each completed source burst, consequently to:

– the programmed source burst; either in fixed addressing mode or in contiguous-data incremented mode. If contiguously incremented (GPDMA_CxTR1.SINC = 1), then the additional address offset value is the programmed burst size, as defined by GPDMA_CxTR1.SBL_1[5:0] and GPDMA_CxTR1.SDW_LOG2[1:0]

– the additional source incremented/decremented offset value as programmed by GPDMA_CxBR1.SDEC and GPDMA_CxTR3.SAO[12:0].

– once/if completed source block transfer, for a channel x with 2D addressing capability (x = 12 to 15). additional block repeat source incremented/decremented offset value as programmed by GPDMA_CxBR1.BRSDEC and GPDMA_CxBR2.BRSAO[15:0]

In linked-list mode, after a LLI data transfer is completed, this register is automatically updated by GPDMA from the memory, provided the LLI is set with GPDMA_CxLLR.USA = 1.

Note: A source address must be aligned with the programmed data width of a source burst (SA[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0]). Else, a user setting error is reported and no transfer is issued.

When the source block size is not a multiple of the source burst size and is a multiple of the source data width, the last programmed source burst is not completed and is internally shorten to match the block size. In this case, the additional GPDMA_CxTR3.SAO[12:0] is not applied.

19.8.15 GPDMA channel x destination address register (GPDMA_CxDAR)

Address offset: 0xA0 + 0x80 * x (x = 0 to 15)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register configures the destination start address of a transfer.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1, and continuously updated by hardware, in order to reflect the address of the next burst transfer to the destination.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by GPDMA from the memory if GPDMA_CxLLR.UDA = 1.

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

Bits 31:0 **DA[31:0]**: destination address

This field is the pointer to the address from which the next data is written.

During the channel activity, depending on the destination addressing mode (GPDMA_CxTR1.DINC), this field is kept fixed or incremented by the data width (GPDMA_CxTR1.DDW_LOG2[1:0]) after each burst destination data, reflecting the next address from which data is written.

During the channel activity, this address is updated after each completed destination burst, consequently to:

– the programmed destination burst; either in fixed addressing mode or in contiguous-data incremented mode. If contiguously incremented (GPDMA_CxTR1.DINC = 1), then the additional address offset value is the programmed burst size, as defined by GPDMA_CxTR1.DBL_1[5:0] and GPDMA_CxTR1.DDW_LOG2[1:0]

– the additional destination incremented/decremented offset value as programmed by GPDMA_CxBR1.DDEC and GPDMA_CxTR3.DAO[12:0].

– once/if completed destination block transfer, for a channel x with 2D addressing capability (x = 12 to 15), the additional block repeat destination incremented/decremented offset value as programmed by GPDMA_CxBR1.BRDDEC and GPDMA_CxBR2.BRDAO[15:0]

In linked-list mode, after a LLI data transfer is completed, this register is automatically updated by the GPDMA from the memory, provided the LLI is set with GPDMA_CxLLR.UDA = 1.

Note: A destination address must be aligned with the programmed data width of a destination burst (DA[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0]). Else, a user setting error is reported and no transfer is issued.

19.8.16 GPDMA channel x transfer register 3 (GPDMA_CxTR3)

Address offset: 0xA4 + 0x80 * x (x = 12 to 15)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the GPDMA from the memory if GPDMA_CxLLR.UT3 = 1.

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

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:16 **DAO[12:0]**: destination address offset increment

The destination address, pointed by GPDMA_CxDAR, is incremented or decremented (depending on GPDMA_CxBR1.DDEC) by this offset DAO[12:0] for each programmed destination burst. This offset is not including and is added to the programmed burst size when the completed burst is addressed in incremented mode (GPDMA_CxTR1.DINC = 1).

Note: A destination address offset must be aligned with the programmed data width of a destination burst (DAO[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0]). Else, a user setting error is reported and no transfer is issued.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:0 **SAO[12:0]**: source address offset increment

The source address, pointed by GPDMA_CxSAR, is incremented or decremented (depending on GPDMA_CxBR1.SDEC) by this offset SAO[12:0] for each programmed source burst. This offset is not including and is added to the programmed burst size when the completed burst is addressed in incremented mode (GPDMA_CxTR1.SINC = 1).

Note: A source address offset must be aligned with the programmed data width of a source burst (SAO[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0]). Else a user setting error is reported and none transfer is issued.

When the source block size is not a multiple of the destination burst size, and is a multiple of the source data width, then the last programmed source burst is not completed and is internally shorten to match the block size. In this case, the additional GPDMA_CxTR3.SAO[12:0] is not applied.

19.8.17 GPDMA channel x block register 2 (GPDMA_CxBR2)

Address offset: 0xA8 + 0x80 * x (x = 12 to 15)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register controls the transfer of a channel x at a 2D/repeated block level.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block, 2D/repeated block, LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the GPDMA from the memory if GPDMA_CxLLR.UB2 = 1.

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

Bits 31:16 **BRDAO[15:0]**: Block repeated destination address offset

For a channel with 2D addressing capability, this field is used to update (by addition or subtraction depending on GPDMA_CxBR1.BRDDEC) the current destination address (GPDMA_CxDAR) at the end of a block transfer.

Note: A block repeated destination address offset must be aligned with the programmed data width of a destination burst (BRDAO[2:0] versus GPDMA_CxTR1.DDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued. BRDAO[15:0] must be set to 0 in peripheral flow-control mode (if GPDMA_CxTR2.PFREQ = 1).

Bits 15:0 **BRSAO[15:0]**: Block repeated source address offset

For a channel with 2D addressing capability, this field is used to update (by addition or subtraction depending on GPDMA_CxBR1.BRSDEC) the current source address (GPDMA_CxSAR) at the end of a block transfer.

Note: A block repeated source address offset must be aligned with the programmed data width of a source burst (BRSAO[2:0] versus GPDMA_CxTR1.SDW_LOG2[1:0]). Else a user setting error is reported and no transfer is issued. BRSAO[15:0] must be set to 0 in peripheral flow-control mode (if GPDMA_CxTR2.PFREQ = 1).

19.8.18 GPDMA channel x linked-list address register (GPDMA_CxLLR)

Address offset: 0xCC + 0x80 * x (x = 0 to 11)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register configures the data structure of the next LLI in the memory and its address pointer. A channel transfer is completed when this register is null.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the GPDMA from the memory if GPDMA_CxLLR.ULL = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UT1	UT2	UB1	USA	UDA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ULL
rw	rw	rw	rw	rw											rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LA[15:2]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bit 31 **UT1**: Update GPDMA_CxTR1 from memory

This bit controls the update of GPDMA_CxTR1 from the memory during the link transfer.
0: no GPDMA_CxTR1 update
1: GPDMA_CxTR1 update

Bit 30 **UT2**: Update GPDMA_CxTR2 from memory

This bit controls the update of GPDMA_CxTR2 from the memory during the link transfer.
0: no GPDMA_CxTR2 update
1: GPDMA_CxTR2 update

Bit 29 **UB1**: Update GPDMA_CxBR1 from memory

This bit controls the update of GPDMA_CxBR1 from the memory during the link transfer.
If UB1 = 0 and if GPDMA_CxLLR ≠ 0, the linked-list is not completed.
GPDMA_CxBR1.BNDT[15:0] is then restored to the programmed value after data transfer is completed and before the link transfer.
0: no GPDMA_CxBR1 update from memory (GPDMA_CxBR1.BNDT[15:0] restored if any link transfer)
1: GPDMA_CxBR1 update

Bit 28 **USA**: update GPDMA_CxSAR from memory

This bit controls the update of GPDMA_CxSAR from the memory during the link transfer.
0: no GPDMA_CxSAR update
1: GPDMA_CxSAR update

Bit 27 **UDA**: Update GPDMA_CxDAR register from memory

This bit is used to control the update of GPDMA_CxDAR from the memory during the link transfer.

0: no GPDMA_CxDAR update

1: GPDMA_CxDAR update

Bits 26:17 Reserved, must be kept at reset value.

Bit 16 **ULL**: Update GPDMA_CxLLR register from memory

This bit is used to control the update of GPDMA_CxLLR from the memory during the link transfer.

0: no GPDMA_CxLLR update

1: GPDMA_CxLLR update

Bits 15:2 **LA[15:2]**: pointer (16-bit low-significant address) to the next linked-list data structure

If UT1 = UT2 = UB1 = USA = UDA = ULL = 0 and if LA[15:2] = 0, the current LLI is the last one. The channel transfer is completed without any update of the linked-list GPDMA register file.

Else, this field is the pointer to the memory address offset from which the next linked-list data structure is automatically fetched from, once the data transfer is completed, in order to conditionally update the linked-list GPDMA internal register file (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR, and GPDMA_CxLLR).

Caution: The user must program this pointer not to exceed the 64-Kbyte addressable space defined by the link base address register GPDMA_CxLBAR. The complete linked-list data structure must be included in the 64-Kbyte addressable space.

Note: The user must program the pointer to be 32-bit aligned. The two low-significant bits are write ignored.

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

19.8.19 GPDMA channel x alternate linked-list address register (GPDMA_CxLLR)

Address offset: 0xCC + 0x80 * x (x = 12 to 15)

Reset value: 0x0000 0000

This register is secure or nonsecure depending on the secure state of channel x (GPDMA_SECCFGR.SECx), and privileged or unprivileged, depending on the privileged state of channel x (GPDMA_PRIVCFGR.PRIVx).

This register configures the data structure of the next LLI in the memory and its address pointer. A channel transfer is completed when this register is null.

This register must be written when GPDMA_CxCR.EN = 0.

This register is read-only when GPDMA_CxCR.EN = 1.

This register must be written when the channel is completed (then the hardware has deasserted GPDMA_CxCR.EN). A channel transfer can be completed and programmed at different levels: block or LLI or full linked-list.

In linked-list mode, during the link transfer, this register is automatically updated by the GPDMA from the memory if GPDMA_CxLLR.ULL = 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UT1	UT2	UB1	USA	UDA	UT3	UB2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ULL
rw	rw	rw	rw	rw	rw	rw									rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LA[15:2]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bit 31 **UT1**: Update GPDMA_CxTR1 from memory

This bit controls the update of GPDMA_CxTR1 from the memory during the link transfer.

0: no GPDMA_CxTR1 update

1: GPDMA_CxTR1 update

Bit 30 **UT2**: Update GPDMA_CxTR2 from memory

This bit controls the update of GPDMA_CxTR2 from the memory during the link transfer.

0: no GPDMA_CxTR2 update

1: GPDMA_CxTR2 update

Bit 29 **UB1**: Update GPDMA_CxBR1 from memory

This bit controls the update of GPDMA_CxBR1 from the memory during the link transfer.
If UB1 = 0 and if GPDMA_CxLLR ≠ 0, the linked-list is not completed.
GPDMA_CxBR1.BNDT[15:0] is then restored to the programmed value after data transfer is completed and before the link transfer.

0: no GPDMA_CxBR1 update from memory (GPDMA_CxBR1.BNDT[15:0] restored if any link transfer)

1: GPDMA_CxBR1 update

Bit 28 **USA**: update GPDMA_CxSAR from memory

This bit controls the update of GPDMA_CxSAR from the memory during the link transfer.

0: no GPDMA_CxSAR update

1: GPDMA_CxSAR update

Bit 27 **UDA**: Update GPDMA_CxDAR register from memory
This bit is used to control the update of GPDMA_CxDAR from the memory during the link transfer.
0:no GPDMA_CxDAR update
1: GPDMA_CxDAR update

Bit 26 **UT3**: Update GPDMA_CxTR3 from memory
This bit controls the update of GPDMA_CxTR3 from the memory during the link transfer.
0: no GPDMA_CxTR3 update
1: GPDMA_CxTR3 update

Bit 25 **UB2**: Update GPDMA_CxBR2 from memory
This bit controls the update of GPDMA_CxBR2 from the memory during the link transfer.
0: no GPDMA_CxBR2 update
1: GPDMA_CxBR2 update

Bits 24:17 Reserved, must be kept at reset value.

Bit 16 **ULL**: Update GPDMA_CxLLR register from memory
This bit is used to control the update of GPDMA_CxLLR from the memory during the link transfer.
0: no GPDMA_CxLLR update
1: GPDMA_CxLLR update

Bits 15:2 **LA[15:2]**: pointer (16-bit low-significant address) to the next linked-list data structure
If UT1 = UT2 = UB1 = USA = UDA = ULL = 0 and if LA[15:2] = 0, the current LLI is the last one. The channel transfer is completed without any update of the linked-list GPDMA register file.
Else, this field is the pointer to the memory address offset from which the next linked-list data structure is automatically fetched from, once the data transfer is completed, in order to conditionally update the linked-list GPDMA internal register file (GPDMA_CxTR1, GPDMA_CxTR2, GPDMA_CxBR1, GPDMA_CxSAR, GPDMA_CxDAR, and GPDMA_CxLLR).

Caution: The user must program this pointer not to exceed the 64-Kbyte addressable space defined by the link base address register GPDMA_CxLBAR. The complete linked-list data structure must be included in the 64-Kbyte addressable space.

Note: The user must program the pointer to be 32-bit aligned. The two low-significant bits are write ignored.

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

19.8.20 GPDMA register map

Table 107. GPDMA register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	GPDMA_ SECCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	GPDMA_ PRIVCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 107. GPDMA register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x08	GPDMA_ RCFGLOCKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCK15	LOCK14	LOCK13	LOCK12	LOCK11	LOCK10	LOCK9	LOCK8	LOCK7	LOCK6	LOCK5	LOCK4	LOCK3	LOCK2	LOCK1	LOCK0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	GPDMA_MISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MIS15	MIS14	MIS13	MIS12	MIS11	MIS10	MIS9	MIS8	MIS7	MIS6	MIS5	MIS4	MIS3	MIS2	MIS1	MIS0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	GPDMA_SMISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MIS15	MIS14	MIS13	MIS12	MIS11	MIS10	MIS9	MIS8	MIS7	MIS6	MIS5	MIS4	MIS3	MIS2	MIS1	MIS0
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14 - 0x4C	Reserved	Reserved																															
0x50+0x80 * x (x = 0 to 15)	GPDMA_ CxLBAR	LBA[31:16]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																
0x5C+0x80 * x (x = 0 to 15)	GPDMA_CxFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																		0	0	0	0	0	0	0								
0x60+0x80 * x (x = 0 to 15)	GPDMA_CxSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FIFOL[7:0]								Res.	TOF	SUSPF	USEF	ULEF	DTEF	HTF	TCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDLEF
	Reset value									0	0	0	0	0	0	0	0		0	0	0	0	0	0	0								1
0x64+0x80 * x (x = 0 to 15)	GPDMA_CxCR	Res.	Res.	Res.	Res	Res.	Res.	Res.	Res.	PRIO [1:0]		Res.	Res.	Res.	Res.	LAP	LSM	Res.	TOIE	SUSPIE	USEIE	ULEIE	DTEIE	HTIE	TCIE	Res.	Res.	Res.	Res.	Res.	SUSP	RESET	EN
	Reset value									0	0					0	0		0	0	0	0	0	0	0						0	0	0
0x90+0x80 * x (x = 0 to 15)	GPDMA_CxTR1	DSEC	DAP	Res.	Res.	DHX	DBX	DBL_1[5:0]						DINC	Res.	DDW_LOG2 [1:0]		SSEC	SAP	SBX	PAM[1:0]		Res.	SBL_1[5:0]						SINC	Res.	SDW_LOG2 [1:0]	
	Reset value	0	0			0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0
0x94+0x80 * x (x = 0 to 15)	GPDMA_CxTR2	TCEM[1:0]		Res.	Res.	Res.	Res.	TRIGPOL [1:0]		Res.	TRIGSEL[6:0]							TRIGM [1:0]		Res.	PFREQ	BREQ	DREQ	SWREQ	Res.	REQSEL[7:0]							
	Reset value	0	0					0	0		0	0	0	0	0	0	0	0	0		0	0	0	0		0	0	0	0	0	0	0	0
0x98+0x80 * x (x = 0 to 11)	GPDMA_CxBR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BNDT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x98+0x80 * x (x = 12 to 15)	GPDMA_CxBR1	BRDDEC	BRSDEC	DDEC	SDEC	Res.	BRC[10:0]											BNDT[15:0]															
	Reset value	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x9C+0x80 * x (x = 0 to 15)	GPDMA_CxSAR	SA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xA0+0x80 * x (x = 0 to 15)	GPDMA_CxDAR	DA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xA4+0x80 * x (x = 12 to 15)	GPDMA_CxTR3	Res.	Res.	Res.	DAO[12:0]													Res.	Res.	Res.	SAO[12:0]												
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0
0xA8+0x80 * x (x = 12 to 15)	GPDMA_CxBR2	BRDAO[15:0]																BRSAO[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xCC+0x80 * x (x = 0 to 11)	GPDMA_CxLLR	UT1	UT2	UB1	USA	UDA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ULL	LA[15:2]														Res.	Res.
	Reset value	0	0	0	0	0											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0xCC+0x80 * x (x = 12 to 15)	GPDMA_CxLLR	UT1	UT2	UB1	USA	UDA	UT3	UB2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ULL	LA[15:2]														Res.	Res.
	Reset value	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

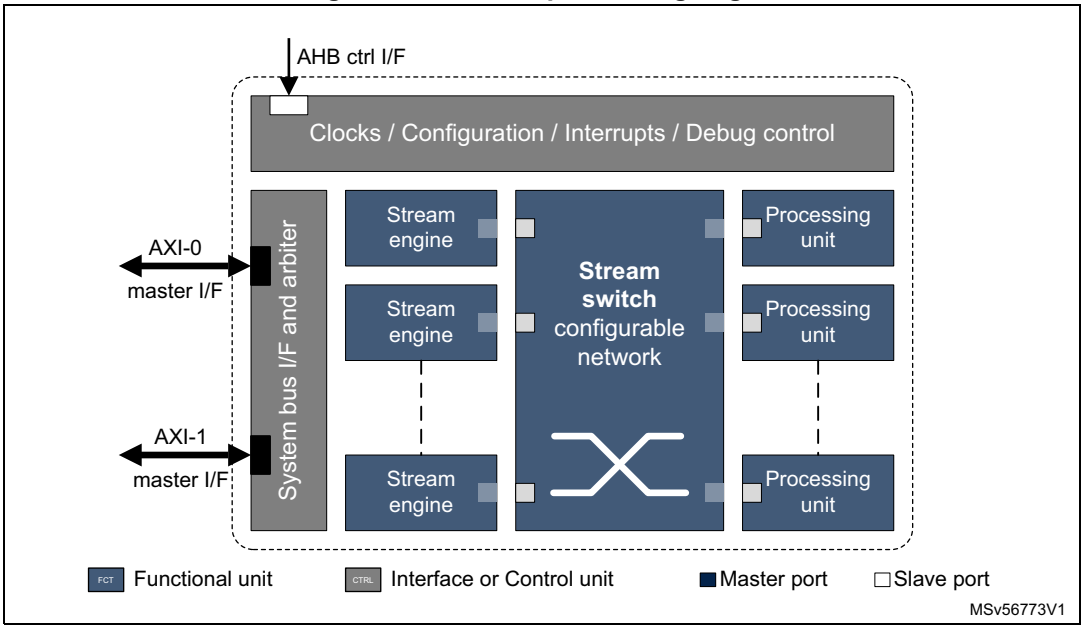
Refer to *Section 2.3: Memory organization* for the register boundary addresses.

20 Neural-ART accelerator™ (NPU)

20.1 NPU introduction

Neural-ART accelerator is a branded family of design-time parametric and runtime reconfigurable neural processing unit (NPU) cores. These cores are specifically designed to accelerate (in area-constrained embedded and IoT devices) the inference execution of a wide range of quantized convolutional neural network (CNN) models.

Figure 114. Stream processing engine



Stream processing

The ST stream processing technology allows the creation of on-demand performance- and area-tailored instances to address artificial intelligence (AI) applications, from the lowest to the highest end class.

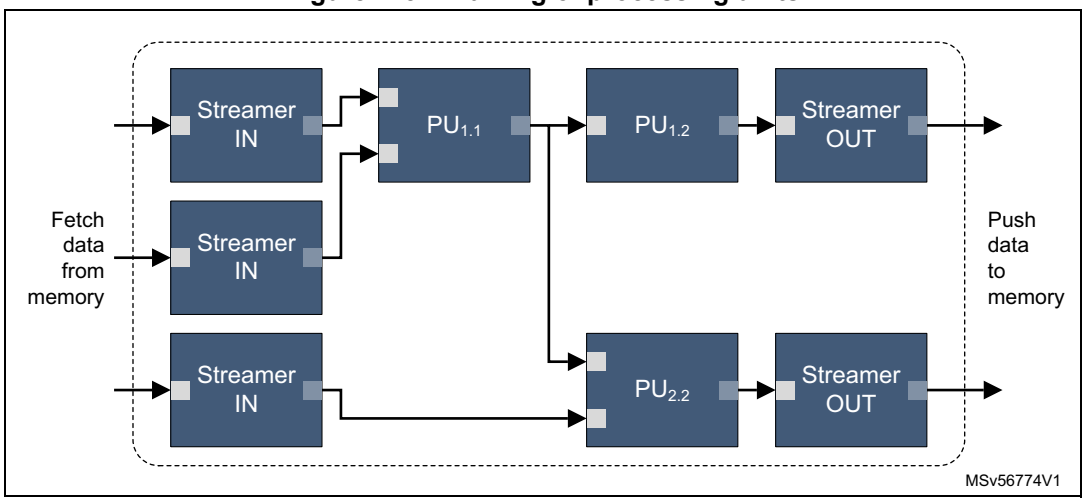
The ST stream processing technology is deemed far more flexible than a hardware data path, and more power-efficient than an on-chip memory bus. The core is a configurable data flow fabric, also known as stream switch, which can be programmed at runtime to create arbitrary virtual chains of processing units (PUs), implementing the sequence of operations to execute. Multiple chains can be active at the same time, provided there are enough available hardware resources.

In the virtual chains, data flow unidirectionally from one PU to the other, hence a processing chain cannot be faster than the slowest element in the chain. A back pressure mechanism controls the data flow in a virtual chain, and data streams can be easily forked or joined to feed multiple destinations, enabling multicasting, context switching, and filtering.

Chaining as many accelerators as possible in a single virtual chain reduces the number of useless and power consuming transfers from/to memory, thus reducing the overall memory traffic and the power consumption.

The resulting IP communicates with the rest of the system through stream engine units. These are smart half-DMAs that can read, and write the data to/from external memory to transfer from/to the IP internal buffers. Although not primarily intended for that, memory-to-memory transfers are possible, connecting two stream engines back to back.

Figure 115. Chaining of processing units



Epoch definition

The execution of CNN models is organized into indivisible processing sequences called epochs. A hardware epoch is defined by the sequence that starts with the configuration of the NPU by the host CPU, and ends with an interrupt sent back to the CPU. The host CPU can then trigger another hardware epoch or execute some software computing for the layers not accelerated by the hardware.

20.2 NPU implementation

The Neural-ART 14 NPU uses a 4CA configuration with the functional units listed in *Table 108*.

Table 108. Neural-ART 14 NPU configuration

Count	Unit	Description
1	STRSWITCH	Central data stream switch
2	DECUN	*Decompression unit*
4	CONVACC	*Convolutional accelerator*
2	POOL	*Pooling unit*
2	ACTIV	*Activation unit*
4	ARITH	*Arithmetic unit* for scalar arithmetic operators
1	RECBUF	*Reconfigurable buffer* (FIFO smart buffer)
10	STRENG	*Stream engines*, with embedded cipher
2	BUSIF	64-bit AXI interface
1	EPOCH	*Epoch controller*

Table 108. Neural-ART 14 NPU configuration (continued)

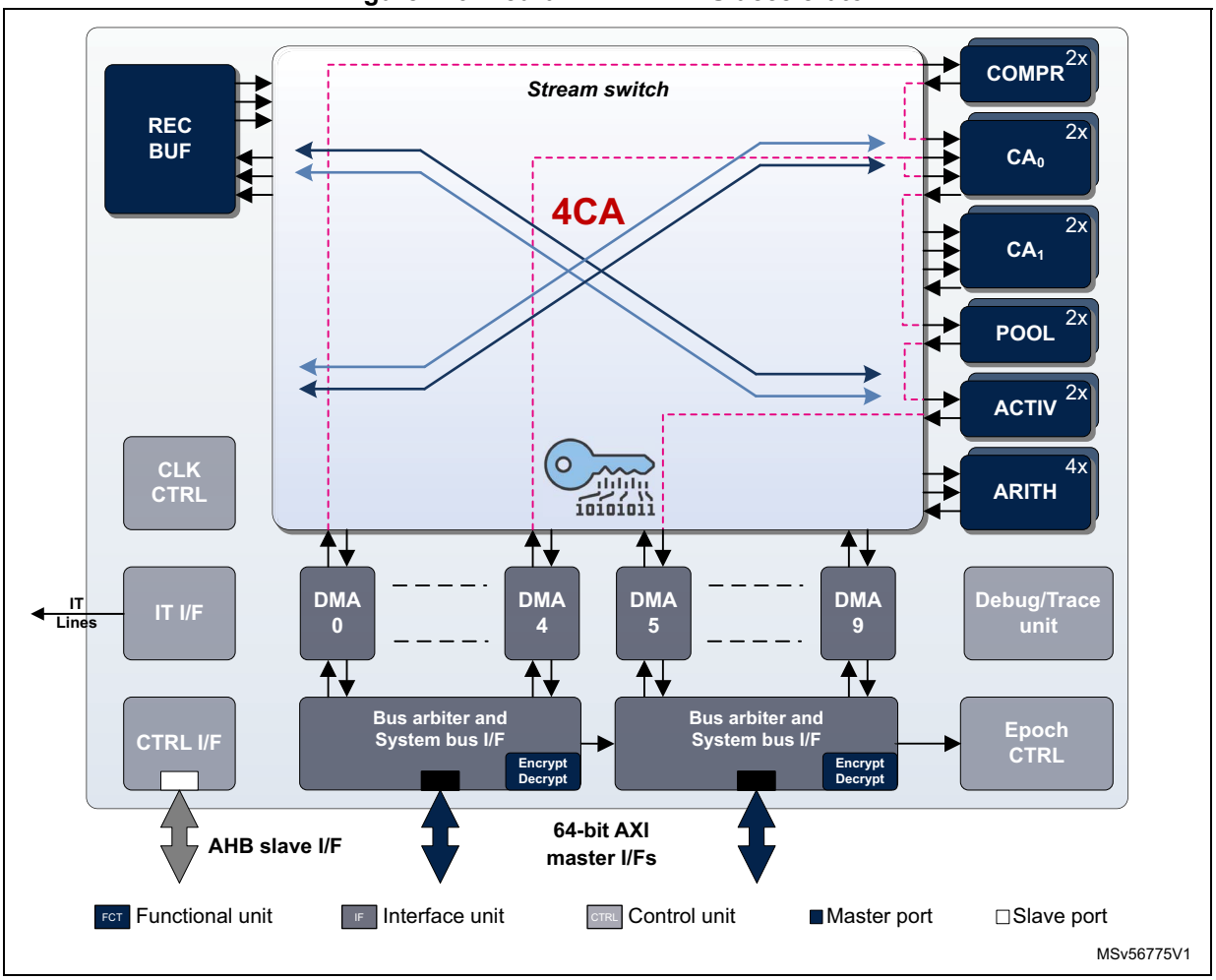
Count	Unit	Description
1	DEBUG	Debug controller
-	SERVICES	Configuration network, Clock and reset manager, Interrupt controller

20.3 NPU functional description

20.3.1 NPU block diagram

The internal organization of the NPU is shown in *Figure 116*.

Figure 116. Neural-ART 14 NPU accelerator



When the NPU is configured and started, it autonomously fetches data from the external memory to feed its internal dynamically interconnected processing units. Similarly, the result of the operation sequence is flushed to the external memory.

Inputs/sources can be the activations/features, but also parameters, such as weights and

biases constant data hosted in a memory-mapped nonvolatile memory. When the data-stream processing pipes are configured and started, the settings cannot be modified until the end-of-transfer event notifications of all DMA-outs and sinks.

20.3.2 NPU pins and internal signals

The NPU connects to the system through:

- One AHB port, used for configuring and controlling the execution of the AI model
- Two fully symmetrical AXI ports, used for fetching data and weights, and flushing outputs.

One interrupt signal fires when the processing for the current epoch is complete. Some additional signals are used for clock, reset, test, and debug. The complete list is given in *Table 109*.

Table 109. Neural-ART 14 NPU signals

Signal	Direction	Width	Description
ipu _clk	In	1	Main clock
ipu_rst_n	In	1	Reset
host_int	Out	4	Host interrupts
dbg_freeze_async	In	1	Debug freeze input signal (active high)
ext_triggers_async	In	4	External synchronization triggers (line based)
trig_sig	Out	8	Trigger signals to an external trace unit (available only in configurations including debug and trace unit)
AXI 0	AXI master port 0		See Arm specification
AXI 1	AXI master port 1		
AHB	AHB slave port		
ATP	In	26	Memory test port
TBYPASS	In	26	
TBISTx	In	26	Memory BIST
HS	In	26	Memory speed select
STDBY	In	26	Memory standby
PSWLARGEMA	In	26	Memory power enable
PSWLARGEMP	In	26	
PSWSMALLMA	In	26	
PSWSMALLMP	In	26	
SLEEP	In	26	Memory sleep

20.3.3 Configuration network

Each of the functional units, whether accelerator or interface, has a configuration space made of a collection of configuration registers. These registers are accessible through the AHB interface. Their detailed description is out of the scope of the present document.

The NPU embeds a configuration network that provides host processor access to the configuration space of the different units in the Neural-ART core. This configuration network connects to the system backbone through a 32-bit AHB5-lite port. It supports a single 32-bit wide read and write transaction, with no pipelining nor outstanding transactions.

20.3.4 Clock/reset manager

The purpose of this unit is twofold:

1. Clock control

   The NPU uses a single functional clock provided by the RCC. This input clock is fed to the clock control unit from where it is gated and fed to the different units of the IP.

   The clock unit generates clocks not gated, and clocks gated by default that need to be enabled after reset.

   Clock gating is controlled by a configuration register bank, accessible via the configuration network, and programmed by the control code generated by the NPU compiler tool chain.

2. Reset control

   The NPU uses a single functional reset provided by the RCC. This reset signal is synchronized inside the IP and fed to the different functional units.

   Setting the clear bit in the control configuration register triggers a global NPU reset.

20.3.5 Stream link

A stream link is a unidirectional interconnection to transport data streams from accelerator units, stream engines, and interfaces to the stream switch and vice versa.

The stream consists of data, messages, or commands, and additional signaling information.

- Data are transported over three 8-bit wide data channels that are either handled individually or can be interpreted as a single 24-bit data value.
- Transported data can use different formats. A raster scan data stream allows the automatic detection of the geometrical dimensions of the received feature data tensor (width and height). A raw data format requires to program additional parameters to handle correctly the received data stream.
- Signaling information is used to identify valid information on the stream link. It indicates the type and format of information transported, and whether this is the first or the last data transmitted.

20.3.6 Stream switch

This switch dynamically connects output stream link ports to input stream link ports, depending on the configuration data written to the configuration registers.

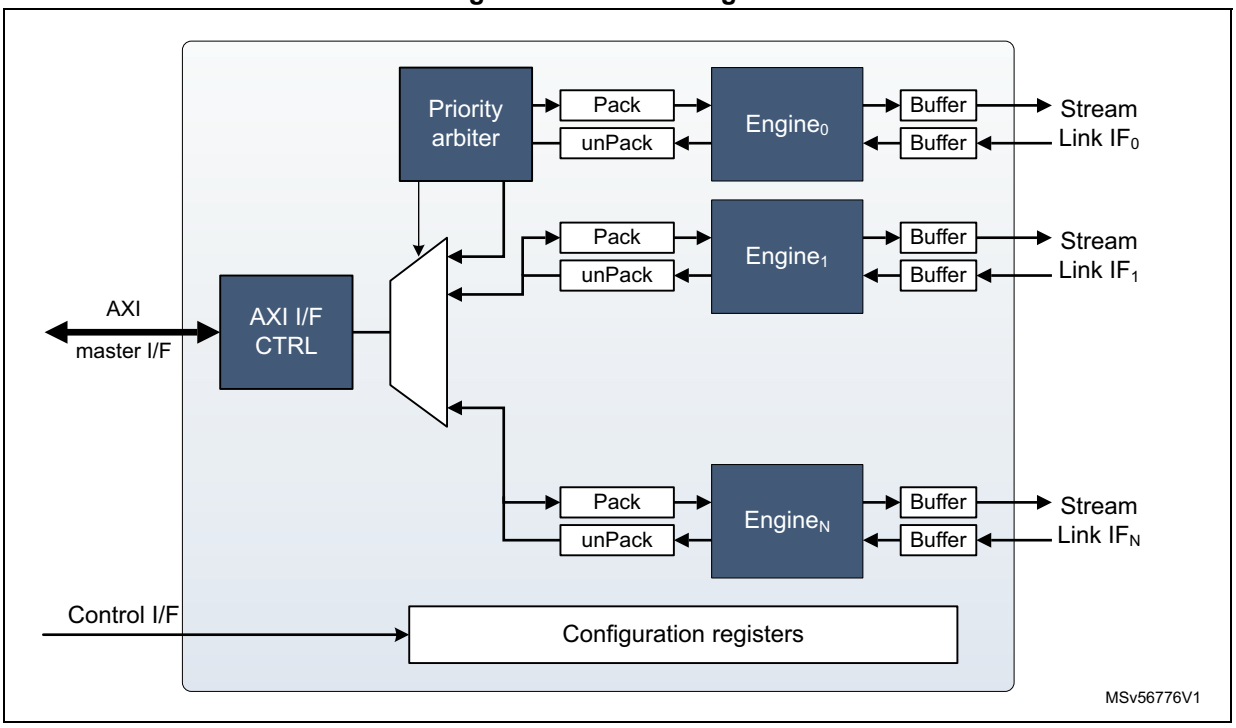
All input stream link ports can forward their data streams to one or multiple (multicast) output ports at the same time, allowing replication of the data stream.

Two stream links can be connected to a single output port of the switch. In this case, the output data are managed using a time-division context switch approach.

20.3.7 Stream engines

These engines manage the transfer from the system to the processing units and vice versa.

Figure 117. Stream engines



A stream engine connects to the stream switch by one input and one output stream link.

The stream engine offers a high configurability on read/write memory patterns:

- 2D and 3D data representations
- Gap management (interleaving)
- High flexibility of data format (size, rounding, subsampling or resampling, MSB or LSB alignment with sign extension).

Linked lists are supported.

20.3.8 Encryption/Decryption unit

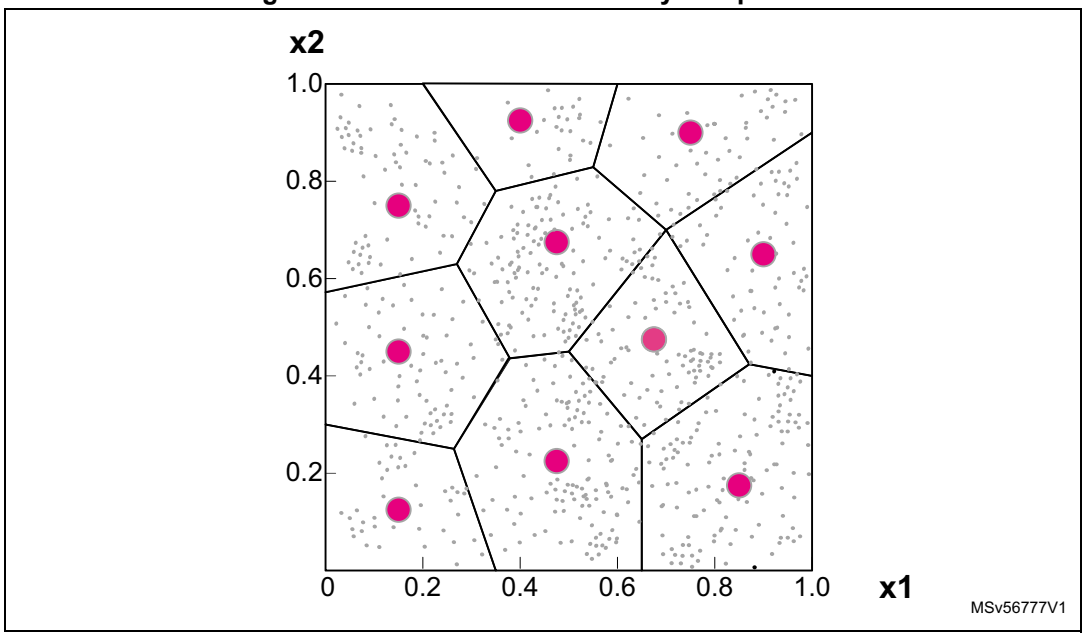
A low overhead/latency encryption/decryption unit based on Keccak-p[200] SHA-3 algorithm cipher, with a programmable number of rounds, is integrated in the bus interface, and can be shared between the different stream engines. It supports both weights and activation decryption and encryption.

20.3.9 Decompression unit

This unit supports on-the-fly decoding for scalar quantized data associated with kernels weights (with arbitrary quantization function), and decoding of vector quantized groups of weights.

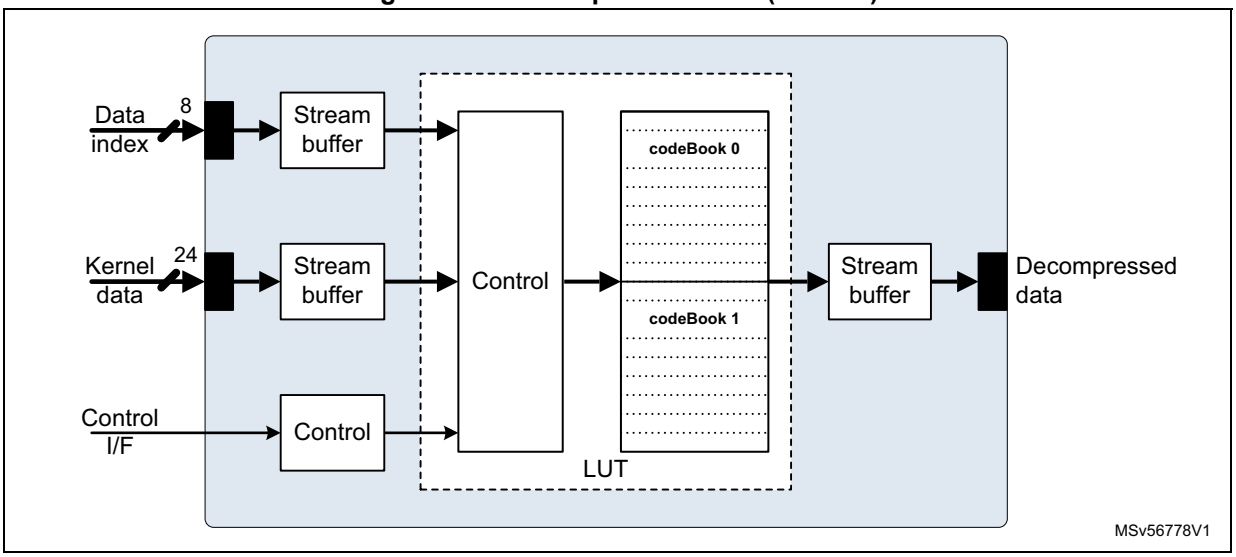
An n-dimensional space can be described in a lossy way, by subdividing it into subvolumes, each described by its centroid. The subvolumes are obtained to minimize a predefined distortion function. The whole signal space can be represented with huge savings in terms of storage, as it is possible to represent the space of interest using only centroid values.

Figure 118. Two dimensional lossy compression



Quantization functions and/or vector quantizer code books (CB) are defined with an offline tool part of the NPU compiler toolchain. The encoder maps the input range into a finite range of rational values, called the code book. Any value stored by the CB is called a code vector. A code vector is made by up to eight code words. Each code book can store up to 256 code vectors (CV).

Figure 119. Decompression unit (DECUN)



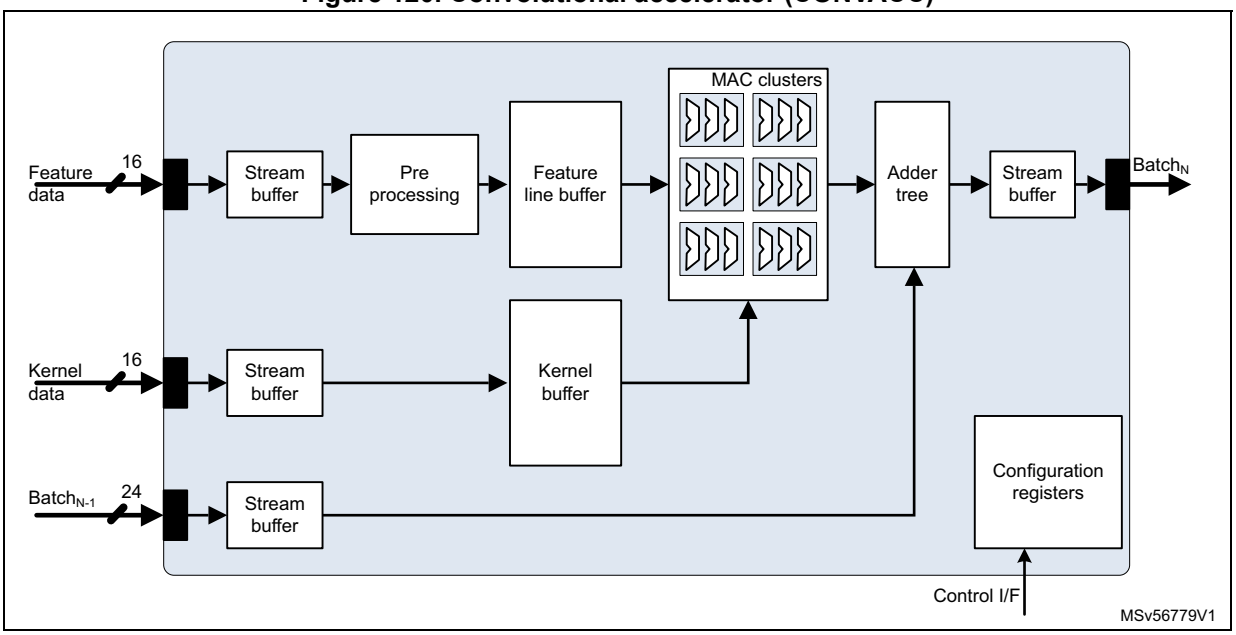
The decompression unit connects to the stream switch by two input stream links and one output stream link.

- iSL_0: receives the 8-bit indexes to decompress the data.
- iSL_1: receives the data to write in the code book, sent as raw data.

20.3.10 Convolutional accelerator

This is the core of the NPU. It implements clusters of multiply-accumulate (MAC) engines supporting fixed-point acceleration of the 3x3 convolutions used by many of the neural network models. It can be configured as 16- or 8-bit MACs, depending on which fixed-point precision for kernels and feature data is considered sufficient.

Figure 120. Convolutional accelerator (CONVACC)



The convolution accelerator connects to the stream switch by three input stream links and one output stream link.

- iSL_0: receives the feature data as 8- or 16-bit raster scan or raw data. Data go through a bit rate adaptation stream buffer and is then transmitted to the feature data preprocessing unit before it is stored in the feature line buffer.
- iSL_1: receives the kernel data as 8- or 16-bit raster scan or raw data.
- oSL: provides convolution results as 8- or 16-bit raster scan or raw data stream, with up to 24-bit wide signed data values.
- iSL_2: can receive intermediate 24-bit signed accumulator values to be accumulated with the convolution results.

Each convolution accelerator supports signed or unsigned arithmetic. It includes eighteen 16-bit MAC units grouped as six clusters of three MAC units. Each MAC performs a single 16x16-, two 16x8-, or four 8x8-bit multiply accumulate operation, which sums up to a maximum of 18 16x16-, 36 16x8-, or 72 8x8-bit multiply accumulate operations per clock cycle and per convolution accelerator unit.

The preprocessing unit implements the following sequence:

1. Format detection: determines if the data is raster scan or raw
2. Shift/round/saturation
3. HOR/VER cropping
4. Zero frame generation: on-the-fly extend the feature tensor structure.
5. Iteration control: simplifies the iterative processing of kernels larger than the 3x3 natively supported by the convolutional accelerator.

20.3.11 Pooling unit

This unit supports pooling operations like local 2D windowed (MxN) min, max, average pooling, as well as global max, min, or average pooling.

Supported features include:

- Native pooling windows of dimensions up to 3x3
- Horizontal and vertical strides ranging from 1 to 15
- Batch size ranging from 1 to 8
- Left, right, and top padding ranging from 0 to 7
- Bottom padding, depending on the window height

Figure 121. Example of pooling operations

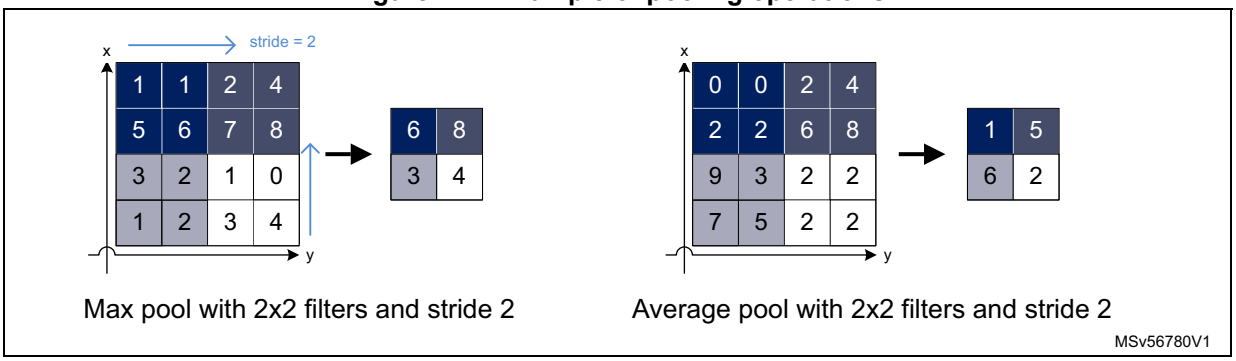
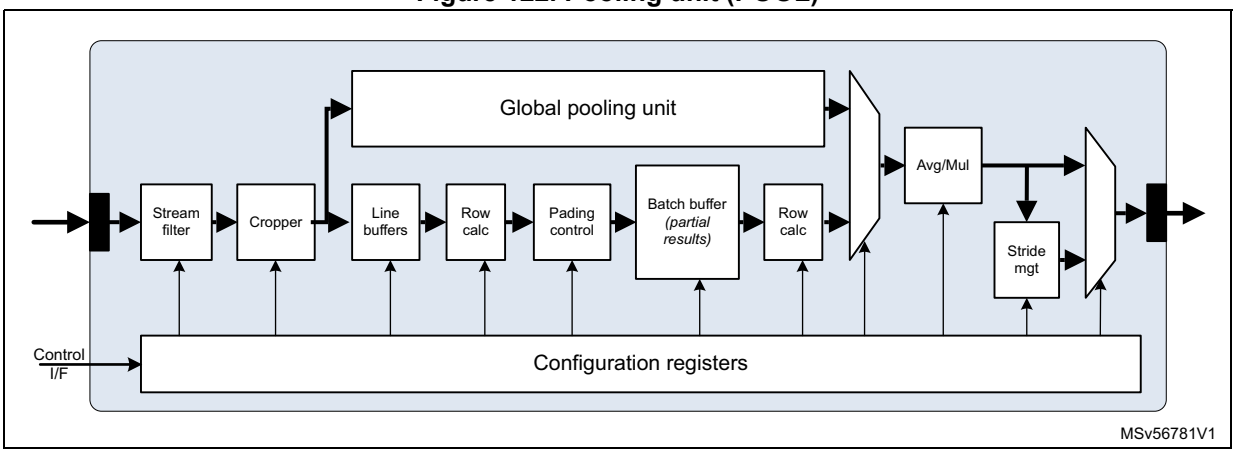


Figure 122. Pooling unit (POOL)



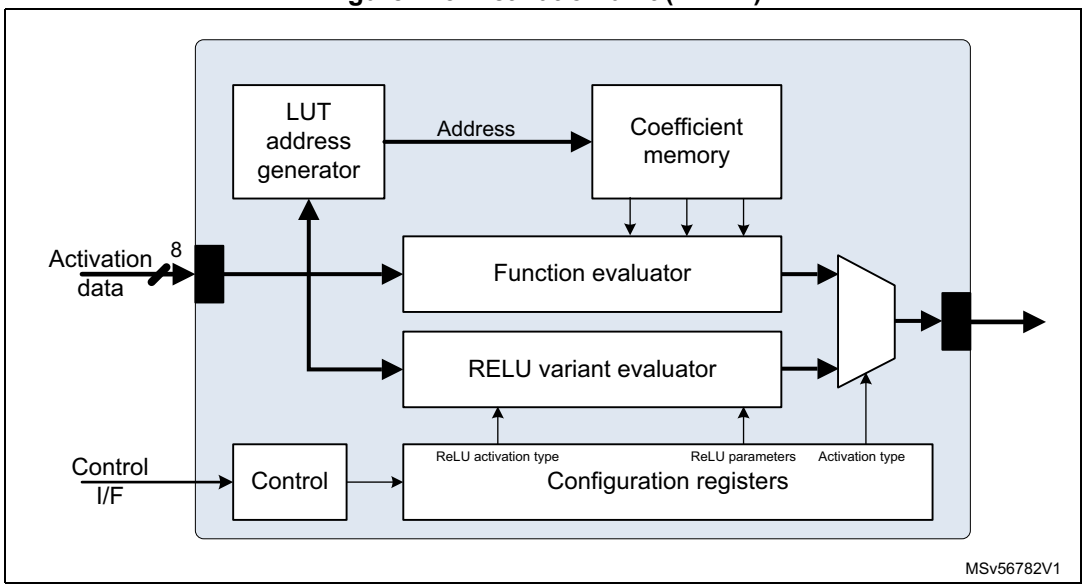
The pooling unit connects to the stream switch by one input and one output stream link:

- iSL_0: receives the activation input values as raster scan or raw data. Input is saturated to 16-bit.
- oSL: produces the pooled output as raw data.

20.3.12 Activation unit

This unit implements the activation functions associated with convolutional neural networks. It provides a dedicated path to compute ReLU-like activations such as plain, parametric, and thresholded ReLUs. There is also a generic function evaluator using a second-degree piece-wise polynomial approximation ($y = ax^2 + bx + c$ or $y = (ax + b) * x + c$). It uses a hierarchical uniform segmentation scheme to implement the evaluator.

Figure 123. Activation unit (ACTIV)



The unit connects to the stream switch by one input stream link and one output stream link.

- iSL_0: receives the activation input values as raster scan or raw data. Input is saturated to 16-bit.
- oSL: produces the result output as raw data.

Table 110 provides examples of usual activation functions.

Table 110. Activation functions example

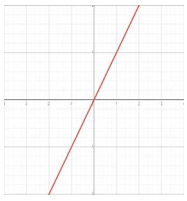
Function	Plot	Equation	Derivative
Identity		$f(x) = x$	$f(x) = 1$

Table 110. Activation functions example (continued)

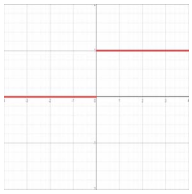
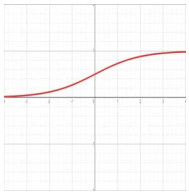
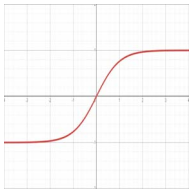
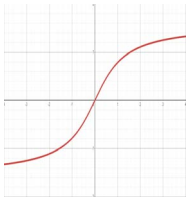
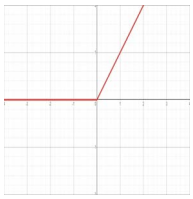
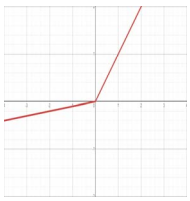
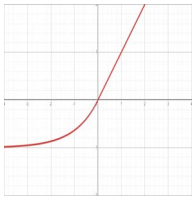
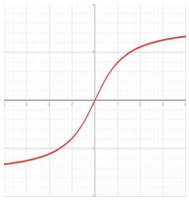
Function	Plot	Equation	Derivative
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ \infty & \text{for } x = 0 \end{cases}$
Logistic		$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1-f(x))$
Hyperbolic tangent		$f(x) = \tanh(x) = \left(\frac{2}{1+e^{-2x}} - 1\right)$	$f'(x) = 1 - f(x)^2$
Arctan		$f(x) = \operatorname{atan}(x)$	$f'(x) = \frac{1}{x^2+1}$
Rectified linear unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parametric rectified linear unit (PReLU)		$f(x) = \begin{cases} ax & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} a & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$

Table 110. Activation functions example (continued)

Function	Plot	Equation	Derivative
Exponential linear unit (ELU)		$f(x) = \begin{cases} a(e^x - 1) \text{ for } x < 0 \\ x \text{ for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + a \text{ for } x < 0 \\ 1 \text{ for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \ln(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$

20.3.13 Arithmetic unit

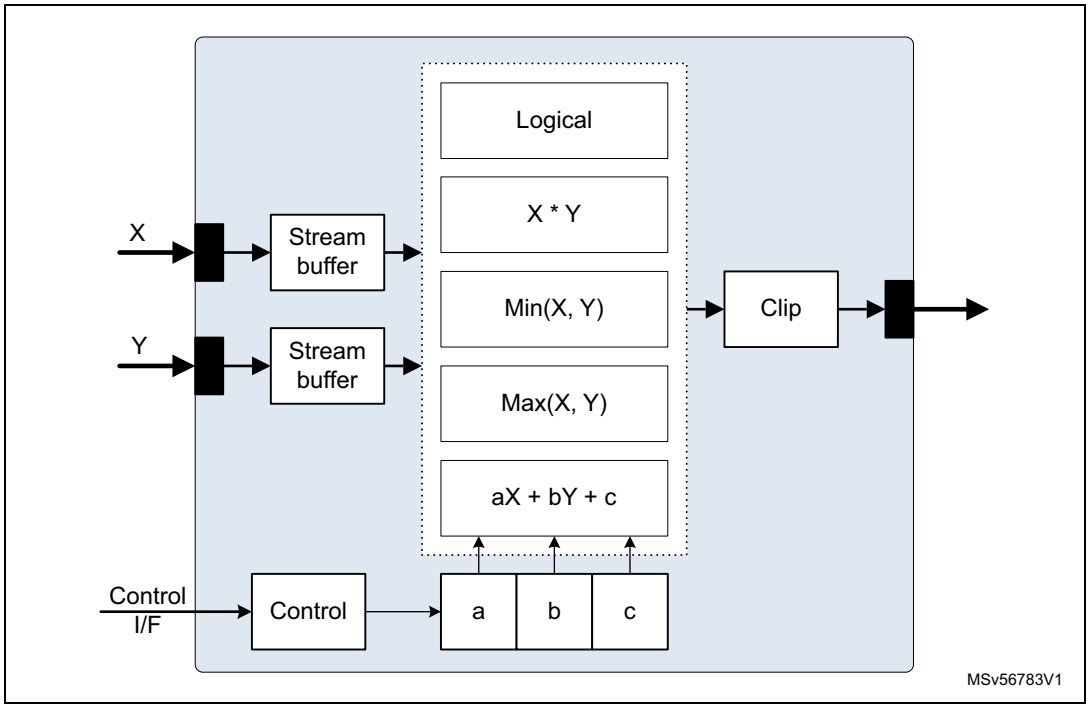
The arithmetic unit can handle outputs coming directly from the convolution accelerators. The main function is aX + bY+ c, where X and Y are input data streams, supplied via two input stream link interfaces and a, b and c are constants that are either vectors or scalars, which are preloaded into an internal memory via the configuration interface.

The arithmetic unit can be configured to perform other operations:

- x * y;
- MIN(x,y), MAX(x,y)
- Logical {AND, OR, NOT(X)}
- Bitwise {AND, OR, XOR, NOT(x)}
- Comparison operations like =, <, ≤, >, ≥
- Math operations like ABS(x), SIGN(x), and CLIP(x) with the 16-bit signed clip ranges supplied by the a coeff (min) and c coeff (max)

Based on the choice and type of coefficients, the unit can also perform element-wise operations like x - y, x + y, and similar ones.

Figure 124. Arithmetic unit (ARITH)



This unit connects to the stream switch by two input stream links and one output stream link.

- $iSL_{0,1}$: receives the input values, denoted X & Y, as raster scan or raw data. Input is saturated to 16-bit.
- oSL: produces the result output as raw data.

20.3.14 Reconfigurable buffer

The amount of data to process is usually divided using channel-wise data segmentation strategies. The data are then organized in execution epochs and channel data blocks along the three spatial dimensions. Data blocks are created considering the number of channels and the size of the incoming data. The partial data coming out from the different data blocks must then be reorganized and used in the next processing.

The reconfigurable buffer takes and stores the input data before making them available to the following units, which is useful to avoid deadlocks.

20.3.15 Epoch controller

This controller makes the NPU operation independent from the host CPU, which just provides a pointer to a 64-bit aligned “binary blob” placed in a memory space accessible through the AXI IF. The binary blob can be encrypted.

The core of the epoch controller is an FSM that decodes the control microinstructions, to appropriately configure all the processing units involved in the model execution using direct access to the NPU internal configuration bus.

The synchronization with the NPU is interrupt-based. An available step-by-step mode allows the execution of one microinstruction at a time. The execution is triggered by a write in a dedicated bit in the IRQ register.

The controller frees the AHB control bus for other uses, and improves the performance through a faster NPU programming. The binary blob is fetched through the AXI interface, which provides a larger bandwidth than the AHB. The direct access to the internal configuration bus allows bypassing all the bus external structure, where transactions from the host processor need to pass through.

20.3.16 Interrupt controller

Functional units can generate level-sensitive, active high, interrupts. These are collected by the interrupt controller, routed to four master interrupt lines, and forwarded to the host CPU.

When a master interrupt fires, the host CPU must read the INTREG register to identify the source of the current interrupt signaling. Interrupt routing to the four master interrupt lines and selective enabling are controlled by the INTORMSK and INTANDMSK mask registers, which have as many bits as interrupt sources.

- INTORMSK: an interrupt is forwarded if any of the unmasked interrupts is active.
- INTANDMSK: an interrupt is forwarded if all the unmasked interrupts are active.

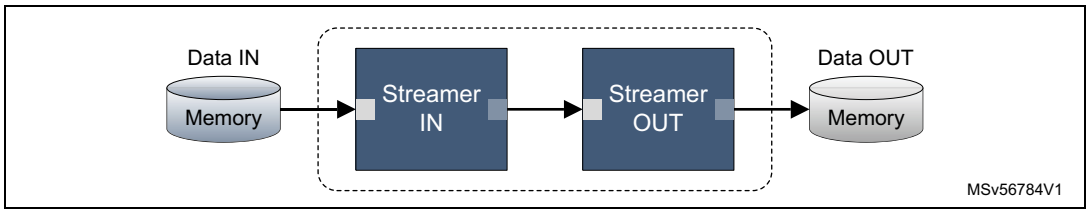
20.4 Configuration examples

The following figures illustrate typical examples of streaming processing supported by the Neural-ART 14 NPU.

20.4.1 DMA transfer

The simplest possible chain involves one input streamer and one output streamer connected back-to-back, which (although not primarily intended for that) configures the NPU as a high-bandwidth DMA.

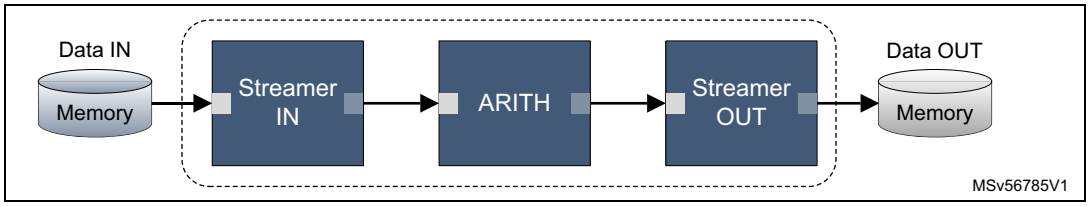
Figure 125. Memory to memory transfer



20.4.2 Simple processing

The simplest processing chain involves one input streamer and one output streamer, with one processing unit in between.

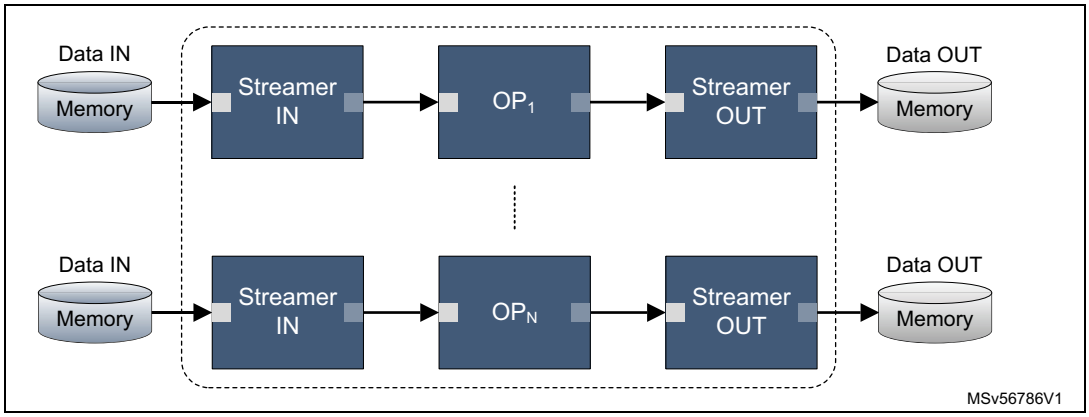
Figure 126. Simple processing



20.4.3 Multiple processing

Implements several independent simple processing chains, like the one in *Figure 126*.

Figure 127. Multiple processing

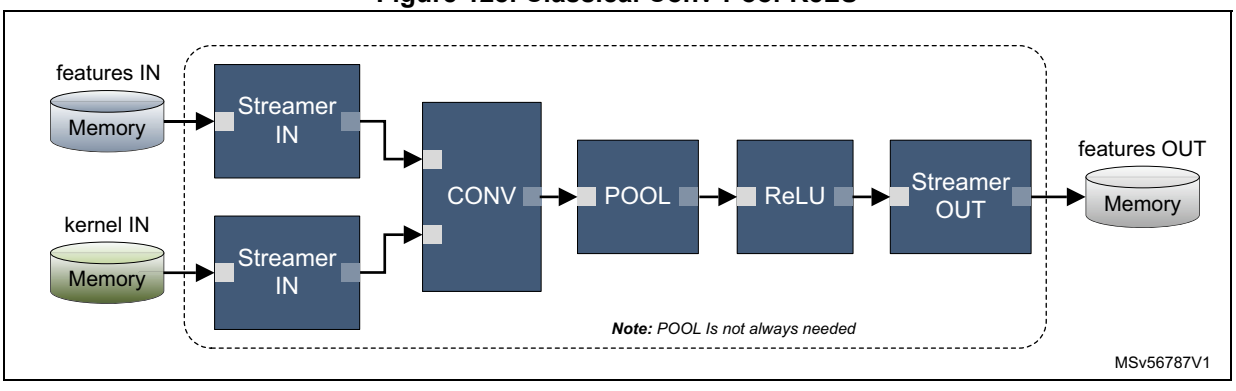


The number of independent chains is limited by the available hardware resources. It cannot exceed half the number of available stream engines, that is, five pairs of stream engines on Neural-ART 14, thus five chains.

20.4.4 Conv-Pool-ReLU

The virtual chain implementing the classical Conv-Pool-ReLU processing is illustrated in *Figure 128*.

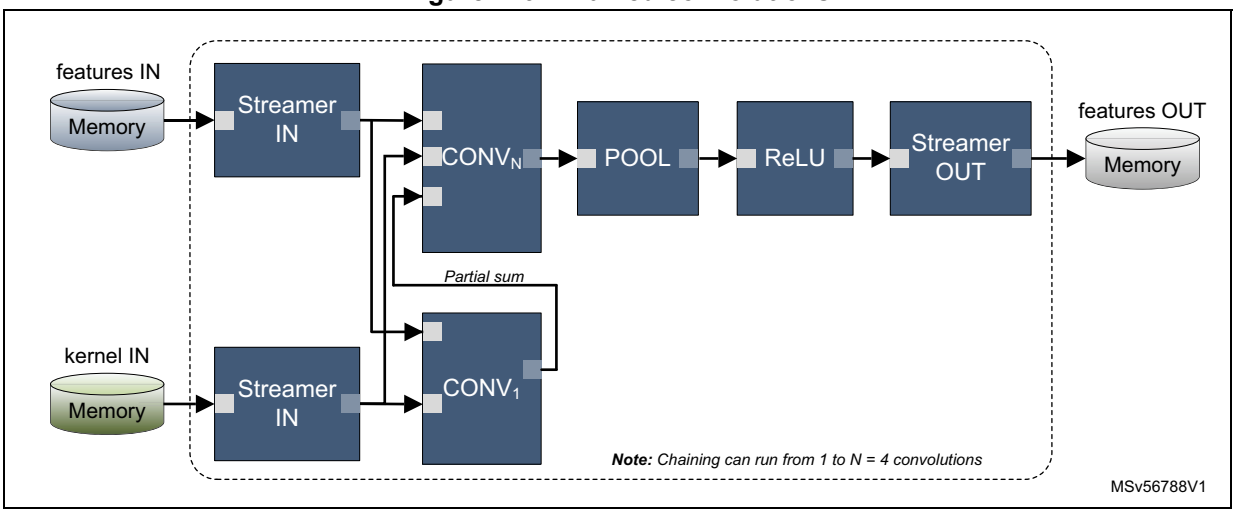
Figure 128. Classical Conv-Pool-ReLU



20.4.5 Chained convolutions

Figure 129 shows an example of chained convolutions.

Figure 129. Chained convolutions



Neural-ART 14 allows chaining up to four convolutions.

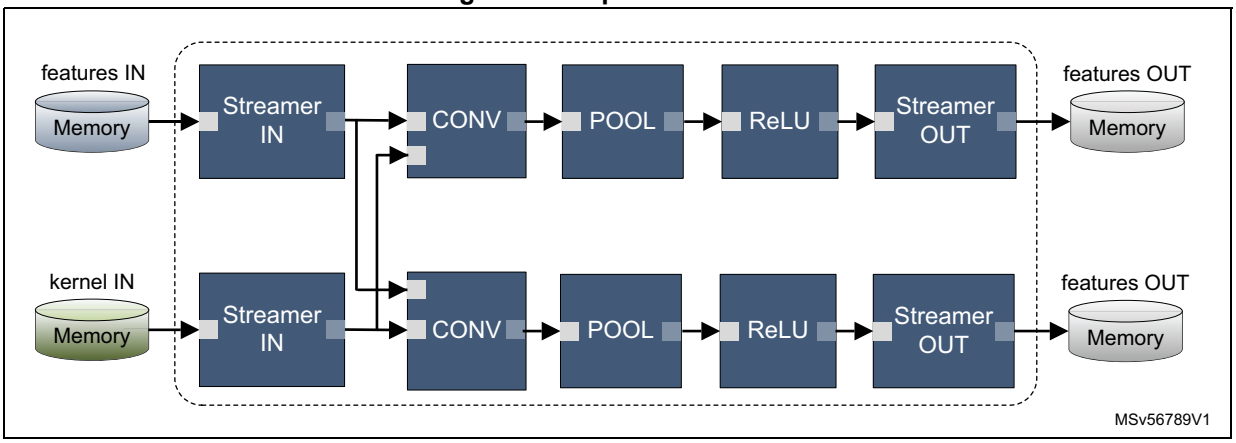
It is possible to use S CONVACCs in series, which requires (S times) more feature data bandwidth, but the same bandwidth for accumulation and output data as a single accelerator.

It is also possible to use P CONVACCs in parallel (see *Figure 127*), which reuse the same feature data requiring P times the amount of accumulation data and output data bandwidth.

20.4.6 Split convolutions

Figure 130 shows an example of split convolution.

Figure 130. Split convolution



20.5 Address space

The configuration bus address mapping depends on the specific NPU instance configuration. Each of the functional units has a 4-Kbyte configuration space, which yields a 128-Kbyte configuration space for the total IP, organized as shown in *Table 111*.

Table 111. Neural-ART 14 functional units memory base addresses(1)

Offset	Unit	Description
0x0000	CLKCTRL	Clock and reset manager
0x1000	INTCTRL	Interrupt controller
0x2000	BUSIF0	Bus interface 0
0x3000	BUSIF1	Bus interface 1
0x4000	STRSWITCH	Stream switch
0x5000	STRENG0	Stream engine 0
0x6000	STRENG1	Stream engine 1
0x7000	STRENG2	Stream engine 2
0x8000	STRENG3	Stream engine 3
0x9000	STRENG4	Stream engine 4
0xA000	STRENG5	Stream engine 5
0xB000	STRENG6	Stream engine 6
0xC000	STRENG7	Stream engine 7
0xD000	STRENG8	Stream engine 8
0xE000	STRENG9	Stream engine 9
0xF000	CONVACC0	Convolutional accelerator unit 1
0x10000	CONVACC1	Convolutional accelerator unit 2
0x11000	CONVACC2	Convolutional accelerator unit 3
0x12000	CONVACC3	Convolutional accelerator unit 4
0x13000	DECUN0	Decompression unit 0
0x14000	DECUN1	Decompression unit 1
0x15000	ACTIV0	Activation unit 0
0x16000	ACTIV1	Activation unit 1
0x17000	ARITH0	Arithmetic unit 0
0x18000	ARITH1	Arithmetic unit 1
0x19000	ARITH2	Arithmetic unit 2
0x1A000	ARITH3	Arithmetic unit 3
0x1B000	POOL0	Pooling unit 0
0x1C000	POOL1	Pooling unit 1
0x1D000	RECBUF0	Reconfigurable buffer 0

Table 111. Neural-ART 14 functional units memory base addresses(1) (continued)

Offset	Unit	Description
0x1E000	EPOCHCTRL0	Epoch controller 0
0x1F000	DEBUG0	Debug and Trace unit 0

1. The base address is the offset relative to the first address location of the Neural-ART core, defined at integration level. Refer to the memory map.

20.6 System integration

20.6.1 System considerations

The amount of memory required by the NPU depends on the complexity (number of layers to execute, size of the input data) and target performance of the selected model of neural network. This amount can vary by 1.5 orders of magnitude.

To reach peak performance, the system memory must be configured with enough independent memory banks. These must have a separate AXI slave port on the system interconnect to guarantee the maximum internal memory bandwidth and to benefit of the AXI bus parallelism.

In addition to the on-chip memory, dedicated external memory interfaces are needed to access a nonvolatile memory, where the model parameters (weights and biases) are hosted. Additional RAMs can be used to address large(r) neural networks that cannot fit entirely in the embedded memory. The response time (latency) and the available bandwidth of these interfaces is a limiting factor in the achievable frame rate and efficiency of the NPU processing units (such as MACs) utilization.

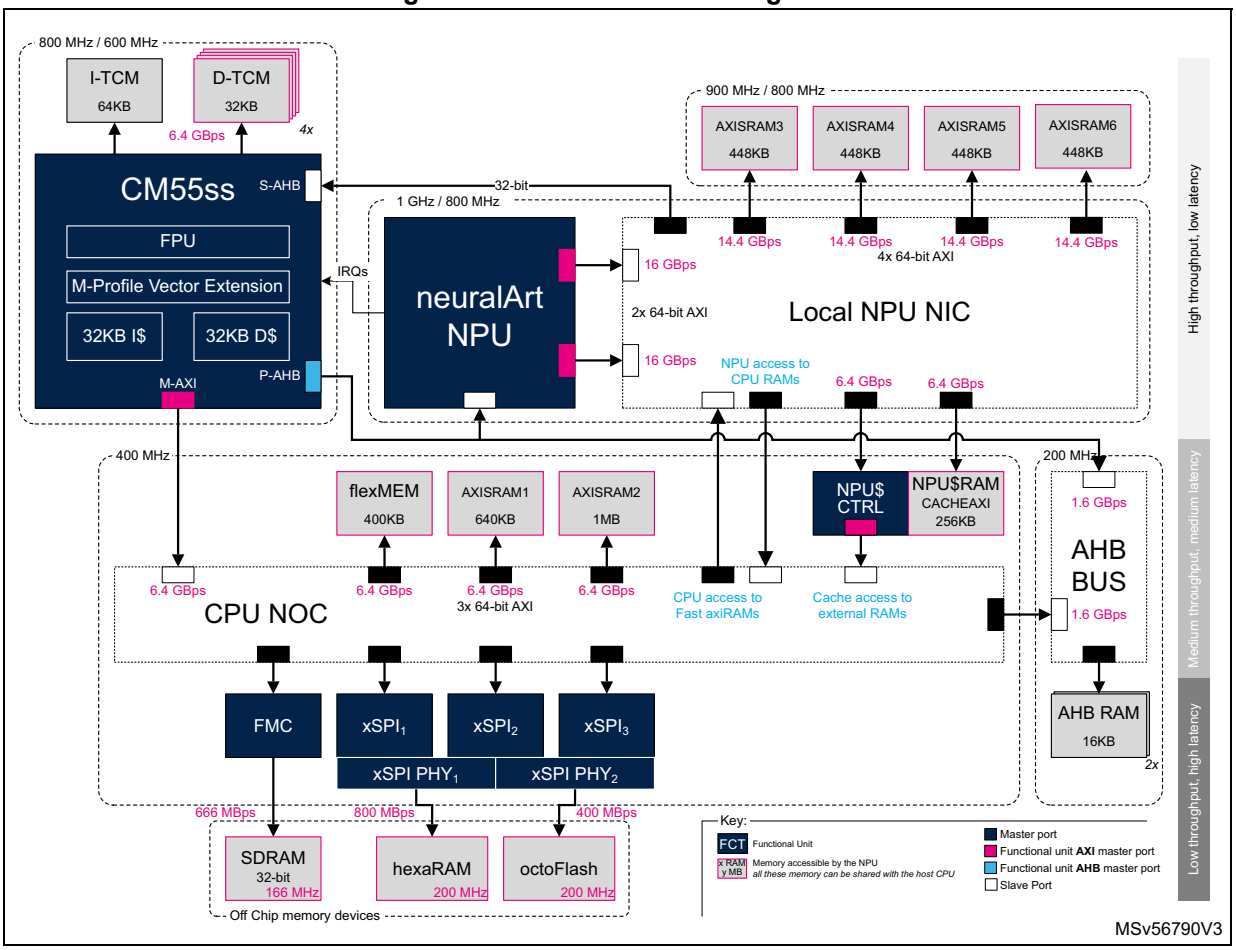
20.6.2 Architecture intent

The system architecture is designed to provide 4.2 Mbytes of contiguous system memory that can be seamlessly shared between the host and the NPU subsystems. At the same time, it offers privileged access to a meaningful high(er)-speed portion of these 4.2 Mbytes to the NPU.

A dedicated NPU cache (CACHEAXI) allows the mitigation of the bandwidth limitation when accessing external memories. When not used for caching, it can be configured as an additional RAM.

The Neural-ART 14 NPU simply connects to the system by two master AXI 64-bit ports that provide the high throughput needed for efficient acceleration of neural network models. Synchronization with the Cortex-M host CPU is through a slave AHB 32-bit port for NPU configuration, and interrupts connected to the NVIC.

Figure 131. Neural-ART 14 integration



Security

The Neural-ART 14 is not TrustZone aware. Therefore, multicontext and multitenancy are supported through the implementation of isolation compartments separated by CIDs. The AHB control interface is protected by a RISUP placed upstream. A RIMU is placed in the downstream of the AXI master interfaces, to assign CID values, so that NPU could only access the sole memories protected with that CID. Refer to the security chapter for further information.

Control interface

The Cortex-M55 host CPU configures the Neural-ART 14 NPU subsystem through the AHB slave port, which is connected to one of the ports of the AHB system bus matrix.

Data interface

The two 64-bit wide master AXI-4 ports of the NPU connect to a high-speed local interconnect that provides four 448-Kbyte banks of fast memory with a privileged, high-throughput / low-latency, asynchronous access. Each bank is implemented as four 112-Kbyte physical memory cuts, served by a full-duplex memory controller. This offers

14.4 Gbyte/s, provided reads/writes are done to different memory cuts, distant by at least 112 Kbytes.

The high-speed local NIC also connects to the main NOC through an asynchronous bridge, which gives access to medium throughput / medium latency on-chip memories:

- The system memory, organized in three memory banks.
  - A 400-Kbyte- bank, known as flexMem, which can, at setup, be allocated to either extend ITCM or DTCM, or left to the system.
  - A 624-Kbyte bank, subdivided into eight 256-Kbyte memory cuts. A bandwidth of 6.4 Gbyte/s is achievable provided reads/writes are distant by at least 256 Kbytes.
  - A 1-Mbyte bank, subdivided into four 256-Kbyte memory cuts. A bandwidth of 6.4 Gbyte/s is achievable provided reads/writes are distant by at least 256 Kbytes.
- The 256 Kbytes of the NPU cache memory (when cache is not enabled),

Finally, the NOC provides additional lower bandwidth / higher latency access to external memories connected to the FMC or xSPI controllers, with bandwidths of respectively 664 and 800 Mbyte/s.

The NPU does not have access to VENC or AHB memories.

Hardware triggers

The NPU provides four external asynchronous hardware trigger inputs. One of these is connected to the DCMIPP. The control SW can enable it to implement a CPU-less synchronization scheme between the camera pipeline and the NPU. This ancillary benefit of this pure hardware synchronization scheme is to reduce the size of the memory buffers used, as it is no longer needed to store a complete frame buffer.

Note: This synchronization scheme cannot be used if the model of CNN to execute requires preserving the input layer for the whole duration of the inference.

Virtual memPool

The actual amount of internal memory allocated to the NPU operation and the organization thereof is configurable by software.

Depending on the model of neural network to be executed, and of the size of the memory buffers it requires, the NPU compiler can dynamically decide to, either:

- access the on-chip memories in parallel, thus improving the bandwidth of the data transfers,
- consider them as fewer, virtual larger banks[(a)], to keep the activation data in internal memories for the sake of the performance.

The compiler can create one or more virtual memory pools, which can group two or more of the contiguous memories accessible through the bus. This decision is frequency a/o protocol agnostic and only cares about address ranges. A virtual memory pool can thus group as many memory banks as needed and can extend from axiRAM1 (main NOC, clocked at 400 MHz) to axiRAM6 (NPU NIC, clocked at 1 GHz).

External memories accessible through the FMC or xSPI interfaces are not contiguous with any other on-chip memory and cannot be part of a virtual memory pool.

a. This grouping of two or more contiguous memory banks is known as a “virtual memory pool”.

21 Chrom-ART Accelerator controller (DMA2D)

21.1 DMA2D introduction

The Chrom-ART Accelerator (DMA2D) is a specialized DMA dedicated to image manipulation. It can perform the following operations:

- Filling a part or the whole of a destination image with a specific color
- Copying a part or the whole of a source image into a part or the whole of a destination image
- Copying a part or the whole of a source image into a part or the whole of a destination image with a pixel format conversion
- Blending a part and/or two complete source images with different pixel format and copy the result into a part or the whole of a destination image with a different color format

All the classical color coding schemes are supported from 4-bit up to 32-bit per pixel with indexed or direct color mode, including block based YCbCr to handle JPEG decoder output. The DMA2D has its own dedicated memories for CLUTs (color look-up tables).

21.2 DMA2D main features

The main DMA2D features are:

- Single AXI master bus architecture
- AHB slave programming interface supporting 8-,16-,32-bit accesses (except for CLUT accesses which are 32-bit)
- User-programmable working area size
- User-programmable offset for sources and destination areas expressed in pixels or bytes
- User-programmable sources and destination addresses on the whole memory space
- Up to two sources with blending operation
- Alpha value that can be modified (source value, fixed value, or modulated value)
- User programmable source and destination color format
- Up to 12 color formats supported from 4-bit up to 32-bit per pixel with indirect or direct color coding
- Block based (8x8) YCbCr support with 4:4:4, 4:2:2 and 4:2:0 chroma sub-sampling factors
- 2 internal memories for CLUT storage in indirect color mode
- Automatic CLUT loading or CLUT programming via the CPU
- User programmable CLUT size
- Internal timer to control AXI bandwidth
- 6 operating modes: register-to-memory, memory-to-memory, memory-to-memory with pixel format conversion, memory-to-memory with pixel format conversion and blending, memory-to memory with pixel format conversion, blending and fixed color foreground, and memory-to memory with pixel format conversion, blending and fixed color background.
- Area filling with a fixed color

- Copy from an area to another
- Copy with pixel format conversion between source and destination images
- Copy from two sources with independent color format and blending
- Output buffer byte swapping to support refresh of displays through parallel interface
- Abort and suspend of DMA2D operations
- Watermark interrupt on a user programmable destination line
- Interrupt generation on bus error or access conflict
- Interrupt generation on process completion

21.3 DMA2D functional description

21.3.1 General description

The DMA2D controller performs direct memory transfer. As an AXI master, it can take the control of the AXI bus matrix to initiate AXI transactions.

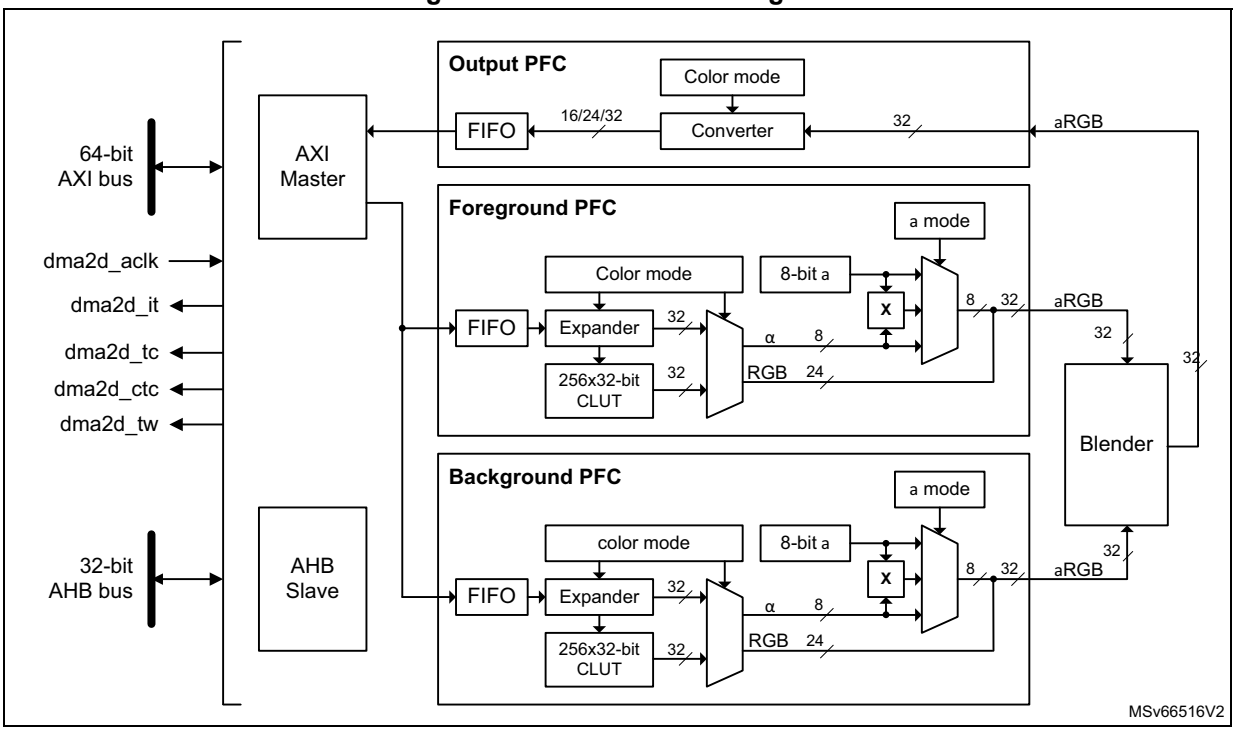
The DMA2D can operate in the following modes:

- Register-to-memory
- Memory-to-memory
- Memory-to-memory with pixel format conversion
- Memory-to-memory with pixel format conversion and blending
- Memory-to-memory with pixel format conversion, blending and fixed color foreground
- Memory-to-memory with pixel format conversion, blending and fixed color background

The AHB slave port is used to program the DMA2D controller.

The block diagram of the DMA2D is shown in *Figure 132*.

Figure 132. DMA2D block diagram



21.3.2 DMA2D internal signals

The internal signals of the DMA2D are given in the table below:

Table 112. DMA2D internal signals

Names	Signal type	Description
dma2d_aclk	Input	DMA2D AXI clock
dma2d_it	Output	DMA2D global interrupt request
dma2d_ctc	Output	DMA2D CLUT transfer complete trigger
dma2d_tc	Output	DMA2D transfer complete trigger
dma2d_tw	Output	DMA2D transfer watermark trigger

Table 113. DMA2D trigger interconnections

Trigger name	Direction	Trigger source/destination
dma2d_ctc	Output	hpdma_trigsel[12]
		gpdma_trigsel[12]
dma2d_tc	Output	hpdma_trigsel[13]
		gpdma_trigsel[13]
dma2d_tw	Output	hpdma_trigsel[14]
		gpdma_trigsel[14]

21.3.3 DMA2D control

The DMA2D controller is configured through DMA2D_CR. The user application can perform the following operations:

- Select the operating mode.
- Enable/disable the DMA2D interrupt.
- Start/suspend/abort ongoing data transfers.

21.3.4 DMA2D foreground and background FIFOs

The DMA2D foreground (FG) FG FIFO and background (BG) FIFO fetch the input data to be copied and/or processed.

The FIFOs fetch the pixels according to the color format defined in their respective pixel format converter (PFC).

They are programmed through the following control registers:

- DMA2D foreground memory address register (DMA2D_FGMAR)
- DMA2D foreground offset register (DMA2D_FGOR)
- DMA2D background memory address register (DMA2D_BGMAR)
- DMA2D background offset register (DMA2D_BGBOR)
- DMA2D number of lines register (number of lines and pixel per lines) (DMA2D_NLR)

When the DMA2D operates in register-to-memory mode, none of the FIFOs is activated.

When the DMA2D operates in memory-to-memory mode (no pixel format conversion nor blending operation), only the FG FIFO is activated, and acts as a buffer.

When the DMA2D operates in memory-to-memory operation with pixel format conversion (no blending operation), the BG FIFO is not activated.

21.3.5 DMA2D foreground and background PFC

DMA2D foreground pixel format converter (PFC) and background pixel format converter perform the pixel format conversion to generate a 32-bit per pixel value. The PFC can also modify the alpha channel.

The first PFC stage converts the color format. The original color format of the foreground and background pixels are configured through CM[3:0] in DMA2D_FGPFCCR and DMA2D_BGPFCCR, respectively.

The supported input formats are given in the table below.

Table 114. Supported color mode in input

CM[3:0]	Color mode
0000	ARGB8888
0001	RGB888
0010	RGB565
0011	ARGB1555
0100	ARGB4444
0101	L8

Table 114. Supported color mode in input (continued)

CM[3:0]	Color mode
0110	AL44
0111	AL88
1000	L4
1001	A8
1010	A4
1011	YCbCr (only for foreground)

The color format are coded as follows:

- Alpha value field: transparency
  0xFF value corresponds to an opaque pixel and 0x00 to a transparent one.
- R field for Red
- G field for Green
- B field for Blue
- L field: luminance
  This field is the index to a CLUT to retrieve the three/four RGB/ARGB components.

If the original format was direct color mode (ARGB/RGB), the extension to 8-bit per channel is performed by copying the MSBs into the LSBs. This ensures a perfect linearity of the conversion.

If the original format does not include an alpha channel, the alpha value is automatically set to 0xFF (opaque).

If the original format is indirect color mode (L/AL), a CLUT is required, and each pixel format converter is associated with a 256 entry 32-bit CLUT.

For the specific alpha mode A4 and A8, no color information is stored nor indexed. The color to be used for the image generation is fixed and defined in DMA2D_FGCOLR for foreground pixels, and in DMA2D_BGCOLR for background pixels.

The order of the fields in the system memory is defined in the table below.

Table 115. Data order in memory

Color mode	@ + 3	@ + 2	@ + 1	@ + 0
ARGB8888	A_0[7:0]	R_0[7:0]	G_0[7:0]	B_0[7:0]
RGB888	B_1[7:0]	R_0[7:0]	G_0[7:0]	B_0[7:0]
	G_2[7:0]	B_2[7:0]	R_1[7:0]	G_1[7:0]
	R_3[7:0]	G_3[7:0]	B_3[7:0]	R_2[7:0]
RGB565	R_1[4:0]G_1[5:3]	G_1[2:0]B_1[4:0]	R_0[4:0]G_0[5:3]	G_0[2:0]B_0[4:0]
ARGB1555	A_1[0]R_1[4:0]G_1[4:3]	G_1[2:0]B_1[4:0]	A_0[0]R_0[4:0]G_0[4:3]	G_0[2:0]B_0[4:0]
ARGB4444	A_1[3:0]R_1[3:0]	G_1[3:0]B_1[3:0]	A_0[3:0]R_0[3:0]	G_0[3:0]B_0[3:0]
L8	L_3[7:0]	L_2[7:0]	L_1[7:0]	L_0[7:0]
AL44	A_3[3:0]L_3[3:0]	A_2[3:0]L_2[3:0]	A_1[3:0]L_1[3:0]	A_0[3:0]L_0[3:0]

Table 115. Data order in memory (continued)

AL88	A_1[7:0]	L_1[7:0]	A_0[7:0]	L_0[7:0]
L4	L_7[3:0]L_6[3:0]	L_5[3:0]L_4[3:0]	L_3[3:0]L_2[3:0]	L_1[3:0]L_0[3:0]
A8	A_3[7:0]	A_2[7:0]	A_1[7:0]	A_0[7:0]
A4	A_7[3:0]A_6[3:0]	A_5[3:0]A_4[3:0]	A_3[3:0]A_2[3:0]	A_1[3:0]A_0[3:0]

The 24-bit RGB888 aligned on 32-bit is supported through the ARGB8888 mode.

Once the 32-bit value is generated, the alpha channel can be modified according to AM[1:0] in DMA2D_FGPFCCR or DMA2D_BGPFCCR, as shown in *Table 116*.

One of the following happens for the alpha channel:

- It is kept as it is (no modification).
- It is replaced by ALPHA[7:0] value i nDMA2D_FGPFCCR/DMA2D_BGPFCCR.
- It is replaced by the original alpha value multiplied by ALPHA[7:0] in DMA2D_FGPFCCR/DMA2D_BGPFCCR divided by 255.

Table 116. Alpha mode configuration

AM[1:0]	Alpha mode
00	No modification
01	Replaced by value in DMA2D_xxPFCCR
10	Replaced by original value multiplied by the value in DMA2D_xxPFCCR / 255
11	Reserved

Note: To support the alternate format, the incoming alpha value can be inverted setting AI in DMA2D_FGPFCCR or DMA2D_BGPFCCR. This applies also to the Alpha value stored in the DMA2D_FGPFCCR or DMA2D_BGPFCCR, and in the CLUT.

The R and B fields can also be swapped setting RBS in DMA2D_FGPFCCR or DMA2D_BGPFCCR. This applies also to the RGB order used in the CLUT, and in DMA2D_FGCOLR or DMA2D_BGCOLR.

21.3.6 DMA2D foreground and background CLUT interface

The CLUT interface manages the CLUT memory access and the automatic loading of the CLUT.

Three kinds of accesses are possible:

- CLUT read by the PFC during pixel format conversion operation
- CLUT accessed through the AHB slave port when the CPU is reading or writing data into the CLUT
- CLUT written through the AXI master port when an automatic loading of the CLUT is performed

The CLUT memory loading can be done in two different ways:

- Automatic loading

  The following sequence must be followed to load the CLUT:

a) Program the CLUT address into DMA2D_FGCMAR (foreground CLUT) or DMA2D_BGCMAR (background CLUT).

b) Program the CLUT size with CS[7:0] field in DMA2D_FGPFCCR (foreground CLUT), or DMA2D_BGPFCCR (background CLUT).

c) Set START in DMA2D_FGPFCCR (foreground CLUT) or DMA2D_BGPFCCR (background CLUT) to start the transfer. During this automatic loading process, the CLUT is not accessible by the CPU. If a conflict occurs, a CLUT access error interrupt is raised assuming CAEIE = 1 in DMA2D_CR.

- Manual loading

  The application has to program the CLUT manually through the DMA2D AHB slave port to which the local CLUT memory is mapped.The foreground CLUT (FGCLUT) is located at address offset 0x0400 and the background CLUT (BGCLUT) at address offset 0x0800.

The CLUT format can be 24 or 32 bits. It is configured through CCM in DMA2D_FGPFCCR (foreground CLUT) or DMA2D_BGPFCCR (background CLUT) as shown in table below.

Table 117. Supported CLUT color mode

CCM	CLUT color mode
0	32-bit ARGB8888
1	24-bit RGB888

The way the CLUT data are organized in the system memory is specified in the table below.

Table 118. CLUT data order in memory

CLUT Color mode	@ + 3	@ + 2	@ + 1	@ + 0
ARGB8888	A_0[7:0]	R_0[7:0]	G_0[7:0]	B_0[7:0]
RGB888	B_1[7:0]	R_0[7:0]	G_0[7:0]	B_0[7:0]
	G_2[7:0]	B_2[7:0]	R_1[7:0]	G_1[7:0]
	R_3[7:0]	G_3[7:0]	B_3[7:0]	R_2[7:0]

21.3.7 DMA2D blender

The DMA2D blender blends the source pixels by pair to compute the resulting pixel.

The blending is performed according to the following equation:

$$\text{with } \alpha_{Mult} = \frac{\alpha_{FG} \cdot \alpha_{BG}}{255}$$

$$\alpha_{OUT} = \alpha_{FG} + \alpha_{BG} - \alpha_{Mult}$$

$$C_{OUT} = \frac{C_{FG}.\alpha_{FG} + C_{BG}.\alpha_{BG} - C_{BG}.\alpha_{Mult}}{\alpha_{OUT}} \quad \text{with C = R or G or B}$$

Division is rounded to the nearest lower integer

No configuration register is required by the blender. The blender use depends on the DMA2D operating mode defined by MODE[2:0] in DMA2D_CR.

21.3.8 DMA2D output PFC

The output PFC performs the pixel format conversion from 32 bits to the output format defined by CM[2:0] in DMA2D_OPFCCR.

Supported output formats are given in the table below

Table 119. Supported color mode in output

CM[2:0]	Color mode
000	ARGB8888
001	RGB888
010	RGB565
011	ARGB1555
100	ARGB4444

Note: To support the alternate format, the calculated alpha value can be inverted setting AI bit in DMA2D_OPFCCR. This applies also to the Alpha value used in DMA2D_OCOLR.

The R and B fields can also be swapped setting RBS in DMA2D_OPFCCR. This applies also to the RGB order used in DMA2D_OCOLR.

21.3.9 DMA2D output FIFO

The output FIFO programs the pixels according to the color format defined in the output PFC.

The destination area is defined through the following registers:

- DMA2D output memory address register (DMA2D_OMAR)
- DMA2D output offset register (DMA2D_OOR**)**
- DMA2D number of lines register (number of lines and pixel per lines) (DMA2D_NLR**)**

If the DMA2D operates in register-to-memory mode, the configured output rectangle is filled by the color specified in DMA2D_OCOLR which contains a fixed 32-, 24-, or 16-bit value. The format is selected by CM[2:0] in DMA2D_OPFCCR.

The data are stored into the memory in the order defined in the table below.

Table 120. Data order in memory

Color mode	@ + 3	@ + 2	@ + 1	@ + 0
ARGB8888	A_0[7:0]	R_0[7:0]	G_0[7:0]	B_0[7:0]
RGB888	B_1[7:0]	R_0[7:0]	G_0[7:0]	B_0[7:0]
	G_2[7:0]	B_2[7:0]	R_1[7:0]	G_1[7:0]
	R_3[7:0]	G_3[7:0]	B_3[7:0]	R_2[7:0]
RGB565	R_1[4:0]G_1[5:3]	G_1[2:0]B_1[4:0]	R_0[4:0]G_0[5:3]	G_0[2:0]B_0[4:0]
ARGB1555	A_1[0]R_1[4:0]G_1[4:3]	G_1[2:0]B_1[4:0]	A_0[0]R_0[4:0]G_0[4:3]	G_0[2:0]B_0[4:0]
ARGB4444	A_1[3:0]R_1[3:0]	G_1[3:0]B_1[3:0]	A_0[3:0]R_0[3:0]	G_0[3:0]B_0[3:0]

The RGB888 aligned on 32 bits is supported through the ARGB8888 mode.

21.3.10 DMA2D output FIFO byte reordering

The output FIFO bytes can be reordered to support display frame buffer update through a parallel interface (F(S)MC) directly from the DMA2D.

The reordering of bytes can be done using:

- RBS bit to swap red and blue component
- SB bit to swap byte two by two in the output FIFO

When the byte swapping is activated (SB = 1 in DMA2D_OPFCCR), the number of pixel per line (PL field in DMA2D_NLR) must be even, and the output memory address (MA field in DMA2D_OMAR) must be even. The output line offset computed in bytes (resulting from LOM field in DMA2D_CR and LO field in DMA2D_OOR values) must also be even. If not a configuration error is detected.

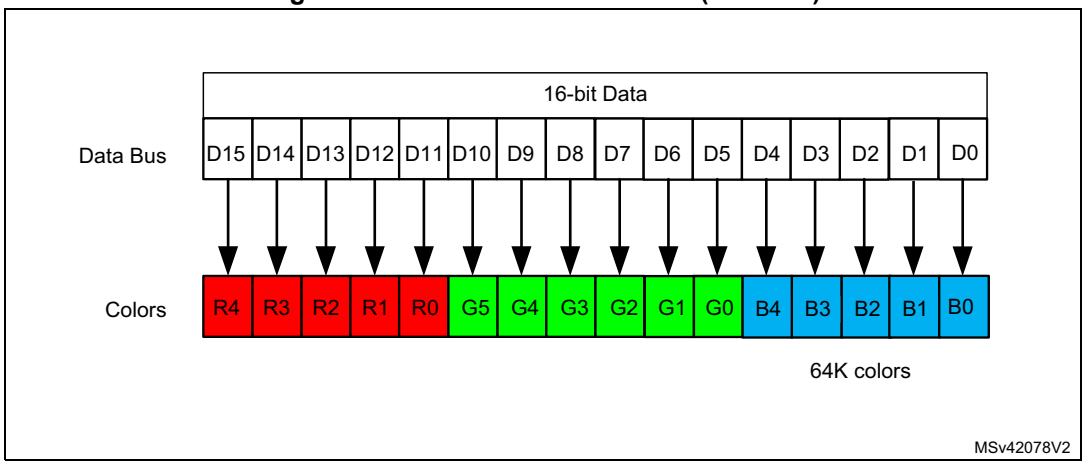
Table 121. Standard data order in memory

Color Mode	@ + 3	@ + 2	@ + 1	@ + 0
RGB888	B_1[7:0]	R_0[7:0]	G_0[7:0]	B_0[7:0]
	G_2[7:0]	B_2[7:0]	R_1[7:0]	G_1[7:0]
	R_3[7:0]	G_3[7:0]	B_3[7:0]	R_2[7:0]
RGB565	R_1[4:0]G_1[5:3]	G_1[2:0]B_1[4:0]	R_0[4:0]G_0[5:3]	G_0[2:0]B_0[4:0]

16-bit mode (RGB565)

This mode is supported without byte reordering by the DMA2D.

Figure 133. Intel 8080 16-bit mode (RGB565)



18/24-bit mode (RGB888)

This mode needs data reordering:

1. The red and the blue have to be swapped (setting the RBS bit).
2. MSB and the LSB bytes of an half-word must be swapped (setting SB bit).

Figure 134. Intel 8080 18/24-bit mode (RGB888)

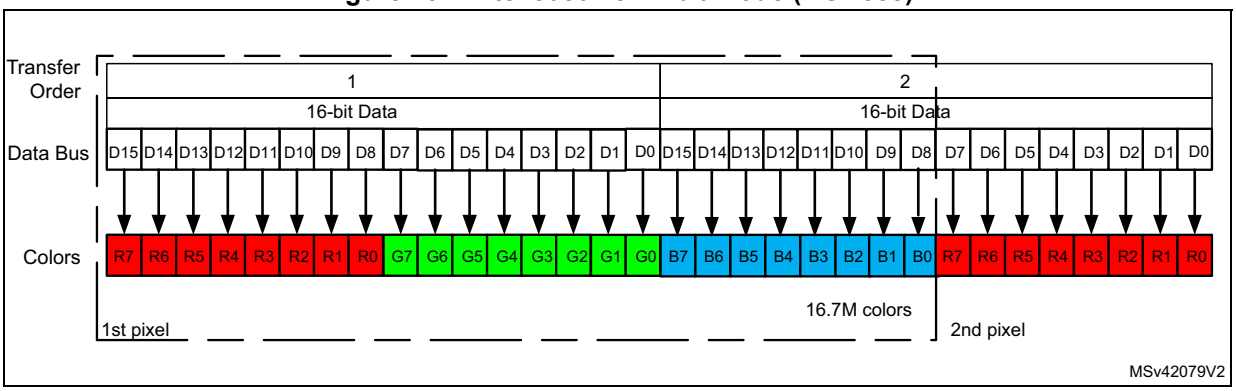


Table 122. Output FIFO byte reordering steps

Steps	@ + 3	@ + 2	@ + 1	@ + 0
Original data ordering	B_1[7:0]	R_0[7:0]	G_0[7:0]	B_0[7:0]
	G_2[7:0]	B_2[7:0]	R_1[7:0]	G_1[7:0]
	R_3[7:0]	G_3[7:0]	B_3[7:0]	R_2[7:0]
Setting the RBS bit				
Data ordering after Red and Blue swap (RBS set)	R_1[7:0]	B_0[7:0]	G_0[7:0]	R_0[7:0]
	G_2[7:0]	R_2[7:0]	B_1[7:0]	G_1[7:0]
	B_3[7:0]	G_3[7:0]	R_3[7:0]	B_2[7:0]
Setting the SB bit				

Table 122. Output FIFO byte reordering steps (continued)

Steps	@ + 3	@ + 2	@ + 1	@ + 0
Data ordering after byte swapping (SB set)	B_0[7:0]	R_1[7:0]	R_0[7:0]	G_0[7:0]
	R_2[7:0]	G_2[7:0]	G_1[7:0]	B_1[7:0]
	G_3[7:0]	B_3[7:0]	B_2[7:0]	R_3[7:0]

21.3.11 DMA2D AXI master port timer

An 8-bit timer is embedded into the AXI master port to provide an optional limitation of the bandwidth on the crossbar. This timer is clocked by the AXI clock and counts a dead time between two consecutive accesses. This limits the bandwidth usage.

The timer enabling and dead time value are configured through DMA2D_AMPTCR.

21.3.12 DMA2D transactions

DMA2D transactions consist of a sequence of a given number of data transfers. Number of data and width can be programmed by software.

Each DMA2D data transfer is composed of up to four steps:

1. Data loading from the memory location pointed by DMA2D_FGMAR and pixel format conversion as defined in DMA2D_FGCR
2. Data loading from a memory location pointed by DMA2D_BGMAR and pixel format conversion as defined in DMA2D_BGCR
3. Blending of all retrieved pixels according to the alpha channels resulting of PFC operation on alpha values
4. Pixel format conversion of resulting pixels according to DMA2D_OPFCCR and programming of data to the memory location addressed through DMA2D_OMAR

21.3.13 DMA2D configuration

Both source and destination data transfers can target peripherals and memories in the whole 4-Gbyte memory area, at addresses ranging between 0x0000 0000 and 0xFFFF FFFF.

The DMA2D can operate in any of the following modes selected through MODE[2:0] in DMA2D_CR:

- Register-to-memory
- Memory-to-memory
- Memory-to-memory with PFC
- Memory-to-memory with PFC and blending
- Memory-to-memory with PFC, blending and fixed FG color
- Memory-to-memory with PFC, blending and fixed BG color

Register-to-memory

The register-to-memory mode is used to fill a user defined area with a predefined color. The color format is set in DMA2D_OPFCCR.

The DMA2D does not perform any data fetching from any source. It just writes the color defined in DMA2D_OCOLR to the area located at the address pointed by DMA2D_OMAR and defined in DMA2D_NLR and DMA2D_OOR.

Memory-to-memory

In memory-to-memory mode, the DMA2D does not perform any graphical data transformation. The foreground input FIFO acts as a buffer, and data are transferred from the source memory location defined in DMA2D_FGMAR, to the destination memory location pointed by DMA2D_OMAR.

The color mode programmed by CM[3:0] in DMA2D_FGPFCCR defines the number of bits per pixel for both input and output.

The size of the area to be transferred is defined by DMA2D_NLR and DMA2D_FGOR for the source, and by DMA2D_NLR and DMA2D_OOR for the destination.

Memory-to-memory with PFC

In this mode, the DMA2D performs a pixel format conversion of the source data, and stores them in the destination memory location.

The size of the areas to be transferred are defined by DMA2D_NLR and DMA2D_FGOR for the source, and by DMA2D_NLR and DMA2D_OOR for the destination.

Data are fetched from the location defined in DMA2D_FGMAR, and processed by the foreground PFC. The original pixel format is configured through DMA2D_FGPFCCR.

If the original pixel format is direct-color mode, the color channels are all expanded to 8 bits.

If the pixel format is indirect color mode, the associated CLUT has to be loaded into the CLUT memory.

The CLUT loading can be done automatically by following the sequence below:

1. Set the CLUT address into DMA2D_FGCMAR.
2. Set the CLUT size with CS[7:0] in DMA2D_FGPFCCR.
3. Set the CLUT format (24 or 32 bits) with CCM in DMA2D_FGPFCCR.
4. Start the CLUT loading by setting START in DMA2D_FGPFCCR.

Once the CLUT loading is complete, CTCIF flag in DMA2D_IFR is raised, and an interrupt is generated if CTCIE is set in DMA2D_CR. The automatic CLUT loading process cannot work in parallel with classical DMA2D transfers.

The CLUT can also be filled by the CPU or by any other master through the AHB port. The access to the CLUT is not possible when a DMA2D transfer is ongoing, and uses the CLUT (indirect color format).

In parallel to the color conversion process, the alpha value can be added or changed depending on the value programmed in DMA2D_FGPFCCR. If the original image does not have an alpha channel, a default alpha value of 0xFF is automatically added to obtain a fully opaque pixel.

The alpha value is modified as follows according to AM[1:0] in DMA2D_FGPFCCR:

- It is unchanged.
- It is replaced by ALPHA[7:0] value in DMA2D_FGPFCCR.
- It is replaced by the original value multiplied by ALPHA[7:0] value in DMA2D_FGPFCCR divided by 255.

The resulting 32-bit data are encoded by the output PFC into the format specified by CM[2:0] in DMA2D_OPFCCR. The output pixel format cannot be the indirect mode since no CLUT generation process is supported.

Processed data are written into destination memory location pointed by DMA2D_OMAR.

Memory-to-memory with PFC and blending

In this mode, two sources are fetched in the foreground and background FIFOs from the memory locations defined by DMA2D_FGMAR and DMA2D_BGMAR respectively.

The two pixel format converters have to be configured as described in memory-to-memory mode. Their configurations can be different as each pixel format converter is independent and has its own CLUT memory.

Once each pixel has been converted into 32 bits by its respective PFC, all pixels are blended according to the equation below:

$$\text{with } \alpha_{Mult} = \frac{\alpha_{FG} \cdot \alpha_{BG}}{255}$$

$$\alpha_{OUT} = \alpha_{FG} + \alpha_{BG} - \alpha_{Mult}$$

$$C_{OUT} = \frac{C_{FG}.\alpha_{FG} + C_{BG}.\alpha_{BG} - C_{BG}.\alpha_{Mult}}{\alpha_{OUT}} \quad \text{with C = R or G or B}$$

Division are rounded to the nearest lower integer

The resulting 32-bit pixel value is encoded by the output PFC according to the specified output format, and data are written into the destination memory location pointed by DMA2D_OMAR.

Memory-to-memory with PFC, blending and fixed color FG

In this mode, only one source is fetched in the background FIFO from the memory location defined by DMA2D_BGMAR.

The value of the foreground color is given by DMA2D_FGCOLR, and the alpha value is set to 0xFF (opaque).

The alpha value can be replaced or modified according to AM[1:0] and ALPHA[7:0] in DMA2D_FGPFCCR.

The two pixel format converters have to be configured as described in memory-to-memory mode. Their configurations can be different as each pixel format converter is independent and has its own CLUT memory

Once each pixel has been converted into 32 bits by its respective PFC, all pixels are blended together. The resulting 32-bit pixel value is encoded by the output PFC according to the specified output format. Data are written into the destination memory location pointed by DMA2D_OMAR.

Memory-to-memory with PFC, blending and fixed color BG

In this mode, only open source is fetched in the foreground FIFO from the memory location defined by DMA2D_FGMAR.

The value of the background color is given by DMA2D_BGCOLR, and the alpha value is set to 0xFF (opaque).

The alpha value can be replaced or modified according to AM[1:0] and ALPHA[7:0] in DMA2D_BGPFCCR.

The two pixel format converters have to be configured as described in memory-to-memory mode. Their configurations can be different as each pixel format converter is independent and has its own CLUT memory

Once each pixel has been converted into 32 bits by their respective PFCs, all pixels are blended together. The resulting 32-bit pixel value is encoded by the output PFC according to the specified output format. Data are written into the destination memory location pointed by DMA2D_OMAR.

Configuration error detection

The DMA2D checks that the configuration is correct before any transfer. The configuration error interrupt flag is set by hardware when a wrong configuration is detected when a new transfer/automatic loading starts. An interrupt is then generated if CEIE = 1 in DMA2D_CR.

The wrong configurations that can be detected are listed below:

- Foreground CLUT automatic loading: MA bits in DMA2D_FGCMAR are not aligned with CCM in DMA2D_FGPFCCR.
- Background CLUT automatic loading: MA bits in DMA2D_BGCMAR are not aligned with CCM in DMA2D_BGPFCCR.
- Memory transfer (except in register-to-memory and memory-to-memory with blending and fixed color FG): MA bits in DMA2D_FGMAR are not aligned with CM in DMA2D_FGPFCCR.
- Memory transfer (except in register-to-memory and memory-to-memory with blending and fixed color FG): CM bits in DMA2D_FGPFCCR are invalid.
- Memory transfer (except in register-to-memory and memory-to-memory with blending and fixed color FG): PL bits in DMA2D_NLR are odd while CM is DMA2D_FGPFCCR is A4 or L4.
- Memory transfer (except in register-to-memory and memory-to-memory with blending and fixed color FG): LO bits in DMA2D_FGOR are odd while CM in DMA2D_FGPFCCR is A4 or L4, and LOM in DMA2D_CR is pixel mode.
- Memory transfer (only in blending mode, except in memory-to-memory with blending and fixed color FG): MA bits in DMA2D_BGMAR are not aligned with the CM in DMA2D_BGPFCCR
- Memory transfer (only in blending and blending with fixed color FG): CM bits in DMA2D_BGPFCCR are invalid.
- Memory transfer (only in blending and blending with fixed color FG mode): PL bits in DMA2D_NLR odd while CM inDMA2D_BGPFCCR is A4 or L4.
- Memory transfer (only in blending and blending with fixed color FG): LO bits in DMA2D_BGOR are odd while CM inDMA2D_BGPFCCR is A4 or L4, and LOM in DMA2D_CR is pixel mode.
- Memory transfer (except in memory-to-memory): MA bits in DMA2D_OMAR are not aligned with CM in DMA2D_OPFCCR.
- Memory transfer (except in memory-to-memory): CM bits in DMA2D_OPFCCR are invalid.

- Memory transfer with byte swapping: PL bits in DMA2D_NLR are odd, or MA bits in DMA2D_OMAR are odd, or LO in bytes (resulting from LOM in DMA2D_CR and LO in DMA2D_OOR) are odd while SB = 1 in DMA2D_OPFCCR.
- Memory transfer: NL bits in DMA2D_NLR are all zeros.
- Memory transfer: PL bits in DMA2D_NLR are all zeros.
- Memory transfer: MODE bits in DMA2D_CR are invalid.
- YCbCr format: when a CLUT loading starts setting START in DMA2D_FGPFCCR
- YCbCr format: when memory-to-memory mode is selected
- YCbCr format: when YCbCR4:4:4 is selected, and the sum of number of pixel (PL) and line offset LO is not a multiple of 8 pixels
- YCbCr format: when YCbCr4:2:2 or YCbCr4:2:0 is selected, and the sum of number of pixel (PL) and line offset LO is not a multiple of 16 pixels

21.3.14 YCbCr support

The DMA2D foreground plane can support 8x8 block-based YCbCr as output by the JPEG decoder with different chroma subsampling factors.

The memory organization follows the standard JFIF rules:

- Each of the three color components must be coded on 8-bit.
- Each component must be arranged by blocks of 8x8 (64 bytes) called MCU.

Depending of the chroma subsampling factor, the MCU must be arranged in the memory as described in the table below.

Table 123. MCU order in memory

Sub-sampling	@	@ + 64	@ + 128	@+192	@+256	@ + 320
4:4:4	Y_1	Cb_1	Cr_1	Y_2	Cb_2	Cr_2
4:2:2	Y_1	Y_2	Cb_{12}	Cr_{12}	Y_3	Y_4
4:2:0	Y_1	Y_2	Y_3	Y_4	Cb_{1234}	Cr_{1234}

The chroma subsampling factor is configured through CSS in DMA2D_FGPFCCR.

Once the DMA2D has started with the foreground configured in YCbCr color mode, the first two chroma MCUs are loaded in the foreground CLUT. Once the chroma MCUs are loaded, the DMA2D performs the loading of the Y MCU as for a classical color mode.

21.3.15 DMA2D transfer control (start, suspend, abort, and completion)

Once the DMA2D is configured, the transfer can be launched by setting START in DMA2D_CR. Once the transfer is completed, START is automatically reset, and TCIF flag is raised in DMA2D_ISR. An interrupt can be generated if TCIE is set in DMA2D_CR.

The user application can suspend the DMA2D at any time by setting SUSP in DMA2D_CR. The transaction can be aborted by setting ABORT i nDMA2D_CR, or can be restarted by resetting SUSP in DMA2D_CR.

The user application can abort at any time an ongoing transaction by setting ABORT in DMA2D_CR. In this case, the TCIF flag is not raised.

Automatic CLUT transfers can also be aborted or suspended by using their own START bits in DMA2D_FGPFCCR and DMA2D_BGPFCCR.

21.3.16 Watermark

A watermark can be programmed to generate an interrupt when the last pixel of a given line has been written to the destination memory area.

The line number is defined in LW[15:0] in DMA2D_LWR.

When the last pixel of this line has been transferred, TWIF flag is raised in DMA2D_ISR, and an interrupt is generated if TWIE is set in DMA2D_CR.

21.3.17 Error management

Two kind of errors can be triggered:

- AXI master port errors signaled by TEIF flag in DMA2D_ISR
- Conflicts caused by CLUT access (CPU trying to access the CLUT while a CLUT loading or a DMA2D transfer is ongoing) signaled by CAEIF flag in DMA2D_ISR

Both flags are associated to their own interrupt enable flag in DMA2D_CR to generate an interrupt if need be (TEIE and CAEIE).

21.3.18 AXI dead time

To limit the AXI bandwidth use, a dead time between two consecutive AXI accesses can be programmed. This feature can be enabled by setting EN in DMA2D_AMTCR.

The dead-time value is stored in DT[7:0] in DMA2D_AMTCR. This value represents the guaranteed minimum number of cycles between two consecutive transactions on AXI bus.

The update of the dead-time value while the DMA2D is running is taken into account for the next AXI transfer.

21.4 DMA2D interrupts

An interrupt can be generated on the following events:

- Configuration error
- CLUT transfer complete
- CLUT access error
- Transfer watermark reached
- Transfer complete
- Transfer error

Separate interrupt enable bits are available for flexibility.

Table 124. DMA2D interrupt requests

Interrupt event	Event flag	Enable control bit
Configuration error	CEIF	CEIE
CLUT transfer complete	CTCIF	CTCIE

Table 124. DMA2D interrupt requests (continued)

Interrupt event	Event flag	Enable control bit
CLUT access error	CAEIF	CAEIE
Transfer watermark	TWF	TWIE
Transfer complete	TCIF	TCIE
Transfer error	TEIF	TEIE

21.5 DMA2D registers

21.5.1 DMA2D control register (DMA2D_CR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE[2:0]		
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CEIE	CTCIE	CAEIE	TWIE	TCIE	TEIE	Res.	LOM	Res.	Res.	Res.	ABORT	SUSP	START
		rw	rw	rw	rw	rw	rw		rw				rs	rw	rs

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **MODE[2:0]**: DMA2D mode
This bit is set and cleared by software. It cannot be modified while a transfer is ongoing.
000: Memory-to-memory (FG fetch only)
001: Memory-to-memory with PFC (FG fetch only with FG PFC active)
010: Memory-to-memory with blending (FG and BG fetch with PFC and blending)
011: Register-to-memory (no FG nor BG, only output stage active)
100: Memory-to-memory with blending and fixed color FG (BG fetch only with FG and BG PFC active)
101: Memory-to-memory with blending and fixed color BG (FG fetch only with FG and BG PFC active)
Others: Reserved

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

Bit 13 **CEIE**: Configuration error (CE) interrupt enable
This bit is set and cleared by software.
0: CE interrupt disabled
1: CE interrupt enabled

Bit 12 **CTCIE**: CLUT transfer complete (CTC) interrupt enable
This bit is set and cleared by software.
0: CTC interrupt disabled
1: CTC interrupt enabled

Bit 11 **CAEIE**: CLUT access error (CAE) interrupt enable
This bit is set and cleared by software.
0: CAE interrupt disabled
1: CAE interrupt enabled

Bit 10 **TWIE**: Transfer watermark (TW) interrupt enable
This bit is set and cleared by software.
0: TW interrupt disabled
1: TW interrupt enabled

Bit 9 **TCIE**: Transfer complete (TC) interrupt enable
This bit is set and cleared by software.
0: TC interrupt disabled
1: TC interrupt enabled

Bit 8 **TEIE**: Transfer error (TE) interrupt enable
This bit is set and cleared by software.
0: TE interrupt disabled
1: TE interrupt enabled

Bit 7 Reserved, must be kept at reset value.

Bit 6 **LOM**: Line offset mode
This bit configures how the line offset is expressed (pixels or bytes) for the foreground, background and output.
This bit is set and cleared by software. It can not be modified while a transfer is ongoing.
0: Line offsets expressed in pixels
1: Line offsets expressed in bytes

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **ABORT**: Abort
This bit can be used to abort the current transfer. This bit is set by software, and is automatically reset by hardware when START = 0.
0: No transfer abort requested
1: Transfer abort requested

Bit 1 **SUSP**: Suspend
This bit can be used to suspend the current transfer. This bit is set and reset by software. It is automatically reset by hardware when START = 0.
0: Transfer not suspended
1: Transfer suspended

Bit 0 **START**: Start
This bit can be used to launch the DMA2D according to parameters loaded in the various configuration registers. This bit is automatically reset by the following events:
- at the end of the transfer
- when the data transfer is aborted by the user by setting ABORT in this register
- when a data transfer error occurs
- when the data transfer has not started due to a configuration error, or another transfer operation already ongoing (automatic CLUT loading)

21.5.2 DMA2D interrupt status register (DMA2D_ISR)

Address offset: 0x004

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CEIF	CTCIF	CAEIF	TWIF	TCIF	TEIF
										r	r	r	r	r	r

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

Bit 5 **CEIF**: Configuration error interrupt flag
This bit is set when START is set in DMA2D_CR, DMA2DFGPFCCR or DMA2D_BGPFCCR, and a wrong configuration has been programmed.

Bit 4 **CTCIF**: CLUT transfer complete interrupt flag
This bit is set when the CLUT copy from a system memory area to the internal DMA2D memory is complete.

Bit 3 **CAEIF**: CLUT access error interrupt flag
This bit is set when the CPU accesses the CLUT while the CLUT is being automatically copied from a system memory to the internal DMA2D.

Bit 2 **TWIF**: Transfer watermark interrupt flag
This bit is set when the last pixel of the watermarked line has been transferred.

Bit 1 **TCIF**: Transfer complete interrupt flag
This bit is set when a DMA2D transfer operation is complete (data transfer only).

Bit 0 **TEIF**: Transfer error interrupt flag
This bit is set when an error occurs during a DMA transfer (data transfer or automatic CLUT loading).

21.5.3 DMA2D interrupt flag clear register (DMA2D_IFCR)

Address offset: 0x008

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCEIF	CCTCIF	CAECIF	CTWIF	CTCIF	CTEIF
										rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

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

Bit 5 **CCEIF**: Clear configuration error interrupt flag
Programming this bit to 1 clears the CEIF flag in DMA2D_ISR.

Bit 4 **CCTCIF**: Clear CLUT transfer complete interrupt flag
Programming this bit to 1 clears the CTCIF flag in DMA2D_ISR.

Bit 3 **CAECIF**: Clear CLUT access error interrupt flag
Programming this bit to 1 clears the CAEIF flag in DMA2D_ISR.

Bit 2 **CTWIF**: Clear transfer watermark interrupt flag
Programming this bit to 1 clears the TWIF flag in DMA2D_ISR.

Bit 1 **CTCIF**: Clear transfer complete interrupt flag
Programming this bit to 1 clears the TCIF flag in DMA2D_ISR.

Bit 0 **CTEIF**: Clear transfer error interrupt flag
Programming this bit to 1 clears the TEIF flag in DMA2D_ISR.

21.5.4 DMA2D foreground memory address register (DMA2D_FGMAR)

Address offset: 0x00C

Reset value: 0x0000 0000

This register can only be written when data transfers are disabled. Once the data transfer started, this register is read-only.

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

Bits 31:0 **MA[31:0]**: Memory address, address of the data used for the foreground image
The address alignment must match the image format selected: a 32-bit per pixel format must be 32-bit aligned, a 16-bit per pixel format must be 16-bit aligned, and a 4-bit per pixel format must be 8-bit aligned.

21.5.5 DMA2D foreground offset register (DMA2D_FGOR)

Address offset: 0x010

Reset value: 0x0000 0000

This register can only be written when data transfers are disabled. Once the data transfer started, this register is read-only.

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **LO[15:0]**: Line offset

This field gives the line offset used for the foreground image, expressed:

– in pixels when LOM = 0 in DMA2D_CR. Only LO[13:0] bits are considered, LO[15:14] bits are ignored.

– in bytes when LOM = 1

This value is used for the address generation. It is added at the end of each line to determine the starting address of the next line.

If the image format is 4-bit per pixel, the line offset must be even.

21.5.6 DMA2D background memory address register (DMA2D_BGMAR)

Address offset: 0x014

Reset value: 0x0000 0000

This register can only be written when data transfers are disabled. Once the data transfer started, this register is read-only.

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

Bits 31:0 **MA[31:0]**: Memory address, address of the data used for the background image

The address alignment must match the image format selected: a 32-bit per pixel format must be 32-bit aligned, a 16-bit per pixel format must be 16-bit aligned and a 4-bit per pixel format must be 8-bit aligned.

21.5.7 DMA2D background offset register (DMA2D_BGOR)

Address offset: 0x018

Reset value: 0x0000 0000

This register can only be written when data transfers are disabled. Once the data transfer started, this register is read-only.

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **LO[15:0]**: Line offset

This field gives the line offset used for the background image, expressed:

– in pixels when LOM = 0 in DMA2D_CR. Only LO[13:0] bits are considered, LO[15:14] bits are ignored.

– in bytes when LOM = 1

This value is used for the address generation. It is added at the end of each line to determine the starting address of the next line.

If the image format is 4-bit per pixel, the line offset must be even.

21.5.8 DMA2D foreground PFC control register (DMA2D_FGPFCCR)

Address offset: 0x01C

Reset value: 0x0000 0000

This register can only be written when data transfers are disabled. Once the data transfer started, this register is read-only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ALPHA[7:0]								Res.	Res.	RBS	AI	CSS[1:0]		AM[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CS[7:0]								Res.	Res.	START	CCM	CM[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw			rc_w1	rw	rw	rw	rw	rw

Bits 31:24 **ALPHA[7:0]**: Alpha value

These bits define a fixed alpha channel value which can replace the original alpha value, or be multiplied by the original alpha value, according to the alpha mode selected through AM[1:0] in this register.

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **RBS**: Red/Blue swap

This bit allows to swap Red and Blue to support BGR or ABGR color formats.

0: Regular mode (RGB or ARGB)

1: Swap mode (BGR or ABGR)

Bit 20 **AI**: Alpha inverted

This bit inverts the alpha value.

0: Regular alpha

1: Inverted alpha

Bits 19:18 **CSS[1:0]**: Chroma subsampling

These bits define the chroma subsampling mode for YCbCr color mode.

00: 4:4:4 (no chroma subsampling)

01: 4:2:2

10: 4:2:0

Others: Reserved

Bits 17:16 **AM[1:0]**: Alpha mode
These bits select the alpha channel value to be used for the foreground image.
00: No modification of the foreground image alpha channel value
01: Replace original foreground image alpha channel value by ALPHA[7: 0]
10: Replace original foreground image alpha channel value by ALPHA[7:0] multiplied with original alpha channel value
Others: Reserved

Bits 15:8 **CS[7:0]**: CLUT size
These bits define the size of the CLUT used for the foreground image.
The number of CLUT entries is equal to CS[7:0] + 1.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **START**: Start
This bit can be set to start the automatic loading of the CLUT. It is automatically reset:
- at the end of the transfer
- when the transfer is aborted by the user by setting ABORT in DMA2D_CR
- when a transfer error occurs
- when the transfer has not started due to a configuration error or another transfer operation already ongoing (data transfer or automatic background CLUT transfer)

Bit 4 **CCM**: CLUT color mode
This bit defines the color format of the CLUT.
0: ARGB8888
1: RGB888

Bits 3:0 **CM[3:0]**: Color mode
These bits defines the color format of the foreground image.
0000: ARGB8888
0001: RGB888
0010: RGB565
0011: ARGB1555
0100: ARGB4444
0101: L8
0110: AL44
0111: AL88
1000: L4
1001: A8
1010: A4
1011: YCbCr
Others: Reserved

21.5.9 DMA2D foreground color register (DMA2D_FGCOLR)

Address offset: 0x020

Reset value: 0x0000 0000

This register can only be written when data transfers are disabled. Once the data transfer started, this register is read-only.

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

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **RED[7:0]**: Red value for the A4 or A8 mode of the foreground image
Used also for fixed color FG in memory-to-memory mode with blending and fixed color FG (BG fetch only with FG and BG PFC active).

Bits 15:8 **GREEN[7:0]**: Green value for the A4 or A8 mode of the foreground image
Used also for fixed color FG in memory-to-memory mode with blending and fixed color FG (BG fetch only with FG and BG PFC active).

Bits 7:0 **BLUE[7:0]**: Blue value for the A4 or A8 mode of the foreground image
Used also for fixed color FG in memory-to-memory mode with blending and fixed color FG (BG fetch only with FG and BG PFC active).

21.5.10 DMA2D background PFC control register (DMA2D_BGPFCCR)

Address offset: 0x024

Reset value: 0x0000 0000

This register can only be written when data transfers are disabled. Once the data transfer started, this register is read-only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ALPHA[7:0]								Res.	Res.	RBS	AI	Res.	Res.	AM[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw			rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CS[7:0]								Res.	Res.	START	CCM	CM[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw			rc_w1	rw	rw	rw	rw	rw

Bits 31:24 **ALPHA[7:0]**: Alpha value
These bits define a fixed alpha channel value which can replace the original alpha value, or be multiplied with the original alpha value according to the alpha mode selected with AM[1:0].

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **RBS**: Red/Blue swap
This bit allows to swap Red and Blue to support BGR or ABGR color formats.
0: Regular mode (RGB or ARGB)
1: Swap mode (BGR or ABGR)

Bit 20 **AI**: Alpha Inverted
This bit inverts the alpha value.
0: Regular alpha
1: Inverted alpha

Bits 19:18 Reserved, must be kept at reset value.

Bits 17:16 **AM[1:0]**: Alpha mode
These bits define which alpha channel value to be used for the background image.
00: No modification of the foreground image alpha channel value
01: Replace original background image alpha channel value by ALPHA[7: 0]
10: Replace original background image alpha channel value by ALPHA[7:0] multiplied with original alpha channel value
Others: Reserved

Bits 15:8 **CS[7:0]**: CLUT size
These bits define the size of the CLUT used for the BG.
The number of CLUT entries is equal to CS[7:0] + 1.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **START**: Start
This bit is set to start the automatic loading of the CLUT. This bit is automatically reset:
- at the end of the transfer
- when the transfer is aborted by the user by setting ABORT bit in DMA2D_CR
- when a transfer error occurs
- when the transfer has not started due to a configuration error or another transfer operation already on going (data transfer or automatic background CLUT transfer)

Bit 4 **CCM**: CLUT color mode
These bits define the color format of the CLUT.
0: ARGB8888
1: RGB888

Bits 3:0 **CM[3:0]**: Color mode
These bits define the color format of the foreground image.
0000: ARGB8888
0001: RGB888
0010: RGB565
0011: ARGB1555
0100: ARGB4444
0101: L8
0110: AL44
0111: AL88
1000: L4
1001: A8
1010: A4
Others: Reserved

21.5.11 DMA2D background color register (DMA2D_BGCOLR)

Address offset: 0x028

Reset value: 0x0000 0000

This register can only be written when data transfers are disabled. Once the data transfer started, this register is read-only.

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

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **RED[7:0]**: Red value for the A4 or A8 mode of the background
Used also for fixed color BG in memory-to-memory mode with blending and fixed color BG (FG fetch only with FG and BG PFC active).

Bits 15:8 **GREEN[7:0]**: Green value for the A4 or A8 mode of the background
Used also for fixed color BG in memory-to-memory mode with blending and fixed color BG (FG fetch only with FG and BG PFC active).

Bits 7:0 **BLUE[7:0]**: Blue value for the A4 or A8 mode of the background
Used also for fixed color BG in memory-to-memory mode with blending and fixed color BG (FG fetch only with FG and BG PFC active).

21.5.12 DMA2D foreground CLUT memory address register (DMA2D_FGCMAR)

Address offset: 0x02C

Reset value: 0x0000 0000

This register can only be written when data transfers are disabled. Once the data transfer started, this register is read-only.

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

Bits 31:0 **MA[31:0]**: Memory address
Address of the data used for the CLUT address dedicated to the foreground image.
If the foreground CLUT format is 32-bit, the address must be 32-bit aligned.

21.5.13 DMA2D background CLUT memory address register (DMA2D_BGCMAR)

Address offset: 0x030

Reset value: 0x0000 0000

This register can only be written when transfers are disabled. Once the CLUT transfer started, this register is read-only.

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

Bits 31:0 **MA[31:0]**: Memory address

Address of the data used for the CLUT address dedicated to the background image. If the background CLUT format is 32-bit, the address must be 32-bit aligned.

21.5.14 DMA2D output PFC control register (DMA2D_OPFCCR)

Address offset: 0x034

Reset value: 0x0000 0000

This register can only be written when transfers are disabled. Once the CLUT transfer started, this register is read-only.

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

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

Bit 21 **RBS**: Red/Blue swap

This bit allows to swap Red and Blue to support BGR or ABGR color formats.
0: Regular mode (RGB or ARGB)
1: Swap mode (BGR or ABGR)

Bit 20 **AI**: Alpha Inverted

This bit inverts the alpha value.
0: Regular alpha
1: Inverted alpha

Bits 19:9 Reserved, must be kept at reset value.

Bit 8 **SB**: Swap bytes
When this bit is set, the bytes in the output FIFO are swapped two by two. The number of pixels per line (PL) must be even, and the output memory address (OMAR) must be even.
0: Bytes in regular order in the output FIFO
1: Bytes swapped two by two in the output FIFO

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 **CM[2:0]**: Color mode
These bits define the color format of the output image.
000: ARGB8888
001: RGB888
010: RGB565
011: ARGB1555
100: ARGB4444
Others: Reserved

21.5.15 DMA2D output color register (DMA2D_OCOLR)

Address offset: 0x038

Reset value: 0x0000 0000

The same register is used to show the color values, with different formats depending on the color mode.

This register can only be written when transfers are disabled. Once the CLUT transfer started, this register is read-only.

ARGB8888 or RGB888 color mode

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

Bits 31:24 **ALPHA[7:0]**: Alpha channel value of the output color in ARGB8888 mode (otherwise reserved)

Bits 23:16 **RED[7:0]**: Red value of the output image in ARGB8888 or RGB888 mode

Bits 15:8 **GREEN[7:0]**: Green value of the output image in ARGB8888 or RGB888

Bits 7:0 **BLUE[7:0]**: Blue value of the output image in ARGB8888 or RGB888

21.5.16 DMA2D output color register [alternate] (DMA2D_OCOLR)

Address offset: 0x038

Reset value: 0x0000 0000

The same register is used to show the color values, with different formats depending on the color mode.

This register can only be written when transfers are disabled. Once the CLUT transfer started, this register is read-only.

RGB565 color mode

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RED[4:0]					GREEN[5:0]						BLUE[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:11 **RED[4:0]**: Red value of the output image in RGB565 mode

Bits 10:5 **GREEN[5:0]**: Green value of the output image in RGB565 mode

Bits 4:0 **BLUE[4:0]**: Blue value of the output image in RGB565 mode

21.5.17 DMA2D output color register [alternate] (DMA2D_OCOLR)

Address offset: 0x038

Reset value: 0x0000 0000

The same register is used to show the color values, with different formats depending on the color mode.

This register can only be written when transfers are disabled. Once the CLUT transfer started, this register is read-only.

ARGB1555 color mode

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A	RED[4:0]					GREEN[4:0]					BLUE[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **A**: Alpha channel value of the output color in ARGB1555 mode

Bits 14:10 **RED[4:0]**: Red value of the output image in ARGB1555 mode

Bits 9:5 **GREEN[4:0]**: Green value of the output image in ARGB1555 mode

Bits 4:0 **BLUE[4:0]**: Blue value of the output image in ARGB1555 mode

21.5.18 DMA2D output color register [alternate] (DMA2D_OCOLR)

Address offset: 0x038

Reset value: 0x0000 0000

The same register is used to show the color values, with different formats depending on the color mode.

This register can only be written when transfers are disabled. Once the CLUT transfer started, this register is read-only.

ARGB4444 color mode

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **ALPHA[3:0]**: Alpha channel of the output color value in ARGB4444

Bits 11:8 **RED[3:0]**: Red value of the output image in ARGB4444 mode

Bits 7:4 **GREEN[3:0]**: Green value of the output image in ARGB4444 mode

Bits 3:0 **BLUE[3:0]**: Blue value of the output image in ARGB4444 mode

21.5.19 DMA2D output memory address register (DMA2D_OMAR)

Address offset: 0x03C

Reset value: 0x0000 0000

This register can only be written when transfers are disabled. Once the CLUT transfer started, this register is read-only.

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

Bits 31:0 **MA[31:0]**: Memory address

Address of the data used for the output FIFO.

The address alignment must match the image format selected: a 32-bit per pixel format must be 32-bit aligned and a 16-bit per pixel format must be 16-bit aligned.

21.5.20 DMA2D output offset register (DMA2D_OOR)

Address offset: 0x040

Reset value: 0x0000 0000

This register can only be written when transfers are disabled. Once the CLUT transfer started, this register is read-only.

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **LO[15:0]**: Line offset

This field gives the line offset used for the output, expressed:

- in pixels when LOM = 0 in DMA2D_CR. Only LO[13:0] bits are considered, LO[15:14] bits are ignored.
- in bytes when LOM = 1

This value is used for the address generation. It is added at the end of each line to determine the starting address of the next line.

21.5.21 DMA2D number of line register (DMA2D_NLR)

Address offset: 0x044

Reset value: 0x0000 0000

This register can only be written when transfers are disabled. Once the CLUT transfer started, this register is read-only.

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

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:16 **PL[13:0]**: Pixel per lines per lines of the area to be transferred

If any of the input image format is 4-bit per pixel, pixel per lines must be even.

Bits 15:0 **NL[15:0]**: Number of lines of the area to be transferred.

21.5.22 DMA2D line watermark register (DMA2D_LWR)

Address offset: 0x048

Reset value: 0x0000 0000

This register can only be written when transfers are disabled. Once the CLUT transfer started, this register is read-only.

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **LW[15:0]**: Line watermark for interrupt generation

An interrupt is raised when the last pixel of the watermarked line has been transferred.

21.5.23 DMA2D AXI master timer configuration register (DMA2D_AMTCR)

Address offset: 0x04C

Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **DT[7:0]**: Dead time

Dead time value in the AXI clock cycle inserted between two consecutive accesses on the AXI master port. These bits represent the minimum guaranteed number of cycles between two consecutive AXI accesses.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **EN**: Dead-time functionality enable

21.5.24 DMA2D foreground CLUT (DMA2D_FGCLUTx)

Address offset: 0x400 + 0x4 * x, (x = 0 to 255)

Reset value: 0xXXXX XXXX

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

Bits 31:24 **ALPHA[7:0]**: Alpha
Alpha value for index {x} for the foreground

Bits 23:16 **RED[7:0]**: Red
Red value for index {x} for the foreground

Bits 15:8 **GREEN[7:0]**: Green
Green value for index {x} for the foreground

Bits 7:0 **BLUE[7:0]**: Blue
Blue value for index {x} for the foreground

21.5.25 DMA2D background CLUT (DMA2D_BGCLUTx)

Address offset: 0x800 + 0x4 * x, (x = 0 to 255)

Reset value: 0xXXXX XXXX

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

Bits 31:24 **ALPHA[7:0]**:Alpha
Alpha value for index {x} for the background

Bits 23:16 **RED[7:0]**: Red
Red value for index {x} for the background

Bits 15:8 **GREEN[7:0]**: Green
Green value for index {x} for the background

Bits 7:0 **BLUE[7:0]**: Blue
Blue value for index {x} for the background

21.5.26 DMA2D register map

Table 125. DMA2D register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	DMA2D_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE[2:0]			Res.	Res.	CEIE	CTCIE	CAEIE	TWIE	TCIE	TEIE	Res.	LOM	Res.	Res.	Res.	ABORT	SUSP	START
	Reset value														0	0	0			0	0	0	0	0	0		0				0	0	0
0x004	DMA2D_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CEIF	CTCIF	CAEIF	TWIF	TCIF	TEIF
	Reset value																											0	0	0	0	0	0
0x008	DMA2D_IFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCEIF	CCTCIF	CAECIF	CTWIF	CTCIF	CTEIF
	Reset value																											0	0	0	0	0	0
0x00C	DMA2D_FGMAR	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	DMA2D_FGOR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LO[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	DMA2D_BGMAR	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	DMA2D_BGOR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LO[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	DMA2D_FGPFCCR	ALPHA[7:0]								Res.	Res.	RBS	AI	CSS[1:0]		AM[1:0]		CS[7:0]								Res.	Res.	START	CCM	CM[3:0]			
	Reset value	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0
0x020	DMA2D_FGCOLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RED[7:0]								GREEN[7:0]								BLUE[7:0]							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	DMA2D_BGPFCCR	ALPHA[7:0]								Res.	Res.	RBS	AI	Res.	Res.	AM[1:0]		CS[7:0]								Res.	Res.	START	CCM	CM[3:0]			
	Reset value	0	0	0	0	0	0	0	0			0	0			0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0
0x028	DMA2D_BGCOLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RED[7:0]								GREEN[7:0]								BLUE[7:0]							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	DMA2D_FGCMAR	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	DMA2D_BGCMAR	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	DMA2D_OPFCCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RBS	AI	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SB	Res.	Res.	Res.	Res.	Res.	CM[2:0]		
	Reset value											0	0												0						0	0	0
0x038	DMA2D_OCOLR ARGB8888 or RGB888 color mode	ALPHA[7:0]								RED[7:0]								GREEN[7:0]								BLUE[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	DMA2D_OCOLR RGB565 color mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RED[4:0]					GREEN[6:0]							BLUE[4:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	DMA2D_OCOLR ARGB1555 color mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	A	RED[4:0]					GREEN[4:0]						BLUE[4:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 125. DMA2D register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x038	DMA2D_OCOLR ARGB4444 color mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALPHA[3:0]				RED[3:0]				GREEN[3:0]				BLUE[3:0]			
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	DMA2D_OMAR	MA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x040	DMA2D_OOR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LO[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	DMA2D_NLR	Res.	Res.	PL[13:0]														NL[15:0]															
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048	DMA2D_LWR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LW[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	DMA2D_AMTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res	Res.	Res.	Res.	Res.	DT[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN
	Reset value																	0	0	0	0	0	0	0	0								0
0x050-0x3FC	Reserved	Reserved																															
0x400 + 0x4*x, (x=0 to 255)	DMA2D_FGCLUTx	ALPHA[7:0]								RED[7:0]								GREEN[7:0]								BLUE[7:0]							
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x800 + 0x4*x, (x=0 to 255)	DMA2D_BGCLUTx	ALPHA[7:0]								RED[7:0]								GREEN[7:0]								BLUE[7:0]							
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

22 Chrom-GRC (GFXMMU)

22.1 GFXMMU introduction

The Chrom-GRC (GFXMMU) is a graphical oriented memory management unit aimed to optimize memory usage according to the display shape.

22.2 GFXMMU main features

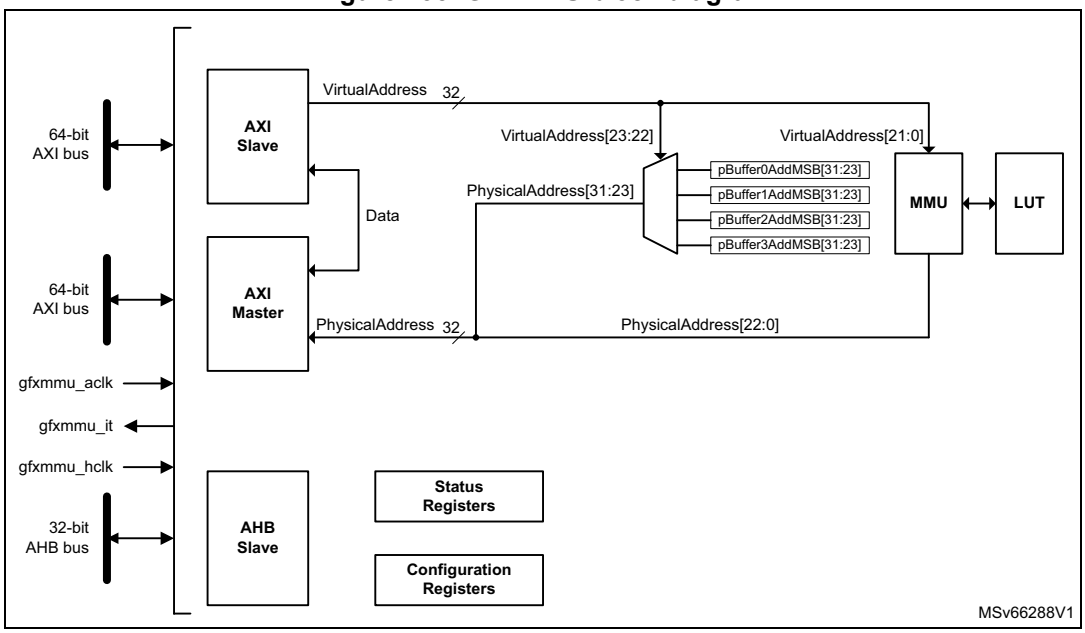
- Fully programmable display shape to physically store only the visible pixel
- Up to 4 virtual buffers
- Each virtual buffer have 3072 or 4096 bytes per line and 1024 lines
- Each virtual buffer can be physically mapped to any system memory
- Packing and unpacking operation to store 32-bit pixel data into 24-bit packed
- Interrupt in case of buffer overflow (1 per buffer)
- Interrupt in case of memory transfer error

22.3 GFXMMU functional and architectural description

The GFXMMU is responsible of address resolution to convert the virtual buffer address into the physical buffer address.

22.3.1 GFXMMU block diagram

Figure 135. GFXMMU block diagram



22.3.2 GFXMMU internal signals

Table 126. GFXMMU internal input/output signals

Internal signal name	Signal type	Description
gfxmmu_aclk	input	GFXMMU AXI clock
gfxmmu_hclk	Input	GFXMMU AHB clock
gfxmmu_it	Output	GFXMMU global interrupt

22.3.3 Virtual memory

The GFXMMU provides a virtual memory space seen by the system masters. This virtual memory space is divided into four virtual buffers.

Virtual buffer

A virtual buffer is seen by any system master as a continuous memory space representing a virtual frame buffer of 1024 lines.

Each line is divided into 256 12-byte or 16-byte blocks depending on the BS bit of the GFXMMU configuration register (GFXMMU_CR).

When the address translation is enabled setting the address translation enable (ATE) bit of the graphic MMU configuration register (GFXMMU_CR), each line can be configured individually to be mapped onto a physical memory.

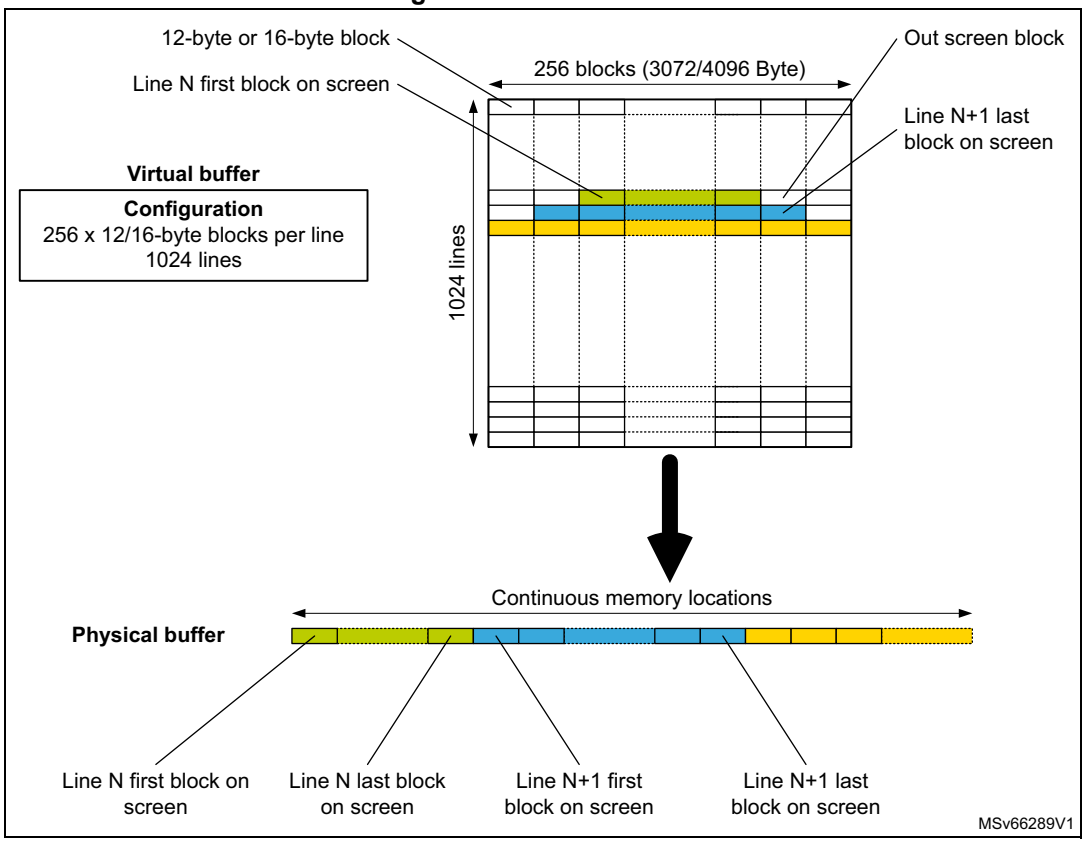
Depending on the display shape and size, only the necessary blocks are mapped to a physical memory location. This mapping is done programming the LUT entry for each line:

- The enable of the line
- The number of the first “visible” block
- The number of the last “visible” block
- The address offset of the line within the physical buffer

The “visible” blocks can be arranged in the physical buffer in a continuous way programming the address offset of each line.

The LUT is common to all the buffers i.e. all the buffers have the same “shape”.

Figure 136. Virtual buffer



Virtual buffer overview

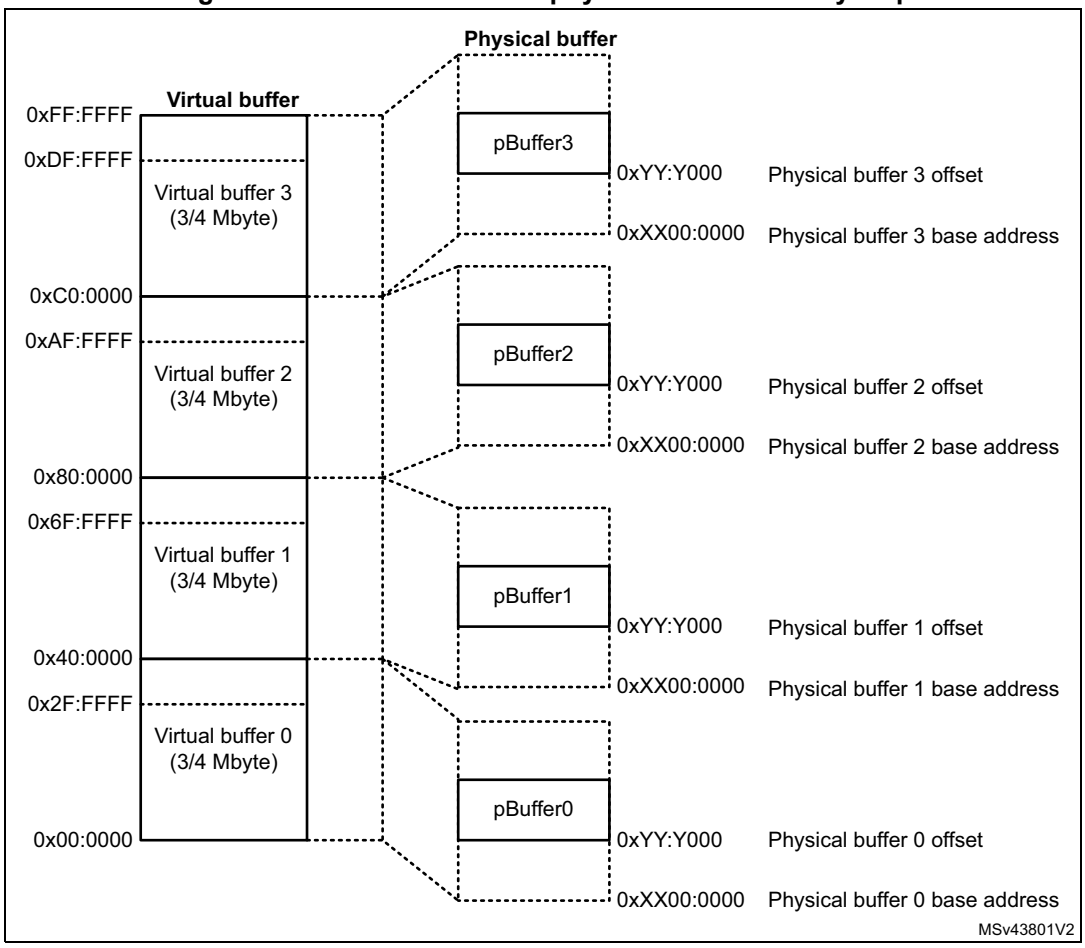
For a frame buffer coded in 32 bpp or 16 bpp, the virtual buffer can be configured to have 12-byte or 16-byte blocks. This results in a virtual frame buffer of 768 x 1024 or 1024 x 1024 pixels for 32 bpp and 1536 x 1024 or 2048 x 1024 for 16 bpp.

For a frame buffer coded in 24 bpp, the virtual buffer must be configured to have 12-byte blocks to have an integer number of pixel per lines. This results in a virtual frame buffer of 1024 x 1024 pixels for 24 bpp.

Each buffer can be physically mapped anywhere in the physical memory thanks to:

- The physical buffer base address (PBBA) field of the GFXMMU buffer x configuration register (GFXMMU_BxCR). It configures the physical location of the 8-Mbyte area where the buffer is mapped.
- The physical buffer location respective to the physical buffer base address is defined by the physical buffer offset (PBO) field of the GFXMMU buffer x configuration register (GFXMMU_BxCR).

Figure 137. Virtual buffer and physical buffer memory map



The buffer can not overflow the 8-Mbyte boundary of the zone defined by its base address. In case of overflow, the buffer x overflow flag (BxOF) of the GFXMMU status register (GFXMMU_SR) is set and an interrupt is generated if the buffer x overflow interrupt enable (BxOIE) bit of the GFXMMU configuration register (GFXMMU_CR) is set.

Virtual buffer application use case

As the physical locations are independently configurable, the four virtual buffers can be physically mapped to non continuous locations. This allows for example to have the four buffers mapped on to four different SDRAM banks and avoid extra precharge cycles accessing the SDRAM.

As a consequence, one buffer must be used by the CPU/Graphic accelerator for frame buffer calculation while an other one must be used by the LTDC.

The two remaining buffers can be used as a graphical library requiring extra drawing buffers.

22.3.4 Packing and unpacking

When addressing 24bpp frame buffers, the virtual buffer must be configured in 12-byte block mode to have an integer number of pixel per block.

Nevertheless, it is possible to access these 24-bit data as if they where unpacked in a 32-bit word.

The packing mode is enabled, per buffer setting the buffer x packing enable (BxPE) bit of the graphic MMU configuration register (GFXMMU_CR).

When the packing mode is enabled and the 12-byte mode is selected, the access to the buffer are done as in 16-byte mode, but either the most significant byte or the less significant byte is ignored to transform each 32-bit word into a 24-bit word.

The ignored byte is selected with the buffer x packing mode (BxPM) bit of the graphic MMU configuration register (GFXMMU_CR).

During write operation MSB or LSB is ignored. During read operation, the ignored MSB or LSB is replaced by the 8-bit default value programmed in the default alpha (DA) field of the graphic MMU default alpha register (GFXMMU_DAR)

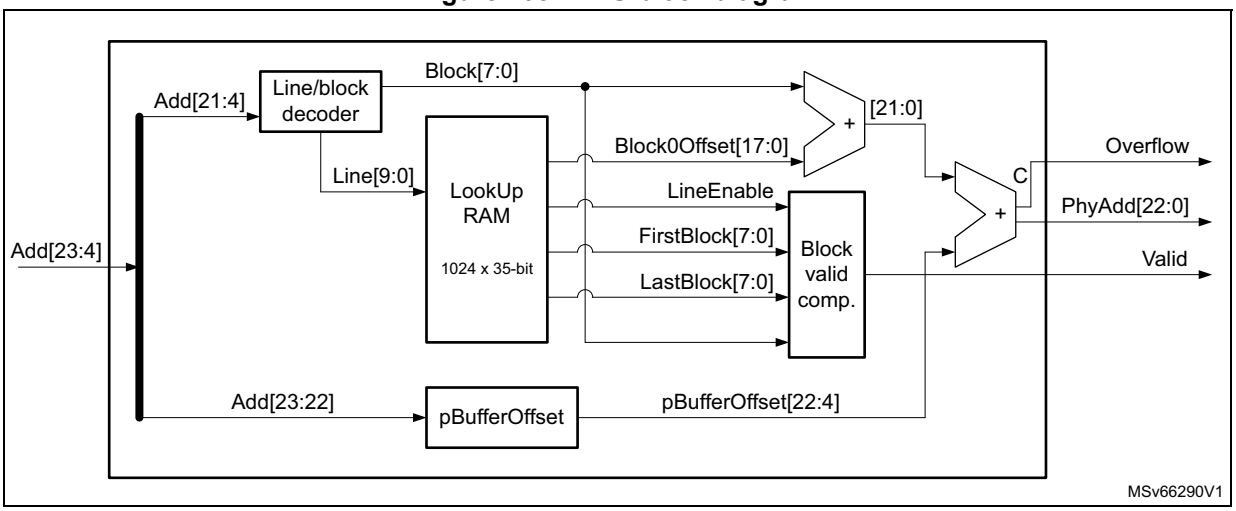
The buffer x packing enable (BxPE) bit has no effect when the graphic MMU is configured in 16-byte block.

22.3.5 MMU architecture

The MMU block is responsible of the address resolution. It receives the 24-bit address and returns the physical 23-bit address and a valid signals to indicate the address is physically mapped or not. The MMU also checks overflow of a area boundary.

The MMU LUT is implemented as a 1024 x 35-bit RAM

Figure 138. MMU block diagram



Line block decoder

The line block decoder is generating the block number and the line number according the address.

Look up RAM

The look up RAM is a 1024 x 35-bit RAM with the following fields:

- 1-bit line enable
- 8-bit first valid block
- 8-bit last valid block
- 18-bit for line offset

As the RAM is bigger than a word, each entry is split into two words on the memory map. The write access are done in two steps:

1. Write the first word with enable/first valid block/last valid block in the GFXMMU_LUTxL memory location (internally buffered).
2. Write the second word with line offset in the GFXMMU_LUTxH memory location (effective write into the memory together with the internally buffered value).

A write in the LUT can happen any time but it can lead to inconsistencies if a master is using the MMU at the same time. As the CPU has the priority during LUT programming, this may slow down MMU calculation.

There is no restriction during read operations, but this may slow down CPU as the MMU has the priority on LUT accesses.

Block validation/comparator

This block is checking if the block is valid.

A block is considered as valid (physically mapped) when:

- Line is enable.
- The block number is greater or equal to the first valid block.
- The block number is lower or equal to the last valid block.

When the block is valid, the physical address generated is considered as correct.

If the result of the MMU evaluation is not valid, the write operations are ignored, and read operations return the default 32-bit value stored in the default value (DV) field of the GFXMMU default value register (GFXMMU_DVR).

Block offset address calculation within the buffer

The block number is added to the line offset to get the offset of the block within the physical buffer.

As a consequence, the line offset stored in the LUT is given by the following formula:

$$\text{Line offset} = [(\text{Number of visible blocks already used}) - (1^{st} \text{ visible block})]$$

with:

- The maximum value for the line offset is when all the block of all the line are used. As the consequence the line offset for the last line can be maximum:
  1023 x 256 = 0x3:FF00
- The minimum value for the line offset is when the last block of the first line is the first valid block: -255 = - 0xFF i.e 0x3:FF01

As the consequence the full range of the line offset entry of the LUT is used.

Carry is not taken into account as this stage to be able to perform negative offset calculations (values from 0x3:FF01 to 0x3:FFFF)

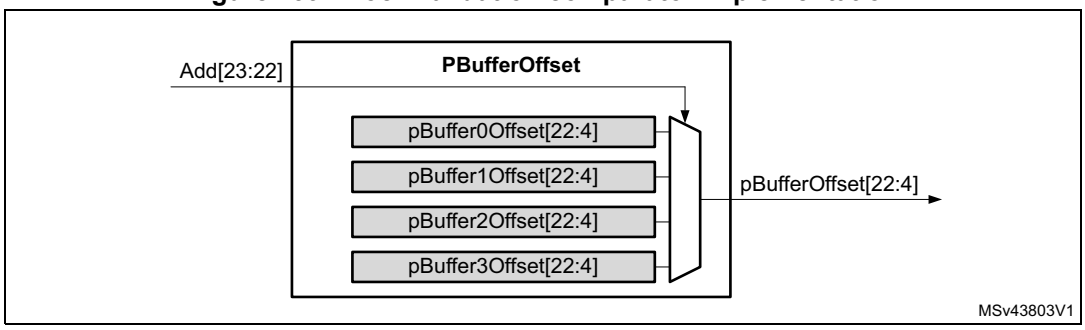
As the block offset is within a 4-Mbyte buffer, the address generated is 22-bit wide.

Block offset address calculation

Once the offset of the block within the buffer as been calculated, this value is added to the offset of the block respective to the physical buffer base address.

The offset of the blocks are defined in registers as shown in *Figure 139*:

Figure 139. Block validation/comparator implementation



The resulting address and the buffer offset address must be on 23-bit.

The carry is taken into account to trigger address overflow. The carry is propagated to the GFXMMU status register (GFXMMU_SR) to set the buffer x overflow flag (BxOF).

Example of calculation

Consider the following configuration for virtual buffer 0:

- 16-byte block mode
- First visible block of line 0: block 7
- Number of visible block in line 0: 10
- First visible block of line 1: block 6
- Number of visible block in line 1: 12
- Address of the physical buffer: 0xC020:0000

The configuration must be:

- The base address of the physical buffer 0: 0xC000:0000
- The offset of buffer 0: 0x20:0000
- First visible block of line 0: block 7
- Last visible block of line 0: block 16
- Block 0 offset of line 0: (0 - 7) = -0x7 = 0x3:FFF9
- First visible block of line 1: block 6
- Last visible block of line 1: block 17
- Block 0 offset of line 1: (10 - 6) = (0xA - 0x6) = 0x4

As a consequence:

- the physical address of block 7 of line 0 is:

0xC000:0000 + 0x20:0000 + (0x3:FFF9 + 0x7 without carry) x 0x10 = 0xC020:0000

- the physical address of block 16 of line 0 is:

0xC000:0000 + 0x20:0000 + (0x3:FFF9 + 0x10 without carry) x 0x10 = 0xC020:0090

- the physical address of block 6 of line 1 is:

0xC000:0000 + 0x20:0000 + (0x4 + 0x6 without carry) x 0x10 = 0xC020:00A0

- the physical address of block 17 of line 1 is:

0xC000:0000 + 0x20:0000 + (0x4 + 0x11 without carry) x 0x10 = 0xC020:0150

For the same example in 12-byte mode, the user has the same configuration regarding the offsets by the addresses are different in the physical buffer:

- the physical address of block 7 of line 0 is:

0xC000:0000 + 0x20:0000 + (0x3:FFF9 + 0x7 without carry) x 0x0C = 0xC020:0000

- the physical address of block 16 of line 0 is:

0xC000:0000 + 0x20:0000 + (0x3:FFF9 + 0x10 without carry) x 0x0C = 0xC020:006C

- the physical address of block 6 of line 1 is:

0xC000:0000 + 0x20:0000 + (0x4 + 0x6 without carry) x 0x0C = 0xC020:0078

- the physical address of block 17 of line 1 is:

0xC000:0000 + 0x20:0000 + (0x4 + 0x11 without carry) x 0x0C = 0xC020:00FC

22.4 GTZC TZIC GFXMMU interrupts

An interrupt can be produced on the following events:

- Buffer 0 overflow
- Buffer 1 overflow
- Buffer 2 overflow
- Buffer 3 overflow
- AXI master error

Separate interrupt enable bits are available for flexibility.

Table 127. GFXMMU interrupt requests

Interrupt event	Event flag	Enable control bit
Buffer 0 overflow	B0OF	B0OIE
Buffer 1 overflow	B1OF	B1OIE
Buffer 2 overflow	B2OF	B2OIE
Buffer 3 overflow	B3OF	B3OIE
AXI master error	AMEF	AMEIE

22.5 GFXMMU registers

22.5.1 GFXMMU configuration register (GFXMMU_CR)

Address offset: 0x0000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
B3PM	B3PE	B2PM	B2PE	B1PM	B1PE	B0PM	B0PE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ATE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BS	Res.	AMEIE	B3OIE	B2OIE	B1OIE	B0OIE
rw									rw		rw	rw	rw	rw	rw

Bit 31 **B3PM**: Buffer 3 packing mode

This bit selects the byte to be removed during packing operations on buffer 3

0: MSB is removed

1: LSB is removed

Bit 30 **B3PE**: Buffer 3 packing enable

This bit enables the packing on buffer 3.

The packing is functional only if the block size is configured in 12-byte mode. In 16-byte mode, this bit is ignored.

0: Packing is disabled

1: Packing is enable

Bit 29 **B2PM**: Buffer 2 packing mode

This bit selects the byte to be removed during packing operations on buffer 2

0: MSB is removed

1: LSB is removed

Bit 28 **B2PE**: Buffer 2 packing enable

This bit enables the packing on buffer 2.

The packing is functional only if the block size is configured in 12-byte mode. In 16-byte mode, this bit is ignored.

0: Packing is disabled

1: Packing is enable

Bit 27 **B1PM**: Buffer 1 packing mode

This bit selects the byte to be removed during packing operations on buffer 1

0: MSB is removed

1: LSB is removed

Bit 26 **B1PE**: Buffer 1 packing enable

This bit enables the packing on buffer 1.

The packing is functional only if the block size is configured in 12-byte mode. In 16-byte mode, this bit is ignored.

0: Packing is disabled

1: Packing is enable

Bit 25 **B0PM**: Buffer 0 packing mode

This bit selects the byte to be removed during packing operations on buffer 0

0: MSB is removed

1: LSB is removed

Bit 24 **B0PE**: Buffer 0 packing enable
This bit enables the packing on buffer 0.
The packing is functional only if the block size is configured in 12-byte mode. In 16-byte mode, this bit is ignored.
0: Packing is disabled
1: Packing is enable

Bits 23:16 Reserved, must be kept at reset value.

Bit 15 **ATE**: Address translation enable
This bit enables the address translation based on the values programmed in the LUT.
0: Address translation is disable
1: Address translation is enable

Bits 14:7 Reserved, must be kept at reset value.

Bit 6 **BS**: Block size
This bit defines the size of the blocks
0: 16-byte blocks
1: 12-byte blocks

Bit 5 Reserved, must be kept at reset value.

Bit 4 **AMEIE**: AXI master error interrupt enable
This bit enables the AXI master error interrupt.
0: Interrupt disable
1: Interrupt enabled

Bit 3 **B3OIE**: Buffer 3 overflow interrupt enable
This bit enables the buffer 3 overflow interrupt.
0: Interrupt disable
1: Interrupt enabled

Bit 2 **B2OIE**: Buffer 2 overflow interrupt enable
This bit enables the buffer 2 overflow interrupt.
0: Interrupt disable
1: Interrupt enabled

Bit 1 **B1OIE**: Buffer 1 overflow interrupt enable
This bit enables the buffer 1 overflow interrupt.
0: Interrupt disable
1: Interrupt enabled

Bit 0 **B0OIE**: Buffer 0 overflow interrupt enable
This bit enables the buffer 0 overflow interrupt.
0: Interrupt disable
1: Interrupt enabled

22.5.2 GFXMMU status register (GFXMMU_SR)

Address offset: 0x0004

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AMEF	B3OF	B2OF	B1OF	B0OF
											r	r	r	r	r

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

Bit 4 **AMEF**: AXI master error flag
This bit is set when an AXI error happens during a transaction. It is cleared by writing 1 to CAMEF.

Bit 3 **B3OF**: Buffer 3 overflow flag
This bit is set when an overflow occurs during the offset calculation of the buffer 3. It is cleared by writing 1 to CB3OF.

Bit 2 **B2OF**: Buffer 2 overflow flag
This bit is set when an overflow occurs during the offset calculation of the buffer 2. It is cleared by writing 1 to CB2OF.

Bit 1 **B1OF**: Buffer 1 overflow flag
This bit is set when an overflow occurs during the offset calculation of the buffer 1. It is cleared by writing 1 to CB1OF.

Bit 0 **B0OF**: Buffer 0 overflow flag
This bit is set when an overflow occurs during the offset calculation of the buffer 0. It is cleared by writing 1 to CB0OF.

22.5.3 GFXMMU flag clear register (GFXMMU_FCR)

Address offset: 0x0008

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CAMEF	CB3OF	CB2OF	CB1OF	CB0OF
											rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

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

Bit 4 **CAMEF**: Clear AXI master error flag
Writing 1 clears the AXI master error flag in the GFXMMU_SR register.

Bit 3 **CB3OF**: Clear buffer 3 overflow flag
Writing 1 clears the buffer 3 overflow flag in the GFXMMU_SR register.

Bit 2 **CB2OF**: Clear buffer 2 overflow flag
Writing 1 clears the buffer 2 overflow flag in the GFXMMU_SR register.

Bit 1 **CB1OF**: Clear buffer 1 overflow flag
Writing 1 clears the buffer 1 overflow flag in the GFXMMU_SR register.

Bit 0 **CB0OF**: Clear buffer 0 overflow flag
Writing 1 clears the buffer 0 overflow flag in the GFXMMU_SR register.

22.5.4 GFXMMU default value register (GFXMMU_DVR)

Address offset: 0x0010

Reset value: 0x0000 0000

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

Bits 31:0 **DV[31:0]**: Default value
This field indicates the default 32-bit value which is returned when a master accesses a virtual memory location not physically mapped.

22.5.5 GFXMMU default alpha register (GFXMMU_DAR)

Address offset: 0x0014

Reset value: 0x0000 0000

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

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **DA[7:0]**: Default alpha
This field indicates the default 8-bit value which is merged with the 24-bit value when a master accesses a virtual memory location in packed mode.

22.5.6 GFXMMU buffer x configuration register (GFXMMU_BxCR)

Address offset: 0x0020 + 0x4 * x, (x = 0 to 3)

Reset value: 0x0000 0000

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

Bits 31:23 **PBBA[31:23]**: Physical buffer base address
Base address MSB of the physical buffer.

Bits 22:4 **PBO[22:4]**: Physical buffer offset
Offset of the physical buffer.

Bits 3:0 Reserved, must be kept at reset value.

22.5.7 GFXMMU LUT entry x low (GFXMMU_LUTxL)

Address offset: 0x1000 + 0x8 * x, (x = 0 to 1023)

Reset value: 0x0000 0000

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

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **LVB[7:0]**: Last valid block
Number of the last valid block of line number X.

Bits 15:8 **FVB[7:0]**: First valid block
Number of the first valid block of line number x.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **EN**: Enable
Line enable.
0: Line is disabled (no MMU evaluation is performed)
1: Line is enabled (MMU evaluation is performed)

22.5.8 GFXMMU LUT entry x high (GFXMMU_LUTxH)

Address offset: 0x1004 + 0x8 * x, (x = 0 to 1023)

Reset value: 0x0000 0000

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

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

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

Bits 17:0 **LO[17:0]**: Line offset

Line offset of line number x expressed in number of blocks

22.5.9 GFXMMU register map

Table 128. GFXMMU register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	GFXMMU_CR	B3PM	B3PE	B2PM	B2PE	B1PM	B1PE	B0PM	B0PE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BS	Res.	AMEIE	B3OIE	B2OIE	B1OIE	B0OIE
	Reset value	0	0	0	0	0	0	0	0									0									0		0	0	0	0	0
0x0004	GFXMMU_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AMEF	B3OF	B2OF	B1OF	B0OF
	Reset value																												0	0	0	0	0
0x0008	GFXMMU_FCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CAMEF	CB3OF	CB2OF	CB1OF	CB0OF
	Reset value																												0	0	0	0	0
0x000C	Reserved	Reserved																															
0x0010	GFXMMU_DVR	DV[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0014	GFXMMU_DAR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x001C	Reserved	Reserved																															
0x0020	GFXMMU_B0CR	PBBA[31:23]								PBO[22:4]																				Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x0024	GFXMMU_B1CR	PBBA[31:23]								PBO[22:4]																				Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x0028	GFXMMU_B2CR	PBBA[31:23]								PBO[22:4]																				Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x002C	GFXMMU_B3CR	PBBA[31:23]								PBO[22:4]																				Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
0x0030-0x0FFC	Reserved	Reserved																															

Table 128. GFXMMU register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1000	GFXMMU_LUT0L	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LVB[7:0]								FVB[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								0
0x1004	GFXMMU_LUT0H	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LO[17:0]																	
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0				
...																																	
0x2FF8	GFXMMU_LUT1023L	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LVB[7:0]								FVB[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								0
0x2FFC	GFXMMU_LUT1023H	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LO[17:0]																	
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0				

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

23 Graphic timer (GFXTIM)

23.1 GFXTIM introduction

The graphic timer (GFXTIM) is a graphic oriented timer allowing smart management of graphical events for frame or line counting.

23.2 GFXTIM main features

- Integrated frame and line-clock generation
- One absolute frame counter with one compare channel
- Two auto-reload relative frame counters
- One line timer with two compare channels
- External tearing-effect line management and synchronization
- Four programmable event generators with external trigger generation
- One watchdog counter

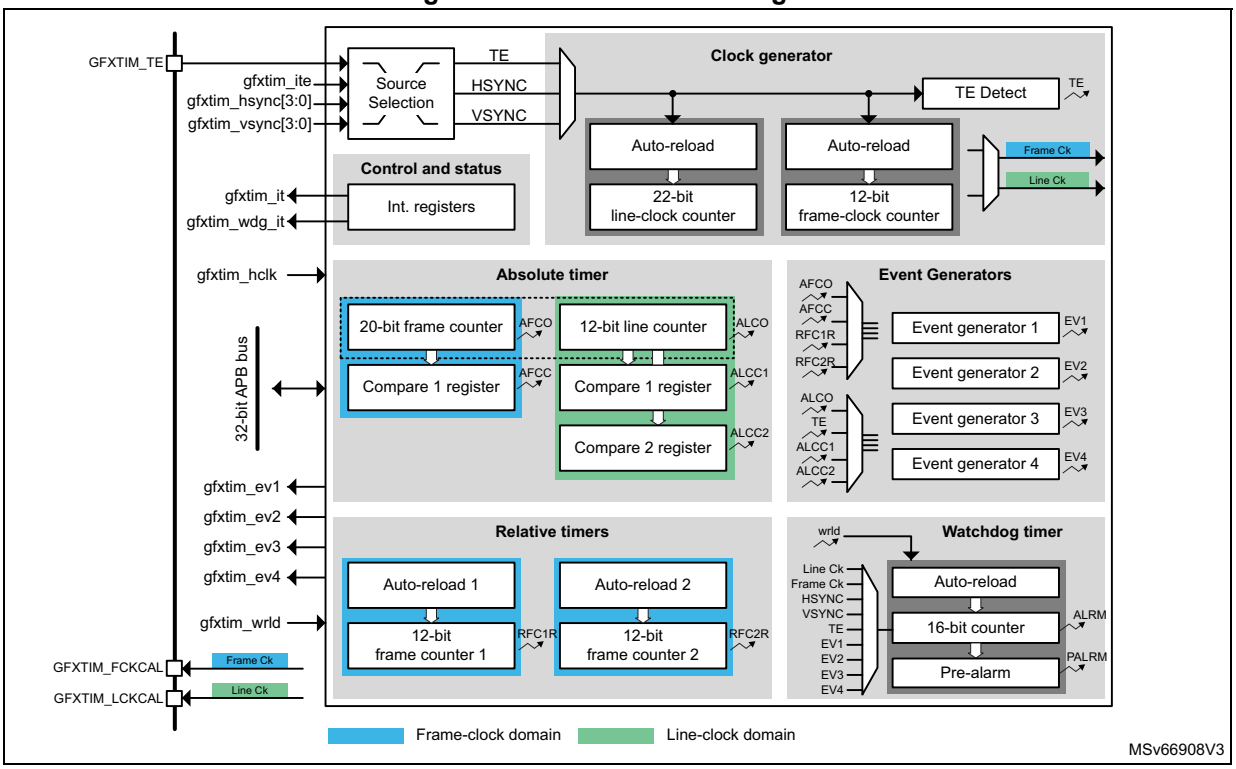
23.3 GFXTIM functional description

23.3.1 Block diagram

The graphic timer is split into six functional blocks

- Clock generator
- Absolute timers
- Relative timers
- Tearing-effect detection
- Event generators
- Watchdog timer

Figure 140. GFXTIM block diagram



23.3.2 GFXTIM pins and internal signals

Table 129. GFXTIM input/output pins

Pin name	Signal type	Description
GFXTIM_TE	Input	Tearing effect
GFXTIM_FCKCAL	Output	Frame-clock calibration output
GFXTIM_LCKCAL	Output	line-clock calibration output

Table 130. GFXTIM internal signals

Signal name	Signal type	Description
gfxtim_hclk	Digital input	Kernel and register interface clock
gfxtim_it	Digital output	Global interrupt
gfxtim_wdg_it	Digital output	Watchdog interrupt
gfxtim_ev1	Digital output	Graphic timer event 1
gfxtim_ev2	Digital output	Graphic timer event 2
gfxtim_ev3	Digital output	Graphic timer event 3
gfxtim_ev4	Digital output	Graphic timer event 4
gfxtim_wrld	Digital input	Watchdog reload
gfxtim_ite	Digital input	Internal tearing effect

Table 130. GFXTIM internal signals (continued)

Signal name	Signal type	Description
gfxtim_hsync[3:0]	Digital input	Horizontal synchronization
gfxtim_vsync[3:0]	Digital input	Vertical synchronization

The table below shows how GFXTIM triggers are connected.

Table 131. GFXTIM trigger interconnections

Trigger name	Direction	Trigger source/destination
gfxtim_ev1	Output	hpdma_trigsel[28]
		gpdma_trigsel[28]
gfxtim_ev2	Output	hpdma_trigsel[27]
		gpdma_trigsel[27]
gfxtim_ev3	Output	hpdma_trigsel[26]
		gpdma_trigsel[26]
gfxtim_ev4	Output	hpdma_trigsel[25]
		gpdma_trigsel[25]
gfxtim_wrld	Input	reserved
gfxtim_ite	Input	reserved
gfxtim_hsync[0]	Input	LCD_HSYNC
gfxtim_hsync[1]	Input	dcmipp_p0_hsync_evt
gfxtim_hsync[2]	Input	dcmipp_p1_hsync_evt
gfxtim_hsync[3]	Input	dcmipp_p2_hsync_evt
gfxtim_vsync[0]	Input	LCD_VSYNC
gfxtim_vsync[1]	Input	dcmipp_p0_vsync_evt
gfxtim_vsync[2]	Input	dcmipp_p1_vsync_evt
gfxtim_vsync[3]	Input	dcmipp_p2_vsync_evt

23.3.3 Clock generator

Two clocks are generated internally to fed the absolute and relative timers:

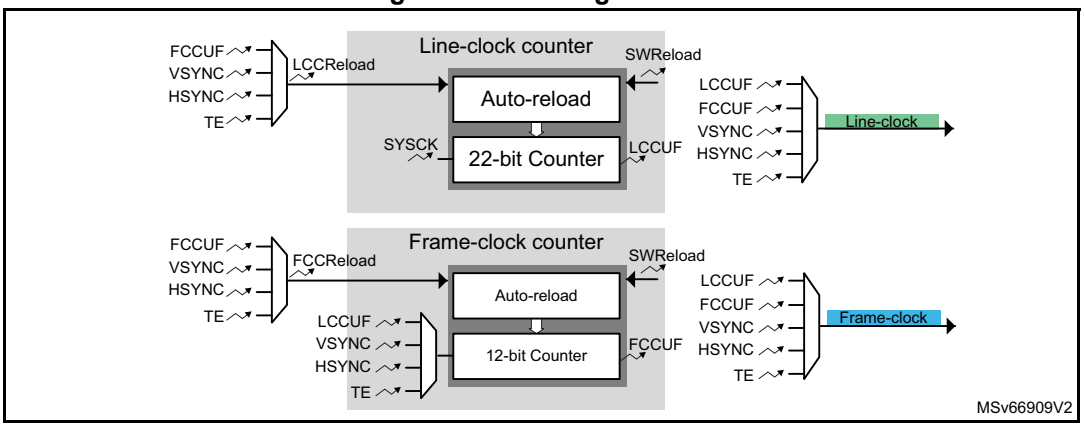
- the frame-clock: clocking frame counters
- the line-clock: clocking line counters

Internal counter for time base generation

The GFXTIM embeds a two-clock generation counter:

- a 22-bit auto-reload down-counter on the system clock
- a 12-bit auto-reload down-counter on selectable internal or external event

Figure 141. Clock generator



Line-clock counter

The line-clock counter is a 22-bit auto-reload down-counter on the system clock.

The line-clock counter is enabled selecting its clock source with LCCCS (line-clock counter clock source) in GFXTIM_CGCR (clock generator control register).

The line-clock counter can be reloaded automatically when one of the following event occurs:

- frame-clock counter underflows
- VSYNC edge (with control of polarity)
- HSYNC edge (with control of polarity)
- TE edge (with control of polarity)

The hardware reload function and source can be selected with LCCHRS (line-clock counter hardware reload source) in GFXTIM_CGCR (clock generator control register).

The line-clock counter can be reloaded by software, setting LCCFR (line-clock counter force reload) in GFXTIM_CGCR (clock generator control register).

The reload value is programmed in GFXTIM_LCCRR (line-clock counter reload register).

Frame-clock counter

The frame-clock counter if an 12-bit auto-reload down-counter clocked by either, TE, HYSNC, VSYNC or a line-clock counter underflow.

The frame-clock counter is enabled selecting its clock source with FCCCS in GFXTIM_CGCR.

The frame-clock counter can be reloaded automatically when one of the following event occurs:

- line-clock counter underflow
- VSYNC edge (with control of polarity)
- HSYNC edge (with control of polarity)
- TE edge (with control of polarity)

The hardware reload function and source can be selected with FCCHRS in GFXTIM_CGCR.

The frame-clock counter can be reloaded by software, setting FCCFR in GFXTIM_CGCR.

The reload value is programmed in GFXTIM_FCCRR.

Clock generation

The line-clock source can be one of the following:

- underflow flag of the internal 22-bit down-counter
- underflow flag of the internal 12-bit down-counter
- HSYNC, VSYNC or TE pin (with control of polarity)

The frame-clock source can be one of the following:

- underflow flag of the internal 22-bit down-counter
- underflow flag of the internal 12-bit down-counter
- HSYNC, VSYNC or TE pin (with control of polarity)

Clock calibration output

For calibration/debug purpose, the frame-clock and line-clock can be output on a specific I/O.

The frame-clock calibration output is enabled by setting FCCOE (frame-clock calibration output enable) in GFXTIM_CR (configuration register).

The line-clock calibration output is enabled by setting LCCOE in GFXTIM_CR.

Synchronization and tearing-effect sources

The GFXTIM can be connected to peripherals providing HSYNC and/or VSYNC synchronization signals, like the LCD-TFT controller or the camera interface.

The source of HSYNC and/or VSYNC is selected through SYNCS (synchronization source) in GFXTIM_CR.

The tearing-effect source can be an external pin or can be provided by the MIPI® DSI Host on system embedding this interface. To extend the orthogonality versus the synchronization signals, the tearing-effect source can be also the selected HSYNC or VSYNC input.

The tearing-effect source can be selected with TES (TE source) in GFXTIM_CR.

23.3.4 Example of clock generator configuration

The clock generator can have several configuration to work:

- in standalone (without any external synchronization)
- with external HSYNC and VSYNC
- with external HSYNC only
- with external VSYNC only
- with external CSYNC (TE) only

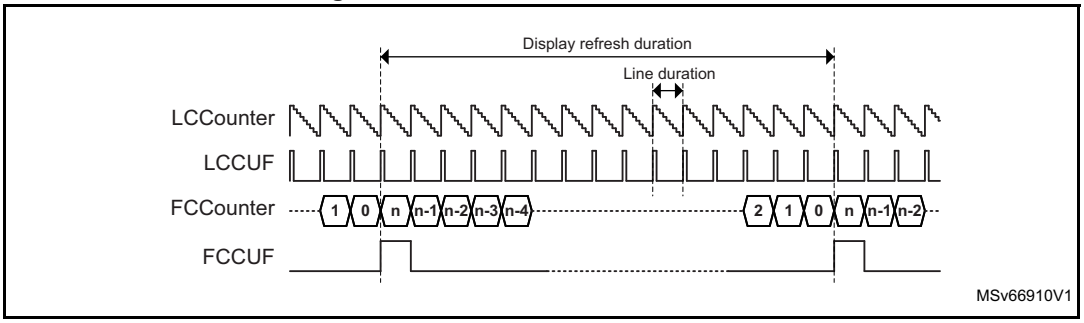
The synchronization signals, HSYNC, VSYNC and TE, are symmetrical in the implementation and can be exchanged if needed.

The set of examples detailed below are given for reference, but other combinations can be programmed int the clock generator.

Standalone

In standalone configuration, the clock generator provides to the GFXTIM, the frame-clock and the line-clock without any external signals.

Figure 142. Waveforms in standalone

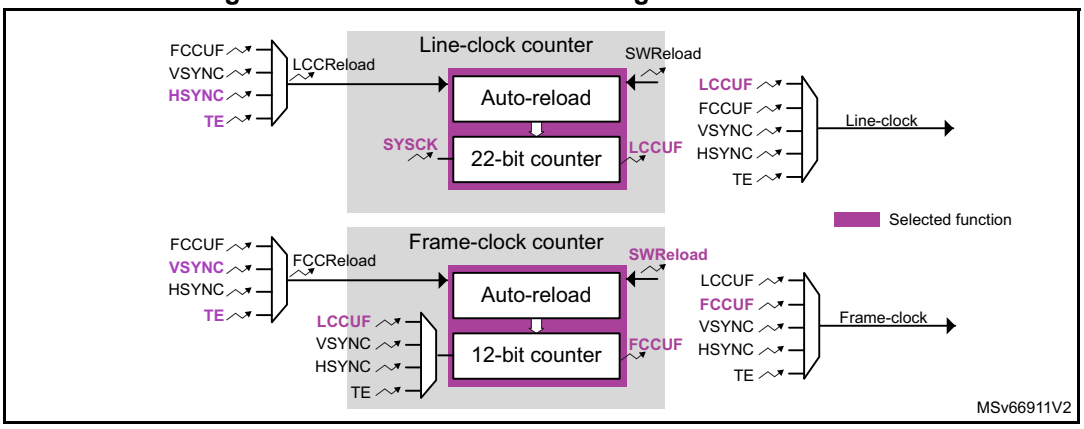


The LCCUF (line-clock counter underflow) event acts as line-clock and the FCCUF (frame-clock counter underflow) acts as frame-clock.

The clock generator can be synchronized by one of the following ways:

- by software reload, setting the FCCFR in GFXTIM_CGCR.
- by hardware reload on an external VSYNC (or TE)

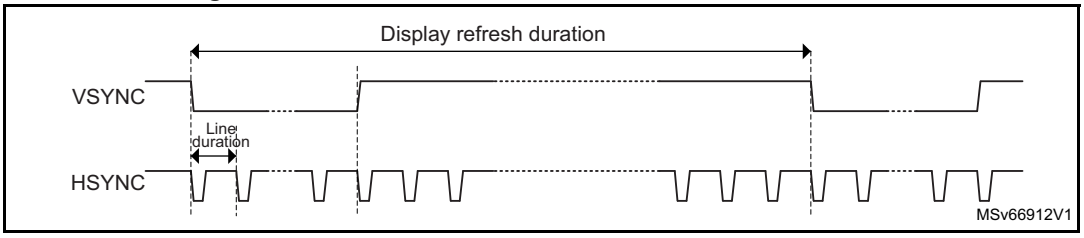
Figure 143. Active counters and signals in standalone



External HSYNC and VSYNC

When using external HSYNC and VSYNC, the counters are not used for the line and frame-clock generation. The clock generator copy directly HSYNC to the line-clock and VSYNC to the frame-clock.

Figure 144. Waveforms with external HSYNC and VSYNC

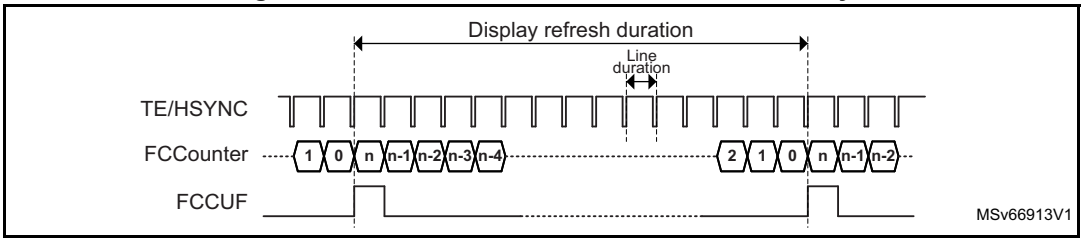


HSYNC acts as line-clock and VSYNC acts as frame-clock.

External HSYNC only

With external HSYNC only, the clock generator provides to the GFXTIM, the frame-clock and the line-clock based only on HSYNC.

Figure 145. Waveforms with external HSYNC only

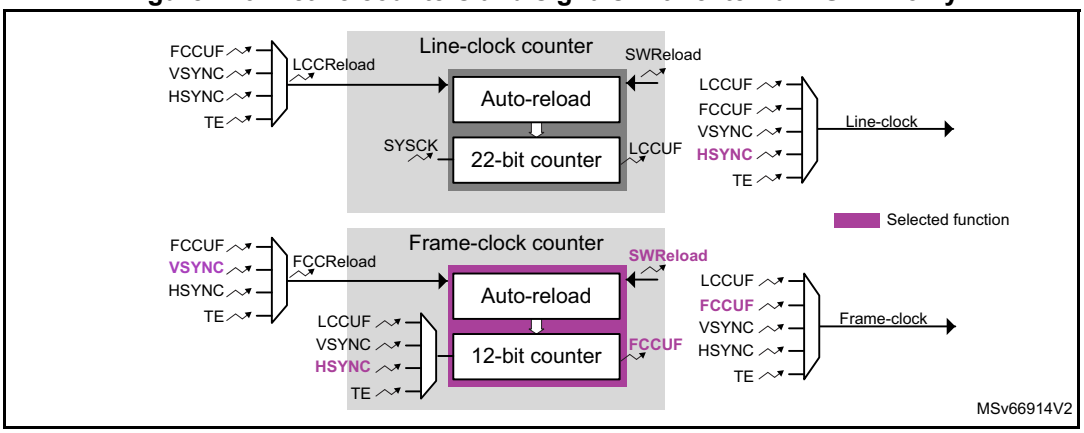


HSYNC acts as line-clock and FCCUF acts as frame-clock.

The clock generator can be synchronized by one of the following ways:

- by software reload, setting FCCFR in GFXTIM_CGCR
- by hardware reload on an external VSYNC (or TE)

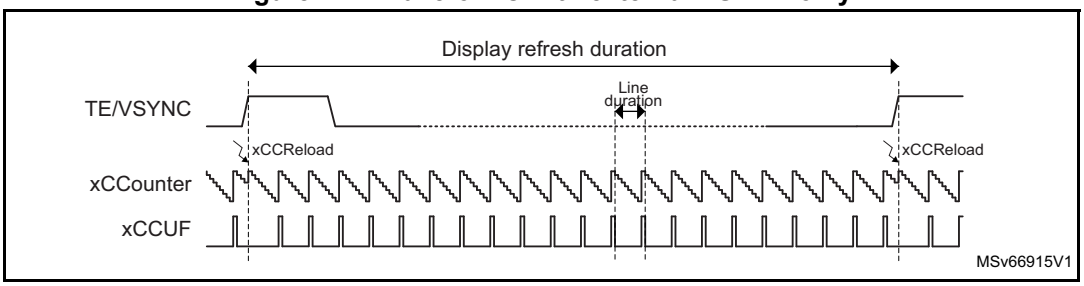
Figure 146. Active counters and signals with external HSYNC only



External VSYNC only

With external VSYNC only, the clock generator provides to the GFXTIM, the frame-clock and the line-clock based only on VSYNC.

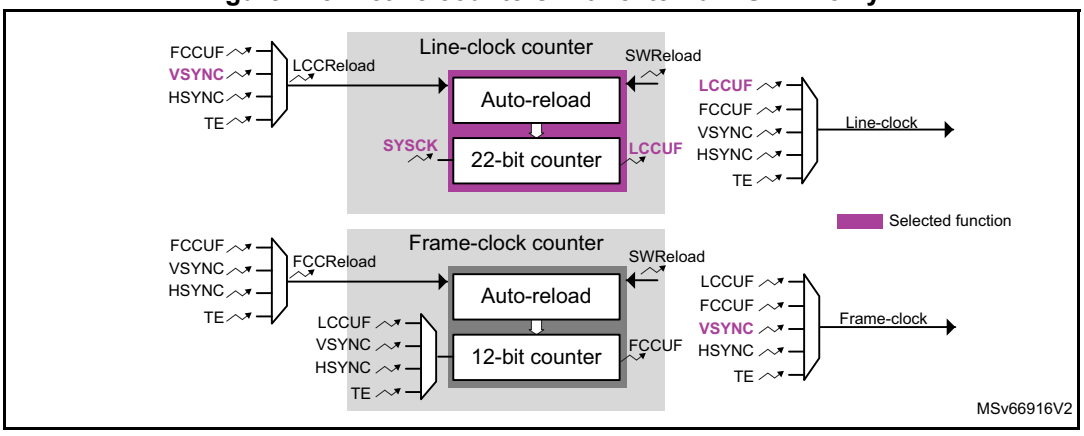
Figure 147. Waveforms with external VSYNC only



LCCUF (line-clock counter underflow) acts as line-clock and VSYNC acts as frame-clock.

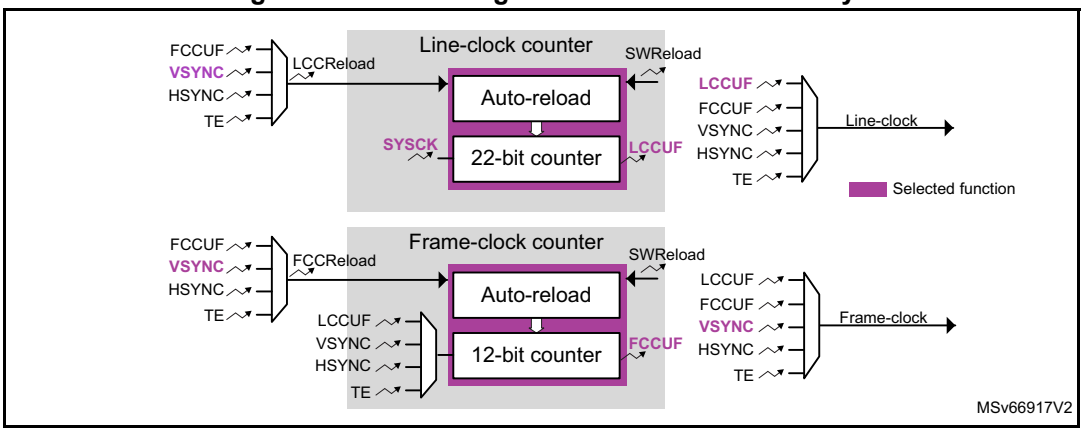
The line-clock counter generating the line-clock is reloaded on VSYNC event.

Figure 148. Active counters with external VSYNC only



The line-clock counter can also act as a prescaler for the frame-clock counter to have a wider range of counting. FCCUF acts as line-clock.

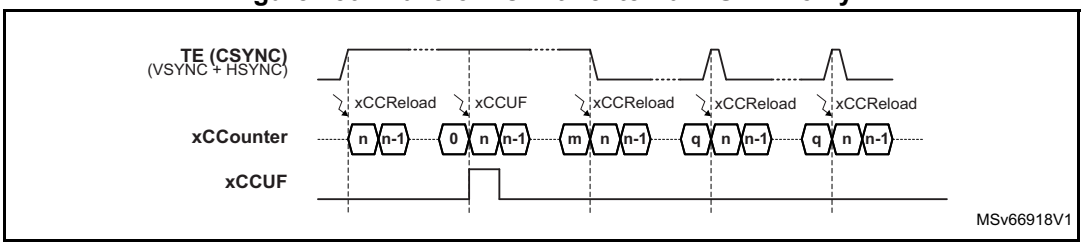
Figure 149. Prescaling when external VSYNC only



External CSYNC only

With external CSYNC only, the clock generator provides to the GFXTIM, the frame-clock and the line-clock based only on CSYNC (in the following figures, CSYNC is input on TE).

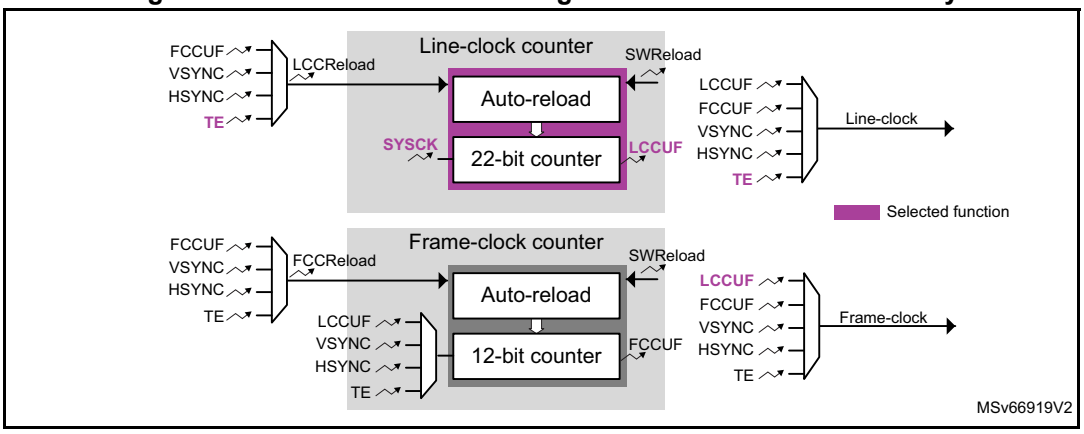
Figure 150. Waveforms with external CSYNC only



The CSYNC (on TE pin in this example) acts as line-clock and LCCUF acts as frame-clock.

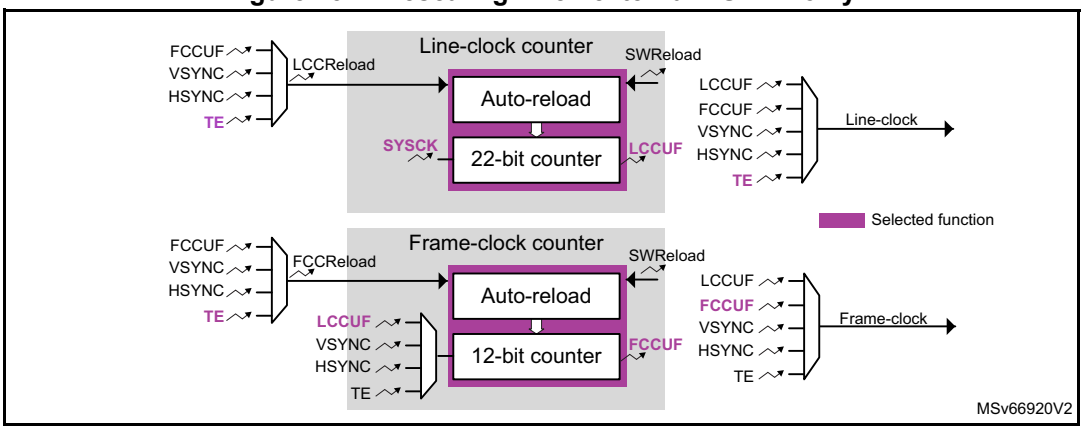
The line-clock counter generating the frame-clock is reloaded on CSYNC event.

Figure 151. Active counters and signals with external CSYNC only



The line-clock counter can also act as a prescaler for the frame-clock counter to have a wider range of counting. FCCUF acts as frame-clock.

Figure 152. Prescaling when external CSYNC only



23.3.5 Absolute timers

Absolute time

The absolute time 32-bit value can be read directly in GFXTIM_ATR (absolute time register) to provide a global time and to generate absolute time accurate event to synchronize the graphical software.

The absolute time is generated by the combination of:

- the absolute frame counter
- the absolute line counter

Absolute frame counter

The absolute frame counter is a 20-bit free running up-counter with a software controlled reset.

It can be enabled by writing 1 in AFCEN (absolute-frame counter enable) of GFXTIM_TCR (timers configuration register).

The counter is incremented at each rising edge of the frame-clock and is reset by setting FAFCR (force absolute-frame counter reset) in GFXTIM_TCR.

In case of overflow of the absolute frame counter, AFCOF (overflow flag) is set in GFXTIM_ISR (interrupt status register), and an interrupt is generated if AFCOIE (overflow interrupt enable) is set in (interrupt enable register).

A 20-bit compare register can generate an absolute-frame counter compare event when the counter value matches the compare value. When a compare event occurs, AFCC1F (compare 1 flag) is set in GFXTIM_ISR, and an interrupt is generated if AFCC1IE (compare 1 interrupt enable) is set in GFXTIM_IER.

The absolute frame counter can be disabled by writing 1 in AFCDIS (counter disable) of GFXTIM_TDR (timers disable register).

The status (enabled/disabled) can be monitored through AFCS (counter status) in GFXTIM_TSR (timers status register).

Absolute line counter

The absolute line counter is 12-bit free running-up counter with an hardware controlled reset.

It can be enabled by writing 1 in ALCEN of GFXTIM_TCR.

The counter is incremented at each rising edge of the line-clock an is reset at each rising edge of the frame-clock or setting FALCR in GFXTIM_TCR).

In case of overflow of the absolute line counter, ALCOF is set in GFXTIM_ISR, and an interrupt is generated if ALCOIE is set in GFXTIM_IER.

Two 12-bit compare registers can generate an absolute line counter compare events when the counter value matches the compare value. When a compare event occurs, ALCC1F or ALCC2F is set in GFXTIM_ISR, and an interrupt is generated if ALCC1IE or ALCC2IE is set in GFXTIM_IER.

The absolute line counter can be disabled writing 1 in ALCDIS of GFXTIM_TDR.

The status (enabled/disabled) can be monitored through ALCS in GFXTIM_TSR.

23.3.6 Relative timers

The relative timers can generate periodic events to synchronize the graphical tasks at frame level.

The relative frame counters are 12-bit down-counting auto-reload timers that are decremented at each rising edge of the frame-clock. The counter is started by writing 1 in RFCxEN (relative frame counter x enable) of GFXTIM_TCR. The starting value is automatically loaded from GFXTIM_RFCxRR (relative frame counter x reload register).

The current relative frame counter value can be read directly in GFXTIM_RFCxR.

Once the counter reaches zero, it is automatically reloaded with the value of GFXTIM_RFCxRR. RFCxRF (reload flag) is set in GFXTIM_ISR is set, and an interrupt is generated if RFCxRIE (interrupt enable) is set in GFXTIM_IER.

Once the reload operation occurs, the timer can be disabled automatically or continue counting if RFCxCM (continuous mode) is set in GFXTIM_TCR.

The status (enabled/disabled) can be monitored through RFCxS in GFXTIM_TSR.

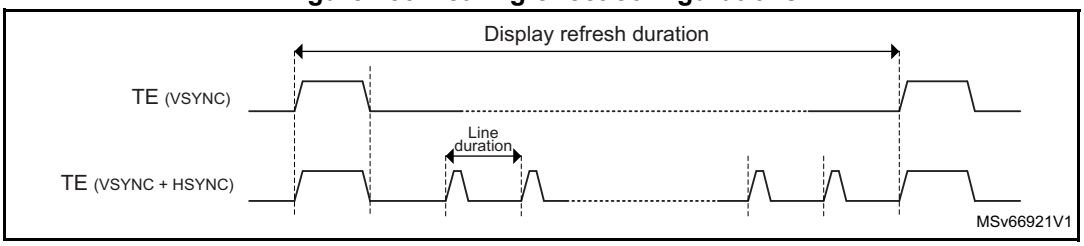
The counter can be reloaded on the fly by setting FRFCRx in GFXTIM_TCR. This force reload neither stop the timer (even if RFCxCM = 0 in GFXTIM_TCR), nor set RFCxRF (reload flag) in GFXTIM_ISR.

The relative frame counter can be disabled by setting RFCxDIS (disable) in GFXTIM_TDR.

23.3.7 Tearing-effect detection

A tearing-effect line can work in one of the two configurations shown in the figure below.

Figure 153. Tearing-effect configurations



A tearing-effect event can be generated on rising or falling edge depending on TEPOL (tearing-effect polarity) in GFXTIM_CR.

When a tearing-effect event is detected, TEF (tearing-event flag) is set in GFXTIM_ISR, and an interrupt is generated if TEIE (interrupt enable) is set in GFXTIM_IER.

23.3.8 Event generator

The event generator can combine timer events into complex events. Up to four combined events can be generated.

The events can be used for:

- interrupt generation
- watchdog clocking
- external trigger generation

A complex event is a combination between a frame event and a line event.

Once a frame event occurs, the GFXTIM waits for the line event to occur before generating the complex event.

The frame event is selected by the corresponding FESx (frame event selection x) in GFXTIM_EVSR (events selection y register). The line event is selected by the corresponding LESx in GFXTIM_EVSR.

The complex event generation is enabled by setting the corresponding EVxEN (event x enable) in GFXTIM_EVCR (event control register).

It is recommended to disable the event generation prior to any event configuration to avoid spurious complex event generation

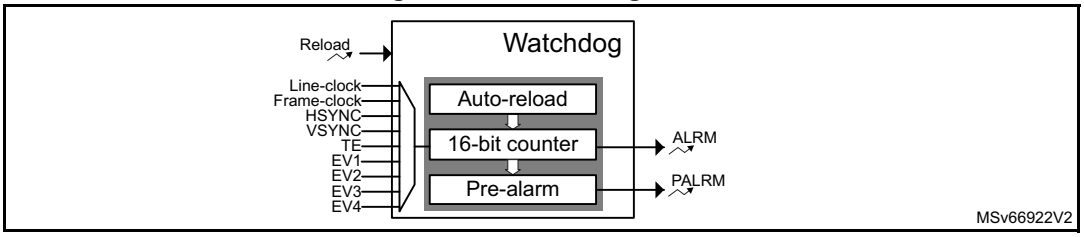
When a complex event occurs, the corresponding EVxF (event x flag) is set in GFXTIM_ISR, and an interrupt is generated if EVxIE (event x interrupt enable) is set in GFXTIM_IER.

Each of the events can be connected to another peripherals (such as DMA) to generate hardware triggers.

23.3.9 Watchdog timer

The watchdog timer is a 16-bit auto-reload down-counter with a programmable clock source.

Figure 154. Watchdog timer



Clock source

The watchdog clock source can be selected through WDGCS (clock source) in GTXTIM_WDGTCR (watchdog timer configuration register), between one of the following:

- line-clock
- frame-clock
- HSYNC
- VSYNC
- TE
- event 1
- event 2
- event 3
- event 4

Startup

The watchdog is started by setting WDGEN in GFXTIM_WDGTCR and stops automatically when reaching 0.

On start, the watchdog counter is automatically loaded with the auto-reload value programmed in GFXTIM_WDGRR (watchdog reload register).

The current watchdog value can be read through GFXTIM_WDGCR.

Auto-reload

The auto-reload can be forced in one of the following ways:

- by software, setting FWDGR (force watchdog reload) in GTXTIM_WDGTCR (watchdog timer configuration register)
- by hardware through an external trigger (gfxtim_wrld signal)

  The polarity of the trigger is configured by WDGHRC (watchdog hardware reload configuration) in GFXTIM_WDGTCR.

Event and interrupt generation

The watchdog can generate two events:

- an alarm when the watchdog down-counter reaches 0

  The watchdog counter is automatically stopped, and WDGAF (alarm flag) is set in GFXTIM_ISR. An interrupt is generated if WDGAIE (alarm interrupt enable) is set in GFXTIM_IER.
- a pre-alarm when the watchdog counter matches the pre-alarm register value in GFXTIM_WDGPAR (watchdog pre-alarm register)

  WDGPF (pre-alarm flag) is set in GFXTIM_ISR, and an interrupt is generated if WDGPIE (pre-alarm interrupt enable) is set in GFXTIM_IER.

23.4 GFXTIM interrupts

An interrupt can be produced on the following events:

- absolute frame counter compare event
- absolute frame counter overflow
- absolute line counter compare events
- absolute line counter overflow
- relative frame counter reload events
- external tearing effect
- combined events
- watchdog alarm event
- watchdog pre-alarm event

Separate interrupt enable bits are available for flexibility.

Table 132. Graphic timer interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method
GFXTIM	Absolute frame counter overflow	AFCOF	AFCOIE	write 1 in CAFCOF
	Absolute frame counter compare 1	AFCC1F	AFCC1IE	write 1 in CAFCC1F
	Absolute line counter overflow	ALCOF	ALCOIE	write 1 in CALCOF
	Absolute line counter compare 1	ALCC1F	ALCC1IE	write 1 in CALCC1F
	Absolute line counter compare 2	ALCC2F	ALCC2IE	write 1 in CALCC2F
	Relative frame counter 1 reload	RFC1RF	RFC1RIE	write 1 in CRFC1RF
	Relative frame counter 2 reload	RFC2RF	RFC2RIE	write 1 in CRFC2RF
	External tearing effect	TEF	TEIE	write 1 in CTEF
	Event 1	EV1F	EV1IE	write 1 in CEV1F
	Event 2	EV2F	EV2IE	write 1 in CEV2F
	Event 3	EV3F	EV3IE	write 1 in CEV3F
	Event 4	EV4F	EV4IE	write 1 in CEV4F

Table 132. Graphic timer interrupt requests (continued)

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method
GFXTIM_WDG	Watchdog alarm	WDGAF	WDGAIE	write 1 in CWDGAF
	Watchdog pre-alarm	WDGPF	WDGPIE	write 1 in CWDGPF

23.5 GFXTIM registers

23.5.1 GFXTIM configuration register (GFXTIM_CR)

Address offset: 0x000

Reset value: 0x0000 0000

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

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

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

Bit 17 **LCCOE**: Line-clock calibration output enable
This bit enables the line-clock output.
0: Line-clock output disabled
1: Line-clock output enabled

Bit 16 **FCCOE**: Frame-clock calibration output enable
This bit enables the frame-clock output.
0: Frame-clock output disabled
1: Frame-clock output enabled

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **SYNCS[1:0]**: Synchronization source
This bitfield selects the synchronization signals (HSYNC and VSYNC) sources.
00: gfxtim_hsync[0] and gfxtim_vsync[0] selected
01: gfxtim_hsync[1] and gfxtim_vsync[1] selected
10: gfxtim_hsync[2] and gfxtim_vsync[2] selected
11: gfxtim_hsync[3] and gfxtim_vsync[3] selected

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **TEPOL**: Tearing--effect polarity
This bit selects the tearing-effect polarity.
0: Tearing effect active on rising edge
1: Tearing effect active on falling edge

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **TES[1:0]**: Tearing source
This bitfield selects the tearing-effect source.
00: TE input pad selected
01: gfxtim_ite selected
10: HSYNC input selected by SYNCS[1:0]
11: VSYNC input selected by SYNCS[1:0]

23.5.2 GFXTIM clock generator configuration register (GFXTIM_CGCR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	FCCHRS[2:0]			Res.	Res.	Res.	FCCFR	Res.	FCCCS[2:0]			Res.	FCS[2:0]		
	rw	rw	rw				w		rw	rw	rw		rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LCCHRS[2:0]			Res.	Res.	Res.	LCCFR	Res.	Res.	Res.	LCCCS	Res.	LCS[2:0]		
	rw	rw	rw				w				rw		rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **FCCHRS[2:0]**: Frame-clock counter hardware reload source
This bitfield configures the hardware reload source for the frame-clock counter.
000: No hardware reload
001: Line- -clock counter underflow
010: HSYNC rising edge
011: HSYNC falling edge
100: VSYNC rising edge
101: VSYNC falling edge
110: TE rising edge
111: TE falling edge

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **FCCFR**: Frame-clock counter force reload
This bit forces frame-clock counter reload
0: No effect
1: Frame-clock counter reload forced

Bit 23 Reserved, must be kept at reset value.

Bits 22:20 **FCCCS[2:0]**: Frame-clock counter clock source
This bitfield configures the clock source for the frame-clock counter.
000: Frame-clock counter disabled
001: Line-clock counter underflow
010: HSYNC rising edge
011: HSYNC falling edge
100: VSYNC rising edge
101: VSYNC falling edge
110: TE rising edge
111: TE falling edge

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **FCS[2:0]**: Frame-clock source
This bitfield configures the frame-clock source.
000: Line-clock counter underflow
001: Frame-clock counter underflow
010: HSYNC rising edge
011: HSYNC falling edge
100: VSYNC rising edge
101: VSYNC falling edge
110: TE rising edge
111: TE falling edge

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **LCCHRS[2:0]**: Line-clock counter hardware reload source
This bitfield configures the hardware reload source for the line-clock counter.
000: No hardware reload
001: Frame-clock counter underflow
010: HSYNC rising edge
011: HSYNC falling edge
100: VSYNC rising edge
101: VSYNC falling edge
110: TE rising edge
111: TE falling edge

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **LCCFR**: Line-clock counter force reload
This bit forces line-clock counter reload.
0: No effect
1: Line-clock counter reload forced

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **LCCCS**: Line-clock counter clock source
This bit configures the clock source for the line-clock counter.
0: Line-clock counter disabled
1: System clock selected

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **LCS[2:0]**: Line-clock source
This bitfield configures the line-clock source.
000: Line-clock counter underflow
001: Frame-clock counter underflow
010: HSYNC rising edge
011: HSYNC falling edge
100: VSYNC rising edge
101: VSYNC falling edge
110: TE rising edge
111: TE falling edge

23.5.3 GFXTIM timers configuration register (GFXTIM_TCR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRFC2R	RFC2CM	RFC2EN	Res.	FRFC1R	RFC1CM	RFC1EN
									w	rw	w		w	rw	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FALCR	ALCEN	Res.	Res.	FAFCR	AFCEN
										w	w			w	w

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **FRFC2R**: Force relative frame counter 2 reload
This bit forces the reload of the relative frame counter 2.
0: No effect
1: Relative frame counter 2 reload forced

Bit 21 **RFC2CM**: Relative frame counter 2 continuous mode
This bit enables the continuous mode of the relative frame counter 2.
0: Relative frame counter 2 is one shot.
1: Relative frame counter 2 is in continuous mode.

Bit 20 **RFC2EN**: Relative frame counter 2 enable
This bit enables the relative frame counter 2.
0: No effect
1: Relative frame counter 2 enabled

Bit 19 Reserved, must be kept at reset value.

Bit 18 **FRFC1R**: Force relative frame counter 1 reload
This bit forces the reload of the relative frame counter 1.
0: No effect
1: Relative frame counter 1 reload forced

Bit 17 **RFC1CM**: Relative frame counter 1 continuous mode
This bit enables the continuous mode of the relative frame counter 1.
0: Relative frame counter 1 is one shot.
1: Relative frame counter 1 is in continuous mode.

Bit 16 **RFC1EN**: Relative frame counter 1 enable
This bit enables the relative frame counter 1.
0: No effect
1: Relative frame counter enabled

Bits 15:6 Reserved, must be kept at reset value.

Bit 5 **FALCR**: Force absolute line counter reset
This bit forces the reset of the absolute line counter.
0: No effect
1: Absolute line counter reset forced

Bit 4 **ALCEN**: Absolute line counter enable
This bit enables the absolute line counter.
0: No effect
1: Absolute line counter enabled

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **FAFCR**: Force absolute frame counter reset
This bit forces the reset of the absolute frame counter.
0: No effect
1: Absolute frame counter reset forced

Bit 0 **AFCEN**: Absolute frame counter enable
This bit enables the absolute frame counter.
0: No effect
1: Absolute frame counter enabled

23.5.4 GFXTIM timers disable register (GFXTIM_TDR)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFC2DIS	Res.	Res.	Res.	RFC1DIS
											w				w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALCDIS	Res.	Res.	Res.	AFCDIS
											w				w

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **RFC2DIS**: Relative frame counter 2 disable
This bit disables the relative frame counter 2.
0: No effect
1: Relative frame counter 2 disabled

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **RFC1DIS**: Relative frame counter 1 disable
This bit disables the relative frame counter 1.
0: No effect
1: Relative frame counter 1 disabled

Bits 15:5 Reserved, must be kept at reset value.

Bit 4 **ALCDIS**: Absolute line counter disable
This bit disables the absolute line counter.
0: No effect
1: Absolute line counter disabled

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **AFCDIS**: Absolute frame counter disable
This bit disables the absolute frame counter.
0: No effect
1: Absolute frame counter disabled

23.5.5 GFXTIM events control register (GFXTIM_EVCR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EV4EN	EV3EN	EV2EN	EV1EN
												rw	rw	rw	rw

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

Bit 3 **EV4EN**: Event 4 enable
This bit enables the complex event 4 generation.
0: Event 4 generation disabled
1: Event 4 generation enabled

Bit 2 **EV3EN**: Event 3 enable
This bit enables the complex event 3 generation.
0: Event 3 generation disabled
1: Event 3 generation enabled

Bit 1 **EV2EN**: Event 2 enable
This bit enables the complex event 2 generation.
0: Event 2 generation disabled
1: Event 2 generation enabled

Bit 0 **EV1EN**: Event 1 enable
This bit enables the complex event 1 generation.
0: Event 1 generation disabled
1: Event 1 generation enabled

23.5.6 GFXTIM events selection register (GFXTIM_EVSR)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	FES4[2:0]			Res.	LES4[2:0]			Res.	FES3[2:0]			Res.	LES3[2:0]		
	rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	FES2[2:0]			Res.	LES2[2:0]			Res.	FES1[2:0]			Res.	LES1[2:0]		
	rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **FES4[2:0]**: Frame-event selection 4
This bitfield defines the frame-event selection for complex event 4 generation.
000: No frame event
001: Absolute frame counter overflow
010: Absolute frame counter compare
100: Relative frame counter 1 reload
101: Relative frame counter 2 reload
Others: Reserved

Bit 27 Reserved, must be kept at reset value.

Bits 26:24 **LES4[2:0]**: Line-event selection 4
This bitfield defines the line-event selection for complex event 4 generation.
000: No line event
001: Absolute line counter overflow
010: Tearing effect
100: Absolute line counter 1 compare
101: Absolute line counter 2 compare
Others: Reserved

Bit 23 Reserved, must be kept at reset value.

Bits 22:20 **FES3[2:0]**: Frame-event selection 3
This bitfield defines the frame-event selection for complex event 3 generation.
000: No frame event
001: Absolute frame counter overflow
010: Absolute frame counter compare
100: Relative frame counter 1 reload
101: Relative frame counter 2 reload
Others: Reserved

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **LES3[2:0]**: Line-event selection 3
This bitfield defines the line-event selection for complex event 3 generation.
000: No line event
001: Absolute line counter overflow
010: Tearing effect
100: Absolute line counter 1 compare
101: Absolute line counter 2 compare
Others: Reserved

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **FES2[2:0]**: Frame-event selection 2
This bitfield defines the frame-event selection for complex event 2 generation.
000: No frame event
001: Absolute frame counter overflow
010: Absolute frame counter compare
100: Relative frame counter 1 reload
101: Relative frame counter 2 reload
Others: Reserved

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **LES2[2:0]**: Line-event selection 2
This bitfield defines the line-event selection for complex event 2 generation.
000: No line event
001: Absolute line counter overflow
010: Tearing effect
100: Absolute line counter 1 compare
101: Absolute line counter 2 compare
Others: Reserved

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **FES1[2:0]**: Frame-event selection 1
This bitfield defines the frame-event selection for complex event 1 generation.
000: No frame event
001: Absolute frame counter overflow
010: Absolute frame counter compare
100: Relative frame counter 1 reload
101: Relative frame counter 2 reload
Others: Reserved

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **LES1[2:0]**: Line-event selection 1
This bitfield defines the line-event selection for complex event 1 generation.
000: No line event
001: Absolute line counter overflow
010: Tearing effect
100: Absolute line counter 1 compare
101: Absolute line counter 2 compare
Others: Reserved

23.5.7 GFXTIM watchdog timer configuration register (GFXTIM_WDGTCR)

Address offset: 0x020

Reset value: 0x0000 0000

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

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **FWDGR**: Force watchdog reload
This bit forces the reload of the graphic watchdog.
0: nNo effect
1: Graphic watchdog reload forced

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

Bits 11:8 **WDGCS[3:0]**: Watchdog clock source
This bitfield selects the watchdog clock source.
0000: Line-clock
0001: Frame-clock
0010: HSYNC rising edge
0011: HSYNC falling edge
0100: VSYNC rising edge
0101: VSYNC falling edge
0110: TE rising edge
0111: TE falling edge
1000: Event 1
1001: Event 2
1010: Event 3
1011: Event 4
Others: Reserved

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **WDGHRC[1:0]**: Watchdog hardware reload configuration
This bitfield configures the watchdog hardware reload.
00: Watchdog hardware reload disabled
01: Watchdog reloaded a rising edge of gfxtim_wrld
10: Watchdog reloaded a falling edge of gfxtim_wrld
11: Reserved

Bit 3 Reserved, must be kept at reset value.

Bit 2 **WDGS**: Watchdog status
This bit returns the status of the graphic watchdog.
0: Graphic watchdog disabled
1: Graphic watchdog enabled

Bit 1 **WDGDIS**: Watchdog disable
This bit disables the graphic watchdog.
0: No effect
1: Graphic watchdog disabled

Bit 0 **WDGEN**: Watchdog enable
This bit enables the graphic watchdog.
0: No effect
1: Graphic watchdog enabled

23.5.8 GFXTIM interrupt status register (GFXTIM_ISR)

Address offset: 0x030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	WDGPF	WDGAF	Res.	Res.	Res.	Res.	EV4F	EV3F	EV2F	EV1F
						r	r					r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	RFC2RF	RFC1RF	Res.	Res.	ALCC2F	ALCC1F	Res.	Res.	Res.	AFCC1F	Res.	TEF	ALCOF	AFCOF
		r	r			r	r				r		r	r	r

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

Bit 25 **WDGPF**: Watchdog pre-alarm flag
This bit indicates that a graphic watchdog pre-alarm occurred.
0: No graphic watchdog pre-alarm occurred.
1: A graphic watchdog pre-alarm occurred.

Bit 24 **WDGAF**: Watchdog alarm flag
This bit indicates that a graphic watchdog alarm occurred.
0: No graphic watchdog alarm occurred.
1: A graphic watchdog alarm occurred.

Bits 23:20 Reserved, must be kept at reset value.

Bit 19 **EV4F**: Event 4 flag
This bit indicates a complex event 4 occurred.
0: No complex event 4 occurred.
1: A complex event 4 occurred.

Bit 18 **EV3F**: Event 3 flag
This bit indicates a complex event 3 occurred.
0: No complex event 3 occurred.
1: A complex event 3 occurred.

Bit 17 **EV2F**: Event 2 flag
This bit indicates a complex event 2 occurred.
0: No complex event 2 occurred.
1: A complex event 2 occurred.

Bit 16 **EV1F**: Event 1 flag
This bit indicates a complex event 1 occurred.
0: No complex event 1 occurred.
1: Complex event 1 occurred.

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

Bit 13 **RFC2RF**: Relative frame counter 2 reload flag
This bit indicates relative frame counter 2 has been reloaded.
0: No reload occurred on relative frame counter 2.
1: A reload on relative frame counter 2 occurred.

Bit 12 **RFC1RF**: Relative frame counter 1 reload flag
This bit indicates relative frame counter 1 has been reloaded.
0: No reload occurred on relative frame counter 1.
1: A reload on relative frame counter 1 occurred.

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **ALCC2F**: Absolute line counter compare 2 flag
This bit indicates match on compare 2 of the absolute line counter.
0: No match occurred on compare 2 of the absolute line counter.
1: A match on compare 2 of the absolute line counter occurred.

Bit 8 **ALCC1F**: Absolute line counter compare 1 flag
This bit indicates match on compare 1 of the absolute line counter.
0: No match occurred on compare 1 of the absolute line counter.
1: A match on compare 1 of the absolute line counter occurred.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **AFCC1F**: Absolute frame counter compare 1 flag
This bit indicates match on compare 1 of the absolute frame counter.
0: No match occurred on compare 1 of the absolute frame counter.
1: A match on compare 1 of the absolute frame counter occurred.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **TEF**: Tearing-effect flag
This bit indicates a tearing effect event occurred.
0: No tearing effect occurred.
1: A tearing effect occurred.

Bit 1 **ALCOF**: Absolute line counter overflow flag
This bit indicates an overflow occurred on the absolute line counter.
0: No overflow occurred on the absolute line counter.
1: A overflow on the absolute line counter occurred.

Bit 0 **AFCOF**: absolute frame counter overflow flag
This bit indicates an overflow occurred on the absolute frame counter.
0: No overflow occurred on the absolute frame counter.
1: An overflow on the absolute frame counter occurred.

23.5.9 GFXTIM interrupt clear register (GFXTIM_ICR)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	CWDGPF	CWDGAF	Res.	Res.	Res.	Res.	CEV4F	CEV3F	CEV2F	CEV1F
						w	w					w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CRFC2RF	CRFC1RF	Res.	Res.	CALCC2F	CALCC1F	Res.	Res.	Res.	CAFCC1F	Res.	CTEF	CALCOF	CAFCOF
		w	w			w	w				w		w	w	w

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

Bit 25 **CWDGPF**: Clear watchdog pre-alarm flag
This bit clears WDGPF in GXFXTIM_ISR.
0: No effect
1: WDGPF cleared

Bit 24 **CWDGAF**: Clear watchdog alarm flag
This bit clears WDGAF in GXFXTIM_ISR.
0: No effect
1: WDGAF cleared

Bits 23:20 Reserved, must be kept at reset value.

Bit 19 **CEV4F**: Clear event 4 flag
This bit clears EV4F in GXFXTIM_ISR.
0: No effect
1: EV4F cleared

Bit 18 **CEV3F**: Clear event 3 flag
This bit clears EV3F in GXFXTIM_ISR.
0: No effect
1: EV3F cleared

Bit 17 **CEV2F**: Clear event 2 flag
This bit clears EV2F in GXFXTIM_ISR.
0: No effect
1: EV2F cleared

Bit 16 **CEV1F**: Clear event 1 flag
This bit EV1F in GXFXTIM_ISR.
0: No effect
1: EV1F cleared

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

Bit 13 **CRFC2RF**: Clear relative frame counter 2 reload flag
This bit clears RFC2RF in GXFXTIM_ISR.
0: No effect
1: RFC2RF cleared

Bit 12 **CRFC1RF**: Clear relative frame counter 1 reload flag
This bit clears RFC1RF in GXTIM_ISR.
0: No effect
1: RFC1RF cleared

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **CALCC2F**: Clear absolute line counter compare 2 flag
This bit clears ALCC2F in GXTIM_ISR.
0: No effect
1: ALCC2F cleared

Bit 8 **CALCC1F**: Clear absolute line counter compare 1 flag
This bit clears ALCC1F in GXTIM_ISR.
0: No effect
1: ALCC1F cleared

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **CAFCC1F**: Clear absolute frame counter compare 1 flag
This bit clears AFCC1F in GXTIM_ISR.
0: No effect
1: AFCC1F cleared

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CTEF**: Clear tearing-effect flag
This bit clears TEF in GXTIM_ISR.
0: No effect
1: TEF cleared

Bit 1 **CALCOF**: Clear absolute line counter overflow flag
This bit clears ALCOF in GXTIM_ISR.
0: No effect
1: ALCOF cleared

Bit 0 **CAFCOF**: Clear absolute frame counter overflow flag
This bit clears AFCOF in GXTIM_ISR.
0: No effect
1: AFCOF cleared

23.5.10 GFXTIM interrupt enable register (GFXTIM_IER)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	WDGPIE	WDGAIE	Res.	Res.	Res.	Res.	EV4IE	EV3IE	EV2IE	EV1IE
						rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	RFC2RIE	RFC1RIE	Res.	Res.	ALCC2IE	ALCC1IE	Res.	Res.	Res.	AFCC1IE	Res.	TEIE	ALCOIE	AFCOIE
		rw	rw			rw	rw				rw		rw	rw	rw

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

Bit 25 **WDGPIE**: Watchdog pre-alarm interrupt enable
This bit enables the watchdog pre-alarm interrupt generation.
0: Watchdog pre-alarm interrupt disabled
1: Watchdog pre-alarm interrupt enabled

Bit 24 **WDGAIE**: Watchdog alarm interrupt enable
This bit enables the watchdog alarm interrupt generation.
0: Watchdog alarm interrupt disabled
1: Watchdog alarm interrupt enabled

Bits 23:20 Reserved, must be kept at reset value.

Bit 19 **EV4IE**: Event 4 interrupt enable
This bit enables the complex event 4 interrupt generation.
0: Event 4 interrupt disabled
1: Event 4 interrupt enabled

Bit 18 **EV3IE**: Event 3 interrupt enable
This bit enables the complex event 3 interrupt generation.
0: Event 3 interrupt disabled
1: Event 3 interrupt enabled

Bit 17 **EV2IE**: Event 2 interrupt enable
This bit enables the complex event 2 interrupt generation.
0: Event 2 interrupt disabled
1: Event 2 interrupt enabled

Bit 16 **EV1IE**: Event 1 interrupt enable
This bit enables the complex event 1 interrupt generation.
0: Event 1 interrupt disabled
1: Event 1 interrupt enabled

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

Bit 13 **RFC2RIE**: Relative frame counter 2 reload interrupt enable
This bit enables the relative frame counter 2 reload interrupt generation.
0: Relative frame counter 2 reload interrupt disabled
1: Relative frame counter 2 reload interrupt enabled

Bit 12 **RFC1RIE**: Relative frame counter 1 reload interrupt enable
This bit enables the relative frame counter 1 reload interrupt generation.
0: Relative frame counter 1 reload interrupt disabled
1: Relative frame counter 1 reload interrupt enabled

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **ALCC2IE**: Absolute line counter compare 2 interrupt enable
This bit enables the absolute line counter compare 2 interrupt generation.
0: Absolute line counter compare 2 interrupt disabled
1: Absolute line counter compare 2 interrupt enabled

Bit 8 **ALCC1IE**: Absolute line counter compare 1 interrupt enable
This bit enables the absolute line counter compare 1 interrupt generation.
0: Absolute line counter compare 1 interrupt disabled
1: Absolute line counter compare 1 interrupt enabled

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **AFCC1IE**: Absolute frame counter compare 1 interrupt enable
This bit enables the absolute frame counter compare interrupt generation.
0: Absolute frame counter compare 1 interrupt disabled
1: Absolute frame counter compare 1 interrupt enabled

Bit 3 Reserved, must be kept at reset value.

Bit 2 **TEIE**: Tearing-effect interrupt enable
This bit enables the Tearing Effect interrupt generation.
0: Tearing-effect interrupt disabled
1: Tearing-effect interrupt enabled

Bit 1 **ALCOIE**: Absolute line counter overflow interrupt enable
This bit enables the absolute line counter overflow interrupt generation.
0: Absolute line counter overflow interrupt disabled
1: Absolute line counter overflow interrupt enabled

Bit 0 **AFCOIE**: Absolute frame counter overflow interrupt enable
This bit enables the absolute frame counter overflow interrupt generation.
0: Absolute frame counter overflow interrupt disabled
1: Absolute frame counter overflow interrupt enabled

23.5.11 GFXTIM timers status register (GFXTIM_TSR)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFC2S	Res.	Res.	Res.	RFC1S
											r				r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALCS	Res.	Res.	Res.	AFCS
											r				r

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **RFC2S**: Relative frame counter 2 status
This bit returns the status of the relative frame counter 2.
0: Relative frame counter 2 disabled
1: Relative frame counter 2 enabled

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **RFC1S**: Relative frame counter 1 status
This bit returns the status of the relative frame counter 1.
0: Relative frame counter 1 disabled
1: Relative frame counter 1 enabled

Bits 15:5 Reserved, must be kept at reset value.

Bit 4 **ALCS**: Absolute line counter status
This bit returns the status of the absolute line counter.
0: Absolute line counter disabled
1: Absolute line counter enabled

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **AFCS**: Absolute frame counter status
This bit returns the status of the absolute frame counter.
0: Absolute frame counter disabled
1: Absolute frame counter enabled

23.5.12 GFXTIM line-clock counter reload register (GFXTIM_LCCRR)

Address offset: 0x040

Reset value: 0x0000 0000

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

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

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

Bits 21:0 **RELOAD[21:0]**: Reload value
Reload value of the line-clock counter.

23.5.13 GFXTIM frame-clock counter reload register (GFXTIM_FCCRR)

Address offset: 0x044

Reset value: 0x0000 0000

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

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

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **RELOAD[11:0]**: Reload value
Reload value of the frame-clock counter.

23.5.14 GFXTIM absolute time register (GFXTIM_ATR)

Address offset: 0x050

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FRAME[19:4]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FRAME[3:0]				LINE[11:0]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 **FRAME[19:0]**: Fame number
Current value of the absolute frame counter.

Bits 11:0 **LINE[11:0]**: Line number
Current value of the absolute line counter.

23.5.15 GFXTIM absolute frame counter register (GFXTIM_AFCR)

Address offset: 0x054

Reset value: 0x0000 0000

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **FRAME[19:0]**: Frame number
Current value of the absolute frame counter.
Note: This bitfield can only be written when the absolute frame counter is disabled.

23.5.16 GFXTIM absolute line counter register (GFXTIM_ALCR)

Address offset: 0x058

Reset value: 0x0000 0000

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

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **LINE[11:0]**: Line number
Current value of the absolute line counter.
Note: This bitfield can only be written when the absolute line counter is disabled.

23.5.17 GFXTIM absolute frame counter compare 1 register (GFXTIM_AFCC1R)

Address offset: 0x060

Reset value: 0x0000 0000

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **FRAME[19:0]**: Frame number
Compare 1 value for the absolute frame counter.

23.5.18 GFXTIM absolute line counter compare 1 register (GFXTIM_ALCC1R)

Address offset: 0x070

Reset value: 0x0000 0000

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

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **LINE[11:0]**: Line number
Compare value 1 for the absolute line counter.

23.5.19 GFXTIM absolute line counter compare 2 register (GFXTIM_ALCC2R)

Address offset: 0x074

Reset value: 0x0000 0000

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

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **LINE[11:0]**: Line number
Compare value 2 for the absolute line counter.

23.5.20 GFXTIM relative frame counter 1 register (GFXTIM_RFC1R)

Address offset: 0x080

Reset value: 0x0000 0000

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

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

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **FRAME[11:0]**: Frame number
Current value of the relative frame counter 1.

23.5.21 GFXTIM relative frame counter 1 reload register (GFXTIM_RFC1RR)

Address offset: 0x084

Reset value: 0x0000 0000

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

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

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **FRAME[11:0]**: Frame reload value
Reload value for the relative frame counter 1.

23.5.22 GFXTIM relative frame counter 2 register (GFXTIM_RFC2R)

Address offset: 0x088

Reset value: 0x0000 0000

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

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

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **FRAME[11:0]**: Frame number
Current value of the relative frame counter 2.

23.5.23 GFXTIM relative frame counter 2 reload register (GFXTIM_RFC2RR)

Address offset: 0x08C

Reset value: 0x0000 0000

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

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

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **FRAME[11:0]**: Frame reload value
Reload value for the relative frame counter 2.

23.5.24 GFXTIM watchdog counter register (GFXTIM_WDGCR)

Address offset: 0x0A0

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VALUE[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **VALUE[15:0]**: Value
Current value of the watchdog counter.

23.5.25 GFXTIM watchdog reload register (GFXTIM_WDGRR)

Address offset: 0x0A4

Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **RELOAD[15:0]**: Reload value
Reload value of the watchdog counter.

23.5.26 GFXTIM watchdog pre-alarm register (GFXTIM_WDGPAR)

Address offset: 0x0A8

Reset value: 0x0000 0000

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **PREALARM[15:0]**: Pre-alarm value
Pre-alarm value of the watchdog counter.

23.5.27 GFXTIM register map

Table 133. GFXTIM register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	GFXTIM_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCCOE	FCCOE	Res.	Res.	Res.	Res.	Res.	Res.	SYNCS [1:0]		Res.	Res.	Res.	TEPOL	Res.	Res.	TES [1:0]	
	Reset value															0	0							0	0				0			0	0
0x004	GFXTIM_CGCR	Res.	FCCHRS [2:0]			Res.	Res.	Res.	FCCFR	Res.	FCCCS [2:0]			Res.	FCS [2:0]			Res.	LCCHRS [2:0]			Res.	Res.	Res.	LCCFR	Res.	Res.	Res.	LCCCS	Res.	LCS[2:0]		
	Reset value		0	0	0				0		0	0	0		0	0	0		0	0	0				0				0		0	0	0

Table 133. GFXTIM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x008	GFXTIM_TCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRFC2R	RFC2CM	RFC2EN	Res.	FRFC1R	RFC1CM	RFC1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FALCR	ALCEN	Res.	Res.	FAFCR	AFCEN
	Reset value										0	0	0		0	0	0											0	0			0	0
0x00C	GFXTIM_TDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFC2DIS	Res.	Res.	Res.	RFC1DIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALCDIS	Res.	Res.	Res.	AFCDIS
	Reset value												0				0												0				0
0x010	GFXTIM_EVCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EV4EN	EV3EN	EV2EN	EV1EN
	Reset value																													0	0	0	0
0x014	GFXTIM_EVSR	Res.	FES4[2:0]			Res.	LES4[2:0]			Res.	FES3[2:0]			Res.	LES3[2:0]			Res.	FES2[2:0]			Res.	LES2[2:0]			Res.	FES1[2:0]			Res.	LES1[2:0]		
	Reset value		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0
0x018-0x01C	Reserved	Reserved																															
0x020	GFXTIM_WDGTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FWDGR	Res.	Res.	Res.	Res.	WDGCS[3:0]				Res.	Res.	WDGHRC[1:0]		Res.	WDGS	WDGDIS	WDGEN
	Reset value																0					0	0	0	0			0	0		0	0	0
0x024-0x02C	Reserved	Reserved																															
0x030	GFXTIM_ISR	Res.	Res.	Res.	Res.	Res.	Res.	WDGPF	WDGAF	Res.	Res.	Res.	Res.	EV4F	EV3F	EV2F	EV1F	Res.	Res.	RFC2RF	RFC1RF	Res.	Res.	ALCC2F	ALCC1F	Res.	Res.	Res.	AFCC1F	Res.	TEF	ALCOF	AFCOF
	Reset value							0	0					0	0	0	0			0	0			0	0				0		0	0	0
0x034	GFXTIM_ICR	Res.	Res.	Res.	Res.	Res.	Res.	CWDGPF	CWDGAF	Res.	Res.	Res.	Res.	CEV4F	CEV3F	CEV2F	CEV1F	Res.	Res.	CRFC2RF	CRFC1RRF	Res.	Res.	CALCC2F	CALCC1F	Res.	Res.	Res.	CAFCC1F	Res.	CTEF	CALCOF	CAFCOF
	Reset value							0	0					0	0	0	0			0	0			0	0				0		0	0	0
0x038	GFXTIM_IER	Res.	Res.	Res.	Res.	Res.	Res.	WDGPIE	WDGAIE	Res.	Res.	Res.	Res.	EV4IE	EV3IE	EV2IE	EV1IE	Res.	Res.	RFC2RIE	RFC1RIE	Res.	Res.	ALCC2IE	ALCC1IE	Res.	Res.	Res.	AFCC1IE	Res.	TEIE	ALCOIE	AFCOIE
	Reset value							0	0					0	0	0	0			0	0			0	0				0		0	0	0
0x03C	GFXTIM_TSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFC2S	Res.	Res.	Res.	RFC1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALCS	Res.	Res.	Res.	AFCS
	Reset value												0				0												0				0
0x040	GFXTIM_LCCRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RELOAD[21:0]																					
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	GFXTIM_FCCRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RELOAD[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x048-0x04C	Reserved	Reserved																															
0x050	GFXTIM_ATR	FRAME[19:0]																				LINE[11:0]											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054	GFXTIM_AFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRAME[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x058	GFXTIM_ALCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LINE[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x05C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Table 133. GFXTIM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x060	GFXTIM_ AFCC1R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRAME[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x064-0x06C	Reserved	Reserved																															
0x070	GFXTIM_ ALCC1R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LINE[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x074	GFXTIM_ ALCC2R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LINE[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x078-0x07C	Reserved	Reserved																															
0x080	GFXTIM_RFC1R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRAME[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x084	GFXTIM_ RFC1RR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRAME[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x088	GFXTIM_RFC2R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRAME[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x08C	GFXTIM_ RFC2RR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FRAME[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x090-0x09C	Reserved	Reserved																															
0x0A0	GFXTIM_ WDGCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VALUE[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A4	GFXTIM_ WDGRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RELOAD[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A8	GFXTIM_ WDGPAR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREALARM[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

24 Nested vectored interrupt controller (NVIC)

24.1 NVIC features

The NVIC includes the following features:

- Up to 196 maskable interrupt channels (not including the Cortex-M55 interrupt lines)
- 16 programmable priority levels (4 bits of interrupt priority are used)
- Low-latency exception and interrupt handling
- Power management control
- Implementation of system control registers

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

All interrupts, including the core exceptions, are managed by the NVIC.

24.1.1 SysTick calibration value register

The SysTick calibration value (SYST_CALIB) is fixed to 0x3E8. It provides a reference timebase of 1 ms based when the SysTick clock frequency is 1 MHz. To match the 1 ms timebase whatever the application frequency, the SysTick reload value must be programmed as follows in the SYST_RVR register:

- If the SysTick clock source is the 100 MHz CPU clock (HCLK):

$$\text{reload value} = (F_{HCLK} \times \text{SYST_CALIB}) - 1$$

- If the SysTick clock source is an external clock:

$$\text{reload value} = ((F_{HCLK} / 8) \times \text{SYST_CALIB}) - 1$$

where F_{HCLK} refers to the AHB frequency expressed in MHz

For example, to achieve a timebase of 1 ms when the SysTick clock source is the 100 MHz HCLK:

$$\text{reload value} = (100 \times \text{SYST_CALIB}) - 1 = 0\text{x}1869\text{F}$$

24.1.2 Interrupt and exception vectors

Exception vectors connected to the NVIC are the following: reset, NMI, HardFault, MemManage, BusFault, UsageFault, SVCall, DebugMonitor, PendSV, and SysTick

Table 134. STM32N6x5/x7xx vector table

NVIC position	Priority	Priority type	Acronym	Description	Address
-	-	-	-	Reserved	0x0000 0000
-	-3	Fixed	Reset	Reset	0x0000 0004

Table 134. STM32N6x5/x7xx vector table (continued)

NVIC position	Priority	Priority type	Acronym	Description	Address
-	-2	Fixed	NMI	Non-maskable interrupt. The RCC clock security system (HSE CSS) is linked to the NMI vector.	0x0000 0008
-	-1	Fixed	HardFault	All classes of fault	0x0000 000C
-	0	Fixed	MemManage	Memory management	0x0000 0010
-	1	Settable	BusFault	Prefetch fault, memory-access fault	0x0000 0014
-	2	Settable	UsageFault	Undefined instruction or illegal state	0x0000 0018
-	-	-	Reserved	Reserved	0x0000 001C 0x0000 002B
-	3	Settable	SVCall	System service call via SWI instruction	0x0000 002C
-	4	Settable	Debug Monitor	Debug monitor	0x0000 0030
-	-	-	Reserved	Reserved	0x0000 0034
-	5	Settable	PendSV	Pendable request for system service	0x0000 0038
-	6	Settable	SysTick	System tick timer	0x0000 003C
0	7	Settable	PVD_PVM	PVDOUT or PVM through the EXTI line	0x0000 0040
-	-	-	-	Reserved	0x0000 0044
2	9	Settable	DTS	Thermal sensor interruption	0x0000 0048
3	10	Settable	RCC	RCC global interrupt	0x0000 004C
4	11	Settable	LOCKUP	LOCKUP - No overstack in Cortex-M55	0x0000 0050
5	12	Settable	CACHE_ECC	Cache ECC error	0x0000 0054
6	13	Settable	TCM_ECC	TCM ECC error	0x0000 0058
7	14	Settable	BCKRAM_ECC	Backup RAM interrupts (SEC and DED)	0x0000 005C
8	15	Settable	FPU	FPU safety flag	0x0000 0060
-	16	-	Reserved (product safety)	Reserved	0x0000 0064
10	17	Settable	RTC_S	RTC secure interrupt	0x0000 0068
11	18	Settable	TAMP	TAMP secure and nonsecure synchronous interrupt line	0x0000 006C
12	19	Settable	RIFSC_TAMPER	RIF can generate an interrupt when a laser attack is detected	0x0000 0070
13	20	Settable	IAC	IAC global interrupt	0x0000 0074
14	21	Settable	RCC_S	RCC global secure interrupt	0x0000 0078
-	22	-	Reserved (product secure)	Reserved	0x0000 007C
16	23	Settable	RTC	RTC interrupt	0x0000 0080
17	24	Settable	Reserved	Reserved	0x0000 0084

Table 134. STM32N6x5/x7xx vector table (continued)

NVIC position	Priority	Priority type	Acronym	Description	Address
18	25	Settable	WDGLS	Independent watchdog interrupt	0x0000 0088
19	26	Settable	WWDG	Window watchdog interrupt	0x0000 008C
20	27	Settable	EXTI0	EXTI line 0 interrupt	0x0000 0090
21	28	Settable	EXTI1	EXTI line 1 interrupt	0x0000 0094
22	29	Settable	EXTI2	EXTI line 2 interrupt	0x0000 0098
23	30	Settable	EXTI3	EXTI line 3 interrupt	0x0000 009C
24	31	Settable	EXTI4	EXTI line 4 interrupt	0x0000 00A0
25	32	Settable	EXTI5	EXTI line 5 interrupt	0x0000 00A4
26	33	Settable	EXTI6	EXTI line 6 interrupt	0x0000 00A8
27	34	Settable	EXTI7	EXTI line 7 interrupt	0x0000 00AC
28	35	Settable	EXTI8	EXTI line 8 interrupt	0x0000 00B0
29	36	Settable	EXTI9	EXTI line 9 interrupt	0x0000 00B4
30	37	Settable	EXTI10	EXTI line 10 interrupt	0x0000 00B8
31	38	Settable	EXTI11	EXTI line 11 interrupt	0x0000 00BC
32	39	Settable	EXTI12	EXTI line 12 interrupt	0x0000 00C0
33	40	Settable	EXTI13	EXTI line 13 interrupt	0x0000 00C4
34	41	Settable	EXTI14	EXTI line 14 interrupt	0x0000 00C8
35	42	Settable	EXTI15	EXTI line 15 interrupt	0x0000 00CC
36	43	Settable	SAES	SAES global interrupt	0x0000 00D0
37	44	Settable	CRYP	CRYP global interrupt	0x0000 00D4
38	45	Settable	PKA	PKA global interrupt	0x0000 00D8
39	46	Settable	HASH	HASH global interrupt	0x0000 00DC
40	47	Settable	RNG	RNG global interrupt	0x0000 00E0
-	48	-	Reserved	Reserved	0x0000 00E4
42	49	Settable	MCE1	MCE1 global interrupt	0x0000 00E8
43	50	Settable	MCE2	MCE2 global interrupt	0x0000 00EC
44	51	Settable	MCE3	MCE3 global interrupt	0x0000 00F0
45	52	Settable	MCE4	MCE4 global interrupt	0x0000 00F4
46	53	Settable	ADC12	ADC1/ADC2 global interrupt	0x0000 00F8
47	54	Settable	CSI	CSI global interrupt	0x0000 00FC
48	55	Settable	DCMIPP	DCMIPP global interrupt	0x0000 0100
-	56	-	Reserved	Reserved	0x0000 0104

Table 134. STM32N6x5/x7xx vector table (continued)

NVIC position	Priority	Priority type	Acronym	Description	Address
-	57	-	Reserved	Reserved	0x0000 0108
-	58	-	Reserved	Reserved	0x0000 010C
52	59	Settable	PAHB_ERR	Write posting errors on Cortex-M55 PAHB interface	0x0000 0110
53	60	Settable	NPU0(1)	NPU mst_ints[0] line	0x0000 0114
54	61	Settable	NPU1(1)	NPU mst_ints[1] line	0x0000 0118
55	62	Settable	NPU2(1)	NPU mst_ints[2] line	0x0000 011C
56	63	Settable	NPU3(1)	NPU mst_ints[3] line	0x0000 0120
57	64	Settable	CACHEAXI(1)	CACHEAXI NPU interrupt cache	0x0000 0124
58	65	Settable	LTDC_LO	LCD low-layer global interrupt	0x0000 0128
59	66	Settable	LTDC_LO_ERR	LCD low-layer error interrupt	0x0000 012C
60	67	Settable	DMA2D	DMA2D global interrupt	0x0000 0130
61	68	Settable	JPEG	JPEG global interrupt	0x0000 0134
62	69	Settable	VENC	VENC global interrupt	0x0000 0138
63	70	Settable	GFXMMU	GFXMMU global interrupt	0x0000 013C
64	71	Settable	GFXTIM	GFXTIM global interrupt	0x0000 0140
65	72	Settable	GPU2D	GPU2D global interrupt	0x0000 0144
66	73	Settable	GPU2D_ERROR	GPU2D error interrupt	0x0000 0148
67	74	Settable	ICACHE	GPU2D cache interrupt	0x0000 014C
68	75	Settable	HPDMA1_CH0	HPDMA1 channel 0 interrupt	0x0000 0150
69	76	Settable	HPDMA1_CH1	HPDMA1 channel 1 interrupt	0x0000 0154
70	77	Settable	HPDMA1_CH2	HPDMA1 channel 2 interrupt	0x0000 0158
71	78	Settable	HPDMA1_CH3	HPDMA1 channel 3 interrupt	0x0000 015C
72	79	Settable	HPDMA1_CH4	HPDMA1 channel 4 interrupt	0x0000 0160
73	80	Settable	HPDMA1_CH5	HPDMA1 channel 5 interrupt	0x0000 0164
74	81	Settable	HPDMA1_CH6	HPDMA1 channel 6 interrupt	0x0000 0168
75	82	Settable	HPDMA1_CH7	HPDMA1 channel 7 interrupt	0x0000 016C
76	83	Settable	HPDMA1_CH8	HPDMA1 channel 8 interrupt	0x0000 0170
77	84	Settable	HPDMA1_CH9	HPDMA1 channel 9 interrupt	0x0000 0174
78	85	Settable	HPDMA1_CH10	HPDMA1 channel 10 interrupt	0x0000 0178
79	86	Settable	HPDMA1_CH11	HPDMA1 channel 11 interrupt	0x0000 017C
80	87	Settable	HPDMA1_CH12	HPDMA1 channel 12 interrupt	0x0000 0180

Table 134. STM32N6x5/x7xx vector table (continued)

NVIC position	Priority	Priority type	Acronym	Description	Address
81	88	Settable	HPDMA1_CH13	HPDMA1 channel 13 interrupt	0x0000 0184
82	89	Settable	HPDMA1_CH14	HPDMA1 channel 14 interrupt	0x0000 0188
83	90	Settable	HPDMA1_CH15	HPDMA1 channel 15 interrupt	0x0000 018C
84	91	Settable	GPDMA1_CH0	GPDMA1 channel 0 interrupt	0x0000 0190
85	92	Settable	GPDMA1_CH1	GPDMA1 channel 1 interrupt	0x0000 0194
86	93	Settable	GPDMA1_CH2	GPDMA1 channel 2 interrupt	0x0000 0198
87	94	Settable	GPDMA1_CH3	GPDMA1 channel 3 interrupt	0x0000 019C
88	95	Settable	GPDMA1_CH4	GPDMA1 channel 4 interrupt	0x0000 01A0
89	96	Settable	GPDMA1_CH5	GPDMA1 channel 5 interrupt	0x0000 01A4
90	97	Settable	GPDMA1_CH6	GPDMA1 channel 6 interrupt	0x0000 01A8
91	98	Settable	GPDMA1_CH7	GPDMA1 channel 7 interrupt	0x0000 01AC
92	99	Settable	GPDMA1_CH8	GPDMA1 channel 8 interrupt	0x0000 01B0
93	100	Settable	GPDMA1_CH9	GPDMA1 channel 9 interrupt	0x0000 01B4
94	101	Settable	GPDMA1_CH10	GPDMA1 channel 10 interrupt	0x0000 01B8
95	102	Settable	GPDMA1_CH11	GPDMA1 channel 11 interrupt	0x0000 01BC
96	103	Settable	GPDMA1_CH12	GPDMA1 channel 12 interrupt	0x0000 01C0
97	104	Settable	GPDMA1_CH13	GPDMA1 channel 13 interrupt	0x0000 01C4
98	105	Settable	GPDMA1_CH14	GPDMA1 channel 14 interrupt	0x0000 01C8
99	106	Settable	GPDMA1_CH15	GPDMA1 channel 15 interrupt	0x0000 01CC
100	107	Settable	I2C1_EV	I2C1 event interrupt	0x0000 01D0
101	108	Settable	I2C1_ER	I2C1 error interrupt	0x0000 01D4
102	109	Settable	I2C2_EV	I2C2 event interrupt	0x0000 01D8
103	110	Settable	I2C2_ER	I2C2 error interrupt	0x0000 01DC
104	111	Settable	I2C3_EV	I2C3 event interrupt	0x0000 01E0
105	112	Settable	I2C3_ER	I2C3 error interrupt	0x0000 01E4
106	113	Settable	I2C4_EV	I2C4 event interrupt	0x0000 01E8
107	114	Settable	I2C4_ER	I2C4 error interrupt	0x0000 01EC
108	115	Settable	I3C1_EV	I3C1 event interrupt	0x0000 01F0
109	116	Settable	I3C1_ER	I3C1 error interrupt	0x0000 01F4
110	117	Settable	I3C2_EV	I3C2 event interrupt	0x0000 01F8
111	118	Settable	I3C2_ER	I3C2 error interrupt	0x0000 01FC
112	119	Settable	TIM1_BRK	TIM1 break interrupt	0x0000 0200

Table 134. STM32N6x5/x7xx vector table (continued)

NVIC position	Priority	Priority type	Acronym	Description	Address
113	120	Settable	TIM1_UP	TIM1 update interrupt	0x0000 0204
114	121	Settable	TIM1_TRG_COM	TIM1 trigger and commutation interrupts	0x0000 0208
115	122	Settable	TIM1_CC	TIM1 capture compare interrupt	0x0000 020C
116	123	Settable	TIM2	TIM2 global interrupt	0x0000 0210
117	124	Settable	TIM3	TIM3 global interrupt	0x0000 0214
118	125	Settable	TIM4	TIM4 global interrupt	0x0000 0218
119	126	Settable	TIM5	TIM5 global interrupt	0x0000 021C
120	127	Settable	TIM6	TIM6 global interrupt	0x0000 0220
121	128	Settable	TIM7	TIM7 global interrupt	0x0000 0224
122	129	Settable	TIM8_BRK	TIM8 break interrupt	0x0000 0228
123	130	Settable	TIM8_UP	TIM18 update interrupt	0x0000 022C
124	131	Settable	TIM8_TRG_COM	TIM8 trigger and commutation interrupts	0x0000 0230
125	132	Settable	TIM8_CC	TIM8 capture compare interrupt	0x0000 0234
126	133	Settable	TIM9	TIM9 global interrupt	0x0000 0238
127	134	Settable	TIM10	TIM10 global interrupt	0x0000 023C
128	135	Settable	TIM11	TIM11 global interrupt	0x0000 0240
129	136	Settable	TIM12	TIM12 global interrupt	0x0000 0244
130	137	Settable	TIM13	TIM13 global interrupt	0x0000 0248
131	138	Settable	TIM14	TIM14 global interrupt	0x0000 024C
132	139	Settable	TIM15	TIM15 global interrupt	0x0000 0250
133	140	Settable	TIM16	TIM16 global interrupt	0x0000 0254
134	141	Settable	TIM17	TIM17 global interrupt	0x0000 0258
135	142	Settable	TIM18	TIM18 global interrupt	0x0000 025C
136	143	Settable	LPTIM1	LPTIM1 global interrupt	0x0000 0260
137	144	Settable	LPTIM2	LPTIM2 global interrupt	0x0000 0264
138	145	Settable	LPTIM3	LPTIM3 global interrupt	0x0000 0268
139	146	Settable	LPTIM4	LPTIM4 global interrupt	0x0000 026C
140	147	Settable	LPTIM5	LPTIM5 global interrupt	0x0000 0270
141	148	Settable	ADF1_FLT0	ADF1 filter 0 global interrupt	0x0000 0274
142	149	Settable	MDF1_FLT0	MDF global Interrupt for filter 0	0x0000 0278
143	150	Settable	MDF1_FLT1	MDF global Interrupt for filter 1	0x0000 027C
144	151	Settable	MDF1_FLT2	MDF global Interrupt for filter 2	0x0000 0280

Table 134. STM32N6x5/x7xx vector table (continued)

NVIC position	Priority	Priority type	Acronym	Description	Address
145	152	Settable	MDF1_FLT3	MDF global Interrupt for filter 3	0x0000 0284
146	153	Settable	MDF1_FLT4	MDF global Interrupt for filter 4	0x0000 0288
147	154	Settable	MDF1_FLT5	MDF global Interrupt for filter 5	0x0000 028C
148	155	Settable	SAI1_A	SAI1 global interrupt A	0x0000 0290
149	156	Settable	SAI1_B	SAI1 global interrupt B	0x0000 0294
150	157	Settable	SAI2_A	SAI2 global interrupt A	0x0000 0298
151	158	Settable	SAI2_B	SAI2 global interrupt B	0x0000 029C
152	159	Settable	SPDIFRX1	SPDIFRX global interrupt	0x0000 02A0
153	160	Settable	SPI1	SPI1 global interrupt	0x0000 02A4
154	161	Settable	SPI2	SPI2 global interrupt	0x0000 02A8
155	162	Settable	SPI3	SPI3 global interrupt	0x0000 02AC
156	163	Settable	SPI4	SPI4 global interrupt	0x0000 02B0
157	164	Settable	SPI5	SPI5 global interrupt	0x0000 02B4
158	165	Settable	SPI6	SPI6 global interrupt	0x0000 02B8
159	166	Settable	USART1	USART1 global interrupt	0x0000 02BC
160	167	Settable	USART2	USART2 global interrupt	0x0000 02C0
161	168	Settable	USART3	USART3 global interrupt	0x0000 02C4
162	169	Settable	UART4	UART4 global interrupt	0x0000 02C8
163	170	Settable	UART5	UART5 global interrupt	0x0000 02CC
164	171	Settable	USART6	UART5 global interrupt	0x0000 02D0
165	172	Settable	UART7	UART5 global interrupt	0x0000 02D4
166	173	Settable	UART8	UART5 global interrupt	0x0000 02D8
167	174	Settable	UART9	UART9 global interrupt	0x0000 02DC
168	175	Settable	USART10	USART10 global interrupt	0x0000 02E0
169	176	Settable	LPUART1	LPUART1 global interrupt	0x0000 02E4
170	177	Settable	XSPI1	XSPI1 global interrupt	0x0000 02E8
171	178	Settable	XSPI2	XSPI2 global interrupt	0x0000 02EC
172	179	Settable	XSPI3	XSPI3 global interrupt	0x0000 02F0
173	180	Settable	FMC	FMC global interrupt	0x0000 02F4
174	181	Settable	SDMMC1	SDMMC1 global interrupt	0x0000 02F8
175	182	Settable	SDMMC2	SDMMC2 global interrupt	0x0000 02FC
176	183	Settable	UCPD1	UCPD global interrupt	0x0000 0300

Table 134. STM32N6x5/x7xx vector table (continued)

NVIC position	Priority	Priority type	Acronym	Description	Address
177	184	Settable	OTG1	USB OTG1 HS global interrupt	0x0000 0304
178	185	Settable	OTG2	USB OTG2 HS global interrupt	0x0000 0308
179	186	Settable	ETH1	Ethernet global interrupt	0x0000 030C
180	187	Settable	FDCAN1_IT0	FDCAN1 interrupt 0	0x0000 0310
181	188	Settable	FDCAN1_IT1	FDCAN1 interrupt 1	0x0000 0314
182	189	Settable	FDCAN2_IT0	FDCAN2 interrupt 0	0x0000 0318
183	190	Settable	FDCAN2_IT1	FDCAN2 interrupt 1	0x0000 031C
184	191	Settable	FDCAN3_IT0	FDCAN3 interrupt 0	0x0000 0320
185	192	Settable	FDCAN3_IT1	FDCAN3 interrupt 1	0x0000 0324
186	193	Settable	FDCAN_CU	Clock calibration unit interrupt line (FDCAN1 only)	0x0000 0328
187	194	Settable	MDIOS	MDIOS global interrupt	0x0000 032C
188	195	Settable	DCMI_PSSI	DCMI/PSSI global interrupt	0x0000 0330
189	196	Settable	WAKEUP_PIN	Wake-up pin interrupts	0x0000 0334
190	197	Settable	CTI0	Debug monitor (Cortex-M55 related)	0x0000 0338
191	198	Settable	CTI1	Debug monitor (Cortex-M55 related)	0x0000 033C
-	199	-	Reserved	Reserved	0x0000 0340
193	200	Settable	LTDC_UP	LCD up-layer global interrupt	0x0000 0344
194	201	Settable	LTDC_UP_ERR	LCD up-layer error interrupt	0x0000 0348

1. Reserved on STM32N6x5xx devices.

25 Extended interrupts and event controller (EXTI)

The extended interrupts and event controller (EXTI) manages the CPU and system wake-up through configurable and direct event inputs. It provides wake-up requests to the power control, and generates a combined secure/nonsecure interrupt request to the CPU NVIC, and events to the CPU event input. An additional event generation block (EVG) is needed to generate the CPU event signal.

The EXTI wake-up requests allow the system to be woken up from Stop modes, and the CPU to be woken up from CStop and CStandby modes.

The interrupt request and event request generation can be used also in Run modes.

25.1 EXTI main features

- EXTI 73 input events supported
- Most event inputs allow to wake up the system
- Some events can only be used to generate a CPU wake-up
- Events without an associated wake-up flag in the peripheral have a flag in the EXTI, and generate a combined secure/nonsecure interrupt to the CPUs from the EXTI
- Events can be used to generate a CPU wake-up event

The asynchronous event inputs are classified in two groups:

- configurable events (signals from I/Os or peripherals able to generate a pulse), with the following features:
  - Selectable active trigger edge
  - Interrupt pending status register bits independent for the rising and falling edge
  - Individual interrupt and event generation mask, used to condition the CPU wake-up, interrupt, and event generation
  - Individual interrupt lines per CPU
  - Software trigger possibility
- direct events (interrupt and wake-up sources from peripherals having an associated flag, which requires to be cleared in the peripheral), with the following features:
  - Fixed rising edge active trigger
  - No interrupt pending status register bit in the EXTI (the interrupt pending status flag is provided by the peripheral generating the event)
  - Individual interrupt and event generation mask, used to condition the CPU wake-up and event generation
  - No software trigger possibility
- Secure events
  - The access to control and configuration bits of secure input events can be made secure and or privileged
- EXTI I/O port selection

25.2 EXTI implementation

Table 135. EXTI implementation

EXTI features(1)	EXTI
EXTI I/O port selection	X
EXTI I/O port to interconnect other peripherals	-
Number of events	73
Security	X

1. X = supported.

25.3 EXTI block diagram

The EXTI consists of the following blocks (see *Figure 155*):

- register block accessed via an AHB interface, which contains all the EXTI registers
- event input trigger block, which provides event input edge trigger logic
- masking block, which provides the event input distribution to the different wake-up, interrupts, and event outputs, plus their masking
- EXTI mux, which provides the I/O port selection on to the EXTI event signal

Figure 155. EXTI block diagram

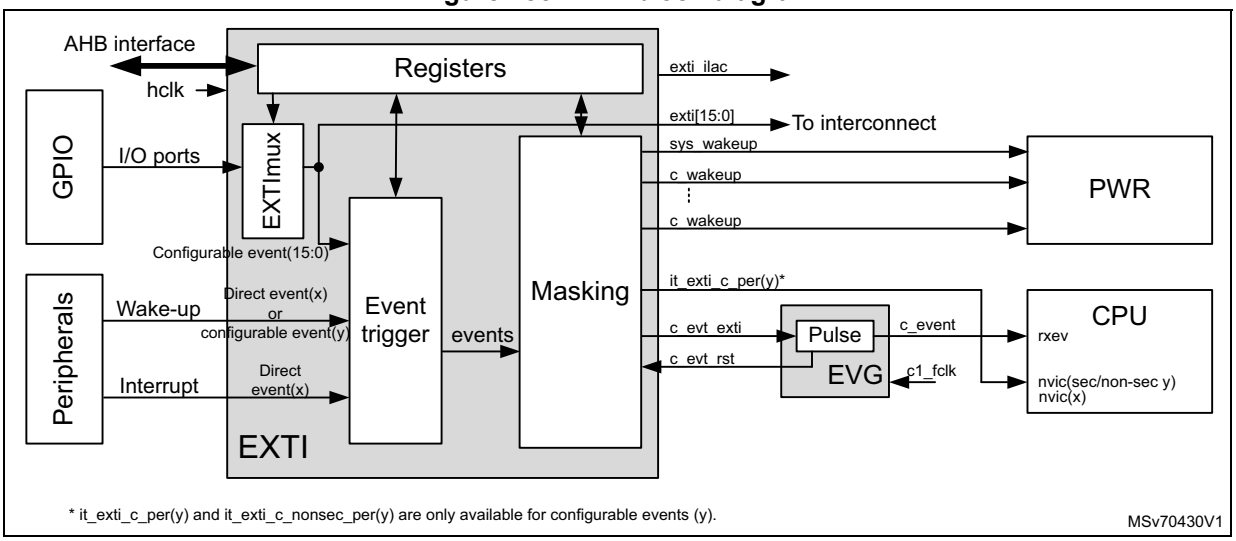


Table 136. EXTI signals

Name	I/O	Description
AHB interface	I/O	EXTI register bus interface. When one event is configured to allow security, the AHB interface support secure accesses.
hclk	I	AHB bus clock and EXTI system clock.
exti_ilac	O	Illegal access event

Table 136. EXTI signals (continued)

Name	I/O	Description
Configurable event(y)	I	Asynchronous wake-up events from peripherals without an associated interrupt and flag in the peripheral
Direct event(x)	I	Synchronous and asynchronous wake-up events from peripherals with an associated interrupt and flag in the peripheral
I/O ports	I	GPIO I/O ports[15:0]
exti[15:0]	O	EXTI GPIO output port to trigger other peripherals
it_exti_c_per(y)	O	Combined secure/nonsecure interrupts to the CPU with configurable event (y)
c_evt_exti	O	High-level sensitive event output for the CPU
c_evt_rst	I	Asynchronous reset input to clear c_evt_exti
sys_wakeup	O	Asynchronous system wake-up request to PWR for ck_sys and hclk
c_wakeup	O	Wake-up request to PWR for the CPU, synchronous to hclk

Table 137. EVG signals

Name	I/O	Description
c_fclk	I	CPU free running clock
c_evt_in	I	High-level sensitive events input from EXTI, asynchronous to CPU clock
c_event	O	Event pulse, synchronous to CPU clock
c_evt_rst	O	Event reset signal, synchronous to CPU clock

25.3.1 EXTI connections between peripherals and CPU

Peripherals able to generate wake-up or interrupt events when the system is in Stop mode or a CPU is in CStop mode, are connected to the EXTI.

Peripheral wake-up signals that generate a pulse, or which do not have interrupt status bits in the peripheral, are connected to an EXTI configurable event input. For these events, the EXTI provides a status pending bit, which requires to be cleared. It is the EXTI interrupt associated with the status bit that interrupts the CPU.

Peripheral interrupt and wake-up signals that have a status bit in the peripheral, which requires to be cleared in the peripheral, are connected to an EXTI direct event input. There is no status pending bit within the EXTI. The interrupt or wake-up is cleared by the CPU in the peripheral. It is the peripheral interrupt that interrupts the CPU directly.

All GPIO ports are inputs for the EXTI multiplexer and used to select a port pin to wake up the system via a configurable event.

EXTI configurable event interrupts are connected to the NVIC. A secure CPU receives both its secure and nonsecure interrupts on the combined secure/nonsecure it_exti_c_per(y) interrupt signal. A nonsecure CPU only receives its nonsecure interrupts on the it_exti_c_per(y) interrupt signal. The CPU has its dedicated individual interrupt signal, which allows CPU interrupt masking based on EXTI_IMRx settings.

The dedicated EXTI/EVG CPU event is connected to the CPU rxev input.

EXTI CPU wake-up signals are connected to the PWR, and are used to wake up the system and CPU subsystem bus clocks.

25.3.2 EXTI wake-up interrupt list

The wake-up sources are listed in the table below. Peripherals able to generate wake-up or interrupt events are connected to the EXTI.

All wake-up sources are able to generate en event to the CPU. The EXTI CPU wake-up signals are connected. They are used to wake up the system, and CPU subsystem bus clocks.

For CPU interrupt handling, see *Section 24: Nested vectored interrupt controller (NVIC)*.

Table 138. EXTI line connection

EXTI index	Acronym	Input type	NVIC connected
0 -15	EXTI[15:0]	Configurable	Yes
16	Reserved	-	-
17	RTC secure wake-up	Direct	Yes
18	RTC nonsecure wake-up	Direct	Yes
19	TAMP wake-up	Direct	Yes
20	OTG1 VBUS plug/unplug	Configurable	Yes
21	OTG2 VBUS plug /unplug	Configurable	Yes
22	I2C1 wake-up	Direct	Yes
23	I2C2 wake-up	Direct	Yes
24	I2C3 wake-up	Direct	Yes
25	I2C4 wake-up	Direct	Yes
26	I3C1 wake-up	Direct	Yes
27	I3C2 wake-up	Direct	Yes
28	USART1 wake-up	Direct	Yes
29	USART2 wake-up	Direct	Yes
30	USART3 wake-up	Direct	Yes
31	UART4 wake-up	Direct	Yes
32	UART5 wake-up	Direct	Yes
33	USART6 wake-up	Direct	Yes
34	UART7 wake-up	Direct	Yes
35	UART8 wake-up	Direct	Yes
36	UART9 wake-up	Direct	Yes
37	USART10 wake-up	Direct	Yes
38	LPUART1 wake-up	Direct	Yes
39	ADF wake-up	Configurable	Yes
40	MDF1 wake-up	Configurable	Yes
41	SPI1 wake-up	Direct	Yes
42	SPI2 wake-up	Direct	Yes

Table 138. EXTI line connection (continued)

EXTI index	Acronym	Input type	NVIC connected
43	SPI3 wake-up	Direct	Yes
44	SPI4 wake-up	Direct	Yes
45	SPI5 wake-up	Direct	Yes
46	SPI6 wake-up	Direct	Yes
47	MDIOS wake-up	Direct	Yes
48	OTG1 wake-up	Direct	Yes
49	OTG2 wake-up	Direct	Yes
50	UCPD wake-up	Direct	Yes
51	ETH1 wake-up	Configurable	Yes
52	LPTIM1 wake-up	Direct	Yes
53	LPTIM2 wake-up	Direct	Yes
54	LPTIM2 _ch1 wake-up	Configurable	Yes
55	LPTIM3 wake-up	Direct	Yes
56	LPTIM3 _ch1 wake-up	Configurable	Yes
57	LPTIM4 wake-up	Direct	Yes
58	LPTIM5 wake-up	Direct	Yes
59	Reserved	-	-
60	Wake-up_PIN1	Direct	Yes
61	Wake-up_PIN2	Direct	Yes
62	Wake-up_PIN3	Direct	Yes
63	Wake-up_PIN4	Direct	Yes
64	WDGLS_EARLY	Direct	Yes
65	LSE_CSS	Direct	Yes
66	PVD wake-up	Configurable	Yes
67	Reserved	-	-
68	VM out wake-up	Configurable	Yes
69	pvm_VDDIO2 wake-up	Configurable	Yes
70	pvm_VDDIO3 wake-up	Configurable	Yes
71	pvm_VDDIO4 wake-up	Configurable	Yes
72	pvm_VDDIO5 wake-up	Configurable	Yes
73	pvm_VDD33USB wake-up	Configurable	Yes
74	pvm_VDDA18ADC wake-up	Configurable	Yes
75	Reserved	-	-
76	Reserved	-	-

Table 138. EXTI line connection (continued)

EXTI index	Acronym	Input type	NVIC connected
77	DTS wake-up	Direct	Yes
78-96	Reserved	-	-

25.4 EXTI functional description

Depending on the EXTI event input type and wake-up targets, different logic implementations are used. The applicable features are controlled from register bits:

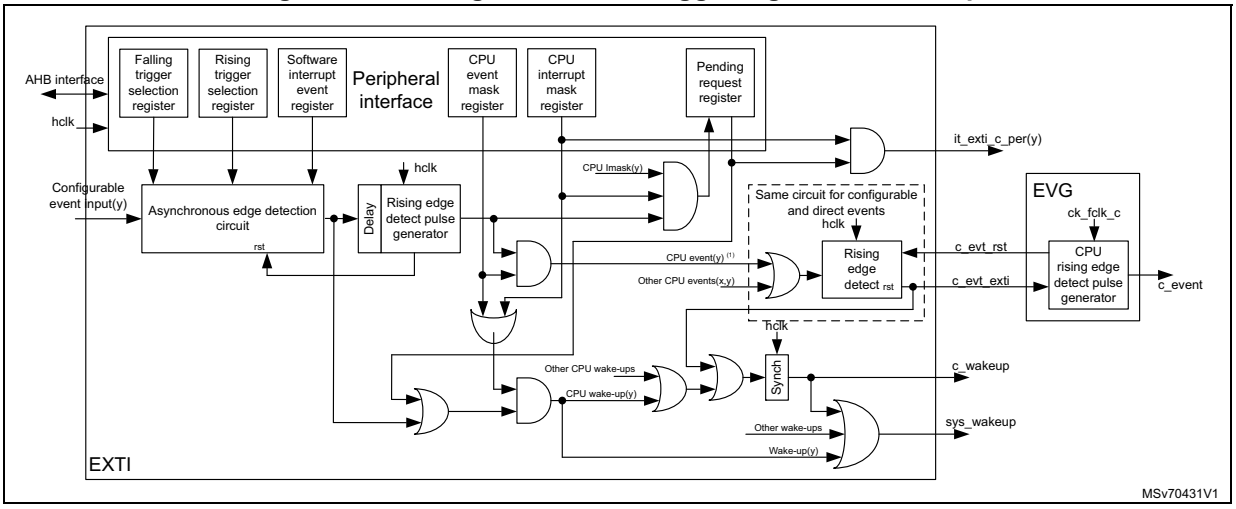
Table 139. EXTI event input configurations and register control

Event input type	Logic implementation	EXTI_RTSR	EXTI_FTSR	EXTI_SWIER	EXTI_RPR	EXTI_FPR	EXTI_SECCFGR	EXTI_PRIVCFGR
Configurable	Configurable event input wake-up logic	X	X	X	X	X	X	X
Direct	Direct event input wake-up logic	-	-	-	-	-	X	X

25.4.1 EXTI configurable event input wake-up

Figure 156 details the logic associated with configurable event inputs that wake up CPU subsystem bus clocks, and generate an EXTI pending flag and combined secure/non-secure interrupt to the CPU and/or a CPU wake-up event.

Figure 156. Configurable event trigger logic CPU wake-up



The software interrupt event register is used to trigger configurable events by software, writing the corresponding register bit, irrespective of the edge selection setting.

Rising and falling edge selection registers are used to enable and to select the configurable event active trigger edge or both edges.

The CPU has its dedicated wake-up (interrupt) mask register, and a dedicated event mask register. The enabled event is used to generate an event on the CPU. All CPU events are ORed together into a single CPU event signal. The event pending registers (EXTI_RPR and EXTI_FPR) are not set for an unmasked CPU event.

The configurable events have unique interrupt pending request register bits. The pending register is only set for an unmasked interrupt. Each configurable event provides dedicated combined secure/nonsecure interrupts to the CPU. The configurable event interrupts need to be acknowledged by software in EXTI_RPR and/or EXTI_FPR.

When a CPU wake-up (interrupt) or CPU event is enabled, the asynchronous edge detection circuit is reset by the clocked delay and rising edge detect pulse generator. This guarantees that the EXTI hclk clock is woken up before the asynchronous edge detection circuit is reset.

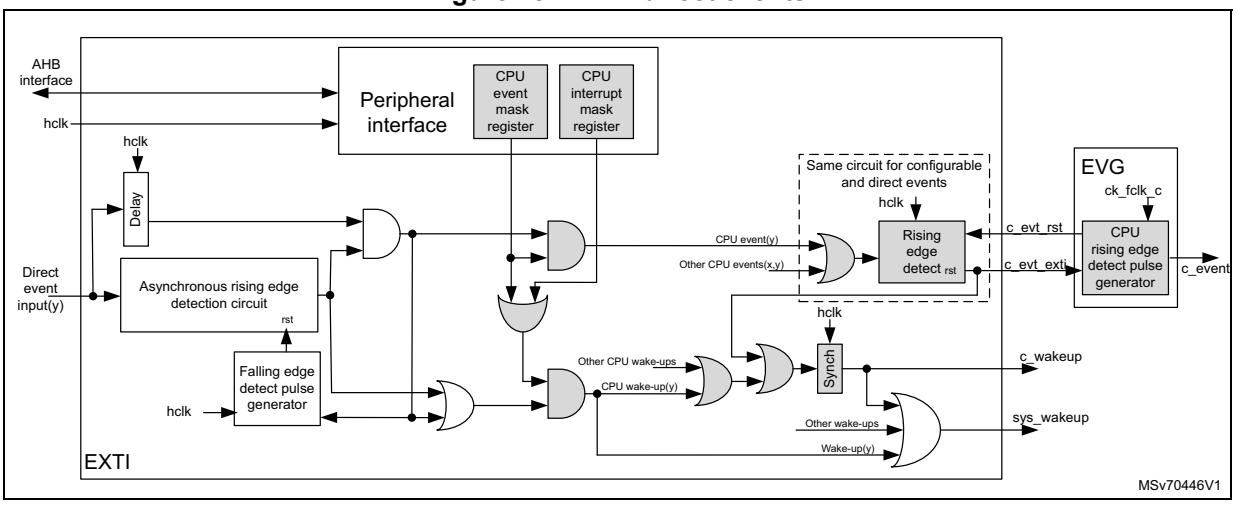
Note: A detected configurable event interrupt pending request can be cleared by the CPU with the correct access permission. The system is not able to enter into low-power modes as long as an interrupt pending request is active.

25.4.2 EXTI direct event input wake-up

Direct events do not have an associated EXTI interrupt. The EXTI only wakes up the system and CPU subsystem clocks, and can generate a CPU wake-up event. The peripheral synchronous interrupt, associated with the direct wake-up event, wakes up the CPU.

The EXTI direct event is able to generate a CPU event that wakes up the CPU. The CPU event may occur before the associated peripheral interrupt flag is set.

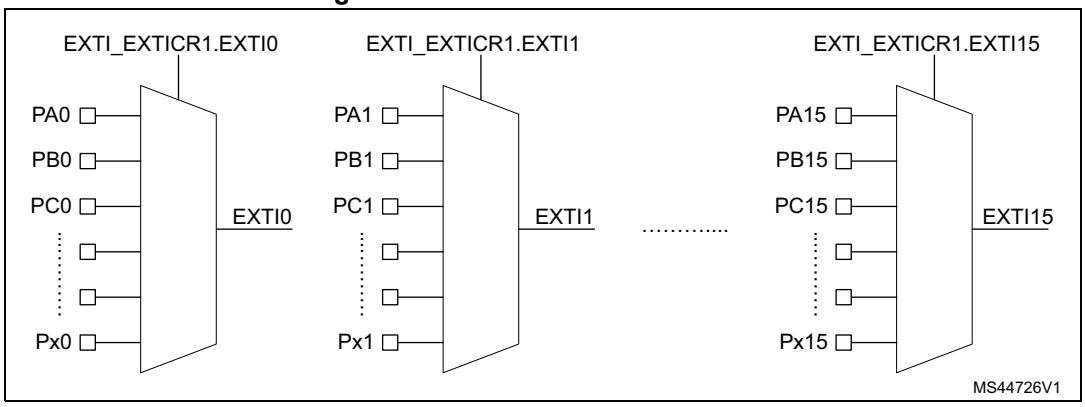
Figure 157. EXTI direct events



25.4.3 EXTI multiplexer selection

The EXTI multiplexer is used to select GPIOs as interrupts and wake-up. The GPIOs are connected via 16 EXTI multiplexers to the first 16 EXTI events as configurable event. The selection of GPIO port as EXTI multiplexer output is controlled by EXTIn_EXTICRx (x = 1 to 4) registers.

Figure 158. EXTI mux GPIO selection



25.5 EXTI functional behavior

The direct event inputs are enabled in the respective peripheral generating the wake-up event. The configurable events are enabled by enabling at least one of the trigger edges.

Once an event input is enabled, the generation of the CPU wake-up is conditioned by the CPU interrupt mask and the CPU event mask.

Table 140. Masking functionality

CPU interrupt enable EXTI_IMR.IMn	CPU event enable EXTI_EMR1.EMn	Configurable event inputs EXTI_RPR.RPIFn EXTI_FPR.FPIFn	exti(n) interrupt(1)	CPU event	CPU wake-up
0	0	No	Masked	Masked	Masked
	1	No	Masked	Yes	Yes
1	0	Status latched	Yes	Masked	Yes(2)
	1	Status latched	Yes	Yes	Yes

1. The CPU dedicated exti(n), combined secure/nonsecure interrupts, goes to the CPU. If no interrupt is required for the CPU, the exti(n) interrupt must be masked in EXTI_IMR.
2. Only if the CPU interrupt is enabled by IMn in EXTI_IMR.

For configurable event inputs, when the enabled edges occur on the event input, an event request is generated. When the associated CPU interrupt is unmasked, the corresponding RPIFn in EXTI_RPR and/or FPIFn in EXTI_FPR is/are set. The CPU subsystem is woken up, and the CPU combined secure/nonsecure interrupt signal is activated.

RPIFn in EXTI_RPR and/or FPIFn in EXTI_FPR must be cleared by software (writing them to 1). This clears the CPU interrupt. An interrupt is only generated on the dedicated interrupt line when it is enabled in EXTI_IMR.

For direct event inputs, when enabled in the associated peripheral, an event request is generated on the rising edge only. There is no corresponding CPU pending bit in the EXTI. When the associated CPU interrupt is unmasked, the corresponding CPU subsystem is woken up. The CPU is woken up (interrupted) by the peripheral synchronous interrupt.

The CPU event must be unmasked to generate an event. When an enabled edge occurs on the event input, a CPU event pulse is generated. There is no event pending bit.

For the configurable event inputs, an event request can be generated by software when writing 1 in EXTI_SWIERx. This generates a rising edge on the event. The rising edge event pending bit is set in EXTI_RPR, whatever the setting in EXTI_RTSR.

25.6 EXTI event protection

The EXTI is able to protect event register bits from being modified by nonsecure and unprivileged accesses. The protection can be activated individually per input event, via EXTI_SECCFGRx and EXTI_PRIVCFGRx. At EXTI level, the protection consists in preventing unauthorized write access to change settings of secure configurable events, change masking of secure input events, and clear pending status of secure input events.

Table 141. Register protection overview

Register name	Access type	Security(1)(2)
EXTI_RTSRx	RW	Secure and privileged can be enabled bit-wise in EXTI_SECCFGRx and EXTI_PRIVCFGRx
EXTI_FTSRx	RW	
EXTI_SWIERx	RW	
EXTI_RPRx	RW	
EXTI_FPRx	RW	
EXTI_SECCFGRx	RW	Always secure and privileged
EXTI_PRIVCFGRx	RW	Always privileged, secure can be enabled bit-wise in EXTI_SECCFGRx
EXTI_EXTICRx	RW	Secure and privileged can be enabled bit-wise in EXTI_SECCFGRx and EXTI_PRIVCFGRx
EXTI_LOCKR	RW	Always secure and privileged
EXTI_IMRx	RW	Secure and privileged can be enabled bit-wise in EXTI_SECCFGRx and EXTI_PRIVCFGRx
EXTI_EMRx	RW	

1. Security is enabled with EXTI_SECCFGRx registers.
2. Privilege is enabled with EXTI_PRIVCFGRx registers.

25.6.1 EXTI register security protection

When security is enabled for an input event, the associated input event configuration and the control bits can only be modified and read by a secure access. A nonsecure write access is discarded and a read returns 0.

When input events are nonsecure, the security is disabled. The associated input event configuration and control bits can be modified and read by a secure access and nonsecure access.

The security configuration in EXTI_SECCFGRx can be globally locked after reset by GLOCK in EXTI_LOCKR.

25.6.2 EXTI register privilege protection

When the privilege is enabled for an input event, the associated input event configuration and control bits can only be modified and read by a privileged access. An unprivileged write access is discarded, and a read returns 0.

When input events are unprivileged, the privilege is disabled. The associated input event configuration and control bits can be modified and read by a privileged or unprivileged access.

The privilege configuration in EXTI_PRIVCFGRx can be globally locked after reset by GLOCK in EXTI_LOCKR.

25.7 EXTI registers

The EXTI register map is divided in the following sections:

Table 142. EXTI register map sections

Address offset	Description
0x000 - 0x01C	General configurable event [31:0] configuration
0x020 - 0x03C	General configurable event [63:32] configuration
0x040 - 0x05C	General configurable event [95:64] configuration
0x060 - 0x06C	EXTI I/O port mux selection
0x070	EXTI protection lock configuration
0x080 - 0x0BC	CPU input event configuration

All registers can be accessed with word (32-bit), half-word (16-bit), and byte (8-bit) access.

25.7.1 EXTI rising trigger selection register (EXTI_RTSR1)

Address offset: 0x000

Reset value: 0x0000 0000

This register contains only bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT21	RT20	Res.	Res.	Res.	Res.
										rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RT15	RT14	RT13	RT12	RT11	RT10	RT9	RT8	RT7	RT6	RT5	RT4	RT3	RT2	RT1	RT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 21:20 **RTx:** Rising trigger event configuration bit of configurable event input x (x = 21 to 20)

0: Rising trigger disabled (for event and interrupt) for input line

1: Rising trigger enabled (for event and interrupt) for input line

- When SECx is disabled in EXTI_SECCFGR1, RTx can be accessed with nonsecure and secure access.
- When SECx is enabled in EXTI_SECCFGR1, RTx can be accessed only with secure access. Nonsecure write to RTx is discarded. Nonsecure read returns 0.
- When PRIVx is disabled in EXTI_PRIVCFGR1, RTx can be accessed with unprivileged and privileged access.
- When PRIVx is enabled in EXTI_PRIVCFGR1, RTx can be accessed only with privileged access. Unprivileged write to RTx is discarded. Unprivileged read returns 0.

Note: The configurable event input is edge triggered. No glitch must be generated on this input. If a rising edge on the configurable event input occurs while writing to the register, the associated pending bit is not set.

Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:0 **RTx:** Rising trigger event configuration bit of configurable event input x (x = 15 to 0)

25.7.2 EXTI falling trigger selection register (EXTI_FTSR1)

Address offset: 0x004

Reset value: 0x0000 0000

This register contains only bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT21	FT20	Res.	Res.	Res.	Res.
										rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FT15	FT14	FT13	FT12	FT11	FT10	FT9	FT8	FT7	FT6	FT5	FT4	FT3	FT2	FT1	FT0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 21:20 **FTx:** Falling trigger event configuration bit of configurable event input x (x = 21 to 20)

0: Falling trigger disabled (for event and interrupt) for input line

1: Falling trigger enabled (for event and interrupt) for input line.

- When SECx is disabled in EXTI_SECCFGR1, FTx can be accessed with non-secure and secure access.
- When SECx is enabled in EXTI_SECCFGR1, FTx can be accessed only with secure access. Nonsecure write to FTx is discarded. Nonsecure read returns 0.
- When PRIVx is disabled in EXTI_PRIVCFGR1, FTx can be accessed with unprivileged and privileged access.
- When PRIVx is enabled in EXTI_PRIVCFGR1, FTx can be accessed only with privileged access. Unprivileged write to FTx is discarded. Unprivileged read returns 0.

Note: The configurable event input is edge triggered. No glitch must be generated on this input. If a falling edge on the configurable event input occurs while writing to the register, the associated pending bit is not set.

Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:0 **FTx:** Falling trigger event configuration bit of configurable event input x (x = 15 to 0)

25.7.3 EXTI software interrupt event register (EXTI_SWIER1)

Address offset: 0x008

Reset value: 0x0000 0000

This register contains only bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI21	SWI20	Res.	Res.	Res.	Res.
										rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWI15	SWI14	SWI13	SWI12	SWI11	SWI10	SWI9	SWI8	SWI7	SWI6	SWI5	SWI4	SWI3	SWI2	SWI1	SWI0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 21:20 **SWIx:** Software interrupt on event x (x = 21 to 20)

0: Writing 0 has no effect.

1: Writing 1 to this bit triggers a rising edge event on event x (bit auto cleared by hardware).

- When SECx is disabled in EXTI_SECCFGR1, SWIx can be accessed with non-secure and secure access.
- When SECx is enabled in EXTI_SECCFGR1, SWIx can be accessed only with secure access. Nonsecure write to SWIx is discarded. Nonsecure read returns 0.
- When PRIVx is disabled in EXTI_PRIVCFGR1,SWIx can be accessed with unprivileged and privileged access.
- When PRIVx is enabled in EXTI_PRIVCFGR1,SWIx can be accessed only with privileged access. Unprivileged write to SWIx is discarded. Unprivileged read returns 0.
- A software interrupt is generated whatever the setting in EXTI_RTSR and EXTI_FTSR. It always returns 0 when read.

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:0 **SWIx:** Software interrupt on event x (x = 15 to 0)

25.7.4 EXTI rising edge pending register (EXTI_RPR1)

Address offset: 0x00C

Reset value: 0x0000 0000

This register contains only bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIF21	RPIF20	Res.	Res.	Res.	Res.
										rc_w1	rc_w1				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RPIF15	RPIF14	RPIF13	RPIF12	RPIF11	RPIF10	RPIF9	RPIF8	RPIF7	RPIF6	RPIF5	RPIF4	RPIF3	RPIF2	RPIF1	RPIF0
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

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

Bits 21:20 **RPIFx:** Configurable event input x rising edge pending bit (x = 21 to 20)

This bit is set when the selected rising edge event or an EXTI_SWIER1 software trigger arrives on the configurable event line. This bit is cleared by writing a 1 into the bit.

0: No rising edge trigger request occurred

1: Rising edge trigger request occurred

- When SECx is disabled in EXTI_SECCFGR1, RPIFx can be accessed with non-secure and secure access.
- When SECx is enabled in EXTI_SECCFGR1, RPIFx can be accessed only with secure access. Nonsecure write to RPIFx is discarded. Nonsecure read returns 0.
- When PRIVx is disabled in EXTI_PRIVCFGR1, RPIFx can be accessed with unprivileged and privileged access.
- When PRIVx is enabled in EXTI_PRIVCFGR1, RPIFx can be accessed only with privileged access. Unprivileged write to RPIFx is discarded. Unprivileged read returns 0.

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:0 **RPIFx:** Configurable event input x rising edge pending bit (x = 15 to 0)

25.7.5 EXTI falling edge pending register (EXTI_FPR1)

Address offset: 0x010

Reset value: 0x0000 0000

This register contains only bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPIF21	FPIF20	Res.	Res.	Res.	Res.
										rc_w1	rc_w1				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FPIF15	FPIF14	FPIF13	FPIF12	FPIF11	FPIF10	FPIF9	FPIF8	FPIF7	FPIF6	FPIF5	FPIF4	FPIF3	FPIF2	FPIF1	FPIF0
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

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

Bits 21:20 **FPIFx:** configurable event input x falling edge pending bit (x = 21 to 20)

This bit is set when the falling edge event arrives on the configurable event line. This bit is cleared by writing a 1 into this bit.

0: No falling edge trigger request occurred

1: Falling edge trigger request occurred

- When SECx is disabled in EXTI_SECCFGR1, FPIFx can be accessed with non-secure and secure access.
- When SECx is enabled in EXTI_SECCFGR1, FPIFx can be accessed only with secure access. Nonsecure write to FPIFx is discarded. Nonsecure read returns 0.
- When PRIVx is disabled in EXTI_PRIVCFGR1, FPIFx can be accessed with unprivileged and privileged access.
- When PRIVx is enabled in EXTI_PRIVCFGR1, FPIFx can be accessed only with privileged access. Unprivileged write to FPIFx is discarded. Unprivileged read returns 0.

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:0 **FPIFx:** Configurable event input x rising edge pending bit (x = 15 to 0)

25.7.6 EXTI security configuration register (EXTI_SECCFGR1)

Address offset: 0x014

Reset value: 0x0000 0000

This register provides write access security and privilege. A nonsecure or unprivileged write access is ignored, and causes the generation of an secure illegal access event. All read return the register data.

Write to this register is ignored if GLOCK is enabled in EXTI_LOCKR.

This register contains only bits for security capable input events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEC 31	SEC 30	SEC 29	SEC 28	SEC 27	SEC 26	SEC 25	SEC 24	SEC 23	SEC 22	SEC 21	SEC 20	SEC 19	SEC 18	SEC 17	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC15	SEC14	SEC13	SEC12	SEC11	SEC10	SEC9	SEC8	SEC7	SEC6	SEC5	SEC4	SEC3	SEC2	SEC1	SEC0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 **SECx:** Security enable on event input x (x = 31 to 17)

0: Event security disabled (nonsecure)

1: Event security enabled (secure)

Bit 16 Reserved, must be kept at reset value.

Bits 15:0 **SECx:** Security enable on event input x (x = 15 to 0)

25.7.7 EXTI privilege configuration register (EXTI_PRIVCFGR1)

Address offset: 0x018

Reset value: 0x0000 0000

This register provides privileged write access protection. An unprivileged write access is discarded, and causes the generation of a privileged illegal access event. An unprivileged read returns the register data.

Write to this register are ignored if GLOCK is enabled in EXTI_LOCKR.

This register contains only bits for security capable input events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRIV31	PRIV30	PRIV29	PRIV28	PRIV27	PRIV26	PRIV25	PRIV24	PRIV23	PRIV22	PRIV21	PRIV20	PRIV19	PRIV18	PRIV17	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIV15	PRIV14	PRIV13	PRIV12	PRIV11	PRIV10	PRIV9	PRIV8	PRIV7	PRIV6	PRIV5	PRIV4	PRIV3	PRIV2	PRIV1	PRIV0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 **PRIVx:** Privilege enable on event input x (x = 31 to 17)

0: Event privilege disabled (unprivileged)

1: Event privilege enabled (privileged)

- When SECx is disabled in EXT_SECCFGR1, PRIVx can be accessed with secure and nonsecure access.
- When SECx is enabled in EXTI_SECCFGR1, PRIVx can only be written with secure access. Nonsecure write to PRIVx is discarded.

Bit 16 Reserved, must be kept at reset value.

Bits 15:0 **PRIVx:** Privilege enable on event input x (x = 15 to 0)

25.7.8 EXTI rising trigger selection register (EXTI_RTSR2)

Address offset: 0x020

Reset value: 0x0000 0000

This register contains only bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT56	Res.	RT54	Res.	Res.	RT51	Res.	Res.	Res.
							rw		rw			rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT40	RT39	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							rw	rw							

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **RT56:** Rising trigger event configuration bit of configurable event input 56

0: Rising trigger disabled (for event and Interrupt) for input line x

1: Rising trigger enabled (for event and Interrupt) for input line x

- When SECx is disabled in EXTI_SECCFGR2, RTx can be accessed with non-secure and secure access.
- When SECx is enabled in EXTI_SECCFGR2, RTx can be accessed only with secure access. Nonsecure write to RTx is discarded. Nonsecure read returns 0.
- When PRIVx is disabled in EXTI_PRIVCFGR2, RTx can be accessed with unprivileged and privileged access.
- When PRIVx is enabled in EXTI_PRIVCFGR2, RTx can be accessed only with privileged access. Unprivileged write to RTx is discarded. Unprivileged read returns 0.

Note: The configurable event input is edge triggered. No glitch must be generated on this input. If a rising edge on the configurable event input occurs while writing to the register, the associated pending bit is not set.

Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

Bit 23 Reserved, must be kept at reset value.

Bit 22 **RT54:** Rising trigger event configuration bit of configurable event input 54

Bits 21:20 Reserved, must be kept at reset value.

Bit 19 **RT51:** Rising trigger event configuration bit of configurable event input 51

Bits 18:9 Reserved, must be kept at reset value.

Bits 8:7 **RTx:** Rising trigger event configuration bit of configurable event input x (x = 40 to 39)

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

25.7.9 EXTI falling trigger selection register (EXTI_FTSR2)

Address offset: 0x024

Reset value: 0x0000 0000

This register contains only bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT56	Res.	FT54	Res.	Res.	FT51	Res.	Res.	Res.
							rw		rw			rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT40	FT39	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							rw	rw							

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **FT56:** Falling trigger event configuration bit of configurable event input 56

0: Falling trigger disabled (for event and Interrupt) for input line

1: Falling trigger enabled (for event and Interrupt) for input line.

- When SECx is disabled in EXTI_SECCFGR2, FTx can be accessed with non-secure and secure access.
- When SECx is enabled in EXTISECCFGR2, FTx can be accessed only with secure access. Nonsecure write to FTx is discarded. Nonsecure read returns 0.
- When PRIVx is disabled in EXTI_PRIVCFGR2, FTx can be accessed with unprivileged and privileged access.
- When PRIVx is enabled in EXTI_PRIVCFGR2, FTx can be accessed only with privileged access. Unprivileged write to FTx is discarded. Unprivileged read returns 0.

Note: The configurable event input is edge triggered. No glitch must be generated on this input. If a falling edge on the configurable event input occurs while writing to the register, the associated pending bit is not set.

Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

Bit 23 Reserved, must be kept at reset value.

Bit 22 **FT54:** Falling trigger event configuration bit of configurable event input 54

Bits 21:20 Reserved, must be kept at reset value.

Bit 19 **FT51:** Falling trigger event configuration bit of configurable event input 51

Bits 18:9 Reserved, must be kept at reset value.

Bits 8:7 **FTx:** Falling trigger event configuration bit of configurable event input x (x = 40 to 39)

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

25.7.10 EXTI software interrupt event register (EXTI_SWIER2)

Address offset: 0x028

Reset value: 0x0000 0000

This register contains only bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI56	Res.	SWI54	Res.	Res.	SWI51	Res.	Res.	Res.
							rw		rw			rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI40	SWI39	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							rw	rw							

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **SWI56:** Software interrupt on event 56

0: Writing 0 has no effect.

1: Writing 1 to this bit triggers a rising edge event on event x (bit auto cleared by hardware).

- When SECx is disabled in EXTI_SECCFGR2, SWIx can be accessed with non-secure and secure access.
- When SECx is enabled in EXTI_SECCFGR2, SWIx can be accessed only with secure access. Nonsecure write to SWIx is discarded. Nonsecure read returns 0.
- When PRIVx is disabled in EXTI_PRIVCFGR2, SWIx can be accessed with unprivileged and privileged access.
- When PRIVx is enabled in EXTI_PRIVCFGR2, SWIx can be accessed only with privileged access. Unprivileged write to SWIx is discarded. Unprivileged read returns 0.
- A software interrupt is generated whatever the setting in EXTI_RTSR and EXTI_FTSR. It always returns 0 when read.

Bit 23 Reserved, must be kept at reset value.

Bit 22 **SWI54:** Software interrupt on event 54

Bits 21:20 Reserved, must be kept at reset value.

Bit 19 **SWI51:** Software interrupt on event 51

Bits 18:9 Reserved, must be kept at reset value.

Bits 8:7 **SWIx:** Software interrupt on event x (x = 40 to 39)

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

25.7.11 EXTI rising edge pending register (EXTI_RPR2)

Address offset: 0x02C

Reset value: 0x0000 0000

This register contains only bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIF56	Res.	RPIF54	Res.	Res.	RPIF51	Res.	Res.	Res.
							rc_w1		rc_w1			rc_w1			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIF40	RPIF39	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							rc_w1	rc_w1							

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **RPIF56:** Configurable event input 56 rising edge pending bit

This bit is set when the selected rising edge event or an EXTI_SWIER2 software trigger arrives on the configurable event line. This bit is cleared by writing a 1 into the bit.

0: No rising edge trigger request occurred

1: Rising edge trigger request occurred

- When SECx is disabled in EXTI_SECCFGR2, RPIFx can be accessed with non-secure and secure access.
- When SECx is enabled in EXTI_SECCFGR2, RPIFx can be accessed only with secure access. Nonsecure write to RPIFx is discarded. Nonsecure read returns 0.
- When PRIVx is disabled in EXTI_PRIVCFGR2, RPIFx can be accessed with unprivileged and privileged access.
- When PRIVx is enabled in EXTI_PRIVCFGR2, this bit can be accessed only with privileged access. Unprivileged write to RPIFx is discarded. Unprivileged read returns 0.

Bit 23 Reserved, must be kept at reset value.

Bit 22 **RPIF54:** Configurable event input 54 rising edge pending bit

Bits 21:20 Reserved, must be kept at reset value.

Bit 19 **RPIF51:** Configurable event input 51 rising edge pending bit

Bits 18:9 Reserved, must be kept at reset value.

Bits 8:7 **RPIFx:** Configurable event input x rising edge pending bit (x = 40 to 39)

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

25.7.12 EXTI falling edge pending register (EXTI_FPR2)

Address offset: 0x030

Reset value: 0x0000 0000

This register contains only bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPIF56	Res.	FPIF54	Res.	Res.	FPIF51	Res.	Res.	Res.
							rc_w1		rc_w1			rc_w1			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPIF40	FPIF39	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							rc_w1	rc_w1							

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **FPIF56:** Configurable event input 56 falling edge pending bit

This bit is set when the falling edge event arrives on the configurable event line. This bit is cleared by writing a 1 into the bit.

0: No falling edge trigger request occurred

1: Falling edge trigger request occurred

- When SECx is disabled in EXTI_SECCFGR2, FPIFx can be accessed with non-secure and secure access.
- When SECx is enabled in EXTI_SECCFGR2, FPIFx can be accessed only with secure access. Nonsecure write to this FPIFx is discarded. Nonsecure read returns 0.
- When PRIVx is disabled in EXTI_PRIVCFGR2, FPIFx can be accessed with unprivileged and privileged access.
- When PRIVx is enabled in EXTI_PRIVCFGR2, FPIFx can be accessed only with privileged access. Unprivileged write to FPIFx is discarded. Unprivileged read returns 0.

Bit 23 Reserved, must be kept at reset value.

Bit 22 **FPIF54:** Configurable event input 54 falling edge pending bit

Bits 21:20 Reserved, must be kept at reset value.

Bit 19 **FPIF51:** Configurable event input 51 falling edge pending bit

Bits 18:9 Reserved, must be kept at reset value.

Bits 8:7 **FPIFx:** Configurable event input x falling edge pending bit (x = 40 to 39)

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

25.7.13 EXTI security enable register (EXTI_SECCFGR2)

Address offset: 0x034

Reset value: 0x0000 0000

This register provides write access security and privilege. A nonsecure or unprivileged write access is ignored, and causes the generation of an secure illegal access event. All read return the register data.

Write to this register is ignored if GLOCK is enabled in EXTI_LOCKR.

This register contains only bits for security capable input events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEC 63	SEC 62	SEC 61	SEC 60	Res.	SEC 58	SEC 57	SEC 56	SEC 55	SEC 54	SEC 53	SEC 52	SEC 51	SEC 50	SEC 49	SEC 48
rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC 47	SEC 46	SEC 45	SEC 44	SEC 43	SEC 42	SEC 41	SEC 40	SEC 39	SEC 38	SEC 37	SEC 36	SEC 35	SEC 34	SEC 33	SEC 32
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **SECx:** Security enable on event input x (x = 63 to 60)

0: Event security disabled (nonsecure)

1: Event security enabled (secure)

Bit 27 Reserved, must be kept at reset value.

Bits 26:0 **SECx:** Security enable on event input x (x = 58 to 32)

25.7.14 EXTI privilege enable register (EXTI_PRIVCFGR2)

Address offset: 0x038

Reset value: 0x0000 0000

This register provides privileged write access protection. An unprivileged write access is discarded, and causes the generation of a privileged illegal access event. An unprivileged read returns the register data.

Write to this register is ignored if GLOCK is enabled in EXTI_LOCKR.

This register contains only bits for security capable input events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRIV 63	PRIV 62	PRIV 61	PRIV 60	Res.	PRIV 58	PRIV 57	PRIV 56	PRIV 55	PRIV 54	PRIV 53	PRIV 52	PRIV 51	PRIV 50	PRIV 49	PRIV 48
rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIV 47	PRIV 46	PRIV 45	PRIV 44	PRIV 43	PRIV 42	PRIV 41	PRIV 40	PRIV 39	PRIV 38	PRIV 37	PRIV 36	PRIV 35	PRIV 34	PRIV 33	PRIV 32
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **PRIVx:** Privilege enable on event input x (x = 63 to 60)

0: Event privilege disabled (unprivileged)

1: Event privilege enabled (privileged)

- When SECx is disabled in EXTI_SECCFGR2, PRIVx can be accessed with secure and nonsecure access.
- When SECx is enabled in EXTI_SECCFGR2, PRIVx can be accessed only with secure access. Nonsecure write to PRIVx is discarded.

Bit 27 Reserved, must be kept at reset value.

Bits 26:0 **PRIVx:** Privilege enable on event input x (x = 58 to 32)

25.7.15 EXTI rising trigger selection register (EXTI_RTSR3)

Address offset: 0x040

Reset value: 0x0000 0000

This register contains only bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	RT74	RT73	RT72	RT71	RT70	RT69	RT68	Res.	RT66	Res.	Res.
					rw	rw	rw	rw	rw	rw	rw		rw		

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:4 **RTx:** Rising trigger event configuration bit of configurable event input x (x = 74 to 68)

0: Rising trigger disabled (for event and interrupt) for input line

1: Rising trigger enabled (for event and interrupt) for input line

- When SECx is disabled in EXTI_SECCFGR3, RTx can be accessed with non-secure and secure access.
- When SECx is enabled in EXTI_SECCFGR3, RTx can be accessed only with secure access. Nonsecure write to RTx is discarded. Nonsecure read returns 0.
- When PRIVx i disabled in EXTI_PRIVCFGR3, RTx can be accessed with unprivileged and privileged access.
- When PRIVx is enabled in EXTI_PRIVCFGR3, RTx can be accessed only with privileged access. Unprivileged write to RTx is discarded. Unprivileged read returns 0.

Note: The configurable event input is edge triggered. No glitch must be generated on this input. If a rising edge on the configurable event input occurs while writing to the register, the associated pending bit is not set.

Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **RT66:** Rising trigger event configuration bit of configurable event input 66

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

25.7.16 EXTI falling trigger selection register (EXTI_FTSR3)

Address offset: 0x044

Reset value: 0x0000 0000

This register contains only bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	FT74	FT73	FT72	FT71	FT70	FT69	FT68	Res.	FT66	Res.	Res.
					rw	rw	rw	rw	rw	rw	rw		rw		

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:4 **FTx:** Falling trigger event configuration bit of configurable event input x (x = 74 to 68)

0: Falling trigger disabled (for event and interrupt) for input line

1: Falling trigger enabled (for event and interrupt) for input line

- When SECx is disabled in EXTI_SECCFGR3, FTx can be accessed with nonsecure and secure access.
- When SECx is enabled in EXTI_SECCFGR3, FTx can be accessed only with secure access. Nonsecure write to FTx is discarded. Nonsecure read returns 0.
- When PRIVx is disabled in EXTI_PRIVCFGR3, FTx can be accessed with unprivileged and privileged access.
- When PRIVx is enabled in EXTI_PRIVCFGR3, FTx can be accessed only with privileged access. Unprivileged write to FTx is discarded. Unprivileged read returns 0.

Note: The configurable event input is edge triggered. No glitch must be generated on this input. If a falling edge on the configurable event input occurs while writing to the register, the associated pending bit is not set.

Rising and falling edge triggers can be set for the same configurable event input. In this case, both edges generate a trigger.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **FT66:** Falling trigger event configuration bit of configurable event input 66

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

25.7.17 EXTI software interrupt event register (EXTI_SWIER3)

Address offset: 0x048

Reset value: 0x0000 0000

This register contains only bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	SWI74	SWI73	SWI72	SWI71	SWI70	SWI69	SWI68	Res.	SWI66	Res.	Res.
					rw	rw	rw	rw	rw	rw	rw		rw		

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:4 **SWIx:** Software interrupt on event x (x = 74 to 68)

0: Writing 0 has no effect.

1: Writing 1 to this bit triggers a rising edge event on event x (bit auto cleared by hardware).

- When SECx is disabled in EXTI_SECCFGR3, SWIx can be accessed with non-secure and secure access.
- When SECx is enabled in EXTI_SECCFGR3, SWIx can be accessed only with secure access. Nonsecure write to SWIx is discarded. Nonsecure read returns 0.
- When PRIVx is disabled in EXTI_PRIVCFGR3, SWIx can be accessed with unprivileged and privileged access.
- When PRIVx is enabled in EXTI_PRIVCFGR3, SWIx can be accessed only with privileged access. Unprivileged write to SWIx is discarded. Unprivileged read returns 0.
- A software interrupt is generated whatever the setting in EXTI_RTSR and EXTI_FTSR. It always returns 0 when read.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **SWI66:** Software interrupt on event 66

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

25.7.18 EXTI rising edge pending register (EXTI_RPR3)

Address offset: 0x04C

Reset value: 0x0000 0000

This register contains only bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	RPIF74	RPIF73	RPIF72	RPIF71	RPIF70	RPIF69	RPIF68	Res.	RPIF66	Res.	Res.
					rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1		rc_w1		

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:4 **RPIFx:** configurable event input x rising edge pending bit (x = 74 to 68)

This bit is set when the selected rising edge event or an EXTI_SWIER3 software trigger arrives on the configurable event line. This bit is cleared by writing a 1 into the bit.

0: No rising edge trigger request occurred

1: Rising edge trigger request occurred

- When SECx is disabled in EXTI_SECCFGR3, RPIFx can be accessed with non-secure and secure access.
- When SECx is enabled in EXTI_SECCFGR3, RPIFx can be accessed only with secure access. Nonsecure write to this RPIFx is discarded. Nonsecure read returns 0.
- When PRIVx is disabled in EXTI_PRIVCFGR3, RPIFx can be accessed with unprivileged and privileged access.
- When PRIVx is enabled in EXTI_PRIVCFGR3, RPIFx can be accessed only with privileged access. Unprivileged write to RPIFx is discarded. Unprivileged read returns 0.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **RPIF66:** configurable event input 66rising edge pending bit

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

25.7.19 EXTI falling edge pending register (EXTI_FPR3)

Address offset: 0x050

Reset value: 0x0000 0000

This register contains only bits for configurable events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	FPIF74	FPIF73	FPIF72	FPIF71	FPIF70	FPIF69	FPIF68	Res.	FPIF66	Res.	Res.
					rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1		rc_w1		

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:4 **FPIFx:** configurable event input x falling edge pending bit (x = 74 to 68)

This bit is set when the falling edge event arrives on the configurable event line. This bit is cleared by writing a 1 into the bit.

0: No falling edge trigger request occurred

1: Falling edge trigger request occurred

- When SECx is disabled in EXTI_SECCFGR3, FPIFx can be accessed with non-secure and secure access.
- When SECx is enabled in EXTISECCFGR3, FPIFx can be accessed only with secure access. Nonsecure write to FPIFx is discarded. Nonsecure read returns 0.
- When PRIVx is disabled in EXTI_PRIVCFGR3, FPIFx can be accessed with unprivileged and privileged access.
- When PRIVx is enabled in EXTI_PRIVCFGR3, FPIFx can be accessed only with privileged access. Unprivileged write to FPIFx is discarded. Unprivileged read returns 0.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **FPIF66:** configurable event input 66 falling edge pending bit

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

25.7.20 EXTI security enable register (EXTI_SECCFGR3)

Address offset: 0x054

Reset value: 0x0000 0000

This register provides write access security and privilege. A nonsecure or unprivileged write access is ignored and causes the generation of an secure illegal access event. All reads return the register data.

Write to this register are ignored if GLOCK is enabled in EXTI_LOCKR.

This register contains only bits for security capable input events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	SEC 77	Res.	Res.	SEC 74	SEC 73	SEC 72	SEC 71	SEC 70	SEC 69	SEC 68	Res.	SEC 66	SEC 65	SEC 64
		rw			rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **SEC77:** Security enable on event input 77
0: Event security disabled (nonsecure)
1: Event security enabled (secure)

Bits 12:11 Reserved, must be kept at reset value.

Bits 10:4 **SECx:** Security enable on event input x (x = 74 to 68)

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **SECx:** Security enable on event input x (x = 66 to 64)

25.7.21 EXTI privilege enable register (EXTI_PRIVCFGR3)

Address offset: 0x058

Reset value: 0x0000 0000

This register provides privileged write access protection. An unprivileged write access is discarded and causes the generation of a privileged illegal access event. An unprivileged read returns the register data.

Write to this register are ignored if GLOCK is enabled in EXTI_LOCKR.

This register contains only bits for security capable Input events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	PRIV 77	Res.	Res.	PRIV 74	PRIV 73	PRIV 72	PRIV 71	PRIV 70	PRIV 69	PRIV 68	Res.	PRIV 66	PRIV 65	PRIV 64
		rw			rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **PRIV77:** Privilege enable on event input 77
0: Event privilege disabled (unprivileged
1: Event privilege enabled (privileged)
- When SECx is disabled in EXTI_SECCFGR3, PRIVx can be accessed with secure and nonsecure access.
- When SECx is enabled in EXTI_SECCFGR3, PRIVx can only be written with secure access. Nonsecure write to PRIVx is discarded.

Bits 12:11 Reserved, must be kept at reset value.

Bits 10:4 **PRIVx:** Privilege enable on event input x (x = 74 to 68)

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **PRIVx:** Privilege enable on event input x (x = 66 to 64)

25.7.22 EXTI external interrupt selection register 1 (EXTI_EXTICR1)

Address offset: 0x060

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXTI3[7:0]								EXTI2[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI1[7:0]								EXTI0[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **EXTI3[7:0]:** EXTI3 GPIO port selection

These bits are written by software to select the source input for EXTI3 external interrupt.
0x00: PA3 pin
0x01: PB3 pin
0x02: PC3 pin
0x03: PD3 pin
0x04: PE3 pin
0x05: PF3 pin
0x06: PG3 pin
0x07: PH3 pin
0x08: PN3 pin
0x09: PO3 pin
0x0A: PP3 pin
0x0B: PQ3 pin
Others: reserved

- When SEC3 is disabled in EXTI_SECCFGR1, this field can be accessed with non-secure and secure access.
- When SEC3 is enabled in EXTI_SECCFGR1, this field can be accessed only with secure access. Nonsecure write is discarded. Nonsecure read returns 0.
- When PRIV3 is disabled in EXTI_PRIVCFGR1, this field can be accessed with privileged and unprivileged access.
- When PRIV3 is enabled in EXTI_PRIVCFGR1, this field can be accessed only with privileged access. Unprivileged write to this field is discarded, and read returns 0.

Bits 23:16 **EXTI2[7:0]:** EXTI2 GPIO port selection

These bits are written by software to select the source input for EXTI2 external interrupt.

0x00: PA2 pin
0x01: PB2 pin
0x02: PC2 pin
0x03: PD2 pin
0x04: PE2 pin
0x05: PF2 pin
0x06: PG2 pin
0x07: PH2 pin
0x08: PN2 pin
0x09: PO2 pin
0x0A: PP2 pin
0x0B: PQ2 pin
Others: reserved

- When SEC2 is disabled in EXTI_SECCFGR1, this field can be accessed with non-secure and secure access.
- When SEC2 is enabled in EXTI_SECCFGR1, this field can be accessed only with secure access. Nonsecure write is discarded, nonsecure read returns 0.
- When PRIV2 is disabled in EXTI_PRIVCFGR1, this field can be accessed with privileged and unprivileged access.
- When PRIV2 is enabled in EXTI_PRIVCFGR1, this field can be accessed only with privileged access. Unprivileged write to this field is discarded and read returns 0.

Bits 15:8 **EXTI1[7:0]:** EXTI1 GPIO port selection

These bits are written by software to select the source input for EXTI1 external interrupt.

0x00: PA1 pin
0x01: PB1 pin
0x02: PC1 pin
0x03: PD1 pin
0x04: PE1 pin
0x05: PF1 pin
0x06: PG1 pin
0x07: PH1 pin
0x08: PN1 pin
0x09: PO1 pin
0x0A: PP1 pin
0x0B: PQ1 pin
Others: reserved

- When SEC1 is disabled in EXTI_SECCFGR1, this field can be accessed with non-secure and secure access.
- When SEC1 is enabled in EXTI_SECCFGR1, this field can be accessed only with secure access. Nonsecure write is discarded, nonsecure read returns 0.
- When PRIV1 is disabled in EXTI_PRIVCFGR1, this field can be accessed with privileged and unprivileged access.
- When PRIV1 is enabled in EXTI_PRIVCFGR1, this field can be accessed only with privileged access. Unprivileged write to this field is discarded and read returns 0.

Bits 7:0 **EXTI0[7:0]:** EXTI0 GPIO port selection.

These bits are written by software to select the source input for EXTI0 external interrupt.

0x00: PA0 pin
0x01: PB0 pin
0x02: PC0 pin
0x03: PD0 pin
0x04: PE0 pin
0x05: PF0 pin
0x06: PG0 pin
0x07: PH0 pin
0x08: PN0 pin
0x09: PO0 pin
0x0A: PP0 pin
0x0B: PQ0 pin
Others: reserved

- When SEC0 is disabled in EXTI_SECCFGR1 this field can be accessed with non-secure and secure access.
- When SEC0 is enabled in EXTI_SECCFGR1, this field can be accessed only with secure access. Nonsecure write is discarded, nonsecure read returns 0.
- When PRIV0 is disabled in EXTI_PRIVCFGR1, this field can be accessed with privileged and unprivileged access.
- When PRIV0 is enabled in EXTI_PRIVCFGR1, this field can be accessed only with privileged access. Unprivileged write to this field is discarded and read returns 0.

25.7.23 EXTI external interrupt selection register 2 (EXTI_EXTICR2)

Address offset: 0x064

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXTI7[7:0]								EXTI6[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI5[7:0]								EXTI4[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **EXTI7[7:0]:** EXTI7 GPIO port selection.

These bits are written by software to select the source input for EXTI7 external interrupt.

0x00: PA7 pin
0x01: PB7 pin
0x02: PC7 pin
0x03: PD7 pin
0x04: PE7 pin
0x05: PF7 pin
0x06: PG7 pin
0x07: PH7 pin
0x08: PN7 pin
0x09: PO7 pin
0x0A: PP7 pin
0x0B: PQ7 pin
Others: reserved

- When SEC7 is disabled in EXTI_SECCFGR1 this field can be accessed with non-secure and secure access.
- When SEC7 is enabled in EXTI_SECCFGR1, this field can be accessed only with secure access. Nonsecure write is discarded, nonsecure read returns 0.
- When PRIV7 is disabled in EXTI_PRIVCFGR1, this field can be accessed with privileged and unprivileged access.
- When PRIV7 is enabled in EXTI_PRIVCFGR1, this field can be accessed only with privileged access. Unprivileged write to this field is discarded and read returns 0.

Bits 23:16 **EXTI6[7:0]:** EXTI6 GPIO port selection.

These bits are written by software to select the source input for EXTI6 external interrupt.

0x00: PA6 pin
0x01: PB6 pin
0x02: PC6 pin
0x03: PD6 pin
0x04: PE6 pin
0x05: PF6 pin
0x06: PG6 pin
0x07: PH6 pin
0x08: PN6 pin
0x09: PO6 pin
0x0A: PP6 pin
0x0B: PQ6 pin
Others: reserved

- When SEC6 is disabled in EXTI_SECCFGR1 this field can be accessed with non-secure and secure access.
- When SEC6 is enabled in EXTI_SECCFGR1, this field can be accessed only with secure access. Nonsecure write is discarded, nonsecure read returns 0.
- When PRIV6 is disabled in EXTI_PRIVCFGR1, this field can be accessed with privileged and unprivileged access.
- When PRIV6 is enabled in EXTI_PRIVCFGR1, this field can be accessed only with privileged access. Unprivileged write to this field is discarded and read returns 0.

Bits 15:8 **EXTI5[7:0]:** EXTI5 GPIO port selection.

These bits are written by software to select the source input for EXTI5 external interrupt.

0x00: PA5 pin
0x01: PB5 pin
0x02: PC5 pin
0x03: PD5 pin
0x04: PE5 pin
0x05: PF5 pin
0x06: PG5 pin
0x07: PH5 pin
0x08: PN5 pin
0x09: PO5 pin
0x0A: PP5 pin
0x0B: PQ5 pin
Others: reserved

- When SEC5 is disabled in EXTI_SECCFGR1 this field can be accessed with non-secure and secure access.
- When SEC5 is enabled in EXTI_SECCFGR1, this field can be accessed only with secure access. Nonsecure write is discarded, nonsecure read returns 0.
- When PRIV5 is disabled in EXTI_PRIVCFGR1, this field can be accessed with privileged and unprivileged access.
- When PRIV5 is enabled in EXTI_PRIVCFGR1, this field can be accessed only with privileged access. Unprivileged write to this field is discarded and read returns 0.

Bits 7:0 **EXTI4[7:0]:** EXTI4 GPIO port selection.

These bits are written by software to select the source input for EXTI4 external interrupt.

0x00: PA4 pin
0x01: PB4 pin
0x02: PC4 pin
0x03: PD4 pin
0x04: PE4 pin
0x05: PF4 pin
0x06: PG4 pin
0x07: PH4 pin
0x08: PN4 pin
0x09: PO4 pin
0x0A: PP4 pin
0x0B: PQ4 pin
Others: reserved

- When SEC4 is disabled in EXTI_SECCFGR1 this field can be accessed with non-secure and secure access.
- When SEC4 is enabled in EXTI_SECCFGR1, this field can be accessed only with secure access. Nonsecure write is discarded, nonsecure read returns 0.
- When PRIV4 is disabled in EXTI_PRIVCFGR1, this field can be accessed with privileged and unprivileged access.
- When PRIV4 is enabled in EXTI_PRIVCFGR1, this field can be accessed only with privileged access. Unprivileged write to this field is discarded and read returns 0.

25.7.24 EXTI external interrupt selection register 3 (EXTI_EXTICR3)

Address offset: 0x068

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXTI11[7:0]								EXTI10[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI9[7:0]								EXTI8[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **EXTI11[7:0]:** EXTI11 GPIO port selection.

These bits are written by software to select the source input for EXTI11 external interrupt.

0x00: PA11 pin
0x01: PB11 pin
0x02: PC11 pin
0x03: PD11 pin
0x04: PE11 pin
0x05: PF11 pin
0x06: PG11 pin
0x07: PH11 pin
0x08: PN11 pin
0x09: PO11 pin
0x0A: PP11 pin
0x0B: PQ11 pin
Others: reserved

- When SEC11 is disabled in EXTI_SECCFGR1 this field can be accessed with non-secure and secure access.
- When SEC11 is enabled in EXTI_SECCFGR1, this field can be accessed only with secure access. Nonsecure write is discarded, nonsecure read returns 0.
- When PRIV11 is disabled in EXTI_PRIVCFGR1, this field can be accessed with privileged and unprivileged access.
- When PRIV11 is enabled in EXTI_PRIVCFGR1, this field can be accessed only with privileged access. Unprivileged write to this field is discarded and read returns 0.

Bits 23:16 **EXTI10[7:0]:** EXTI10 GPIO port selection.

These bits are written by software to select the source input for EXTI10 external interrupt.

0x00: PA10 pin
0x01: PB10 pin
0x02: PC10 pin
0x03: PD10 pin
0x04: PE10 pin
0x05: PF10 pin
0x06: PG10 pin
0x07: PH10 pin
0x08: PN10 pin
0x09: PO10 pin
0x0A: PP10 pin
0x0B: PQ10 pin
Others: reserved

- When SEC10 is disabled in EXTI_SECCFGR1 this field can be accessed with non-secure and secure access.
- When SEC10 is enabled in EXTI_SECCFGR1, this field can be accessed only with secure access. Nonsecure write is discarded, nonsecure read returns 0.
- When PRIV10 is disabled in EXTI_PRIVCFGR1, this field can be accessed with privileged and unprivileged access.
- When PRIV10 is enabled in EXTI_PRIVCFGR1, this field can be accessed only with privileged access. Unprivileged write to this field is discarded and read returns 0.

Bits 15:8 **EXTI9[7:0]:** EXTI9 GPIO port selection.

These bits are written by software to select the source input for EXTI9 external interrupt.

0x00: PA9 pin
0x01: PB9 pin
0x02: PC9 pin
0x03: PD9 pin
0x04: PE9 pin
0x05: PF9 pin
0x06: PG9 pin
0x07: PH9 pin
0x08: PN9 pin
0x09: PO9 pin
0x0A: PP9 pin
0x0B: PQ9 pin
Others: reserved

- When SEC9 is disabled in EXTI_SECCFGR1 this field can be accessed with non-secure and secure access.
- When SEC9 is enabled in EXTI_SECCFGR1, this field can be accessed only with secure access. Nonsecure write is discarded, nonsecure read returns 0.
- When PRIV9 is disabled in EXTI_PRIVCFGR1, this field can be accessed with privileged and unprivileged access.
- When PRIV9 is enabled in EXTI_PRIVCFGR1, this field can be accessed only with privileged access. Unprivileged write to this field is discarded and read returns 0.

Bits 7:0 **EXTI8[7:0]:** EXTI8 GPIO port selection.

These bits are written by software to select the source input for EXTI8 external interrupt.

0x00: PA8 pin
0x01: PB8 pin
0x02: PC8 pin
0x03: PD8 pin
0x04: PE8 pin
0x05: PF8 pin
0x06: PG8 pin
0x07: PH8 pin
0x08: PN8 pin
0x09: PO8 pin
0x0A: PP8 pin
0x0B: PQ8 pin
Others: reserved

- When SEC8 is disabled in EXTI_SECCFGR1 this field can be accessed with non-secure and secure access.
- When SEC8 is enabled in EXTI_SECCFGR1, this field can be accessed only with secure access. Nonsecure write is discarded, nonsecure read returns 0.
- When PRIV8 is disabled in EXTI_PRIVCFGR1, this field can be accessed with privileged and unprivileged access.
- When PRIV8 is enabled in EXTI_PRIVCFGR1, this field can be accessed only with privileged access. Unprivileged write to this field is discarded and read returns 0.

25.7.25 EXTI external interrupt selection register 4 (EXTI_EXTICR4)

Address offset: 0x06C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EXTI15[7:0]								EXTI14[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXTI13[7:0]								EXTI12[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **EXTI15[7:0]:** EXTI15 GPIO port selection.

These bits are written by software to select the source input for EXTI15 external interrupt.

0x00: PA15 pin
0x01: PB15 pin
0x02: PC15 pin
0x03: PD15 pin
0x04: PE15 pin
0x05: PF15 pin
0x06: PG15 pin
0x07: PH15 pin
0x08: PN15 pin
0x09: PO15 pin
0x0A: PP15 pin
0x0B: PQ15 pin
Others: reserved

- When SEC15 is disabled in EXTI_SECCFGR1 this field can be accessed with non-secure and secure access.
- When SEC15 is enabled in EXTI_SECCFGR1, this field can be accessed only with secure access. Nonsecure write is discarded, nonsecure read returns 0.
- When PRIV15 is disabled in EXTI_PRIVCFGR1, this field can be accessed with privileged and unprivileged access.
- When PRIV15 is enabled in EXTI_PRIVCFGR1, this field can be accessed only with privileged access. Unprivileged write to this field is discarded and read returns 0.

Bits 23:16 **EXTI14[7:0]:** EXTI14 GPIO port selection.

These bits are written by software to select the source input for EXTI14 external interrupt.

0x00: PA14 pin
0x01: PB14 pin
0x02: PC14 pin
0x03: PD14 pin
0x04: PE14 pin
0x05: PF14 pin
0x06: PG14 pin
0x07: PH14 pin
0x08: PN14 pin
0x09: PO14 pin
0x0A: PP14 pin
0x0B: PQ14 pin
Others: reserved

- When SEC14 is disabled in EXTI_SECCFGR1 this field can be accessed with non-secure and secure access.
- When SEC14 is enabled in EXTI_SECCFGR1, this field can be accessed only with secure access. Nonsecure write is discarded, nonsecure read returns 0.
- When PRIV14 is disabled in EXTI_PRIVCFGR1, this field can be accessed with privileged and unprivileged access.
- When PRIV14 is enabled in EXTI_PRIVCFGR1, this field can be accessed only with privileged access. Unprivileged write to this field is discarded and read returns 0.

Bits 15:8 **EXTI13[7:0]:** EXTI13 GPIO port selection.

These bits are written by software to select the source input for EXTI13 external interrupt.

0x00: PA13 pin
0x01: PB13 pin
0x02: PC13 pin
0x03: PD13 pin
0x04: PE13 pin
0x05: PF13 pin
0x06: PG13 pin
0x07: PH13 pin
0x08: PO13 pin
0x09: PP13 pin
0x0A: PQ13 pin
0x0B: PN13 pin
Others: reserved

- When SEC13 is disabled in EXTI_SECCFGR1 this field can be accessed with non-secure and secure access.
- When SEC13 is enabled in EXTI_SECCFGR1, this field can be accessed only with secure access. Nonsecure write is discarded, nonsecure read returns 0.
- When PRIV13 is disabled in EXTI_PRIVCFGR1, this field can be accessed with privileged and unprivileged access.
- When PRIV13 is enabled in EXTI_PRIVCFGR1, this field can be accessed only with privileged access. Unprivileged write to this field is discarded and read returns 0.

Bits 7:0 **EXTI12[7:0]:** EXTI12 GPIO port selection.

These bits are written by software to select the source input for EXTI12 external interrupt.

0x00: PA12 pin
0x01: PB12 pin
0x02: PC12 pin
0x03: PD12 pin
0x04: PE12 pin
0x05: PF12 pin
0x06: PG12 pin
0x07: PH12 pin
0x08: PO12 pin
0x09: PP12 pin
0x0A: PQ12 pin
0x0B: PN12 pin
Others: reserved

- When SEC12 is disabled in EXTI_SECCFGR1 this field can be accessed with non-secure and secure access.
- When SEC12 is enabled in EXTI_SECCFGR1, this field can be accessed only with secure access. Nonsecure write is discarded, nonsecure read returns 0.
- When PRIV12 is disabled in EXTI_PRIVCFGR1, this field can be accessed with privileged and unprivileged access.
- When PRIV12 is enabled in EXTI_PRIVCFGR1, this field can be accessed only with privileged access. Unprivileged write to this field is discarded and read returns 0.

25.7.26 EXTI lock register (EXTI_LOCKR)

Address offset: 0x070

Reset value: 0x0000 0000

This register can only be written by secure or privileged access. A write access from any other protection level is discarded, and any read returns zero value. Both illegal write and read access generate an illegal access event.

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

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

Bit 0 **GLOCK:** Global security privilege EXTI_SECCFGRx/PRIVCFGRx
This bit is written once after reset.
0: Security, privilege open can be modified.
1: Security, privilege locked can no longer be modified.

25.7.27 EXTI CPU wake-up with interrupt mask register 1 (EXTI_IMR1)

Address offset: 0x080

Reset value: 0x0000 0000

This register contains only bits for configurable events and direct events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IM31	IM30	IM29	IM28	IM27	IM26	IM25	IM24	IM23	IM22	IM21	IM20	IM19	IM18	IM17	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IM15	IM14	IM13	IM12	IM11	IM10	IM9	IM8	IM7	IM6	IM5	IM4	IM3	IM2	IM1	IM0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 **IMx:** CPU wake-up with interrupt mask on event input x (x = 31 to 17)
0: Wake-up and interrupt from input event x is masked.
1: Wake-up and interrupt from input event x is unmasked. CPU wake-up event and configurable event interrupt generated.

- When SECx is disabled in EXTI_SECCFGR1, IMx can be accessed with nonsecure and secure access.
- When SECx is enabled in EXTI_SECCFGR1, IMx can be accessed only with secure access. Nonsecure write to IMx is discarded. Nonsecure read returns 0.
- When PRIVx is disabled in EXTI_PRIVCFGR1, IMx can be accessed with privileged and unprivileged access.
- When PRIVx is enabled in EXTI_PRIVCFGR1, IMx can be accessed only with privileged access. Unprivileged write to IMx is discarded, and read returns 0.

Bit 16 Reserved, must be kept at reset value.

Bits 15:0 **IMx:** CPU wake-up with interrupt mask on event input x (x = 15 to 0)

25.7.28 EXTI CPU wake-up with event mask register 1 (EXTI_EMR1)

Address offset: 0x084

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EM31	EM30	EM29	EM28	EM27	EM26	EM25	EM24	EM23	EM22	EM21	EM20	EM19	EM18	EM17	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EM15	EM14	EM13	EM12	EM11	EM10	EM9	EM8	EM7	EM6	EM5	EM4	EM3	EM2	EM1	EM0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 **EMx:** CPU wake-up with event on event input x (x = 31 to 17)

0: Wake-up with event request from line x is masked.

1: Wake-up with event request from line x is unmasked.

- When SECx is disabled in EXTI_SECCFGR1, EMx can be accessed with non-secure and secure access.
- When SECx is enabled in EXTI_SECCFGR1, EMx can be accessed only with secure access. Nonsecure write to EMx is discarded. Nonsecure read returns 0.
- When PRIVx is disabled in EXTI_PRIVCFGR1, EMx can be accessed with privileged and unprivileged access.
- When PRIVx is enabled in EXTI_PRIVCFGR1, EMx can be accessed only with privileged access. Unprivileged write to EMx is discarded, and read returns 0.

Bit 16 Reserved, must be kept at reset value.

Bits 15:0 **EMx:** CPU wake-up with interrupt mask on event input x (x = 15 to 0)

25.7.29 EXTI CPU wake-up with interrupt mask register 2 (EXTI_IMR2)

Address offset: 0x090

Reset value: 0x0000 0000

This register contains only bits for configurable events and direct events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IM63	IM62	IM61	IM60	Res.	IM58	IM57	IM56	IM55	IM54	IM53	IM52	IM51	IM50	IM49	IM48
rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IM47	IM46	IM45	IM44	IM43	IM42	IM41	IM40	IM39	IM38	IM37	IM36	IM35	IM34	IM33	IM32
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **IMx:** CPU wake-up with interrupt mask on event input x (x = 63 to 60)

0: Wake-up and interrupt from input event x is masked.

1: Wake-up and interrupt from input event x is unmasked. CPU wake-up event and configurable event interrupt generated.

- When SECx is disabled in EXTI_SECCFGR2, IMx can be accessed with nonsecure and secure access.
- When SECx is enabled in EXTI_SECCFGR2, IMx can be accessed only with secure access. Nonsecure write to IMx is discarded. Nonsecure read returns 0.
- When PRIVx is disabled in EXTI_PRIVCFGR2, IMx can be accessed with privileged and unprivileged access.
- When PRIVx is enabled in EXTI_PRIVCFGR2, IMx can be accessed only with privileged access. Unprivileged write to IMx is discarded, and read returns 0.

Bit 27 Reserved, must be kept at reset value.

Bits 26:0 **IMx:** CPU wake-up with interrupt mask on event input x (x = 58 to 32)

25.7.30 EXTI CPU wake-up with event mask register 2 (EXTI_EMR2)

Address offset: 0x094

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EM63	EM62	EM61	EM60	Res.	EM58	EM57	EM56	EM55	EM54	EM53	EM52	EM51	EM50	EM49	EM48
rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EM47	EM46	EM45	EM44	EM43	EM42	EM41	EM40	EM39	EM38	EM37	EM36	EM35	EM34	EM33	EM32
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **EMx:** CPU wake-up with event on event input x (x = 63 to 60)

0: Wake-up with event request from line x is masked.

1: Wake-up with event request from line x is unmasked.

- When SECx is disabled in EXTI_SECCFGR2, EMx can be accessed with non-secure and secure access.
- When SECx is enabled in EXTI_SECCFGR2, EMx can be accessed only with secure access. Nonsecure write to EMx is discarded. Nonsecure read returns 0.
- When PRIVx is disabled in EXTI_PRIVCFGR2, EMx can be accessed with privileged and unprivileged access.
- When PRIVx is enabled in EXTI_PRIVCFGR2, EMx can be accessed only with privileged access. Unprivileged write to EMx is discarded, and read returns 0.

Bit 27 Reserved, must be kept at reset value.

Bits 26:0 **EMx:** CPU wake-up with interrupt mask on event input x (x = 58 to 0)

25.7.31 EXTI CPU wake-up with interrupt mask register 3 (EXTI_IMR3)

Address offset: 0x0A0

Reset value: 0x0000 0000

This register contains only bits for configurable events and direct events.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	IM77	Res.	Res.	IM74	IM73	IM72	IM71	IM70	IM69	IM68	Res.	IM66	IM65	IM64
		rw			rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **IM77:** CPU wake-up with interrupt mask on event input 77

0: Wake-up and interrupt from input event x is masked.

1: Wake-up and interrupt from input event x is unmasked. CPU wake-up event and configurable event interrupt generated.

- When SECx is disabled in EXTI_SECCFGR3, IMx can be accessed with non-secure and secure access.
- When SECx is enabled in EXTI_SECCFGR3, IMx can be accessed only with secure access. Nonsecure write to IMx is discarded. Nonsecure read returns 0.
- When PRIVx is disabled in EXTI_PRIVCFGR3, IMx can be accessed with privileged and unprivileged access.
- When PRIVx is enabled EXTI_PRIVCFGR3, IMx can be accessed only with privileged access. Unprivileged write to IMx is discarded and read returns 0.

Bits 12:11 Reserved, must be kept at reset value.

Bits 10:4 **IMx:** CPU wake-up with interrupt mask on event input x (x = 74 to 68)

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **IMx:** CPU wake-up with interrupt mask on event input x (x = 66 to 64)

25.7.32 EXTI CPU wake-up with event mask register 3 (EXTI_EMR3)

Address offset: 0x0A4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	EM77	Res.	Res.	EM74	EM73	EM72	EM71	EM70	EM69	EM68	Res.	EM66	EM65	EM64
		rw			rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **EM77:** CPU wake-up with event on event input 77

0: Wake-up with event request from line x is masked.

1: Wake-up with event request from line x is unmasked.

- When SECx is disabled in EXTI_SECCFGR3, EMx can be accessed with non-secure and secure access.
- When SECx is enabled in EXTI_SECCFGR3, EMx can be accessed only with secure access. Nonsecure write to EMx is discarded. Nonsecure read returns 0.
- When PRIVx is disabled in EXTI_PRIVCFGR3, EMx can be accessed with privileged and unprivileged access.
- When PRIVx is enabled in EXTI_PRIVCFGR3, EMx can be accessed only with privileged access. Unprivileged write to EMx is discarded, and read returns 0.

Bits 12:11 Reserved, must be kept at reset value.

Bits 10:4 **EMx:** CPU wake-up with interrupt mask on event input x (x = 74 to 68)

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **EMx:** CPU wake-up with interrupt mask on event input x (x = 66 to 64)

25.7.33 EXTI register map

Table 143. EXTI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	EXTI_RTSR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT21	RT20	Res.	Res.	Res.	Res.	RT[15:0]															
	Reset value											0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	EXTI_FTSR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT21	FT20	Res.	Res.	Res.	Res.	FT[15:0]															
	Reset value											0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	EXTI_SWIER1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI21	SWI20	Res.	Res.	Res.	Res.	SWI[15:0]															
	Reset value											0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	EXTI_RPR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIF21	RPIF20	Res.	Res.	Res.	Res.	RPIF[15:0]															
	Reset value											0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	EXTI_FPR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPIF21	FPIF20	Res.	Res.	Res.	Res.	FPIF[15:0]															
	Reset value											0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	EXTI_ SECCFGR1	SEC[31:17]															Res.	SEC[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	EXTI_ PRIVCFGR1	PRIV[31:17]															Res.	PRIV[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	Reserved	Reserved																															
0x020	EXTI_RTSR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT56	Res.	RT54	Res.	Res.	RT51	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT40	RT39	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value								0		0			0												0	0						
0x024	EXTI_FTSR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT56	Res.	FT54	Res.	Res.	FT51	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT40	FT39	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value								0		0			0												0	0						
0x028	EXTI_SWIER2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI56	Res.	SW54	Res.	Res.	SW51	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SW40	SW39	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value								0		0			0												0	0						
0x02C	EXTI_RPR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIF56	Res.	RPIF54	Res.	Res.	RPIF51	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIF40	RPIF39	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value								0		0			0												0	0						
0x030	EXTI_FPR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPIF56	Res.	FPIF54	Res.	Res.	FPIF51	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPIF40	FPIF39	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value								0		0			0												0	0						
0x034	EXTI_ SECCFGR2	SEC[63:60]				Res.	SEC[58:32]																										
	Reset value	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	EXTI_ PRIVCFGR2	PRIV[63:60]				Res.	PRIV[58:32]																										
	Reset value	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	Reserved	Reserved																															

Table 143. EXTI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x040	EXTI_RTSR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RT74	RT73	RT72	RT71	RT70	RT69	RT68	Res.	RT66	Res.	Res.
	Reset value																						0	0	0	0	0	0	0		0		
0x044	EXTI_FTSR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FT74	FT73	FT72	FT71	FT70	FT69	FT68	Res.	FT66	Res.	Res.
	Reset value																						0	0	0	0	0	0	0		0		
0x048	EXTI_SWIER3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWI74	SWI73	SWI72	SWI71	SWI70	SWI69	SWI68	Res.	SWI66	Res.	Res.
	Reset value																						0	0	0	0	0	0	0		0		
0x04C	EXTI_RPR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPIF74	RPIF73	RPIF72	RPIF71	RPIF70	RPIF69	RPIF68	Res.	RPIF66	Res.	Res.
	Reset value																						0	0	0	0	0	0	0		0		
0x050	EXTI_FPR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FPIF74	FPIF73	FPIF72	FPIF71	FPIF70	FPIF69	FPIF68	Res.	FPIF66	Res.	Res.
	Reset value																						0	0	0	0	0	0	0		0		
0x054	EXTI_ SECCFGR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC77	Res.	Res.	SEC74	SEC73	SEC72	SEC71	SEC70	SEC69	SEC68	Res.	SEC66	SEC65	SEC64
	Reset value																			0			0	0	0	0	0	0	0		0	0	0
0x058	EXTI PRIVCFGR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV77	Res.	Res.	PRIV74	PRIV73	PRIV72	PRIV71	PRIV70	PRIV69	PRIV68	Res.	PRIV66	PRIV65	PRIV64
	Reset value																			0			0	0	0	0	0	0	0		0	0	0
0x05C	Reserved	Reserved																															
0x060	EXTI_EXTICR1	EXTI3[7:0]								EXTI2[7:0]								EXTI1[7:0]					EXTI0[7:0]										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x064	EXTI_EXTICR2	EXTI7[7:0]								EXTI6[7:0]								EXTI5[7:0]					EXTI4[7:0]										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x068	EXTI_EXTICR3	EXTI11[7:0]								EXTI10[7:0]								EXTI9[7:0]					EXTI8[7:0]										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x06C	EXTI_EXTICR4	EXTI15[7:0]								EXTI14[7:0]								EXTI13[7:0]					EXTI2[7:0]										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x070	EXTI_LOCKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GLOCK
	Reset value																																0
0x074 to 0x07C	Reserved	Reserved																															
0x080	EXTI_IMR1	IM[31:17]															Res.	IM[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084	EXTI_EMR1	EM[31:17]															Res.	EM[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08C	Reserved	Reserved																															

Table 143. EXTI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x090	EXTI_IMR2	IM[63:60]				Res.	IM[58:32]																										
	Reset value	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x094	EXTI_EMR2	EM[63:60]				Res.	EM[58:32]																										
	Reset value	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x09C	Reserved	Reserved																															
0x0A0	EXTI_IMR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IM77	Res.	Res.	IM74	IM73	IM72	IM71	IM70	IM69	IM68	Res.	IM66	IM65	IM64
	Reset value																			0			0	0	0	0	0	0	0		0	0	0
0x0A4	EXTI_EMR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EM77	Res.	Res.	EM74	EM73	EM72	EM71	EM70	EM69	EM68	Res.	EM66	EM65	EM64
	Reset value																			0			0	0	0	0	0	0	0		0	0	0

Refer to *Section 2.3* for the register boundary addresses.

26 Cyclic redundancy check calculation unit (CRC)

26.1 CRC introduction

The CRC (cyclic redundancy check) calculation unit is used to get a CRC code from 8-, 16- or 32-bit data word and a generator polynomial.

Among other applications, CRC-based techniques are used to verify data transmission or storage integrity. In the scope of the functional safety standards, they offer a means of verifying the flash memory integrity. The CRC calculation unit helps compute a signature of the software during runtime, to be compared with a reference signature generated at link time and stored at a given memory location.

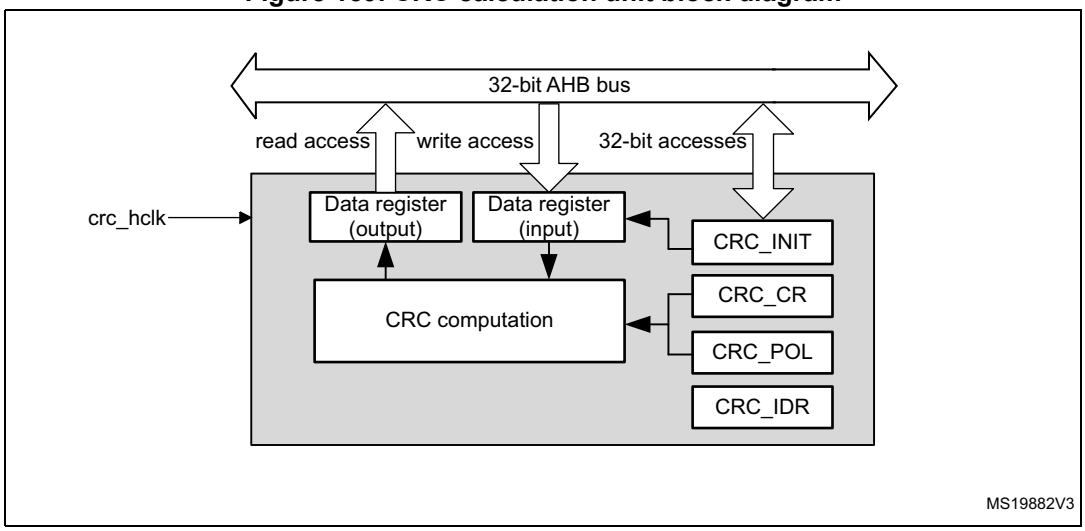
26.2 CRC main features

- Uses CRC-32 (Ethernet) polynomial: 0x4C11DB7
  $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$
- Alternatively, uses fully programmable polynomial with programmable size (7, 8, 16, 32 bits)
- Handles 8-,16-, 32-bit data size
- Programmable CRC initial value
- Single input/output 32-bit data register
- Input buffer to avoid bus stall during calculation
- CRC computation done in 4 AHB clock cycles (HCLK) for the 32-bit data size
- General-purpose 8-bit register (can be used for temporary storage)
- Reversibility options on I/O data
- Accessed through AHB slave peripheral by 32-bit words only, with the exception of CRC_DR register that can be accessed by words, right-aligned half-words and right-aligned bytes

26.3 CRC functional description

26.3.1 CRC block diagram

Figure 159. CRC calculation unit block diagram



26.3.2 CRC internal signals

Table 144. CRC internal input/output signals

Signal name	Signal type	Description
crc_hclk	Digital input	AHB clock

26.3.3 CRC operation

The CRC calculation unit has a single 32-bit read/write data register (CRC_DR). It is used to input new data (write access), and holds the result of the previous CRC calculation (read access).

Each write operation to the data register creates a combination of the previous CRC value (stored in CRC_DR) and the new one. CRC computation is done on the whole 32-bit data word or byte by byte depending on the format of the data being written.

The CRC_DR register can be accessed by word, right-aligned half-word and right-aligned byte. When the RTYPE_IN bit is set in CRC_CR, non-word accesses to CRC_DR register are ignored. For the other registers only 32-bit accesses are allowed.

The duration of the computation depends on input data width:

- 4 AHB clock cycles for 32 bits
- 2 AHB clock cycles for 16 bits
- 1 AHB clock cycles for 8 bits

An input buffer allows a second data to be immediately written without waiting for any wait-states due to the previous CRC calculation.

The data size can be dynamically adjusted to minimize the number of write accesses for a given number of bytes. For instance, a CRC for 5 bytes can be computed with a word write followed by a byte write.

The input data can be reversed to manage the various endianness schemes. The reversing operation can be performed on 8 bits, 16 bits and 32 bits depending on the REV_IN[1:0] bits in the CRC_CR register. If RTYPE_IN bit is set in this register, the input reversing operation can be performed with a byte or half-word granularity, but not with single bit granularity.

For example, with RTYPE_IN cleared, 0x1A2B3C4D input data are used for CRC calculation as:

- 0x58D43CB2 with bit-reversal done by byte (REV_IN[1:0] = 01)
- 0xD458B23C with bit-reversal done by half-word (REV_IN[1:0] = 10)
- 0xB23CD458 with bit-reversal done on the full word (REV_IN[1:0] = 11)

With RTYPE_IN set, 0x1A2B3C4D input data are used for CRC calculation as:

- 0x3C4D1A2B with half-word reversal done by word (REV_IN[1:0] = 01)
- 0x4D3C2B1A with byte-reversal done by word (REV_IN[1:0] = 10)

The output data can also be reversed by setting the REV_OUT[1:0] bits in the CRC_CR register. If RTYPE_OUT is set in this register, the output reversing operation is performed with a byte or half-word granularity, but not with single bit granularity.

With RTYPE_OUT cleared, the operation is done at bit level. For example, 0x11223344 output data are converted to 0x22CC4488.

With RTYPE_OUT set, 0x11223344 output data are converted as:

- 0x33441122 with half-word reversal done by word (REV_IN[1:0] = 01)
- 0x44332211 with byte-reversal done by word (REV_IN[1:0] = 10)

The CRC calculator can be initialized to a programmable value using the RESET control bit in the CRC_CR register (the default value is 0xFFFFFFFF).

The initial CRC value can be programmed with the CRC_INIT register. The CRC_DR register is automatically initialized upon CRC_INIT register write access.

The CRC_IDR register can be used to hold a temporary value related to CRC 7-, 8-, 16-, and 32-bit calculations. It is not affected by the RESET bit in the CRC_CR register.

Polynomial programmability

The polynomial coefficients are fully programmable through the CRC_POL register, and the polynomial size can be configured to be 7-, 8-, 16-, or 32-bits by programming the POLYSIZE[1:0] bits in the CRC_CR register. Even polynomials are not supported.

Note: The type of an even polynomial is $X+X^2+..+X^n$, while the type of an odd polynomial is $1+X+X^2+..+X^n$.

If the CRC data is less than 32-bit, its value can be read from the least significant bits of the CRC_DR register.

To obtain a reliable CRC calculation, the change on-fly of the polynomial value or size can not be performed during a CRC calculation. As a result, if a CRC calculation is ongoing, the application must either reset it or perform a CRC_DR read before changing the polynomial.

The default polynomial value is the CRC-32 (Ethernet) polynomial: 0x4C11DB7.

26.4 CRC registers

The CRC_DR register can be accessed by words, right-aligned half-words and right-aligned bytes when RTYPE bit is cleared in CRC_CR. When RTYPE is set, only word accesses are allowed. For the other registers only 32-bit accesses are allowed.

26.4.1 CRC data register (CRC_DR)

Address offset: 0x00

Reset value: 0xFFFF FFFF

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

Bits 31:0 **DR[31:0]:** Data register bits

This register is used to write new data to the CRC calculator.

It holds the previous CRC calculation result when it is read.

If the data size is less than 32 bits, the least significant bits are used to write/read the correct value.

26.4.2 CRC independent data register (CRC_IDR)

Address offset: 0x04

Reset value: 0x0000 0000

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

Bits 31:0 **IDR[31:0]**: General-purpose 32-bit data register bits

These bits can be used as a temporary storage location for four bytes.

This register is not affected by CRC resets generated by the RESET bit in the CRC_CR register

26.4.3 CRC control register (CRC_CR)

Address offset: 0x08

Reset value: 0x0000 0000

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

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **RTYPE_OUT**: Reverse type output
This bit controls the reversal granularity of the output data.
0: Bit level output
1: Byte or half-word level output

Bit 9 **RTYPE_IN**: Reverse type input
This bit controls the reversal granularity of the input data.
0: Bit level input
1: Byte or half-word level input

Bits 8:7 **REV_OUT[1:0]**: Reverse output data
This bitfield controls the reversal of the bit order of the output data.
00: Bit order not affected (RTYPE_OUT = 0 or 1)
01: Bit-reversed output format (RTYPE_OUT = 0) or half-word reversal done by word (RTYPE_OUT = 1)
10: Bit order not affected (RTYPE_OUT = 0) or byte reversal done by word (RTYPE_OUT = 1)
11: Bit order not affected (RTYPE_OUT = 0 or 1)

Bits 6:5 **REV_IN[1:0]**: Reverse input data
This bitfield controls the reversal of the bit order of the input data
00: Bit order not affected (RTYPE_IN = 0 or 1)
01: Bit reversal done by byte (RTYPE_IN = 0) or half-word reversal done by word (RTYPE_IN = 1)
10: Bit reversal done by half-word (RTYPE_IN = 0) or byte reversal done by word (RTYPE_IN = 1)
11: Bit reversal done by word (RTYPE_IN = 0) or bit order is not affected (RTYPE_IN = 1)

Bits 4:3 **POLYSIZE[1:0]**: Polynomial size

These bits control the size of the polynomial.

00: 32 bit polynomial

01: 16 bit polynomial

10: 8 bit polynomial

11: 7 bit polynomial

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **RESET**: RESET bit

This bit is set by software to reset the CRC calculation unit and set the data register to the value stored in the CRC_INIT register. This bit can only be set, it is automatically cleared by hardware.

26.4.4 CRC initial value (CRC_INIT)

Address offset: 0x10

Reset value: 0xFFFF FFFF

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

Bits 31:0 **CRC_INIT[31:0]**: Programmable initial CRC value

This register is used to write the CRC initial value.

26.4.5 CRC polynomial (CRC_POL)

Address offset: 0x14

Reset value: 0x04C1 1DB7

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

Bits 31:0 **POL[31:0]**: Programmable polynomial

This register is used to write the coefficients of the polynomial to be used for CRC calculation.

If the polynomial size is less than 32 bits, the least significant bits have to be used to program the correct value.

26.4.6 CRC register map

Table 145. CRC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	CRC_DR	DR[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x04	CRC_IDR	IDR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	CRC_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTYPE_OUT	RTYPE_IN	REV_OUT[1:0]		REV_IN[1:0]		POLYSIZE[1:0]		Res.	Res.	RESET
	Reset value																						0	0	0	0	0	0	0	0			0
0x10	CRC_INIT	CRC_INIT[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x14	CRC_POL	POL[31:0]																															
	Reset value	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	1	0	0	0	1	1	1	0	1	1	0	1	1	0	1	1	1

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

27 Flexible memory controller (FMC)

The flexible memory controller (FMC) includes three memory controllers:

- the NOR/PSRAM memory controller
- the NAND memory controller
- the synchronous DRAM (SDRAM/Mobile LPSDR SDRAM) controller

27.1 FMC main features

The FMC functional block makes the interface with: synchronous and asynchronous static memories, SDRAM memories, and NAND flash memory. Its main purposes are:

- to translate AXI transactions into the appropriate external device protocol
- to meet the access time requirements of the external memory devices

All external memories share the addresses, data and control signals with the controller. Each external device is accessed by means of a unique Chip Select. The FMC performs only one access at a time to an external device.

The main features of the FMC are the following:

- Interface with static-memory mapped devices including:
  - Static random access memory (SRAM)
  - NOR flash memory/One NAND flash memory
  - PSRAM (4 memory subregions)
  - Ferroelectric RAM (FRAM)
  - NAND flash memory with ECC hardware to check up to 8 Kbytes of data
- Interface with synchronous DRAM (SDRAM/Mobile LPSDR SDRAM) memories
- Burst mode support for faster access to synchronous devices such as NOR flash memory, PSRAM and SDRAM)
- Programmable continuous clock output for asynchronous and synchronous accesses
- 8-,16- or 32-bit wide data bus
- Independent chip-select control for each memory region
- Independent configuration for each memory region
- Write enable and byte lane select outputs for use with PSRAM, SRAM and SDRAM devices
- External asynchronous wait control

At startup the FMC pins must be configured by the user application. The FMC input/output pins which are not used by the application can be used for other purposes.

After reset, the FMC is disabled. Once all the used memory controllers are configured, the FMC must be enabled by setting the FMCEN bit.

The FMC registers that define the external device type and associated characteristics are set at boot time and do not change until the next reset or power-up. If some registers have to be re-configured, the FMC must be disabled (FMCEN bit set to 0x0). However, only few bits can be changed on the fly:

- FMCEN bit
- ECCEN and PBEN bits in FMC_PCR register
- FMC_CSQICR and FMC_CSQIER register bits
- FMC command sequencer registers and FMC BCH registers
- WEN bit in the FMC_PCR register
- MODE[2:0], DS1/DS2, NRFS[3:0] and MRD[13:0] bits in FMC_SDCMR register
- REIE and CRE bits in the FMC_SDRTR register.

If some parameters have to be modified while the FMC is enabled, the FMC must be first disabled so that no further access is allowed to any memory controller during register modification. Once all needed configurations are updated, enable the FMC as follows:

1. Disable FMC by resetting the FMCEN bit.
2. Wait ISOST[1:0] bits in FMC_SR register to be set to 0b11.
3. Wait until PEF flag is set in the FMC_SR register.
4. Program the registers with new values.
5. Enable the FMC by setting the FMCEN bit.

When the SDRAM controller is available, if the SDCLK clock ratio or refresh rate has to be modified after the initialization phase, the following procedure must be followed.

1. Stop by software, the AXI traffic from/to the SDRAM devices only.
2. Put all the SDRAM devices in self refresh and wait for the CMDOK command completion flag.
3. Change the clock ratio to the new value.
4. Send a normal mode command (NRM) to exit from SR and wait for the CMDOK command completion flag.
5. Send a load mode register command (LMR) as needed to update devices timing (optional)
6. Resume the AXI traffic to SDRAM devices.

27.2 FMC implementation

Table 146. FMC features

Modes/features	FMC
NAND flash chip-select outputs	4
NAND flash ready/busy	1
NAND/NOR flash with common WAIT pin	No

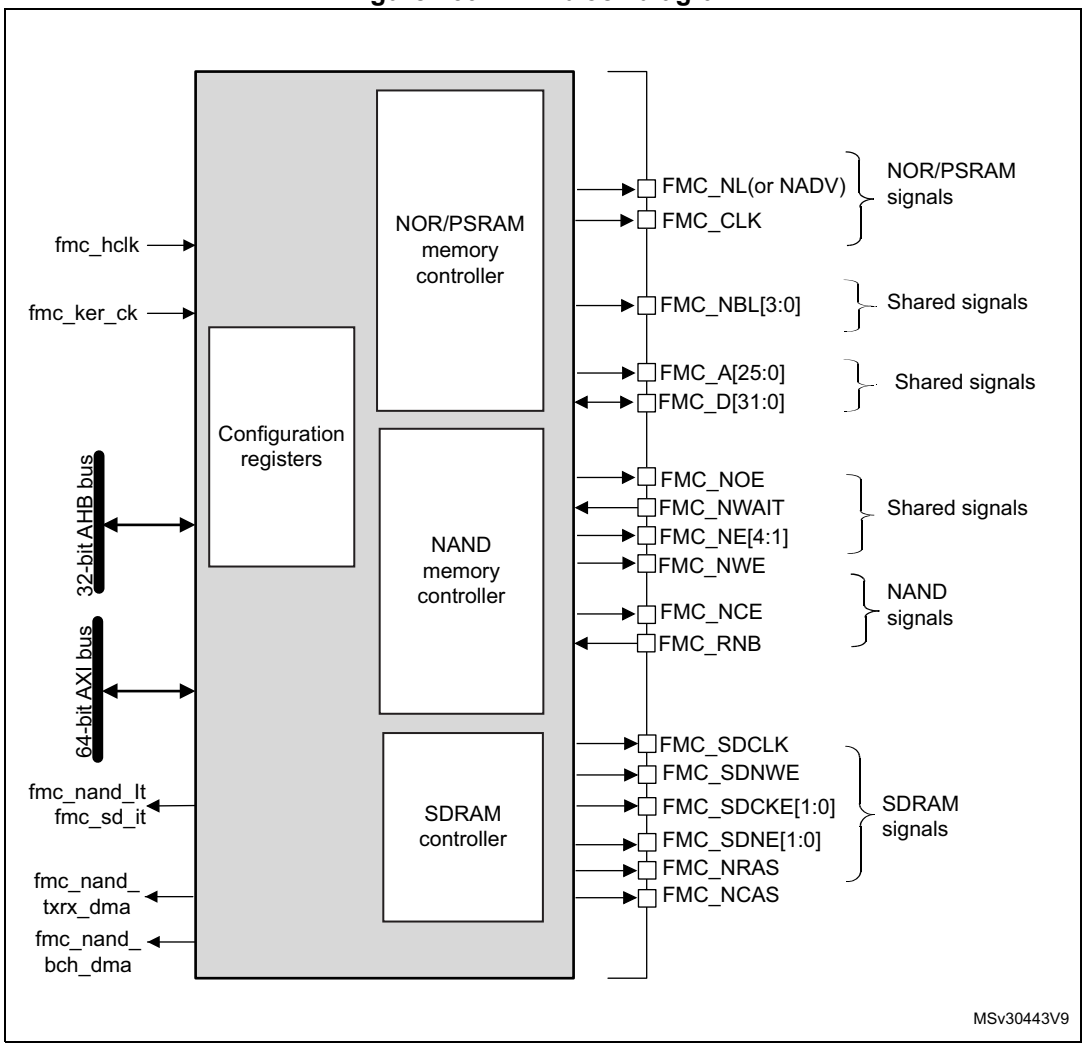
27.3 FMC block diagram

The FMC consists of the following main blocks:

- The NOR flash/PSRAM/SRAM controller
- The NAND controller
- The SDRAM controller
- The AXI interface
- The AHB interface (including the FMC configuration registers)

The block diagram is shown in the below figure.

Figure 160. FMC block diagram



27.4 FMC internal signals

Table 147 gives the list of FMC internal signals. FMC pins (or external signals) are described in *Section 27.8.1: External memory interface signals*.

Table 147. FMC internal signals

Names	Signal type	Description
fmc_nand_it	Digital output	FMC NAND flash interrupt
fmc_sd_it	Digital output	FMC SDRAM interrupt
fmc_ker_ck	Digital input	FMC kernel clock
fmc_hclk	Digital input	FMC interface clock
fmc_nand_txrx_dma	Digital output	DMA request for reading or writing data from/to NAND flash memory
fmc_nand_bch_dma	Digital output	DMA request for BCH results

27.5 AHB interface

The AHB slave interface allows internal CPUs to configure the FMC registers.

The AHB clock (HCLK) is the reference clock for the FMC register accesses.

The FMC registers must be accessed by words (32 bits). For non 32-bit AHB accesses, FMC does not return an AHB bus error and the register value can be corrupted.

27.6 AXI interface

The AXI slave interface allows internal CPUs and other bus master peripherals to access the external memories.

AXI transactions are translated into the external device protocol. As the AXI data bus is 64-bit wide, the AXI transactions might be split into several consecutive 32- or 16- or 8-bit accesses according to data size accesses.

The FMC generates an AXI slave error when one of the following conditions is met:

- When reading or writing to an FMC memory region (memory region 1 to 4) which is not enabled
- When reading or writing to any FMC memory region while the FMC is disabled (FMCEN bit is reset)
- When reading or writing by the command sequencer to NAND memory region while it is disabled
- When reading or writing to the NOR flash memory region while the MBKEN bit is reset in the FMC_BCRx register
- When writing to a write-protected SDRAM device (WP bit set in the SDRAM_SDCRx register)
- When the SDRAM address range is violated (access to a reserved address range)
- When reading or writing to an SDRAM device with the SDEN bit cleared in the FMC_SDCRx register

The FMC generates an AXI decoder error in the following conditions:

- When ADDR[31:28] address bits are not supported by the FMC memory region base address following the BMAP[1:0] bits configuration
- and when an access to an SDRAM or a NAND memory region is performed while the SDRAM or the NAND controller is not present.

The kernel clock for the FMC is the asynchronous fmc_ker_ck clock (refer the Reset and Clock control (RCC) section for details on fmc_ker_ck clock source selection).

27.6.1 Supported memories and transactions

General transaction rules

The requested AXI transaction data size can be 8-, 16-, 32- or 64-bit wide whereas the accessed external device has a fixed data width. This may lead to inconsistent transfers.

Therefore, some simple transaction rules must be followed:

- AXI transaction size and memory data size must be equal to prevent issues from occurring.
- AXI transaction size is greater than the memory size:

  In this case, the FMC splits the AXI transaction into smaller consecutive memory accesses to meet the external data width.
- AXI transaction size is smaller than the memory size:

  The transfer may or not be consistent depending on the type of external device:
  - Accesses to devices that have the byte select feature (SRAM, ROM, PSRAM, SDRAM)

    In this case, the FMC allows read/write transactions and accesses the right data through its byte lanes NBL[3:0].

    Bytes to be written are addressed by NBL[3:0].

    All memory bytes are read (NBL[3:0] are driven low during read transaction) and the useless ones are discarded.
  - Accesses to devices that do not have the byte select feature (NOR and NAND flash memories)

    This situation occurs when a byte access is requested to a 16-bit wide flash memory. Since the device cannot be accessed in Byte mode (only 16-bit words can be read/written from/to the flash memory), write transactions are not allowed while read transactions are allowed (the controller reads the entire 16-bit memory word and uses only the required byte).

Wrap support for NOR flash/PSRAM

Synchronous memories must be configured in Linear burst mode of undefined length as not all masters can issue wrap transactions.

If a master generates a wrap transaction:

- The read operation is split into two linear burst transactions.
- The write operation is split into two linear burst transactions.

Configuration registers

The FMC can be configured through a set of registers. Refer to *Section 27.8.6*, for a detailed description of the NOR flash/PSRAM controller registers. Refer to *Section 27.9.9*, for a detailed description of the NAND flash registers and to *Section 27.10.5* for a detailed description of the SDRAM controller registers.

27.7 External device address mapping

From the FMC point of view, the external memory is divided into fixed-size memory regions of 256 Mbytes each (see *Table 148*):

- Memory region 1 is used to address up to 4 NOR flash memories or PSRAM devices. This region is split into four NOR/PSRAM subregions with four dedicated Chip Selects, as follows:
  - Subregion 1 - NOR/PSRAM 1
  - Subregion 2 - NOR/PSRAM 2
  - Subregion 3- NOR/PSRAM 3
  - Subregion 4 - NOR/PSRAM 4

Note: The Memory region 1 type attribute must be declared as 'device'.

- Memory region 5 and 6 are used to address SDRAM devices (one device per region).

For each memory region, the type of memory to be used can be configured by the user application through the Configuration register.

Table 148. FMC memory regions (default mapping)

Address	Memory region	Supported memory type
0x6000 0000 - 0x6FFF FFFF	Memory region 1	NOR flash memory/PSRAM/SRAM
0xC000 0000 - 0xCFFF FFFF	Memory region 5	SDRAM device 1
0xD000 0000 - 0xDFFF FFFF	Memory region 6	SDRAM device 2

The mapping of FMC memory regions can be modified through the BMAP[1:0] bits in FMC_BCR1 register. *Table 149* shows the configuration to swap the NOR/PSRAM and SDRAM memory regions or remap SDRAM device 1 and 2. Thus allowing to access the SDRAM devices at two different address mapping.

Table 149. FMC memory region remap using BMAP[1:0]

FMC memory region	BMAP[1:0] = 00 (Default mapping)	BMAP[1:0] = 01	BMAP[1:0] = 10
Memory region 1	NOR/PSRAM/SRAM	SDRAM device 1	NOR/PSRAM/SRAM
Memory region 5	SDRAM device 1	NOR/PSRAM/SRAM	SDRAM device 1
Memory region 6	SDRAM device 2	SDRAM device 2	SDRAM device 2

27.7.1 NOR/PSRAM address mapping

ADDR[27:26] bits are used to select one of the four memory subregions as shown in *Table 150*.

Table 150. NOR/PSRAM subregion selection

ADDR[27:26](1)	Selected subregion
00	Subregion 1 - NOR/PSRAM 1
01	Subregion 2 - NOR/PSRAM 2
10	Subregion 3- NOR/PSRAM 3
11	Subregion 4- NOR/PSRAM 4

1. ADDR are internal address lines that are translated to external memory.

The ADDR[25:0] bits contain the external memory address. Since ADDR is a byte address whereas the memory is addressed at word level, the address actually issued to the memory varies according to the memory data width, as shown in the following table.

Table 151. NOR/PSRAM External memory address

Memory width(1)	Data address issued to the memory	Maximum memory capacity (bits)
8-bit	ADDR[25:0]	64 Mbytes * 8 = 512 Mbit
16-bit	ADDR[25:1] >> 1	64 Mbytes/2 * 16 = 512 Mbit
32-bit	ADDR[25:2] >> 2	64 Mbytes/4 * 32 = 512 Mbit

1. In case of a 16-bit external memory width, the FMC internally uses ADDR[25:1] to generate the address for external memory FMC_A[24:0]. In case of a 32-bit memory width, the FMC internally uses ADDR[25:2] to generate the external address.
Whatever the external memory width, FMC_A[0] must be connected to external memory address A[0].

27.7.2 NAND flash memory address mapping

The 4 Kbyte NAND flash address bits are mapped according to *Table 152*.

Table 152. NAND access memory map

Address field	FMC signal name	Function	Detailed description
ADDR[10]	NCE	NAND chip enable selection	00: NCE1
ADDR[9:8]	-	Reserved	-
ADDR[7]	-	Timing selection	0: Common space, timing defined by FMC_PMEM 1: Attribute space, timing defined by FMC_PATT
ADDR[6]	-	Reserved	-
ADDR[5]	A[17]	Address (ALE)	00: Data section 01: Command section 10: Address section 11: reserved
ADDR[4](1)	A[16]	Command (CLE)	

1. Four address LSBs are don't care: each section has 16 bytes mapped to the alias region.

The application software uses the 3 sections to access the NAND flash memory:

- **To send a command to NAND flash memory**, the software must write the command value to any memory location in the command section.
- **To specify the NAND flash address that must be read or written**, the software must write the address value to any memory location in the address section. Since an address can be 4 or 5 bytes long (depending on the actual memory size), several consecutive write operations to the address section are required to specify the full address.
- **To read or write data,** the software reads or writes the data from/to any memory location in the data section.

Since the NAND flash memory automatically increments addresses, there is no need to increment the address of the data section to access consecutive memory locations.

Note: The NAND region has to be accessed as a device and not as memory. Write accesses to command and address sections must be done in Byte mode and aligned with the section base address (offset = 0).

Read and Write accesses to NAND data section must be in Byte mode for 8-bit NAND flash memories and Byte or Half-word mode for 16-bit NAND flash memories.

Read and Write bursts must start at section base address (offset = 0) and must not cross the 16-byte section boundary.

27.7.3 SDRAM address mapping

Two SDRAM regions are available as indicated in *Table 153*.

Table 153. SDRAM device selection

Selected device	Control register	Timing register
SDRAM device 1	FMC_SDCR1	FMC_SDTR
SDRAM device 2	FMC_SDCR2	

The following table shows SDRAM mapping for an 13-bit row, a 11-bit column and 4 internal bank configuration.

Table 154. SDRAM address mapping

Memory width(1)	Internal bank	Row address	Column address(2)	Maximum memory capacity (Mbytes)
8-bit	ADDR[25:24]	ADDR[23:11]	ADDR[10:0]	64 Mbytes: 4 * 8K * 2K
16-bit	ADDR[26:25]	ADDR[24:12]	ADDR[11:1]	128 Mbytes: 4 * 8K * 2K * 2
32-bit	ADDR[27:26]	ADDR[25:13]	ADDR[12:2]	256 Mbytes: 4 * 8K * 2K * 4

1. When interfacing with a 16-bit memory, the FMC internally uses the ADDR[11:1] internal address lines to generate the external address. When interfacing with a 32-bit memory, the FMC internally uses ADDR[12:2] lines to generate the external address. Whatever the memory width, FMC_A[0] has to be connected to the SDRAM A[0].
2. The AutoPrecharge is not supported. However, FMC_A[10] must be connected to SDRAM A[0].

The ADDR[27:0] bits are translated to external SDRAM address depending on the SDRAM controller configuration:

- Row size:11, 12 or 13 bits
- Column size: 8, 9, 10 or 11 bits
- Number of internal banks: two or four internal banks

The following tables show the SDRAM address mapping versus the SDRAM controller configuration.

Table 155. SDRAM address mapping with 8-bit data bus width(1)(2)

Row size configuration	ADDR (Internal address lines)																										
	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
11-bit row size configuration	Res.							Bank [1:0]		Row[10:0]											Column[7:0]							
	Res.						Bank [1:0]			Row[10:0]											Column[8:0]							
	Res.					Bank [1:0]				Row[10:0]											Column[9:0]							
	Res.				Bank [1:0]					Row[10:0]											Column[10:0]							
12-bit row size configuration	Res.						Bank [1:0]			Row[11:0]											Column[7:0]							
	Res.					Bank [1:0]				Row[11:0]											Column[8:0]							
	Res.				Bank [1:0]					Row[11:0]											Column[9:0]							
	Res.			Bank [1:0]						Row[11:0]											Column[10:0]							
13-bit row size configuration	Res.					Bank [1:0]				Row[12:0]											Column[7:0]							
	Res.				Bank [1:0]					Row[12:0]											Column[8:0]							
	Res.			Bank [1:0]						Row[12:0]											Column[9:0]							
	Res.		Bank [1:0]							Row[12:0]											Column[10:0]							

1. BANK[1:0] are the Bank Address BA[1:0]. When only 2 internal banks are used, BA1 must always be cleared.
2. Access to Reserved (Res.) address range generates an AXI slave error.

Table 156. SDRAM address mapping with 16-bit data bus width(1)(2)

Row size Configuration	ADDR (address Lines)																										
	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
11-bit row size configuration	Res.						Bank [1:0]		Row[10:0]										Column[7:0]									BM0(3)
	Res.						Bank [1:0]		Row[10:0]										Column[8:0]									BM0
	Res.						Bank [1:0]		Row[10:0]										Column[9:0]									BM0
	Res.						Bank [1:0]		Row[10:0]										Column[10:0]									BM0
12-bit row size configuration	Res.						Bank [1:0]		Row[11:0]										Column[7:0]									BM0
	Res.						Bank [1:0]		Row[11:0]										Column[8:0]									BM0
	Res.						Bank [1:0]		Row[11:0]										Column[9:0]									BM0
	Res.						Bank [1:0]		Row[11:0]										Column[10:0]									BM0
13-bit row size configuration	Res.						Bank [1:0]		Row[12:0]										Column[7:0]									BM0
	Res.						Bank [1:0]		Row[12:0]										Column[8:0]									BM0
	Res.						Bank [1:0]		Row[12:0]										Column[9:0]									BM0
	Res.						Bank [1:0]		Row[12:0]										Column[10:0]									BM0

1. BANK[1:0] are the Bank Address BA[1:0]. When only 2 internal banks are used, BA1 must always be cleared.
2. Access to Reserved space (Res.) generates an AXI Slave error.
3. BM0: is the byte mask for 16-bit access.

Table 157. SDRAM address mapping with 32-bit data bus width(1)(2)

Row size configuration	ADDR (address Lines)																										
	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
11-bit row size configuration	Res.					Bank [1:0]		Row[10:0]											Column[7:0]								BM[1:0](3)	
	Res.					Bank [1:0]		Row[10:0]											Column[8:0]								BM[1:0	
	Res.					Bank [1:0]		Row[10:0]											Column[9:0]								BM[1:0	
	Res.					Bank [1:0]		Row[10:0]											Column[10:0]								BM[1:0	

Table 157. SDRAM address mapping with 32-bit data bus width(1)(2) (continued)

Row size configuration	ADDR (address Lines)																											
	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
12-bit row size configuration	Res.				Bank [1:0]		Row[11:0]												Column[7:0]								BM[1:0	
	Res.				Bank [1:0]		Row[11:0]												Column[8:0]								BM[1:0	
	Res.				Bank [1:0]		Row[11:0]												Column[9:0]								BM[1:0	
	Res.				Bank [1:0]		Row[11:0]												Column[10:0]								BM[1:0	
13-bit row size configuration	Res.				Bank [1:0]		Row[12:0]												Column[7:0]								BM[1:0	
	Res.				Bank [1:0]		Row[12:0]												Column[8:0]								BM[1:0	
	Res.				Bank [1:0]		Row[12:0]												Column[9:0]								BM[1:0	
	Bank [1:0]				Row[12:0]														Column[10:0]								BM[1:0	

1. BANK[1:0] are the Bank Address BA[1:0]. When only 2 internal banks are used, BA1 must always be cleared.
2. Access to Reserved space (Res.) generates an AXI slave error.
3. BM[1:0]: is the byte mask for 32-bit access.

27.8 NOR flash/PSRAM controller

The FMC generates the appropriate signal timings to drive the following types of memories:

- Asynchronous SRAM, FRAM and ROM
  - 8 bits
  - 16 bits
  - 32 bits
- PSRAM (CellularRAM™)
  - Asynchronous mode
  - Burst mode for synchronous accesses with configurable option to split burst access when crossing boundary page for CRAM 1.5.
  - Multiplexed or non-multiplexed
- NOR flash memory
  - Asynchronous mode
  - Burst mode for synchronous accesses
  - Multiplexed or non-multiplexed

The FMC outputs a unique Chip Select signal, NE[4:1], per memory region. All the other signals (addresses, data and control) are shared.

The FMC supports a wide range of devices through a programmable timings among which:

- Programmable wait-states (up to 15)
- Programmable bus turnaround cycles (up to 15)
- Programmable output enable and write enable delays (up to 15)
- Independent read and write timings and protocol to support the widest variety of memories and timings
- Programmable continuous clock (FMC_CLK) output.

The FMC output Clock (FMC_CLK) is a submultiple of the fmc_ker_ck clock. It can be delivered to the selected external device either during synchronous accesses only or during asynchronous and synchronous accesses depending on the CCKEN bit configuration in the FMC_BCR1 register:

- If the CCLKEN bit is reset, the FMC generates the clock (FMC_CLK) only during synchronous accesses (Read/write transactions).
- If the CCLKEN bit is set, the FMC generates a continuous clock during asynchronous and synchronous accesses. To generate the FMC_CLK continuous clock, memory region 1 must be configured in synchronous mode (see *Section 27.8.6: NOR/PSRAM controller registers*). Since the same clock is used for all synchronous memories, when a continuous output clock is generated and synchronous accesses are performed, the AXI data size has to be the same as the memory data width (MWID[1:0]) otherwise the FMC_CLK frequency is changed depending on AXI data transaction (refer to *Section 27.8.5: Synchronous transactions* for FMC_CLK divider ratio formula).

The size of each memory region is fixed and equal to 64 Mbytes. Each memory region is configured through dedicated registers (see *Section 27.8.6: NOR/PSRAM controller registers*).

The programmable memory parameters include access times (see *Table 158*) and support for wait management (for PSRAM and NOR flash accessed in burst mode).

Table 158. Programmable NOR/PSRAM access parameters

Parameter	Function	Access mode	Unit	Min.	Max.
Address setup	Duration of the address setup phase	Asynchronous	FMC clock cycle (fmc_ker_ck)	0	15
Address hold	Duration of the address hold phase	Asynchronous, muxed I/Os	FMC clock cycle (fmc_ker_ck)	1	15
NBL setup	Duration of the Byte lanes setup phase	Asynchronous	FMC clock cycle (fmc_ker_ck)	0	3
Data setup	Duration of the data setup phase	Asynchronous	FMC clock cycle (fmc_ker_ck)	1	256
Data hold	Duration of the data hold phase	Asynchronous	FMC clock cycle (fmc_ker_ck)	0	3
Bust turn	Duration of the bus turnaround phase	Asynchronous and synchronous read	FMC clock cycle (fmc_ker_ck)	0	15

Table 158. Programmable NOR/PSRAM access parameters (continued)

Parameter	Function	Access mode	Unit	Min.	Max.
Clock divide ratio	Number of FMC clock cycles (fmc_ker_ck) to build one memory clock cycle (CLK)	Synchronous	FMC clock cycle (fmc_ker_ck)	2	16
Data latency	Number of clock cycles to issue to the memory before the first data of the burst	Synchronous	Memory clock cycle (fmc_ker_ck)	2	17

27.8.1 External memory interface signals

Table 159, *Table 160* and *Table 161* list the signals that are typically used to interface with NOR flash memory, SRAM and PSRAM.

Note: The prefix "N" identifies the signals which are active low.

NOR flash memory, non-multiplexed I/Os

Table 159. Non-multiplexed I/O NOR flash memory

FMC signal name	I/O	Function
CLK	O	Clock (for synchronous access)
A[25:0]	O	Address bus
D[31:0]	I/O	Bidirectional data bus
NE[x]	O	Chip Select, x = 1 to 4
NOE	O	Output enable
NWE	O	Write enable
NL(=NADV)	O	Latch enable (this signal is called address valid, NADV, by some NOR flash devices)
NWAIT	I	NOR flash wait input signal to the FMC

The maximum capacity is 512 Mbits (26 address lines).

NOR flash memory, 16-bit multiplexed I/Os

Table 160. 16-bit multiplexed I/O NOR flash memory

FMC signal name	I/O	Function
CLK	O	Clock (for synchronous access)
A[25:16]	O	Address bus
AD[15:0]	I/O	16-bit multiplexed, bidirectional address/data bus (the 16-bit address A[15:0] and data D[15:0] are multiplexed on the databus)
NE[x]	O	Chip Select, x = 1 to 4
NOE	O	Output enable
NWE	O	Write enable

Table 160. 16-bit multiplexed I/O NOR flash memory (continued)

FMC signal name	I/O	Function
NL(=NADV)	O	Latch enable (this signal is called address valid, NADV, by some NOR flash devices)
NWAIT	I	NOR flash wait input signal to the FMC

The maximum capacity is 512 Mbits.

PSRAM/SRAM/FRAM, non-multiplexed I/Os

Table 161. Non-multiplexed I/Os PSRAM/SRAM

FMC signal name	I/O	Function
CLK	O	Clock (only for PSRAM synchronous access)
A[25:0]	O	Address bus
D[31:0]	I/O	Data bidirectional bus
NE[x]	O	Chip Select, x = 1 to 4 (called NCE by PSRAM (CellularRAM™, CRAM))
NOE	O	Output enable
NWE	O	Write enable
NL(= NADV)	O	Address valid only for PSRAM input (memory signal name: NADV)
NWAIT	I	PSRAM wait input signal to the FMC
NBL[3:0]	O	Byte lane output (upper and lower byte enable)

The maximum capacity is 512 Mbits.

PSRAM, 16-bit multiplexed I/Os

Table 162. 16-bit multiplexed I/O PSRAM

FMC signal name	I/O	Function
CLK	O	Clock (for synchronous access)
A[25:16]	O	Address bus
AD[15:0]	I/O	16-bit multiplexed, bidirectional address/data bus (the 16-bit address A[15:0] and data D[15:0] are multiplexed on the databus)
NE[x]	O	Chip Select, x = 1 to 4 (called NCE by PSRAM (CellularRAM™, CRAM))
NOE	O	Output enable
NWE	O	Write enable
NL(= NADV)	O	Address valid PSRAM input (memory signal name: NADV)
NWAIT	I	PSRAM wait input signal to the FMC
NBL[3:0]	O	Byte lane output (upper and lower byte enable)

The maximum capacity is 512 Mbits (26 address lines).

27.8.2 Supported memories and transactions

Table 163 below shows an example of the supported devices, access modes and transactions when the memory data bus is 16-bit wide for NOR flash memory, PSRAM and SRAM. The transactions not allowed (or not supported) by the FMC are shown in gray in this example.

Table 163. NOR flash/PSRAM: Example of supported memories and transactions

Device	Mode	R/W	AXI data size	Memory data size	Allowed/ not allowed	Comments
NOR flash (muxed I/Os and non-muxed I/Os)	Asynchronous	R	8	16	Y	-
	Asynchronous	W	8	16	N	-
	Asynchronous	R	16	16	Y	-
	Asynchronous	W	16	16	Y	-
	Asynchronous	R	32	16	Y	Split into 2 FMC accesses
	Asynchronous	W	32	16	Y	Split into 2 FMC accesses
	Asynchronous	R	64	16	Y	Split into 4 FMC accesses
	Asynchronous	W	64	16	Y	Split into 4 FMC accesses
	Asynchronous page	R	-	16	N	Mode is not supported
	Synchronous	R	8	16	N	-
	Synchronous	R	16	16	Y	-
	Synchronous	R	32	16	Y	-
	Synchronous	R	64	16	Y	Split into 4 FMC accesses

Table 163. NOR flash/PSRAM: Example of supported memories and transactions

Device	Mode	R/W	AXI data size	Memory data size	Allowed/ not allowed	Comments
PSRAM (multiplexed I/Os and non-multiplexed I/Os)	Asynchronous	R	8	16	Y	-
	Asynchronous	W	8	16	Y	Use of byte lanes NBL[3:0]
	Asynchronous	R	16	16	Y	-
	Asynchronous	W	16	16	Y	-
	Asynchronous	R	32	16	Y	Split into 2 FMC accesses
	Asynchronous	W	32	16	Y	Split into 2 FMC accesses
	Asynchronous	R	64	16	Y	Split into 4 FMC accesses
	Asynchronous	W	64	16	Y	Split into 4 FMC accesses
	Asynchronous page	R	-	16	N	Mode is not supported
	Synchronous	R	8	16	N	-
	Synchronous	R	16	16	Y	-
	Synchronous	R	32/64	16	Y	-
	Synchronous	W	8	16	Y	Use of byte lanes NBL[3:0]
	Synchronous	W	16/32/64	16	Y	-
SRAM and ROM	Asynchronous	R	8 / 16	16	Y	-
	Asynchronous	W	8 / 16	16	Y	Use of byte lanes NBL[3:0]
	Asynchronous	R	32	16	Y	Split into 2 FMC accesses
	Asynchronous	W	32	16	Y	Split into 2 FMC accesses
	Asynchronous	R	64	16	Y	Split into 4 FMC accesses
	Asynchronous	W	64	16	Y	Split into 4 FMC accesses

27.8.3 General timing rules

Signals synchronization

- All controller output signals change on the rising edge of the fmc_ker_ck clock
- In synchronous mode (read or write), all output signals change on the rising edge of fmc_ker_ck clock.
- NBL signals

  They are applicable to memories which support byte enable.
  - The NBL signals are driven low:

    from the start of a read memory transfer and optionally earlier according to NBLSET, when the device requires some setup time from NBL to NCE,

    for bytes to be written during data transfer and according to the first data bytes during the address transfer.
  - The NBL signals are driven high two FMC_CLK clock cycles after the completion of the read or write transfer.

27.8.4 NOR flash/PSRAM controller asynchronous transactions

Asynchronous static memories (NOR flash, PSRAM, SRAM, FRAM)

- Signals are synchronized by the internal clock fmc_ker_ck. This clock is not issued to the memory.
- The FMC always samples the data before deasserting the Chip Select signal NE. This guarantees that the memory data hold timing constraint is met (minimum chip enable high to data transition is usually 0 ns).
- If the extended mode is enabled (EXTMOD bit is set in the FMC_BCRx register), up to four extended modes (A, B, C and D) are available. It is possible to mix A, B, C and D modes for read and write operations. For example, read operation can be performed in mode A and write in mode B. The extended mode is not compatible with the muxed mode (EXTMOD bit must be kept cleared in muxed mode).
- If the extended mode is disabled (EXTMOD bit is reset in the FMC_BCRx register), the FMC can operate in mode 1 or mode 2 as follows:
  - Mode 1 is the default mode when SRAM/PSRAM memory type is selected (MTYP[1:0] = 0x0 or 0x01 in the FMC_BCRx register)
  - Mode 2 is the default mode when NOR memory type is selected (MTYP[1:0] = 0x10 in the FMC_BCRx register).

Mode 1 - SRAM/FRAM/PSRAM (CRAM)

The next figures show the read and write transactions for the supported modes followed by the required configuration of FMC_BCRx, and FMC_BTRx/FMC_BWTRx registers.

Figure 161. Mode 1 read access waveforms

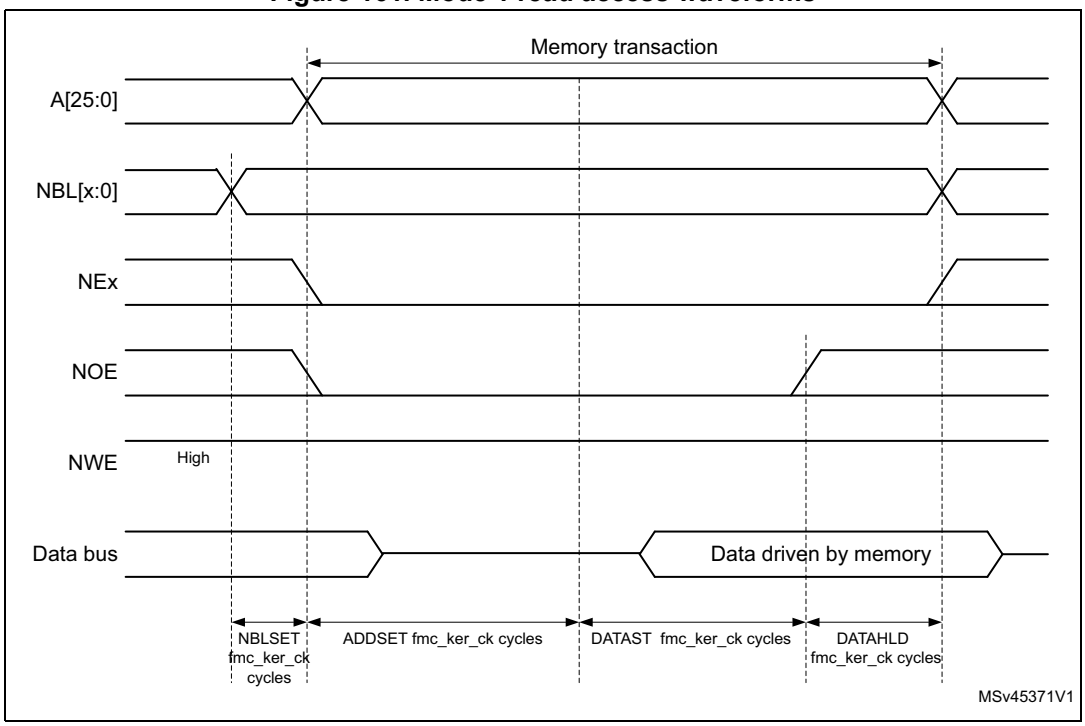
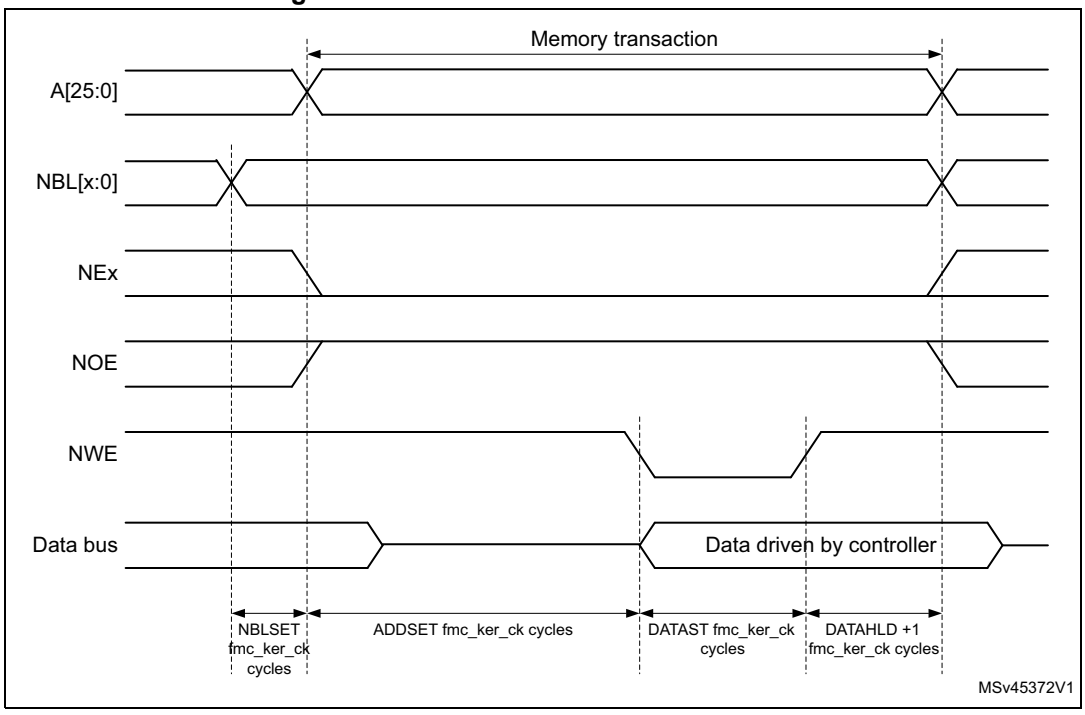


Figure 162. Mode 1 write access waveforms



The DATAHLD time at the end of the read and write transaction guarantee the address and data hold time after the NOE/NWE rising edge. For write, the DATAST value must be greater than zero (DATAST > 0).

Table 164. FMC_BCRx bitfields (mode 1)

Bit number	Bit name	Value to set
31	Reserved	0x0
30-24	Reserved	0x00
23:22	NBLSET[1:0]	As needed
21:20	CSCOUNT	As needed
19	CBURSTRW	0x0 (no effect in asynchronous mode)
18:16	CPSIZE	0x0 (no effect in asynchronous mode)
15	ASYNCWAIT	Set if the memory supports this feature. Otherwise keep cleared.
14	EXTMOD	0x0
13	WAITEN	0x0 (no effect in asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don’t care
10	Reserved	0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1
6	Reserved	0x0
5-4	MWID[1:0]	As needed
3-2	MTYP	As needed, exclude 0x2 (NOR flash memory)
1	MUXE	0x0
0	MBKEN	0x1

Table 165. FMC_BTRx bitfields (mode 1)

Bit number	Bit name	Value to set
31:30	DATAHLD	Duration of the Data hold phase (DATAHLD fmc_ker_ck cycles) for read accesses.
29-28	ACCMOD	Don’t care
27-24	DATLAT	Don’t care
23-20	CLKDIV	Don’t care
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15-8	DATAST	Duration of the second access phase (DATAST fmc_ker_ck cycles).

Table 165. FMC_BTRx bitfields (mode 1) (continued)

Bit number	Bit name	Value to set
7-4	ADDHLD	Don't care
3-0	ADDSET	Duration of the first access phase (ADDSET fmc_ker_ck cycles). Minimum value for ADDSET is 0.

Mode A - SRAM/FRAM/PSRAM (CRAM) OE toggling

Figure 163. Mode A read access waveforms

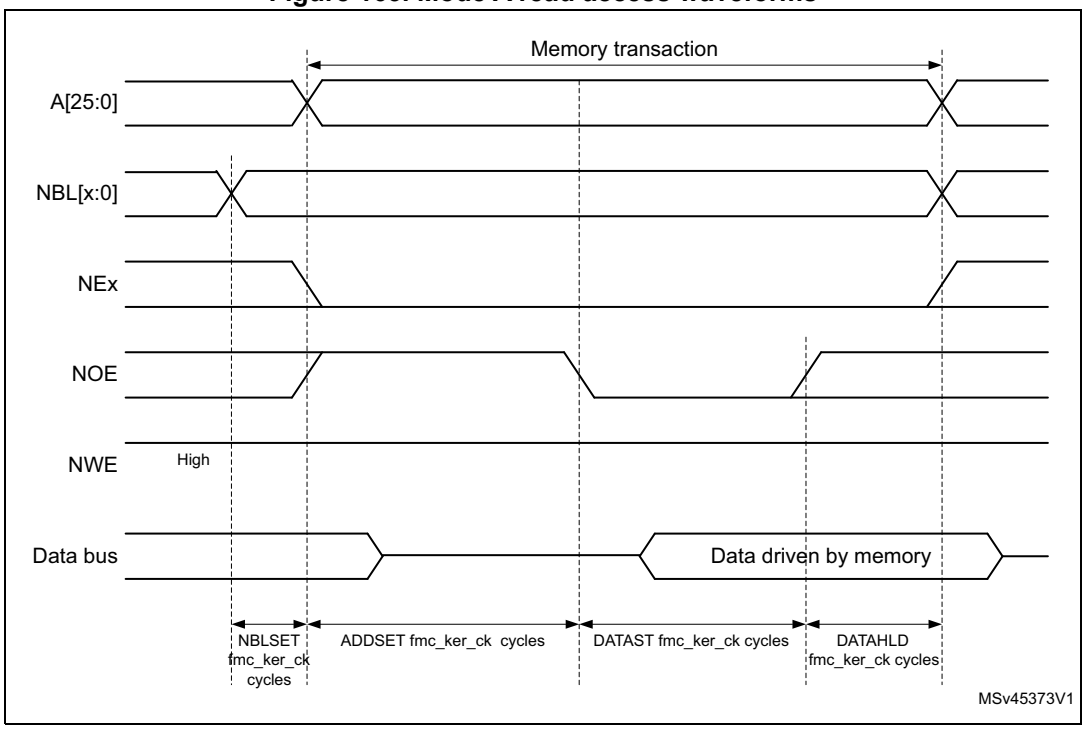
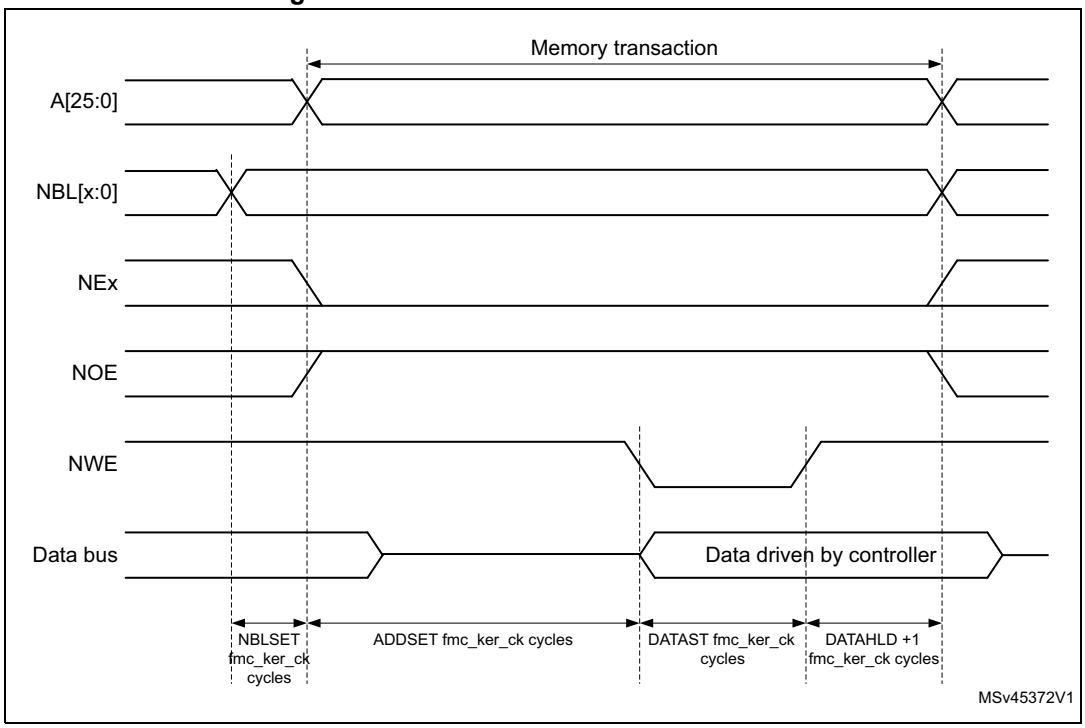


Figure 164. Mode A write access waveforms



The differences compared with mode 1 are the toggling of NOE and the independent read and write timings.

Table 166. FMC_BCRx bitfields (mode A)

Bit number	Bit name	Value to set
31	Reserved	0x0
30-24	Reserved	0x00
23:22	NBLSET[1:0]	As needed
21:20	CSCOUNT	As needed
19	CBURSTRW	0x0 (no effect in asynchronous mode)
18:16	CPSIZE	0x0 (no effect in asynchronous mode)
15	ASYNCWAIT	Set if the memory supports this feature. Otherwise keep cleared.
14	EXTMOD	0x1
13	WAITEN	0x0 (no effect in asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don’t care
10	Reserved	0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0

Table 166. FMC_BCRx bitfields (mode A) (continued)

Bit number	Bit name	Value to set
7	Reserved	0x1
6	Reserved	0x0
5-4	MWID[1:0]	As needed
3-2	MTYP	As needed, exclude 0x2 (NOR flash memory)
1	MUXEN	0x0
0	MBKEN	0x1

Table 167. FMC_BTRx bitfields (mode A)

Bit number	Bit name	Value to set
31:30	DATAHLD	Duration of the Data hold phase (DATAHLD fmc_ker_ck cycles for read accesses).
29-28	ACCMOD	0x0
27-24	DATLAT	Don’t care
23-20	CLKDIV	Don’t care
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15-8	DATAST	Duration of the second access phase (DATAST fmc_ker_ck cycles) for read accesses.
7-4	ADDHLD	Don’t care
3-0	ADDSET	Duration of the first access phase (ADDSET fmc_ker_ck cycles) for read accesses. Minimum value for ADDSET is 0.

Table 168. FMC_BWTRx bitfields (mode A)

Bit number	Bit name	Value to set
31:30	DATAHLD	Duration of the Data hold phase (DATAHLD+1 fmc_ker_ck cycles for write accesses).
29-28	ACCMOD	0x0
27:20	Reserved	0x0
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15-8	DATAST	Duration of the second access phase (DATAST fmc_ker_ck cycles) for write accesses.
7-4	ADDHLD	Don’t care
3-0	ADDSET	Duration of the first access phase (ADDSET fmc_ker_ck cycles) for write accesses. Minimum value for ADDSET is 0.

Mode 2/B - NOR flash

Figure 165. Mode 2 and mode B read access waveforms

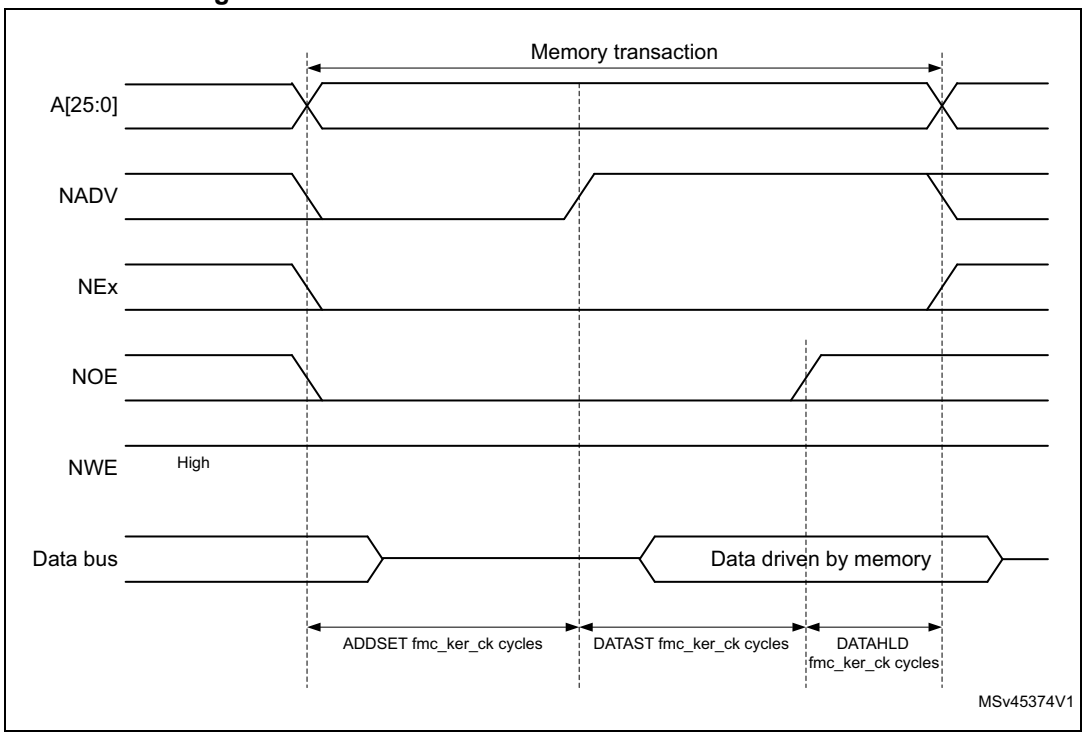


Figure 166. Mode 2 write access waveforms

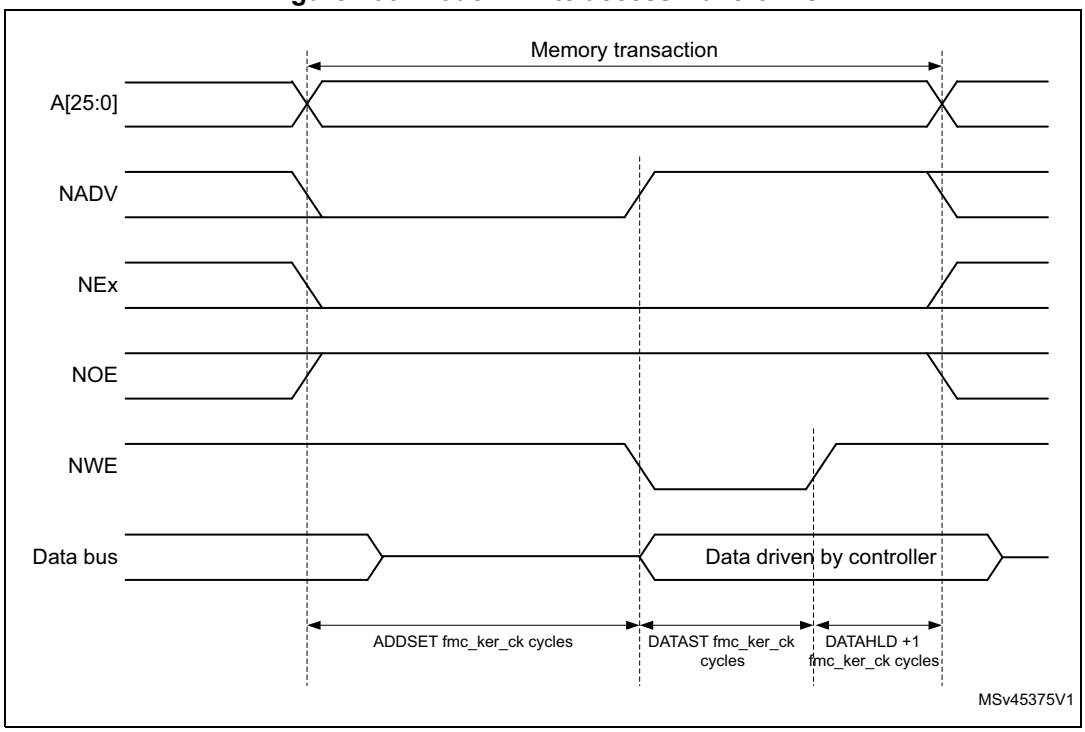
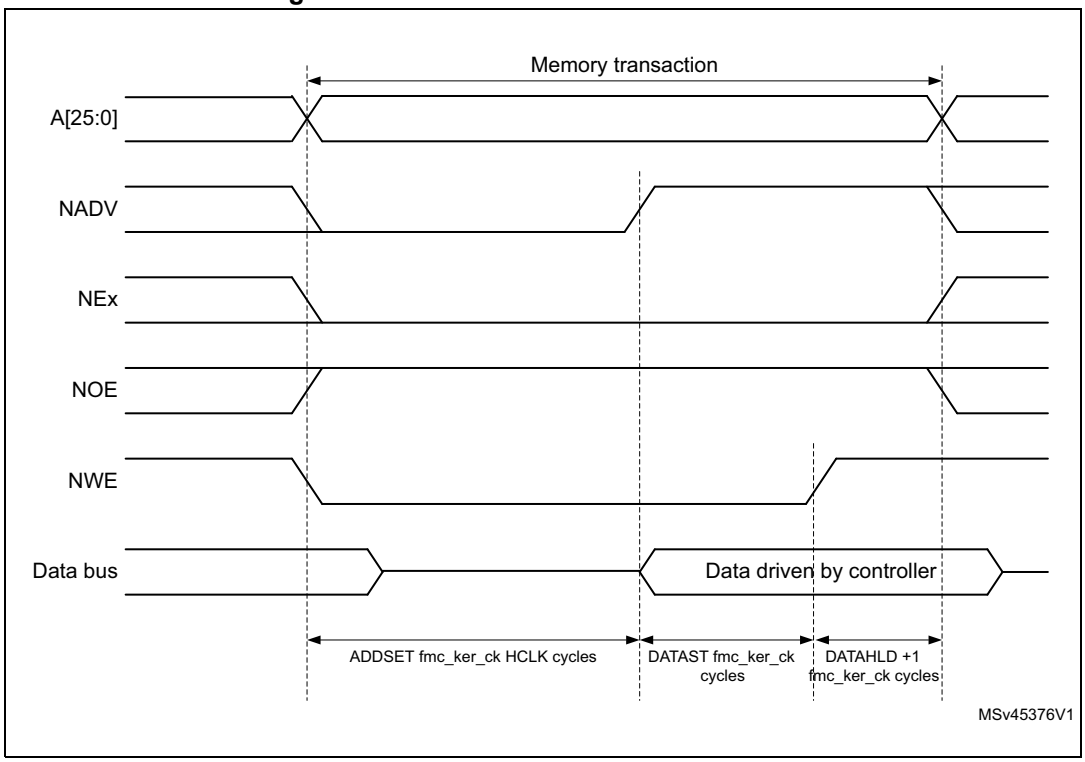


Figure 167. Mode B write access waveforms



The differences with mode 1 are the toggling of NWE and the independent read and write timings when extended mode is set (mode B).

Table 169. FMC_BCRx bitfields (mode 2/B)

Bit number	Bit name	Value to set
31	Reserved	0x0
30-24	Reserved	0x000
23:22	NBLSET[1:0]	As needed
21:20	CSCOUNT	As needed
19	CBURSTRW	0x0 (no effect in asynchronous mode)
18:16	CPSIZE	0x0 (no effect in asynchronous mode)
15	ASYNCWAIT	Set if the memory supports this feature. Otherwise keep cleared.
14	EXTMOD	0x1 for mode B, 0x0 for mode 2
13	WAITEN	0x0 (no effect in asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don’t care
10	Reserved	0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0

Table 169. FMC_BCRx bitfields (mode 2/B) (continued)

Bit number	Bit name	Value to set
7	Reserved	0x1
6	Reserved	0x0
5-4	MWID[1:0]	As needed
3-2	MTYP	0x2 (NOR flash memory)
1	MUXEN	0x0
0	MBKEN	0x1

Table 170. FMC_BTRx bitfields (mode 2/B)

Bit number	Bit name	Value to set
31:30	DATAHLD	Duration of the Data hold phase (DATAHLD fmc_ker_ck cycles for read accesses).
29-28	ACCMOD	0x1 if extended mode is set
27-24	DATLAT	Don’t care
23-20	CLKDIV	Don’t care
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15-8	DATAST	Duration of the access second phase (DATAST fmc_ker_ck cycles) for read accesses.
7-4	ADDHLD	Don’t care
3-0	ADDSET	Duration of the access first phase (ADDSET fmc_ker_ck cycles) for read accesses. Minimum value for ADDSET is 0.

Table 171. FMC_BWTRx bitfields (mode 2/B)

Bit number	Bit name	Value to set
31:30	DATAHLD	Duration of the Data hold phase (DATAHLD+1 fmc_ker_ck cycles for write accesses).
29-28	ACCMOD	0x1 if extended mode is set
27:20	Reserved	0x0
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15-8	DATAST	Duration of the access second phase (DATAST fmc_ker_ck cycles) for write accesses.
7-4	ADDHLD	Don’t care
3-0	ADDSET	Duration of the access first phase (ADDSET fmc_ker_ck cycles) for write accesses. Minimum value for ADDSET is 0.

Note: The FMC_BWTRx register is valid only if the extended mode is set (mode B), otherwise its content is don’t care.

Mode C - NOR flash - OE toggling

Figure 168. Mode C read access waveforms

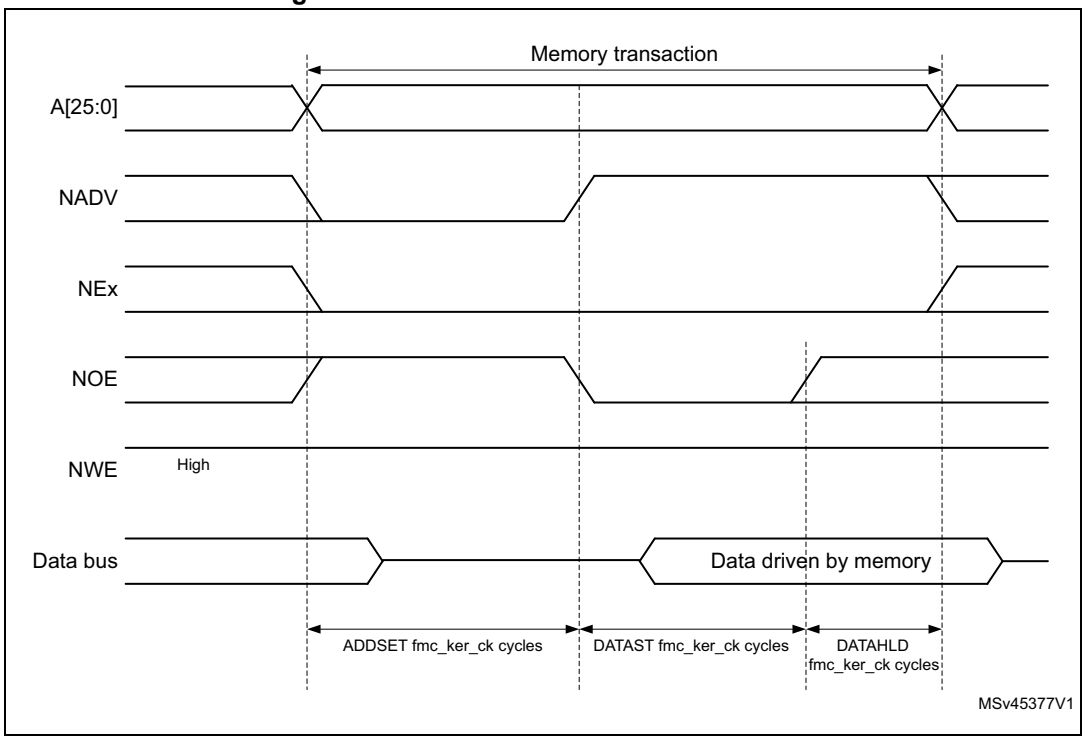
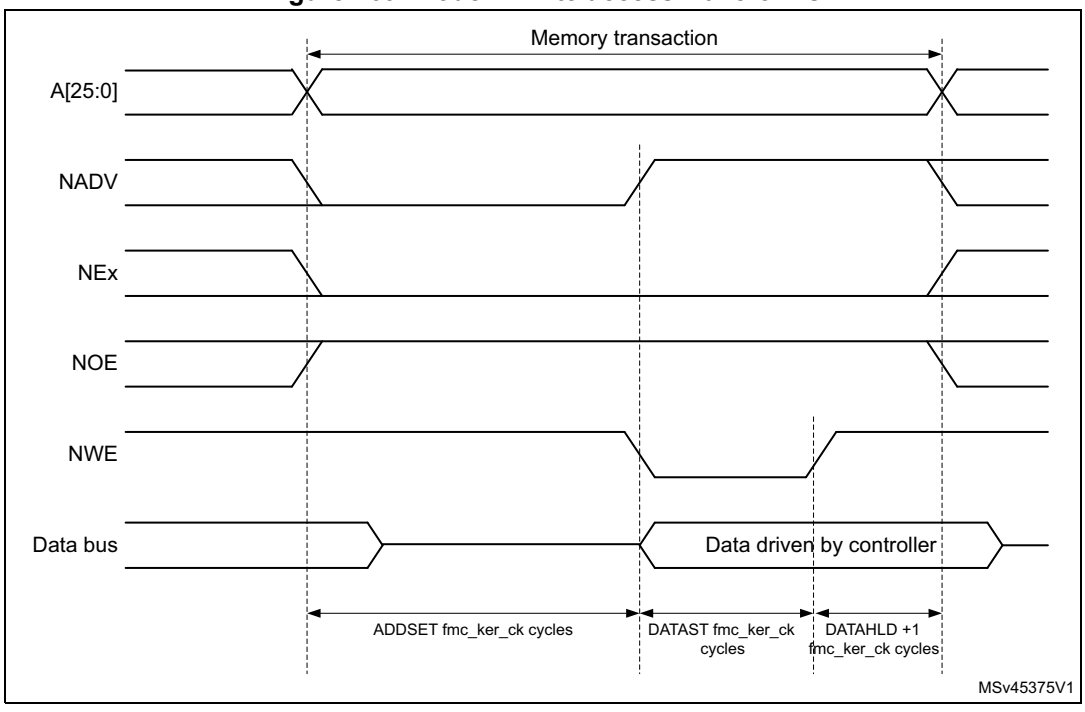


Figure 169. Mode C write access waveforms



The differences compared with mode 1 are the toggling of NOE and the independent read and write timings.

Table 172. FMC_BCRx bitfields (mode C)

Bit No.	Bit name	Value to set
31	Reserved	0x0
30-24	Reserved	0x00
23:22	NBLSET[1:0]	As needed
21:20	CSCOUNT	As needed
19	CBURSTRW	0x0 (no effect in asynchronous mode)
18:16	CPSIZE	0x0 (no effect in asynchronous mode)
15	ASYNCWAIT	Set if the memory supports this feature. Otherwise keep cleared.
14	EXTMOD	0x1
13	WAITEN	0x0 (no effect in asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don’t care
10	Reserved	0x0
9	WAITPOL	Meaningful only if bit 15 is 1
8	BURSTEN	0x0
7	Reserved	0x1
6	Reserved	0x0
5-4	MWID[1:0]	As needed
3-2	MTYP[1:0]	0x02 (NOR flash memory)
1	MUXEN	0x0
0	MBKEN	0x1

Table 173. FMC_BTRx bitfields (mode C)

Bit No.	Bit name	Value to set
31:30	DATAHLD	Duration of the Data hold phase (DATAHLD fmc_ker_ck cycles for read accesses).
29-28	ACCMOD	0x2
27-24	DATLAT	0x0
23-20	CLKDIV	0x0
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15-8	DATAST	Duration of the second access phase (DATAST fmc_ker_ck cycles) for read accesses.
7-4	ADDHLD	Don’t care
3-0	ADDSET	Duration of the first access phase (ADDSET fmc_ker_ck cycles) for read accesses. Minimum value for ADDSET is 0.

Table 174. FMC_BWTRx bitfields (mode C)

Bit No.	Bit name	Value to set
31:30	DATAHLD	Duration of the Data hold phase (DATAHLD+1 fmc_ker_ck cycles for write accesses).
29-28	ACCMOD	0x2
27:20	Reserved	0x0
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15-8	DATAST	Duration of the second access phase (DATAST fmc_ker_ck cycles) for write accesses.
7-4	ADDHLD	Don't care
3-0	ADDSET	Duration of the first access phase (ADDSET fmc_ker_ck cycles) for write accesses. Minimum value for ADDSET is 0.

Mode D - asynchronous access with extended address

Figure 170. Mode D read access waveforms

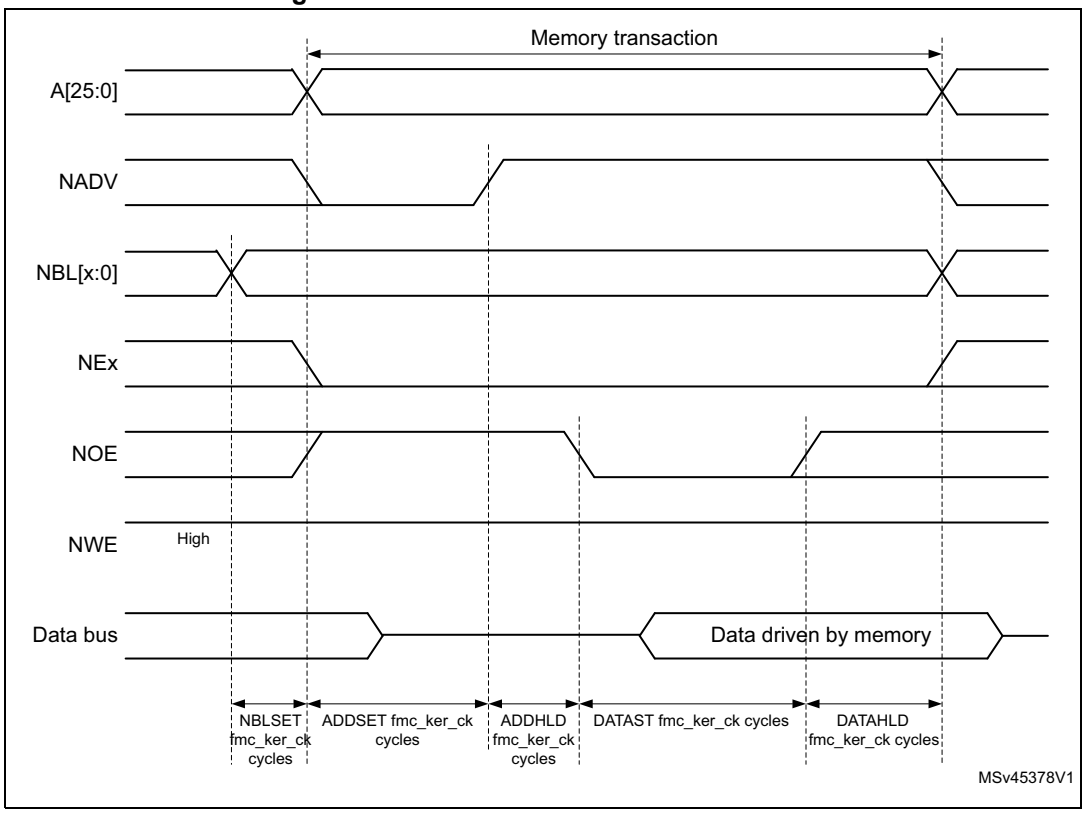
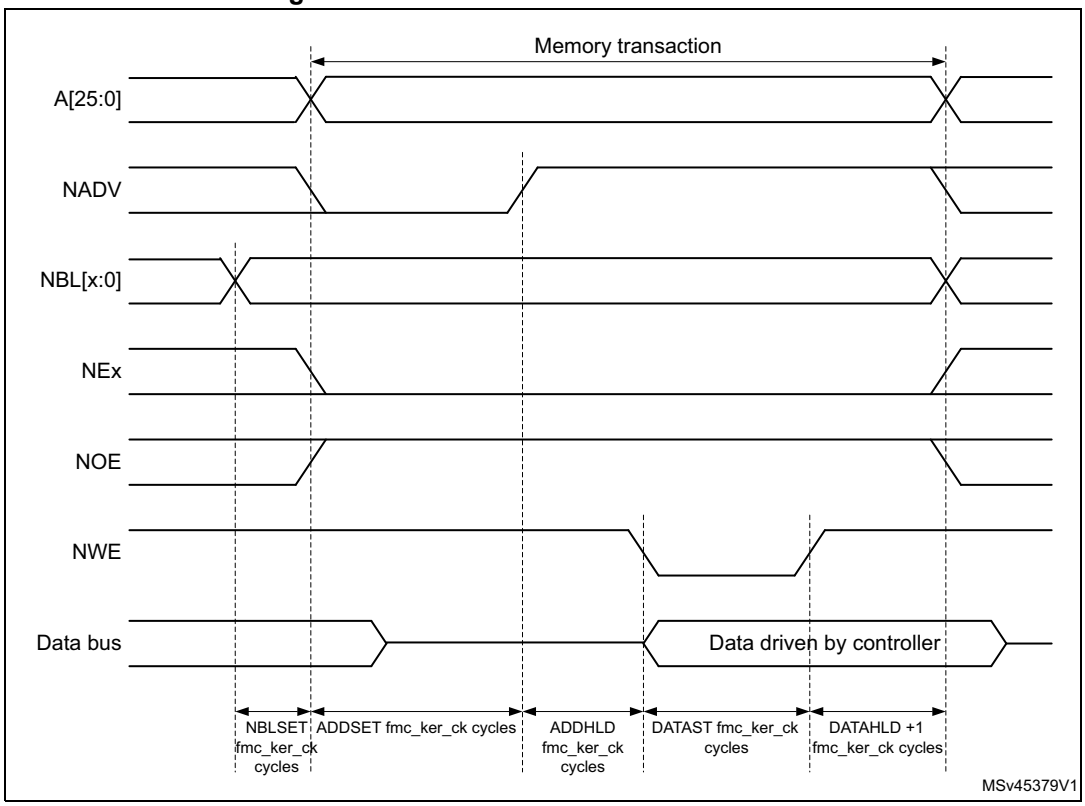


Figure 171. Mode D write access waveforms



The differences with mode 1 are the toggling of NOE that goes on toggling after NADV changes and the independent read and write timings.

Table 175. FMC_BCRx bitfields (mode D)

Bit No.	Bit name	Value to set
31	Reserved	0x0
30-24	Reserved	0x00
23:22	NBLSET[1:0]	As needed
21:20	CSCOUNT	As needed
19	CBURSTRW	0x0 (no effect in asynchronous mode)
18:16	CPSIZE	0x0 (no effect in asynchronous mode)
15	ASYNCWAIT	Set if the memory supports this feature. Otherwise keep cleared. The ASYNCWAIT feature is not supported in mode D mode.
14	EXTMOD	0x1
13	WAITEN	0x0 (no effect in asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don’t care
10	Reserved	0x0
9	WAITPOL	Meaningful only if bit 15 is 1

Table 175. FMC_BCRx bitfields (mode D) (continued)

Bit No.	Bit name	Value to set
8	BURSTEN	0x0
7	Reserved	0x1
6	Reserved	0x0
5-4	MWID[1:0]	As needed
3-2	MTYP[1:0]	As needed
1	MUXEN	0x0
0	MBKEN	0x1

Table 176. FMC_BTRx bitfields (mode D)

Bit No.	Bit name	Value to set
31:30	DATAHLD	Duration of the Data hold phase (DATAHLD fmc_ker_ck cycles for read accesses).
29-28	ACCMOD	0x3
27-24	DATLAT	Don’t care
23-20	CLKDIV	Don’t care
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15-8	DATAST	Duration of the second access phase (DATAST fmc_ker_ck cycles) for read accesses.
7-4	ADDHLD	Duration of the middle phase of the read access (ADDHLD fmc_ker_ck cycles)
3-0	ADDSET	Duration of the first access phase (ADDSET fmc_ker_ck cycles) for read accesses. Minimum value for ADDSET is 1.

Table 177. FMC_BWTRx bitfields (mode D)

Bit No.	Bit name	Value to set
31:30	DATAHLD	Duration of the Data hold phase (DATAHLD+1 fmc_ker_ck cycles for write accesses).
29-28	ACCMOD	0x3
27:20	Reserved	0x0
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15-8	DATAST	Duration of the second access phase (DATAST + 1 fmc_ker_ck cycles) for write accesses.
7-4	ADDHLD	Duration of the middle phase of the write access (ADDHLD fmc_ker_ck cycles)
3-0	ADDSET	Duration of the first access phase (ADDSET fmc_ker_ck cycles) for write accesses. Minimum value for ADDSET is 1.

Muxed mode - multiplexed asynchronous access to NOR flash memory

Figure 172. Muxed read access waveforms

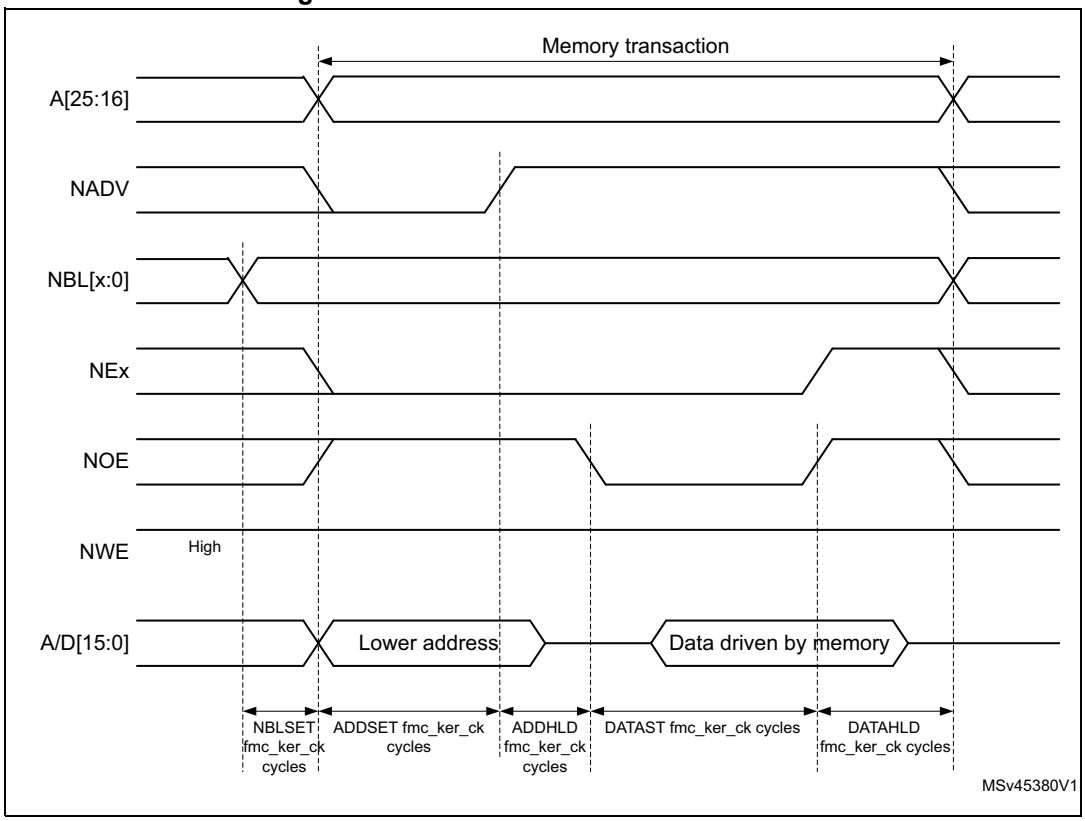
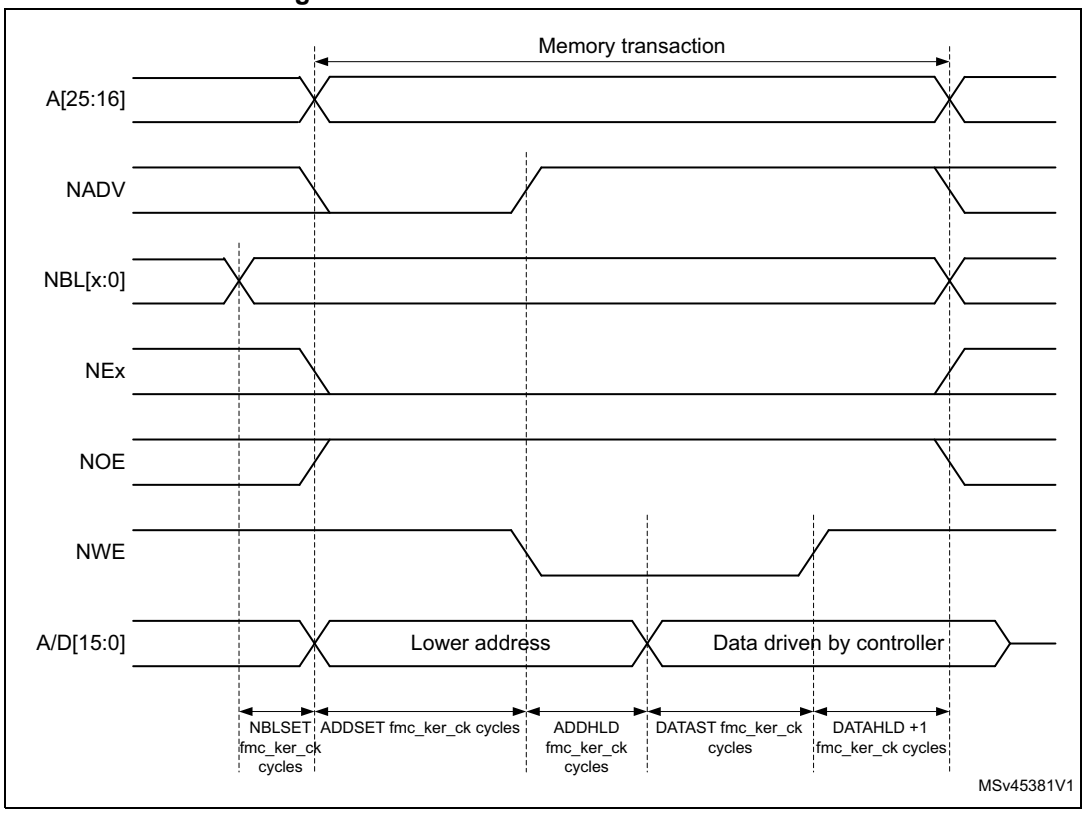


Figure 173. Muxed write access waveforms



The difference with mode D is the drive of the lower address byte(s) on the data bus.

Table 178. FMC_BCRx bitfields (muxed mode)

Bit No.	Bit name	Value to set
31	Reserved	0x0
30-24	Reserved	0x00
23:22	NBLSET[1:0]	As needed
21:20	CSCOUNT	As needed
19	CBURSTRW	0x0 (no effect in asynchronous mode)
18:16	CPSIZE	0x0 (no effect in asynchronous mode)
15	ASYNCWAIT	Set if the memory supports this feature. Otherwise keep cleared. The ASYNCWAIT feature is not supported in muxed mode.
14	EXTMOD	0x0
13	WAITEN	0x0 (no effect in asynchronous mode)
12	WREN	As needed
11	WAITCFG	Don’t care
10	Reserved	0x0
9	WAITPOL	Meaningful only if bit 15 is 1

Table 178. FMC_BCRx bitfields (muxed mode) (continued)

Bit No.	Bit name	Value to set
8	BURSTEN	0x0
7	Reserved	0x1
6	Reserved	0x0
5-4	MWID[1:0]	As needed
3-2	MTYP[1:0]	0x2 (NOR flash memory)
1	MUXEN	0x1
0	MBKEN	0x1

Table 179. FMC_BTRx bitfields (muxed mode)

Bit No.	Bit name	Value to set
31:30	DATAHLD	Duration of the Data hold phase (DATAHLD fmc_ker_ck cycles for read accesses, DATAHLD+1 fmc_ker_ck cycles for write accesses).
29-28	ACCMOD	0x0
27-24	DATLAT	Don’t care
23-20	CLKDIV	Don’t care
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15-8	DATAST	Duration of the second access phase (DATAST fmc_ker_ck cycles for read accesses and DATAST+1 fmc_ker_ck cycles for write accesses).
7-4	ADDHLD	Duration of the middle phase of the access (ADDHLD fmc_ker_ck cycles).
3-0	ADDSET	Duration of the first access phase (ADDSET fmc_ker_ck cycles). Minimum value for ADDSET is 1.

WAIT management for asynchronous accesses

If the asynchronous memory asserts the WAIT signal to indicate that it is not yet ready to accept or to provide data, the ASYNCWAIT bit has to be set in FMC_BCRx register.

If the WAIT signal is active (high or low depending on the WAITPOL bit), the second access phase (Data setup phase), programmed by the DATAST bits, is extended until WAIT becomes inactive. Unlike the data setup phase, the first access phases (Address setup and Address hold phases), programmed by the ADDSET and ADDHLD bits, are not WAIT sensitive and so they are not prolonged.

The data setup phase must be programmed so that WAIT can be detected 4 fmc_ker_ck cycles before the end of the memory transaction. The following cases must be considered:

1. The memory asserts the WAIT signal aligned to NOE/NWE which toggles:

$$DATAST \geq (4 \times FMCCLK) + max_wait_assertion_time$$

2. The memory asserts the WAIT signal aligned to NEx (or NOE/NWE not toggling):
   if

$$max_wait_assertion_time > address_phase + hold_phase$$

   then:

$$DATAST \geq (4 \times FMCCLK) + (max_wait_assertion_time - address_phase - hold_phase$$

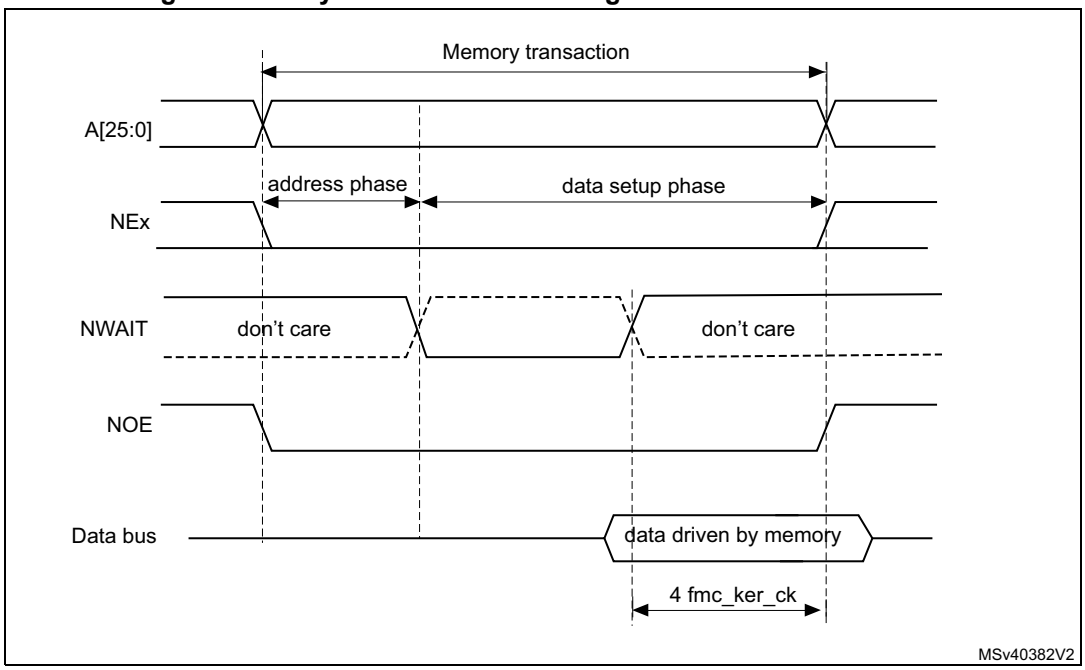
   otherwise

$$DATAST \geq 4 \times FMCCLK$$

   where max_wait_assertion_time is the maximum time taken by the memory to assert the WAIT signal once NEx/NOE/NWE is low.

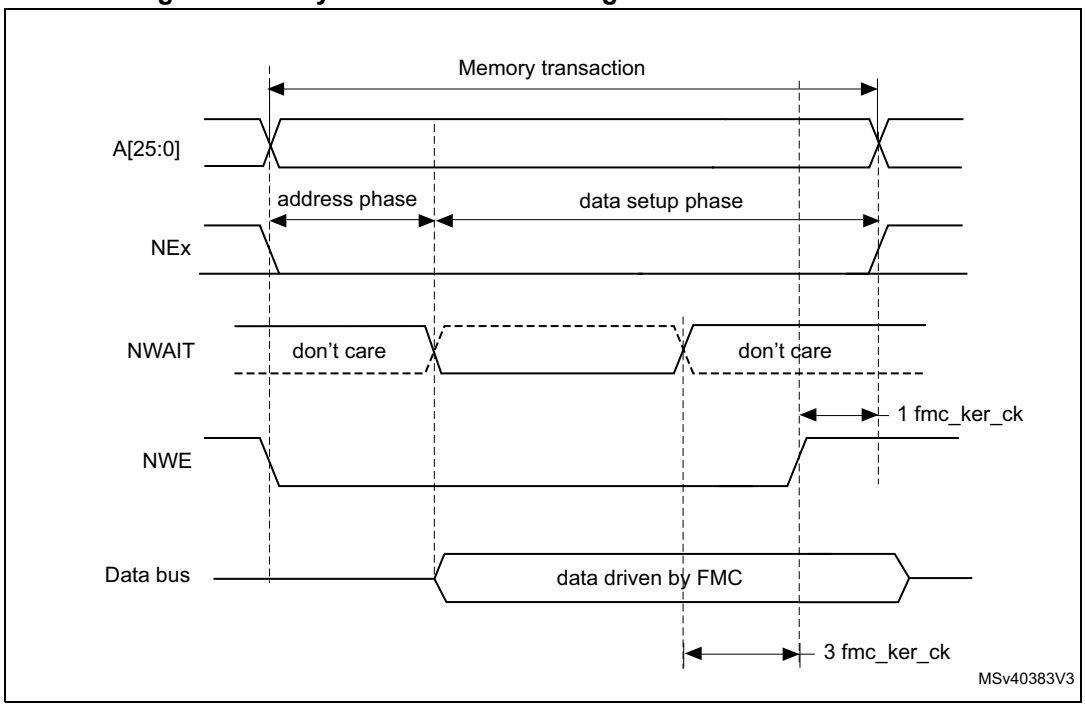
Figure 174 and *Figure 175* show the number of fmc_ker_ck clock cycles that are added to the memory access phase after WAIT is released by the asynchronous memory (independently of the above cases).

Figure 174. Asynchronous wait during a read access waveforms



1. NWAIT polarity depends on WAITPOL bit setting in FMC_BCRx register.

Figure 175. Asynchronous wait during a write access waveforms



1. NWAIT polarity depends on WAITPOL bit setting in FMC_BCRx register.

CellularRAM™ (PSRAM) refresh management

The CellularRAM™ does not allow maintaining the chip select signal (NE) low for longer than the t_{CEM} timing specified in memory device. This timing can be programmed in the FMC_CFGR register. It defines the maximum duration of the NE low pulse (expressed in fmc_ker_ck cycles) in synchronous mode for PSRAM read or write accesses.

27.8.5 Synchronous transactions

The memory clock, FMC_CLK, is a submultiple of fmc_ker_ck. It depends on the value of CLKDIV.

NOR flash memories specify a minimum time from NADV assertion to FMC_CLK high. To meet this constraint, the FMC does not issue the clock to the memory during the first internal clock cycle of the synchronous access (before NADV assertion). This guarantees that the rising edge of the memory clock occurs in the middle of the NADV low pulse.

Data latency versus NOR memory latency

The data latency is the number of cycles to wait before sampling the data. The DATLAT value must be consistent with the latency value specified in the NOR flash configuration register. The FMC does not include the clock cycle when NADV is low in the data latency count.

Caution: Some NOR flash memories include the NADV Low cycle in the data latency count, so that the exact relation between the NOR flash latency and the FMC DATLAT parameter can be either:

- NOR flash latency = (DATLAT + 2) CLK clock cycles
- or NOR flash latency = (DATLAT + 3) CLK clock cycles

Some recent memories assert NWAIT during the latency phase. In such cases DATLAT can be set to its minimum value. As a result, the FMC samples the data and waits long enough to evaluate if the data are valid. Thus the FMC detects when the memory exits latency and real data are processed.

Other memories do not assert NWAIT during latency. In this case the latency must be set correctly for both the FMC and the memory, otherwise invalid data are mistaken for good data, or valid data are lost in the initial phase of the memory access.

Single-burst transfer

When the selected memory region is configured in burst mode for synchronous accesses, if for example a single-burst transaction is requested on 16-bit memories, the FMC performs a burst transaction of length 1 (if the AXI transfer is 16 bits), or length 2 (if the AXI transfer is 32 bits) and deassert the Chip Select signal when the last data is strobed.

Such transfers are not the most efficient in terms of cycles compared to asynchronous read operations. Nevertheless, a random asynchronous access first requires to re-program the memory access mode, which altogether lasts longer.

Cross boundary page for CellularRAM™ 1.5

CellularRAM™ 1.5 does not allow burst access to cross the page boundary. The FMC allows to split automatically the burst access when the memory page size is reached by configuring the CPSIZE bits in the FMC_BCR1 register following the memory page size.

Wait management

For synchronous NOR flash memories, NWAIT is evaluated after the programmed latency period, which corresponds to (DATLAT+2) FMC_CLK clock cycles.

If NWAIT is active (low level when WAITPOL = 0, high level when WAITPOL = 1), wait-states are inserted until NWAIT is inactive (high level when WAITPOL = 0, low level when WAITPOL = 1).

When NWAIT is inactive, the data is considered valid either immediately (bit WAITCFG = 1) or on the next clock edge (bit WAITCFG = 0).

During wait-state insertion via the NWAIT signal, the controller continues to send clock pulses to the memory, keeping the Chip Select and output enable signals valid. It does not consider the data as valid.

In burst mode, there are two timing configurations for the NOR flash NWAIT signal:

- The flash memory asserts the NWAIT signal one data cycle before the wait-state (default after reset).
- The flash memory asserts the NWAIT signal during the wait-state

The FMC supports both NOR flash wait-state configurations, for each Chip Select, thanks to the WAITCFG bit in the FMC_BCRx registers (x = 0 to 3).

Figure 176. Wait configuration waveforms

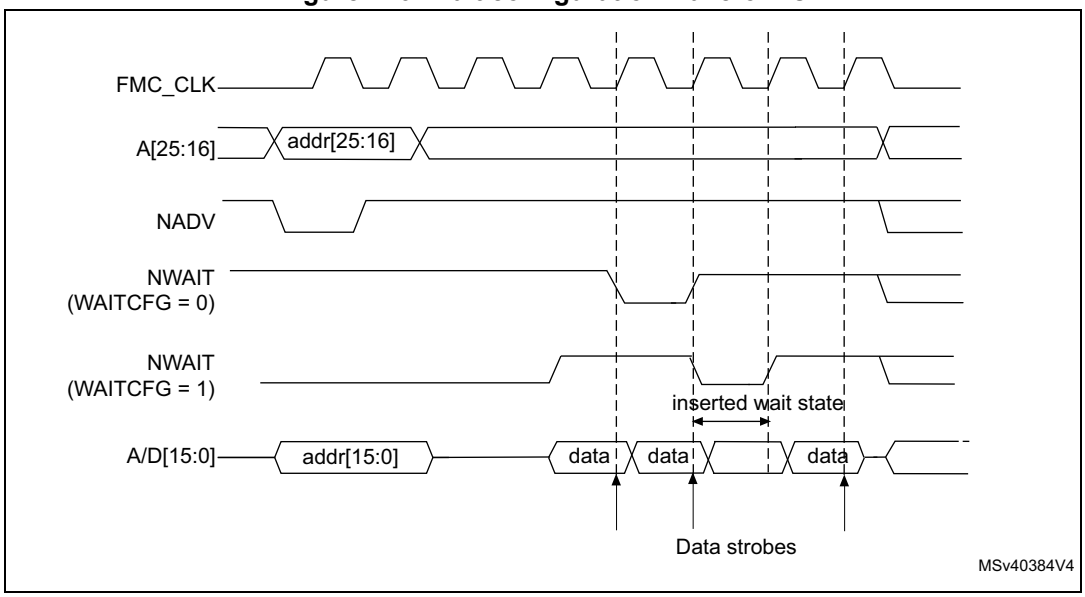


Figure 177. Synchronous multiplexed read mode waveforms - NOR, PSRAM (CRAM)

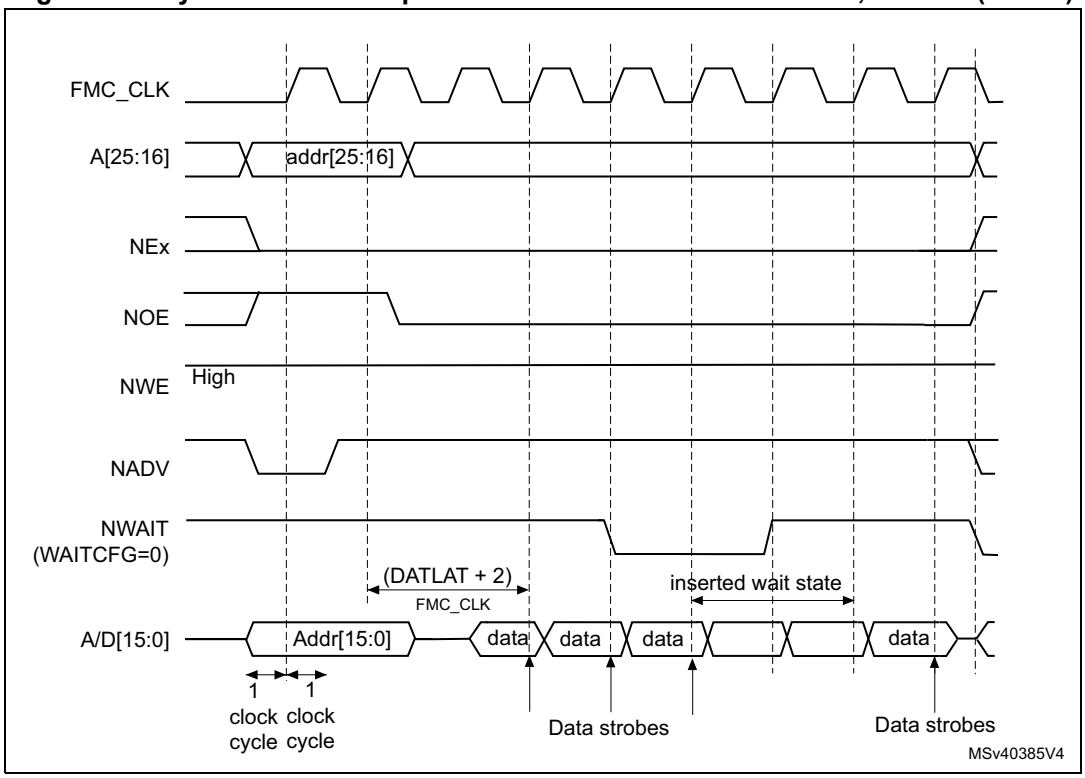


Table 180. FMC_BCRx bitfields (Synchronous multiplexed read mode)

Bit No.	Bit name	Value to set
31	Reserved	0x0
30-24	Reserved	0x00
23:22	NBLSET[1:0]	As needed
21:20	CSCOUNT	As needed
19	CBURSTRW	As needed
18:16	CPSIZE	As needed. (0x1 when using CRAM 1.5)
15	ASYNCWAIT	0x0
14	EXTMOD	0x0
13	WAITEN	to be set if the memory supports this feature, to be kept cleared otherwise
12	WREN	no effect on synchronous read
11	WAITCFG	to be set according to memory
10	Reserved	0x0
9	WAITPOL	to be set according to memory
8	BURSTEN	0x1
7	Reserved	0x1
6	Reserved	0x0

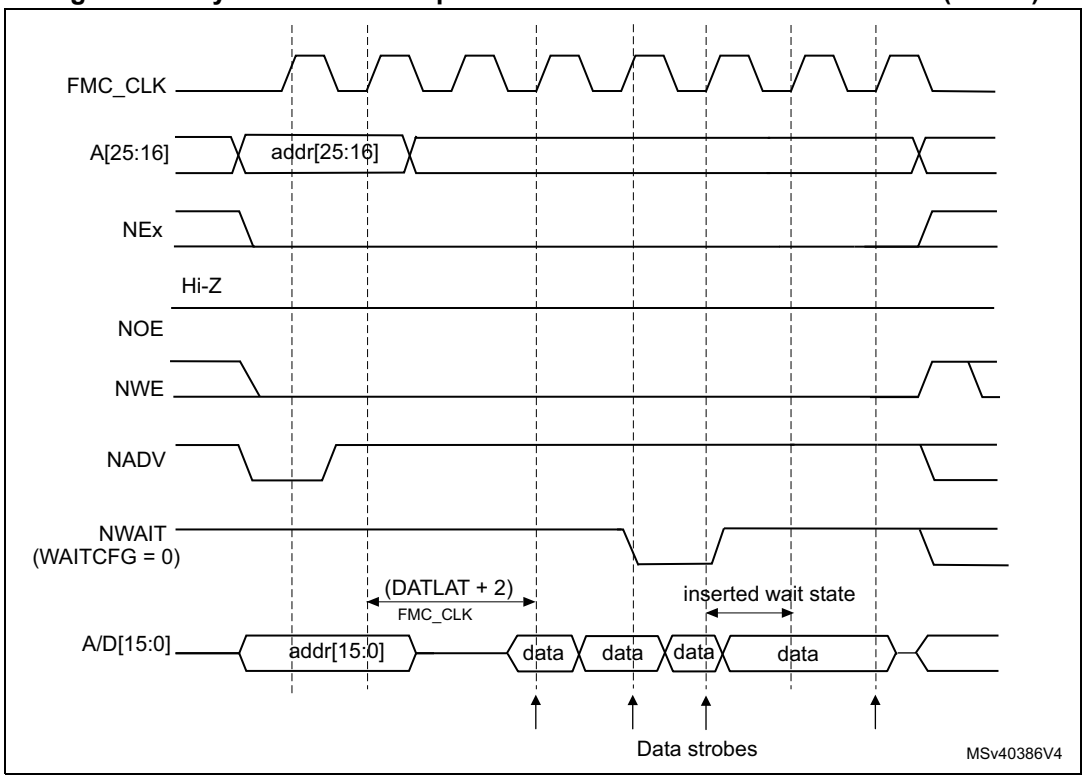
Table 180. FMC_BCRx bitfields (Synchronous multiplexed read mode) (continued)

Bit No.	Bit name	Value to set
5-4	MWID[1:0]	As needed
3-2	MTYP[1:0]	0x1 or 0x2
1	MUXEN	As needed
0	MBKEN	0x1

Table 181. FMC_BTRx bitfields (Synchronous multiplexed read mode)

Bit No.	Bit name	Value to set
31:30	DATAHLD	Don’t care
29:28	ACCMOD	0x0
27-24	DATLAT	Data latency
27-24	DATLAT	Data latency
23-20	CLKDIV	FMC_CLK divider ratio
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15-8	DATAST	Don’t care
7-4	ADDHLD	Don’t care
3-0	ADDSET	Don’t care

Figure 178. Synchronous multiplexed write mode waveforms - PSRAM (CRAM)



1. The memory must issue NWAIT signal one cycle in advance, accordingly WAITCFG must be programmed to 0.

Table 182. FMC_BCRx bitfields (Synchronous multiplexed write mode)

Bit No.	Bit name	Value to set
31	Reserved	0x0
30-24	Reserved	0x00
23:22	NBLSET[1:0]	As needed
21:20	CSCOUNT	As needed
19	CBURSTRW	As needed
18:16	CPSIZE	As needed. (0x1 when using CRAM 1.5)
15	ASYNCWAIT	0x0
14	EXTMOD	0x0
13	WAITEN	to be set if the memory supports this feature, to be kept cleared otherwise.
12	WREN	0x1
11	WAITCFG	0x0
10	Reserved	0x0
9	WAITPOL	to be set according to memory
8	BURSTEN	0x1
7	Reserved	0x1

Table 182. FMC_BCRx bitfields (Synchronous multiplexed write mode) (continued)

Bit No.	Bit name	Value to set
6	Reserved	0x0
5-4	MWID[1:0]	As needed
3-2	MTYP[1:0]	0x1
1	MUXEN	As needed
0	MBKEN	0x1

Table 183. FMC_BTRx bitfields (Synchronous multiplexed write mode)

Bit No.	Bit name	Value to set
31:30	DATAHLD	Don't care
29:28	ACCMOD	0x0
27-24	DATLAT	Data latency
23-20	CLKDIV	FMC_CLK divider ratio
19-16	BUSTURN	Time between NEx high to NEx low (BUSTURN fmc_ker_ck)
15-8	DATAST	Don't care
7-4	ADDHLD	Don't care
3-0	ADDSET	Don't care

27.8.6 NOR/PSRAM controller registers

The peripheral registers have to be accessed by words (32-bit)

SRAM/NOR flash chip-select control register for memory region x (FMC_BCRx)

Address offset: 0x00 + 0x08 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 30DB, 0x0000 30D2, 0x0000 30D2, 0x0000 30D2

This register contains the control information of each memory region, used for SRAMs, PSRAM, FRAM and NOR flash memories.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBLSET[1:0]		CSCOUNT1	CSCOUNT0	CBURSTRW	CPSIZE[2:0]		
								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	Res.	WAITPOL	BURSTEN	Res.	Res.	MWID[1:0]		MTYP[1:0]		MUXEN	MBKEN
rw	rw	rw	rw	rw		rw	rw			rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:24 Reserved, must be kept at reset value.

Bits 23:22 **NBLSET[1:0]**: Byte lane (NBL) setup

These bits configure the NBL setup timing from NBLx low to Chip select NEx low.
00: NBL setup time is 0 fmc_ker_ck clock cycle.
01: NBL setup time is 1 fmc_ker_ck clock cycle.
10: NBL setup time is 2 fmc_ker_ck clock cycles.
11: NBL setup time is 3 fmc_ker_ck clock cycles.

Bits 21:20 **CSCOUNT[1:0]:** Chip Select (CS) counter

These bits define the maximum duration of Chip Select low for PSRAM synchronous mode and for each memory region.
The Chip Select counter is loaded into a timer which is decremented using the KCK_FMC while the NE is maintained low during PSRAM read or write access.
When the timer reaches 0, the PSRAM controller splits the current access and toggles NE to allow PSRAM device refresh.
The programmed counter value guarantees a maximum NE pulse width (t_{CEM}) as specified for PSRAM devices.
The counter is reloaded and starts decrementing each time a new access is triggered by a transition of NE from high to low.
0b00: Counter disabled
0b01: NEx deasserted after fmc_ker_ck clock cycle
0b10: NEx deasserted after 64 fmc_ker_ck clock cycles
0b11: NEx deasserted after 256 fmc_ker_ck clock cycles

Note: Most PSRAMs manage refresh transparently and do not require access split. CSCOUNT[1:0] must consequently be left at its default value (counter disabled), otherwise CSCOUNT[1:0] value and access split impact must be checked according to PSRAM timing parameters.

Bit 19 **CBURSTRW:** Write burst enable

For PSRAM (CRAM) operating in burst mode, the bit enables synchronous accesses during write operations. The enable bit for synchronous read accesses is the BURSTEN bit in the FMC_BCRx register.
0: Write operations are always performed in asynchronous mode
1: Write operations are performed in synchronous mode.

Bits 18:16 **CPSIZE[2:0]:** CRAM page size

These bits are used for CellularRAM™ 1.5 which does not allow burst accesses to cross the address boundaries between pages. When these bits are configured, the FMC automatically splits the burst access when the memory page size is reached (refer to memory datasheet for page size).
000: No burst split when crossing page boundary. (default after reset).
001: 128 bytes
010: 256 bytes
011: 512 bytes
100: 1024 bytes
Others: Reserved.

Bit 15 **ASYNCWAIT**: Wait signal during asynchronous transfers

This bit enables/disables wait signal usage by the FMC even during asynchronous transfers.
0: NWAIT signal not taken in to account during asynchronous transfers (default after reset)
1: NWAIT signal taken in to account during asynchronous transfers

Bit 14 **EXTMOD:** Extended mode enable

This bit enables write timing programming for asynchronous accesses inside the FMC_BWTR register, thus resulting in different timings for read and write operations.

0: Values inside FMC_BWTR register not taken into account (default after reset)

1: Values inside FMC_BWTR register taken into account

Note: When the extended mode is disabled, the FMC can operate in mode 1 or mode 2 as follows:

- *Mode 1 is the default mode when the SRAM/PSRAM memory type is selected (MTYP[1:0] =0x0 or 0x01)*
- *Mode 2 is the default mode when the NOR memory type is selected (MTYP[1:0] = 0x10).*

Bit 13 **WAITEN:** Wait enable bit

This bit enables/disables wait-state insertion via the NWAIT signal when accessing the memory in synchronous mode.

0: NWAIT signal disabled (its level not taken into account, no wait-state inserted after the programmed flash latency period)

1: NWAIT signal enabled (its level is taken into account after the programmed latency period to insert wait-states if asserted) (default after reset)

Bit 12 **WREN:** Write enable bit

This bit indicates whether write operations to the memory region by the FMC are enabled/disabled:

0: Write operations to the memory region by the FMC disabled. An AXI slave error is reported,

1: Write operations to the memory region by the FMC enabled (default after reset).

Bit 11 **WAITCFG:** Wait timing configuration

The NWAIT signal indicates whether the data from the memory are valid or if a wait-state must be inserted when accessing the memory in synchronous mode. This configuration bit determines if NWAIT is asserted by the memory one clock cycle before the wait-state or during the wait-state:

0: NWAIT signal active one data cycle before wait-state (default after reset),

1: NWAIT signal active during wait-state (not used for PSRAM).

Bit 10 Reserved, must be kept at reset value.

Bit 9 **WAITPOL:** Wait signal polarity bit

This bit defines the polarity of the wait signal from memory used for either in synchronous or asynchronous mode:

0: NWAIT active low (default after reset),

1: NWAIT active high.

Bit 8 **BURSTEN:** Burst enable bit

This bit enables/disables synchronous accesses during read operations. It is valid only for synchronous memories operating in burst mode:

0: Burst mode disabled (default after reset). Read accesses are performed in asynchronous mode.

1: Burst mode enable. Read accesses are performed in synchronous mode.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **MWID[1:0]:** Memory data bus width
This bit defines the external memory device width, valid for all type of memories.
00: 8 bits
01: 16 bits (default after reset)
10: 32 bits
11: Reserved.

Bits 3:2 **MTYP[1:0]:** Memory type
This bit defines the type of external memory attached to the corresponding memory region:
00: SRAM/ FRAM (default after reset for memory region 2...4)
01: PSRAM (CRAM)/FRAM
10: NOR flash/OneNAND flash (default after reset for memory region 1)
11: Reserved.

Bit 1 **MUXEN:** Address/data multiplexing enable bit
When this bit is set, the address and data values are multiplexed on the data bus, valid only with NOR and PSRAM memories:
0: Address/Data non-multiplexed
1: Address/Data multiplexed on databus (default after reset)

Bit 0 **MBKEN:** Memory region enable bit
This bit enables the memory region. After reset memory region 1 is enabled, all others are disabled. Accessing a disabled memory region causes an ERROR on AXI bus.
0: Corresponding memory region is disabled
1: Corresponding memory region is enabled

SRAM/NOR flash chip-select timing registers for memory region x (FMC_BTRx)

Address offset: 0x04 + 0x8 * (x - 1), (x = 1 to 4)

Reset value: 0x0FFF FFFF

This register contains the control information of each memory region, used for SRAMs, PSRAM and NOR flash memories.If the EXTMOD bit is set in the FMC_BCRx register, then this register is partitioned for write and read access, that is, two registers are available: one to configure read accesses (this register) and one to configure write accesses (FMC_BWTRx registers).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATAHLD[1:0]		ACCMOD[1:0]		DATLAT[3:0]				CLKDIV[3:0]				BUSTURN[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAST[7:0]								ADDHLD[3:0]				ADDSET[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 **DATAHLD[1:0]:** Data Hold phase duration

These bits are written by software to define the duration of the *data hold* phase in fmc_ker_ck cycles (refer to *Figure 174* to *Figure 175*), used in asynchronous accesses:
For read/write accesses:
00: DATAHLD phase duration = fmc_ker_ck clock cycle * 0 (default, read)/1 (default, write)
01: DATAHLD phase duration = fmc_ker_ck clock cycle * 1 (read)/2 (write)
10: DATAHLD phase duration = fmc_ker_ck clock cycle * 2 (read)/3 (write)
11: DATAHLD phase duration = fmc_ker_ck clock cycle * 3 (read)/4 (write)

Bits 29:28 **ACCMOD[1:0]:** Access mode

These bits specify the asynchronous access modes as shown in the timing diagrams. These bits are taken into account only when the EXTMOD bit in the FMC_BCRx register is 1.
00: Access mode A
01: Access mode B
10: Access mode C
11: Access mode D

Bits 27:24 **DATLAT[3:0]:** Data latency for synchronous memory (see note below bit descriptions)

For synchronous access with read/write burst mode enabled (BURSTEN / CBURSTRW bits set), these bits define the number of memory clock cycles (+2) to issue to the memory before reading/writing the first data.
This timing parameter is not expressed in fmc_ker_ck periods, but in FMC_CLK periods.
For asynchronous access, this value is don't care.
0000: Data latency of 2 FMC_CLK clock cycles for first burst access
1111: Data latency of 17 FMC_CLK clock cycles for first burst access (default value after reset)

Bits 23:20 **CLKDIV[3:0]:** Clock divide ratio (for FMC_CLK signal)

These bits define FMC_CLK clock output signal period, expressed in fmc_ker_ck cycles:
0000: FMC_CLK period= 1x fmc_ker_ck period
0001: FMC_CLK period = 2 × fmc_ker_ck periods
0010: FMC_CLK period = 3 × fmc_ker_ck periods
1111: FMC_CLK period = 16 × fmc_ker_ck periods (default value after reset)
This value is don't care when accessing NOR flash memories, SRAM and PSRAM in asynchronous mode.

Note: Refer to Section 27.8.5: Synchronous transactions for FMC_CLK *divider ratio formula).*

Note: CLKDIV[3:0] bits are don't care when CCLKEN bit of FMC_CFGR is set. In this case FMC_CLK clock output is controlled by CLKDIV of FMC_CFGR.

Bits 19:16 **BUSTURN[3:0]:** Bus turnaround phase duration

These bits are written by software to add a delay between the end of current read or write transaction and the next transaction on the same memory region.

This delay allows matching the minimum time between consecutive transactions (t_{EHEL} from NEx high to NEx low) and the maximum time required by the memory to free the data bus after a read access ($t_{EHQZ\ f}$from chip enable high to output Hi-Z). This delay is recommended for mode D and muxed mode. For non-muxed memory, the bus turnaround delay can be set to a minimum value that respects the following condition:

(BUSTRUN + 1)fmc_ker_ck period ≥ $t_{EHELmin}$ max ($t_{EHELmin}$, $t_{EHQZmax)}$

For FRAMs, the bus turnaround delay must be configured to match the minimum precharge time (t_{PC}). This delay is inserted between all consecutive transactions on the same memory region (read/read, write/write, read/write and write/read) to match the t_{PC} memory timing, and the Chip select toggles between consecutive accesses.

(BUSTRUN + 1) fmc_ker_ck period >= t_{PCmin}

0000: BUSTURN phase duration = 1 * fmc_ker_ck clock cycle added

...

1111: BUSTURN phase duration = 16 * fmc_ker_ck clock cycles added (default value after reset)

Bits 15:8 **DATAST[7:0]:** Data-phase duration

These bits are written by software to define the duration of the data phase (refer to *Figure 174* to *Figure 175*) used in asynchronous accesses:

0000 0000: Reserved.

0000 0001: DATAST phase duration = 1 * fmc_ker_ck clock cycles

0000 0010: DATAST phase duration = 2 * fmc_ker_ck clock cycles

...

1111 1111: DATAST phase duration = 255 * fmc_ker_ck clock cycles (default value after reset)

For each memory type and access mode data-phase duration, refer to the respective figure (*Figure 174* to *Figure 175*).

Example of mode 1, write access, DATAST=1:

Data-phase duration= DATAST+1 = 2 *fmc_ker_ck clock cycles.

Note: This value is don't care in synchronous accesses.

Bits 7:4 **ADDHLD[3:0]:** Address-hold phase duration

These bits are written by software to define the duration of the *address hold* phase (refer to *Figure 170* to *Figure 173*) used in mode D or multiplexed accesses:

0000: Reserved.

0001: ADDHLD phase duration =1 * fmc_ker_ck clock cycle

0010: ADDHLD phase duration = 2 * fmc_ker_ck clock cycle

...

1111: ADDHLD phase duration = 15 * fmc_ker_ck clock cycles (default value after reset)

For each access mode address-hold phase duration, refer to the respective figure (*Figure 170* to *Figure 173*).

Note: This value is not used in synchronous accesses, and the address hold phase lasts always 1 memory clock period.

Bits 3:0 **ADDSET[3:0]:** Address setup phase duration

These bits are written by software to define the duration of the *address setup* phase (refer to *Figure 174* to *Figure 175*), used in SRAMs, ROMs and asynchronous NOR flash memories:

0000: ADDSET phase duration = 0 * fmc_ker_ck clock cycle

...

1111: ADDSET phase duration = 15 * fmc_ker_ck clock cycles (default value after reset)

For each access mode address setup phase duration, refer to the respective figure (refer to *Figure 174* to *Figure 175*).

Note: This value is don't care in synchronous accesses.

In muxed mode or mode D, the minimum value for ADDSET is 1.

Note: PSRAMs (CRAMs) have a variable latency due to internal refresh. Therefore these memories issue the NWAIT signal during the whole latency phase to extend the latency as needed.

On PSRAMs (CRAMs), DATLAT must be configured to 0 so that the FMC quickly exits its latency phase, starts sampling NWAIT from memory, and starts read or write operations when the memory is ready.

This method can be used also with the latest generation of synchronous flash memories, which, unlike older flash memories, issue the NWAIT signal (check the corresponding flash memory datasheet).

SRAM/NOR-flash write timing registers for memory region x (FMC_BWTRx)

Address offset: 0x104 + 0x8 * (x - 1), (x = 1 to 4)

Reset value: 0x000F FFFF

This register contains the control information of each memory region. It is used for SRAMs, FRAMs, PSRAMs and NOR flash memories. When the EXTMOD bit is set in the FMC_BCRx register, then this register is active for write access.

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

Bits 31:30 **DATAHLD[1:0]:** Data Hold phase duration

These bits are written by software to define the duration of the *data hold* phase in fmc_ker_ck cycles (refer to *Figure 175*), used in asynchronous write accesses:

00: DATAHLD phase duration = 1 * fmc_ker_ck clock cycle (default)

01: DATAHLD phase duration = 2 * fmc_ker_ck clock cycle

10: DATAHLD phase duration = 3 * fmc_ker_ck clock cycle

11: DATAHLD phase duration = 4 * fmc_ker_ck clock cycle

Bits 29:28 **ACCMOD[1:0]:** Access mode.

These bits specify the asynchronous access modes as shown in the next timing diagrams.These bits are taken into account only when the EXTMOD bit in the FMC_BCRx register is 1.

00: Access mode A

01: Access mode B

10: Access mode C

11: Access mode D

Bits 27:20 Reserved, must be kept at reset value.

Bits 19:16 **BUSTURN[3:0]**: Bus turnaround phase duration

These bits are written by software to add a delay between the end of current write transaction and the next transaction on the same memory region.

For FRAMs, the bus turnaround delay must be configured to match the minimum precharge time (t_{PC}). The bus turnaround delay is inserted between all consecutive transactions on the same memory region (read/read, write/write, read/write and write/read), and the Chip select toggles between consecutive accesses.

(BUSTRUN + 1) fmc_ker_ck period >= t_{PCmin}

0000: BUSTURN phase duration = 1 * fmc_ker_ck clock cycle added

...

1111: BUSTURN phase duration = 16 * fmc_ker_ck clock cycles added (default value after reset)

Bits 15:8 **DATAST[7:0]:** Data-phase duration.

These bits are written by software to define the data phase duration (refer to *Figure 174* to *Figure 175*) used in asynchronous SRAM, PSRAM and NOR flash memory accesses:
0000 0000: Reserved.
0000 0001: DATAST phase duration = 1 * fmc_ker_ck clock cycles
0000 0010: DATAST phase duration = 2 * fmc_ker_ck clock cycles
...
1111 1111: DATAST phase duration = 255 * fmc_ker_ck clock cycles (default value after reset)

Bits 7:4 **ADDHLD[3:0]:** Address-hold phase duration.

These bits are written by software to define the duration of the address hold phase (refer to *Figure 172* to *Figure 173*) used in asynchronous multiplexed accesses:
0000: Reserved.
0001: ADDHLD phase duration = 1 * fmc_ker_ck clock cycle
0010: ADDHLD phase duration = 2 * fmc_ker_ck clock cycle
...
1111: ADDHLD phase duration = 15 * fmc_ker_ck clock cycles (default value after reset)

Note: In synchronous NOR flash accesses, this value is not used, and the address hold phase always lasts 1 flash clock period.

Bits 3:0 **ADDSET[3:0]:** Address setup phase duration.

These bits are written by software to define the duration of the address setup phase in fmc_ker_ck cycles (refer to *Figure 174* to *Figure 175*) used in asynchronous accesses:
0000: ADDSET phase duration = 0 * fmc_ker_ck clock cycle
...
1111: ADDSET phase duration = 15 * fmc_ker_ck clock cycles (default value after reset)

Note: In synchronous accesses, this value is not used, and the address setup phase always lasts 1 flash clock period. In muxed mode, the minimum ADDSET value is 1.

FMC common configuration register (FMC_CFGR)

Address offset: 0x020

Reset value: 0x0000 0000

This register is user to set configuration common to all controllers

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FMCEN	Res.	Res.	Res.	Res.	Res.	BMAP1	BMAP0	Res.	Res.	Res.	CCLKEN	CLKDIV[3:0]			
rw						rw	rw				rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bit 31 **FMCEN:** FMC enable
This bit enables/disables the FMC.
0: FMC disabled
1: FMC enabled

Bits 30:26 Reserved, must be kept at reset value.

Bits 25:24 **BMAP[1:0]:** FMC memory region mapping
These bits allow the remapping of SDRAM device 2 or the swapping of FMC NOR/PSRAM and SDRAM memory regions.
00: Default mapping (refer to *Table 148*)
01: Devices are remapped (refer to *Table 149*)
10: Devices are remapped (refer to *Table 149*)
11: All memory region accesses are ignored. A bus error is generated.

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **CCLKEN:** Continuous clock enable
This bit enables the FMC_CLK clock output to external memory devices.
0: The FMC_CLK is only generated during the synchronous memory access (read/write transaction). The FMC_CLK clock ratio is defined by the programmed CLKDIV value in the FMC_BCRx register (default after reset).
1: The FMC_CLK is generated continuously during asynchronous and synchronous access.
If continuous clock mode is enabled, the synchronous memories connected to all memory regions are clocked by the same clock with the clock ratio defined by CLKDIV in FMC_CFGR. The CLKDIV value in FMC_BWTRx are don't care.

Bits 19:16 **CLKDIV[3:0]:** Clock divide ratio (for FMC_CLK signal)
These bits define the FMC_CLK clock output signal period, expressed in number of fmc_ker_ck cycles:
0000: FMC_CLK period= 1x fmc_ker_ck period
0001: FMC_CLK period = 2 × fmc_ker_ck periods
0010: FMC_CLK period = 3 × fmc_ker_ck periods
1111: FMC_CLK period = 16 × fmc_ker_ck periods (default value after reset)
This value is don't care when accessing NOR flash memories, SRAM and PSRAM in asynchronous mode.
Note: Refer to Section 27.8.5: Synchronous transactions for FMC_CLK divider ratio formula.

Bits 15:0 Reserved, must be kept at reset value.

27.9 NAND flash controller

The FMC generates the appropriate signal timings to drive 8- and 16-bit NAND flash memories

The NAND memory region is configured through dedicated registers (*Section 27.9.9*). The programmable memory parameters include access timings (shown in *Table 184*) and ECC configuration.

The NAND controller supports the following features:

- Programmable page size of 256, 512, 1024, 2048, 4096 or 8192 bytes
- Programmable memory timings
- Programmable Error Correction Capability using BCH code or Hamming code
- Command sequencer: a hardware accelerator for read/write accesses within a page or within a block of configurable size.

  This feature is useful when ECC computation is performed.
- Programmable address offset for ECC bytes programming using the command sequencer
- Interrupt generation on ECC error detection and page or sector transfer complete

Table 184. Programmable NAND flash access parameters

Parameter	Function	Access mode	Unit	Min.	Max.
Memory setup time	Number of clock cycles (fmc_ker_ck) required to set up the address before the command assertion	Read/Write	FMC clock cycle (fmc_ker_ck)	1	256
Memory wait	Minimum duration (in fmc_ker_ck clock cycles) of the command assertion	Read/Write	FMC clock cycle (fmc_ker_ck)	1	256
Memory hold	Number of clock cycles (fmc_ker_ck) during which the address must be held (as well as the data if a write access is performed) after the command deassertion	Read/Write	FMC clock cycle (fmc_ker_ck)	1	256
Memory databus high-Z	Number of clock cycles (fmc_ker_ck) during which the data bus is kept in high-Z state after a write access has started	Write	FMC clock cycle (fmc_ker_ck)	1	256

27.9.1 External memory interface signals

The following tables list the typical signals that are used to interface NAND flash memory.

Note: The “N” prefix identifies the signals which are active low.

8-bit NAND flash memory

Table 185. 8-bit NAND flash memory

FMC signal name	I/O	Function
A[17]	O	NAND flash address latch enable (ALE) signal
A[16]	O	NAND flash command latch enable (CLE) signal
D[7:0]	I/O	8-bit multiplexed, bidirectional address/data bus
NCE	O	Chip Select
NOE(= NRE)	O	Output enable (memory signal name: read enable, NRE)

Table 185. 8-bit NAND flash memory (continued)

FMC signal name	I/O	Function
NWE	O	Write enable
RNB	I	NAND flash Ready/Busy (RNB) input signal to the FMC

Note: Theoretically, there is no capacity limitation as the FMC can manage as many address cycles as needed.

16-bit NAND flash memory

Table 186. 16-bit NAND flash memory

FMC signal name	I/O	Function
A[17]	O	NAND flash address latch enable (ALE) signal
A[16]	O	NAND flash command latch enable (CLE) signal
D[15:0]	I/O	16-bit multiplexed, bidirectional address/data bus
NCE	O	Chip Select
NOE(= NRE)	O	Output enable (memory signal name: read enable, NRE)
NWE	O	Write enable
RNB	I	NAND flash Ready/Busy (RNB) input signal to the FMC

Note: Theoretically, there is no capacity limitation as the FMC can manage as many address cycles as needed.

27.9.2 NAND flash supported memories and transactions

Table 187 shows the supported devices, access modes and transactions. Transactions not allowed (or not supported) by the NAND flash controller are shown in gray.

Table 187. Supported memories and transactions

Device	Mode	R/W	AXI data size	Memory data size	Allowed/ not allowed	Comments
NAND 8-bit	Asynchronous	R	8	8	Y	
	Asynchronous	W	8	8	Y	
	Asynchronous	R	16	8	Y	Split into 2 FMC accesses
	Asynchronous	W	16	8	Y	Split into 2 FMC accesses
	Asynchronous	R	32	8	Y	Split into 4 FMC accesses
	Asynchronous	W	32	8	Y	Split into 4 FMC accesses
	Asynchronous	R	64	8	Y	Split into 8 FMC accesses
	Asynchronous	W	64	8	Y	Split into 8 FMC accesses

Table 187. Supported memories and transactions (continued)

Device	Mode	R/W	AXI data size	Memory data size	Allowed/ not allowed	Comments
NAND 16-bit	Asynchronous	R	8	16	Y	
	Asynchronous	W	8	16	N	
	Asynchronous	R	16	16	Y	
	Asynchronous	W	16	16	Y	
	Asynchronous	R	32	16	Y	Split into 2 FMC accesses
	Asynchronous	W	32	16	Y	Split into 2 FMC accesses
	Asynchronous	R	64	16	Y	Split into 4 FMC accesses
	Asynchronous	W	64	16	Y	Split into 4 FMC accesses

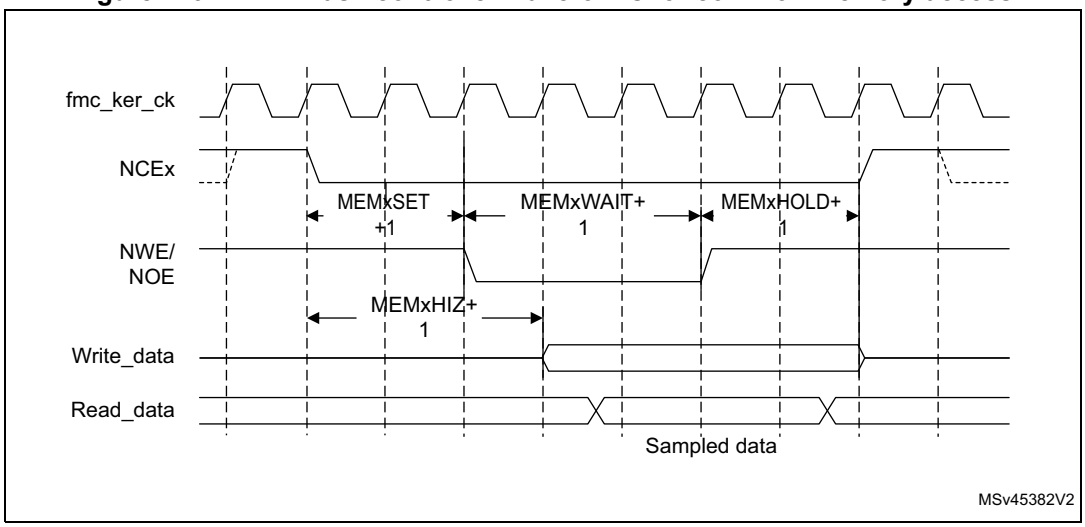
27.9.3 Timing diagrams for NAND flash memory

The NAND flash memory region is managed through a set of registers:

- Control register: FMC_PCR
- Interrupt status register: FMC_SR
- ECC register: FMC_HECCR
- Timing register for Common memory space: FMC_PMEM
- Timing register for Attribute memory space: FMC_PATT

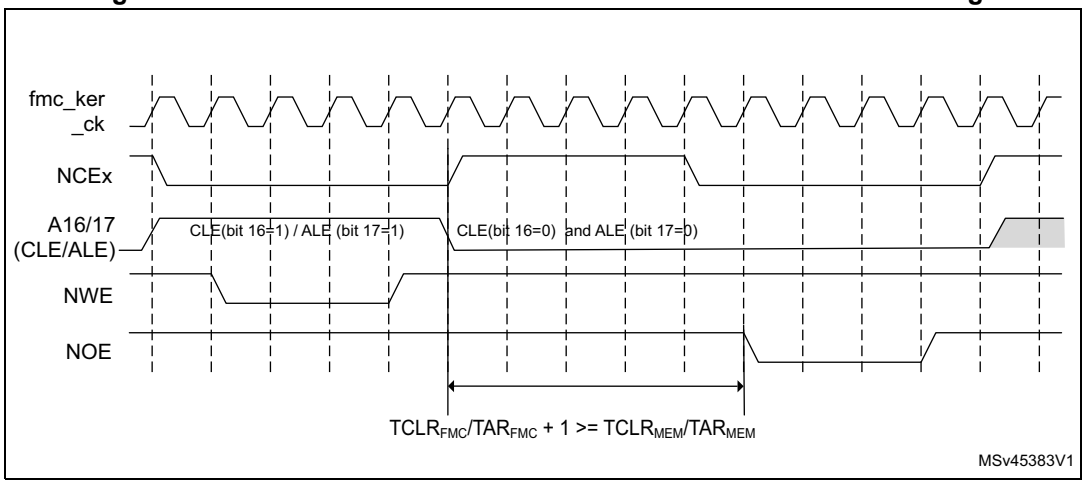
Each timing configuration register contains three parameters used to define the number of fmc_ker_ck cycles for the different phases of any NAND flash access, plus one parameter that defines the timing to start driving the data bus when a write access is performed. *Figure 179* shows the timing parameter definitions for common memory accesses, knowing that Attribute memory space access timings are similar.

Figure 179. NAND flash controller waveforms for common memory access



1. NOE remains high (inactive) during write accesses. NWE remains high (inactive) during read accesses.
2. If PWAITEN = 1, the programmed values of (MEMxSET + MEMxHOLD) must be > 1.

Figure 180. NAND flash controller waveforms for TCLR and TAR timings



1. NOE remains high (inactive) during write accesses. NWE remains high (inactive) during read accesses.

27.9.4 NAND flash operations

The command latch enable (CLE) and address latch enable (ALE) signals of the NAND flash memory device are driven by address signals from the FMC. This means that to send a command or an address to the NAND flash memory, the CPU has to perform a write to a specific address in its memory space.

A typical page read operation from the NAND flash device requires the following steps:

1. Program and enable the NAND memory region by configuring the FMC_PCR and FMC_PMEM (and for some devices, FMC_PATT, see *Section 27.9.5: NAND flash prewait function*) registers according to the characteristics of the NAND flash memory (PWID bits to configure NAND flash data bus width, PWAITEN set or cleared as needed, see *Section 27.7.2: NAND flash memory address mapping* for timing configuration).
2. The CPU performs a byte write to the common memory space, with a data byte value equal to one flash command byte (for example 0x00 for Samsung NAND flash devices). The NAND flash CLE input is active during the write strobe (low pulse on NWE), so that the written byte is interpreted as a command by the NAND flash memory. Once the command is latched by the memory device, it does not need to be written again for the following page read operations.
3. The CPU can send the start address for a read operation by writing a five-byte address to the common memory or the attribute space followed by the second read command. The NAND flash ALE input is active during the write strobe (low pulse on NWE), so that the written bytes are interpreted as the start address for read operations. Using the attribute memory space allows to use a different FMC timing configuration, which can

be used to implement the prewait function required for some NAND flash memories (see details in *Section 27.9.5: NAND flash prewait function*).

4. The controller waits till the NAND flash memory is ready (RNB signal high), before starting a new access to the same memory region or to another memory region.
5. The CPU can then perform byte read operations from the common memory space to read the NAND flash page (data field + Spare field) byte by byte.
6. No CPU command or address write operation is required to read the next NAND flash page. This can be done in three different ways:
   – by simply performing the operation described in step 5,
   – a new random address can be accessed by restarting the operation at step 3,
   – a new command can be sent to the NAND flash device by restarting at step 2.

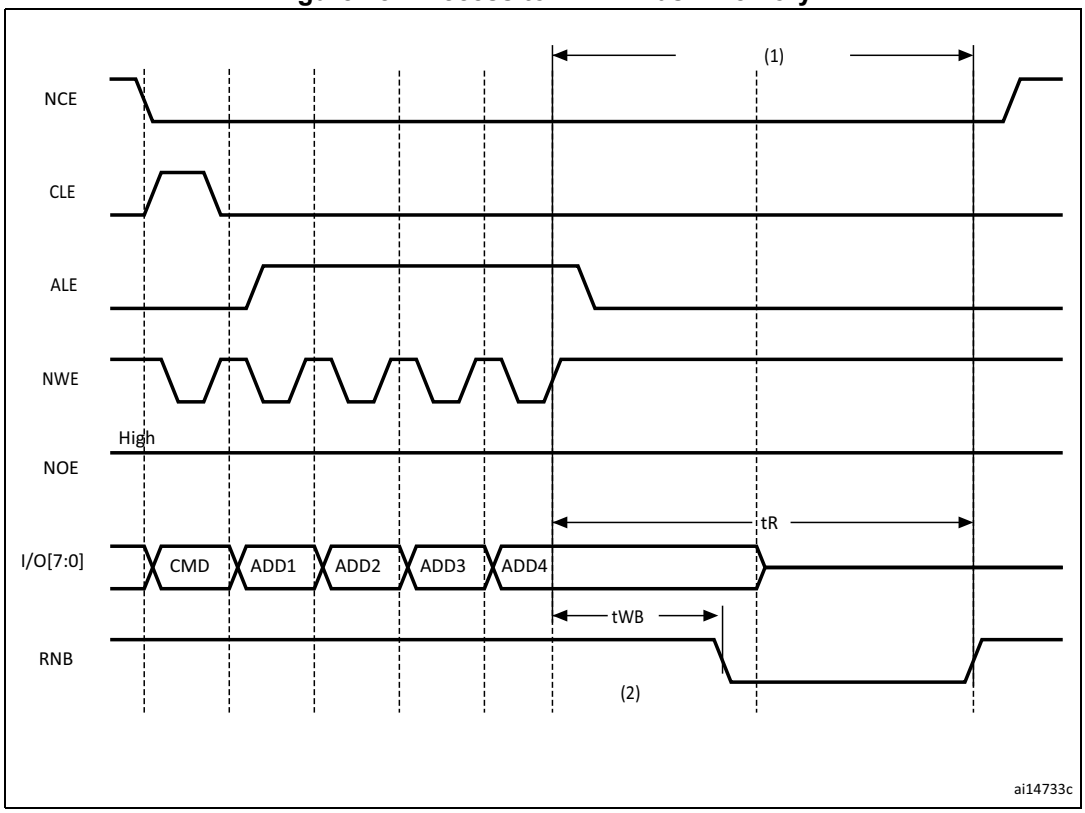
Note: Non-'CE don't care' NAND flash memories are not supported.

ONFI mode 4 and 5 EDO data output cycle timings are not supported.

27.9.5 NAND flash prewait function

The NAND controller must wait for RNB to be asserted low before sending its next command (see *Figure 181*). This timing constraint is set according to t_{WB} using MEMHOLD[7:0] bitfield in FMC_PMEM or ATTHOLD[7:0] in FMC_PATT.

Figure 181. Access to NAND flash memory



1. NCE can be deasserted for NAND flash memories since CE is don't care except for very old NAND flash devices that are not considered in this example.
2. RNB sampling can be delayed after t_{WB} by using ATTHOLD[7:0] bit of FMC_PATT. As an example, if fmc_ker_clk = 14 ns and ATTHOLD[7:0] = 7, the delay is 8*14 = 112 ns. The usual t_{WB} maximum value is 100 ns and the programmed delay must be higher than t_{WB} + 2*fmc_ker_clk.

Any CPU read or write access to the NAND flash memory has a delay of (MEMHOLD[7:0] + 1) * fmc_ker_ck cycles after the NWE signal is asserted.

The ATTHOLD[7:0] bitfield in FMC_PATT register must be determined to satisfy t_{WB} timing constraints and selectively apply larger timing constraints to the last command or address byte in CPU mode. When setting ATTHOLD[7:0], the 2 * fmc_ker_ck delay required for the sampling of RNB by NAND sequencer must also taken into account.

27.9.6 NAND ECC controller

NAND flash memories include a spare memory area (or OOB) at the end of each page. This area is typically used to store some metadata (such as Bad Block Marker information to invalidate the whole page) or for error management functions.

The FMC supports on-the-fly error correction code (ECC) computation during data read/program from/to NAND flash memory. This mechanism reduces the host CPU workload when processing the ECC by software.

The user application can select one out of two algorithms:

- Hamming code algorithm for 1-bit error code correction and 2-bit error detection per 256, 512, 1024, 2048, 4096 or 8192 bytes.
- BCH (Bose, Chaudhuri and Hocquenghem) code for 4-bit error code correction and 8-bit error detection or 8-bit error code correction and 16-bit error detection per 512 bytes.

The principle of both ECC controllers is similar: the ECC modules monitor the NAND flash data bus and read/write signals (NCE and NWE) each time the NAND flash memory region is active. The ECC controllers operate as follows:

- When accessing a NAND flash memory region, the data present on the D[7:0] or D[15:0] bus are latched and used for ECC computation.
- When accessing any other address in NAND flash memory, the ECC logic is idle, and does not perform any operation. As a result, write operations to define commands or addresses to the NAND flash memory are not taken into account for ECC computation.

Once the desired number of bytes has been read/written from/to the NAND flash memory by the host CPU or DMA, the ECC result registers are updated.

Hamming code

The Hamming code implemented can perform 1-bit error correction and 2-bit error detection per 256, 512, 1024, 2048, 4 096 or 8192 bytes read or written from/to the NAND flash memory. It consists in calculating the row and column parity. This algorithm is supported by 8-bit and 16-bit NAND flash memories.

Page write and read sequences with Hamming code are described below.

To perform a page program with Hamming ECC computation:

1. Enable the ECCEN bit in the FMC_PCR register.
2. Write data to the NAND flash memory page. While the NAND page is written, the ECC controller computes the ECC value.
3. Wait until the ECC code is ready (NWRF is set in the FMC_SR register).
4. Read the ECC value available in the FMC_HECCR register and store it in NAND flash spare area.
5. Clear the ECCEN bit and launch the page programming.

To perform a page read with Hamming ECC computation:

1. Enable the ECCEN bit in the FMC_PCR register before reading the NAND page. While the NAND page is read, the ECC block computes the ECC value.
2. Wait until the ECC code is ready (NWRF set in the FMC_SR register).
3. Read the ECC value available in the spare area as well also the ECC value, processed by Hamming controller, available in FMC_HECCR register.
4. If the two ECC values are identical, no correction is required. Otherwise an ECC error occurred, the software correction routine returns information on whether the error can be corrected or not, and corrects it when possible.

Depending on the spare area size, the page can be split into several sectors of identical size. This allows increasing the error correction capability for each page since 1 error can be corrected and 2 errors can be detected for each sector.

Example

For a NAND flash memory with 2 Kbytes + 64 bytes pages, each page can be split into 8 sectors of 256 bytes each. 3 bytes (22 bits) of ECC are required for each sector, which makes 24 bytes (8 * 3 bytes) to be stored in the spare area for the whole page. For the whole 2 Kbyte page, up to 8 errors can be corrected, respectively up to 16 errors detected, provided there is less than 1 error, respectively 2 errors, per sector.

BCH (Bose, Chaudhuri and Hocquenghem) code

To increase the error correction capability, the FMC embeds a BCH (Bose, Chaudhuri and Hocquenghem) encoder and decoder.

It supports 4-bit error code correction and 8-bit error detection or 8-bit error code correction, and 16-bit error detection per sector. BCH encoder/decoder module handles sectors of fixed size, equal to 512 bytes. Each page must consequently be split into 512-byte sectors. The number of parity bytes generated for each sector is given in *Table 188*. For 16-bit NAND flash memories, a padding is added to align parity bytes to an even number (8 bytes and 14 bytes for 4-bit and 8-bit BCH code, respectively).

Table 188. Number of ECC parity bytes per sector

Function	BCH code	
Error correction	4-bit	8-bit
Number of parity bytes	7 bytes	13 bytes

The BCH module detects errors and returns the bit position for each sector. To correct the errors, the software needs to flip the error bits at the provided bit position stored in the FMC_BCHPBRx register.

The BCH ECC controller supports 8-bit and 16-bit NAND flash memories.

During a page program operation, the BCH encoder generates the ECC bytes (7 or 13 bytes) for each sector of 512 bytes that must be written in the NAND flash spare area with a programmable offset. It must take into account some metadata at the beginning of the spare area. This operation must be executed for each sector. The BCH encoder does not process the spare area: metadata must have their own error protection mechanism.

The total number of ECC bytes (N) is given by the below equation:

$$N = \text{Number of ECC bytes /sector} \times \text{Number of sector/page}$$

Example

For an 8-bit NAND flash with 4096-byte pages and an error correction capability of 4-bit per 512 bytes, the number of sector per page is 8 (4096 / 512). The total number of ECC bytes is consequently:

$$N = 7 \times 8 = 56 \text{ bytes}$$

The user application has to make sure that the spare area is large enough to store the metadata and parity bytes.

Page program and read sequences with BCH code are described below.

To perform a page program:

1. Enable write access by setting the WEN bit in the FMC_PCR register.
2. Enable the ECC by setting the ECCEN bit in FMC_PCR register.
3. Write 512 data bytes to the sector in the NAND flash memory page. While the sector is written, the BCH encoder computes the ECC value.
4. Wait until the BCH encoder parity bits are available by monitoring EPBRF bit of FMC_BCHISR register.
5. Read the ECC value available in the FMC_BCHPBRx registers and store it in the NAND flash spare area.
6. Disable ECC by resetting the ECCEN bit in FMC_PCR register.
7. Execute again step 2 to 6 for all sectors to be written in NAND flash memory.
8. Launch the page programming.

To perform a page read:

1. Enable read access by resetting the WEN bit in the FMC_PCR register.
2. Enable the ECC by stetting the ECCEN bit in FMC_PCR register.
3. Read the NAND page sector (512 bytes).
4. Read the corresponding ECC bytes (7 or 13 bytes) from the NAND flash spare area.
5. During the read of data and parity bits, the syndrome is calculated. Then the error location search is launched.
6. Wait until the decoding error is ready (DERF bit set in the FMC_BCHISR register), then read the decoding results (errors detected if any, correctable or not, number of errors, bit position) in FMC_BCHDSR0 and FMC_BCHDSRx registers.
7. Thanks to the decoding results, the software can correct the errors in the sector (when possible).
8. Disable ECC by resetting the ECCEN bit in FMC_PCR register.
9. Execute again step 2 to 7 for all relevant sectors of the page.

27.9.7 FMC command sequencer

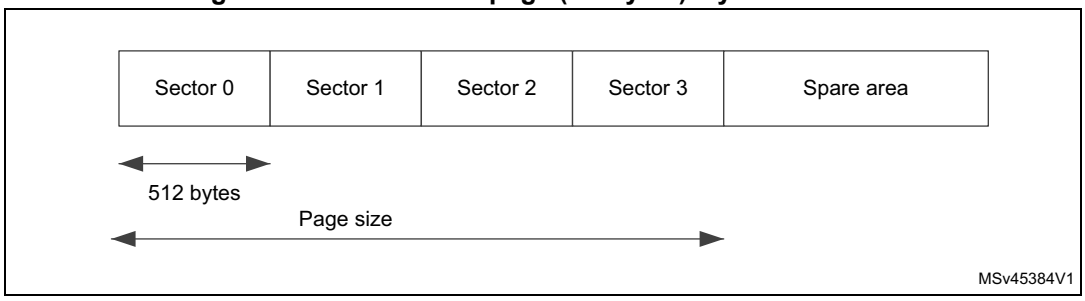
NAND flash page programming and reading can be performed by the CPU as described in the above sections.
Data sector programming/reading as well as ECC byte programming/reading can also be managed automatically by the FMC command sequencer for command and address transfers, and by DMA for data transfers.

The purpose of the command sequencer is to ease the programming/reading of NAND flash pages with ECC computation as well as free the CPU of sequencing tasks. The command sequencer enables the software to work on full NAND flash pages instead of sectors, which is more efficient and essential to achieve high throughput. The sequencer handles all the sectors in the page. It does not support sub-page read or program operations.

One DMA channel is required for write operations and two DMA channels for read operations.

A specific NAND flash page layout must be respected when using the BCH algorithm. See *Figure 182* for an example of 2Kbyte page BCH layout.

Figure 182. NAND flash page (2 Kbytes) layout with BCH



Page program sequence

1. A data buffer located in SRAM has to be initialized with the data to be written to the NAND flash page.
2. The CPU configures the FMC command sequencer registers: page program command (0x80), address bytes, change write column command (0x85), ECC address offset, page size, sector size and sector number.

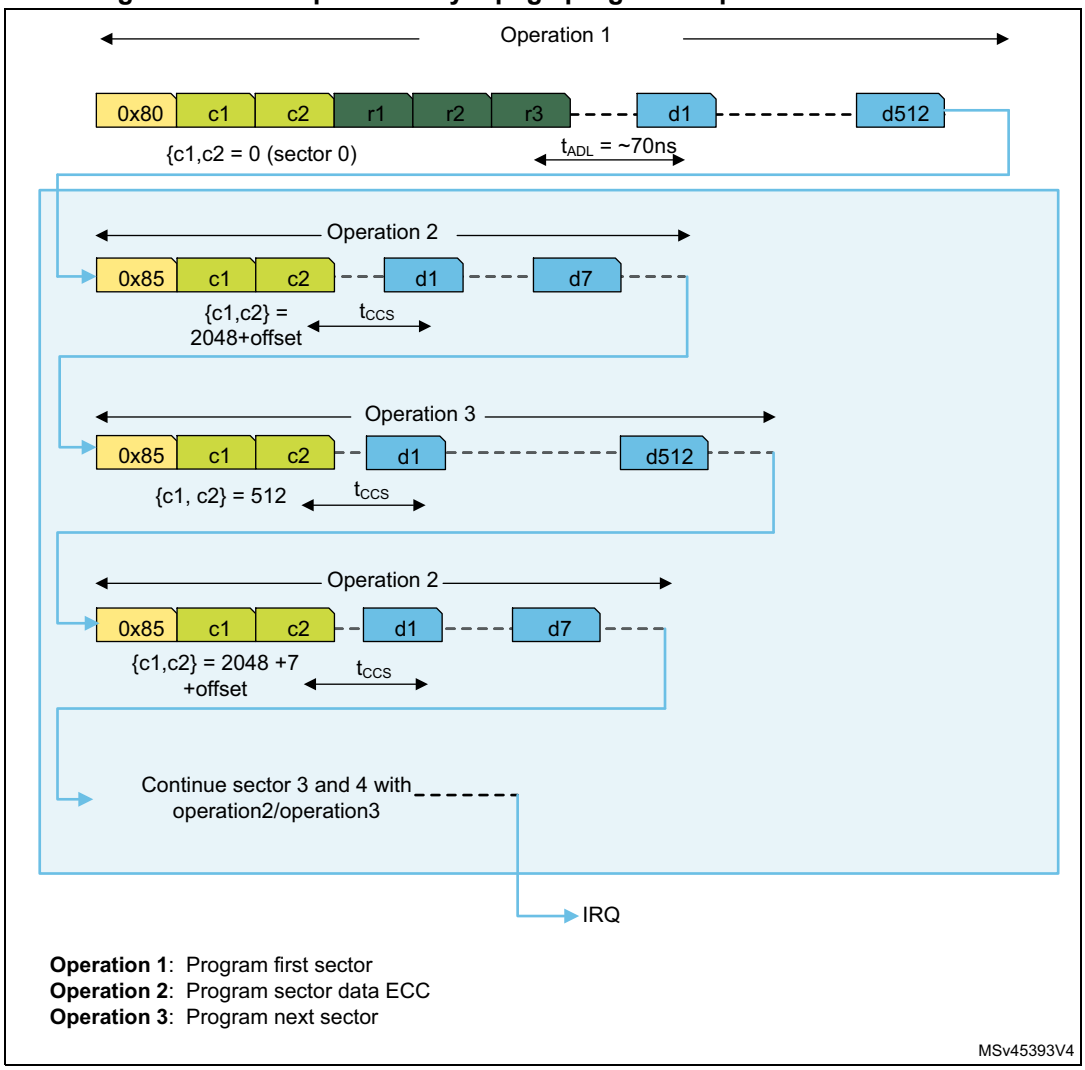
   It also selects the ECC algorithm (Hamming or BCH) and configures one DMA channel to transfer data from the buffer to the NAND flash page.
3. The CPU then launches the sequencer by setting the CSQSTART bit in the FMC_CSQCR register.
4. The command sequencer automatically generates the control and address cycles to the NAND flash memory, following the programmed timings configured either in FMC_PMEM or FMC_PATT register.
5. A DMA request is issued to start data transfer from the data buffer to NAND flash memory.
6. When the full sector is programmed, go to step 7 if ECC is enabled. Otherwise, jump to step 9.
7. The FMC command sequencer automatically issues a change write column command (0x85) followed by two address cycles. The command to be issued and the number of

address cycles are programmed in FMC_CSQCFGR2 register. The address is automatically generated from the programmed ECC address offset.

8. Once the parity bytes are available in ECC encoder (Hamming or BCH) module, they are automatically transferred to the NAND flash spare area. The FMC command sequencer can generate a sector transfer complete interrupt if it has been previously enabled.
9. The sequencer then repeats step 4 (except that it performs only a change write column command instead of a page program command) to step 8 to program the following sectors and their corresponding ECC parity bytes. The ECC address offset is automatically incremented for each sector transfer. When ECC is disabled, the sequencer repeats step 5.
10. The FMC command sequencer can generate a transfer complete interrupt or a transfer complete error interrupt by sector or when all sectors are written.
11. The CPU can program any further metadata in the NAND flash area, taking care not to overwrite parity bytes.
12. The program command (0x10) can then be written to the command section to program the NAND flash memory.
13. An interrupt, if enabled, is generated on the rising edge of the Ready/Busy (RNB) signal to indicate that the page program operation is complete.

Refer to *Figure 183* for an overview of the page program sequence with BCH.

Figure 183. Example of 2 Kbyte page program sequence with 4-bit BCH



Page read sequence

1. A data buffer is allocated in SRAM to store the NAND flash page data. A decoding status buffer is also allocated to store result of ECC decoding.
2. The CPU programs the command sequencer registers: the page read first cycle command (0x00), address bytes, the page read second cycle command (0x30), change read column command (0x05), spare area offset, page size, sector size and sector number.

   It configures the ECC algorithm if required (Hamming or BCH), one DMA channel to handle data transfers from the NAND flash page to the data buffer, and a second DMA

channel to transfer ECC decoding results from ECC status registers to the RAM decoding status buffer.

3. The CPU then launches the FMC sequencer by setting the CSQSTART bit in the FMC_CSQCR register.
4. The FMC command sequencer issues the command and address cycles to the NAND flash memory.
5. The command sequencer waits for a Ready/Busy (RNB) rising edge to start data transfer.
6. It sends a DMA request to launch the data transfer from the NAND flash page to the data buffer.
7. When all sector data are read, go to step 8 if ECC is enabled, otherwise go to step 9. The FMC command sequencer can generate a sector transfer complete interrupt if it has been previously enabled.
8. When ECC status registers are empty (because the Hamming error code or the BCH error position bits of previous sector have already been read), the ECC command sequencer automatically issues a change read column command (0xE0) followed by two address cycles and by the 0xE0 command, to read ECC bytes from NAND flash memory and store them in ECC registers. These commands as well as the number of address cycles are programmed in the FMC_CSQCFGR2 register (see *Figure 184*).

   The address in spare area is extracted from the programmed ECC address offset.

   When the Hamming algorithm is used, a DMA request to the second DMA channel is send to transfer ECC codes (FMC_HPR and FMC_HECCR registers) to the decoding status buffer.

   When BCH algorithm is used, once all the parity bytes have been read, the BCH decoder first processes the syndrome to detect errors and then searches for error positions if any.

   As the BCH decoder is pipelined, it is possible to start reading a sector even if the decoding of the previous sector is not complete.

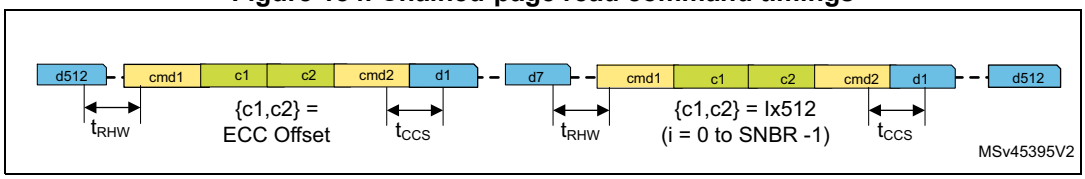
   In parallel of the following steps, when BCH sector decoding is complete, the new error status is stored in the ECC status registers (FMC_BCHDSR0) and a DMA request to the second DMA channel is sent to transfer data from the status registers to the decoding status buffer located in RAM.
9. The FMC command sequencer then repeats step 4 (except that a change read column command is issued instead of the page read first cycle command) to step 8 to read the other sectors and their corresponding ECC bytes. The addresses are issued and incremented following the page size, number of sectors, spare area address offset address and BCH code selection. When the read sequence is complete, the command sequencer waits for t_{RHWL} before issuing a new command sequence (refer to *Figure 183*). When ECC is disabled, the sequencer repeats step 6.
10. Once all sectors within a page and the status registers (FMC_BCHDSRx) are transferred by the command sequencer, the ECC controller can generate the transfer complete interrupt or transfer complete error (when enabled).
11. By reading the decoding status buffer, the CPU can start processing the errors corresponding to each sector. If no error has been detected, the page stored in RAM is validated. Otherwise the CPU corrects the error(s) before validating the page buffer in RAM. If there are too many errors, the CPU can detect the page cannot be corrected and invalidate the data buffer.

Following the selected BCH code, if the number of detected errors exceeds the number of correctable errors, this bit is not significant and the DUE bit is set. If the number of errors the detected by the BCH decoder exceeds the maximum number of detectable errors, DEN and DUE bits are not significant.

The DMA channels can also send interrupts each time a data transfer is complete. This allows the CPU to process error(s) in a sector as soon as it is available.

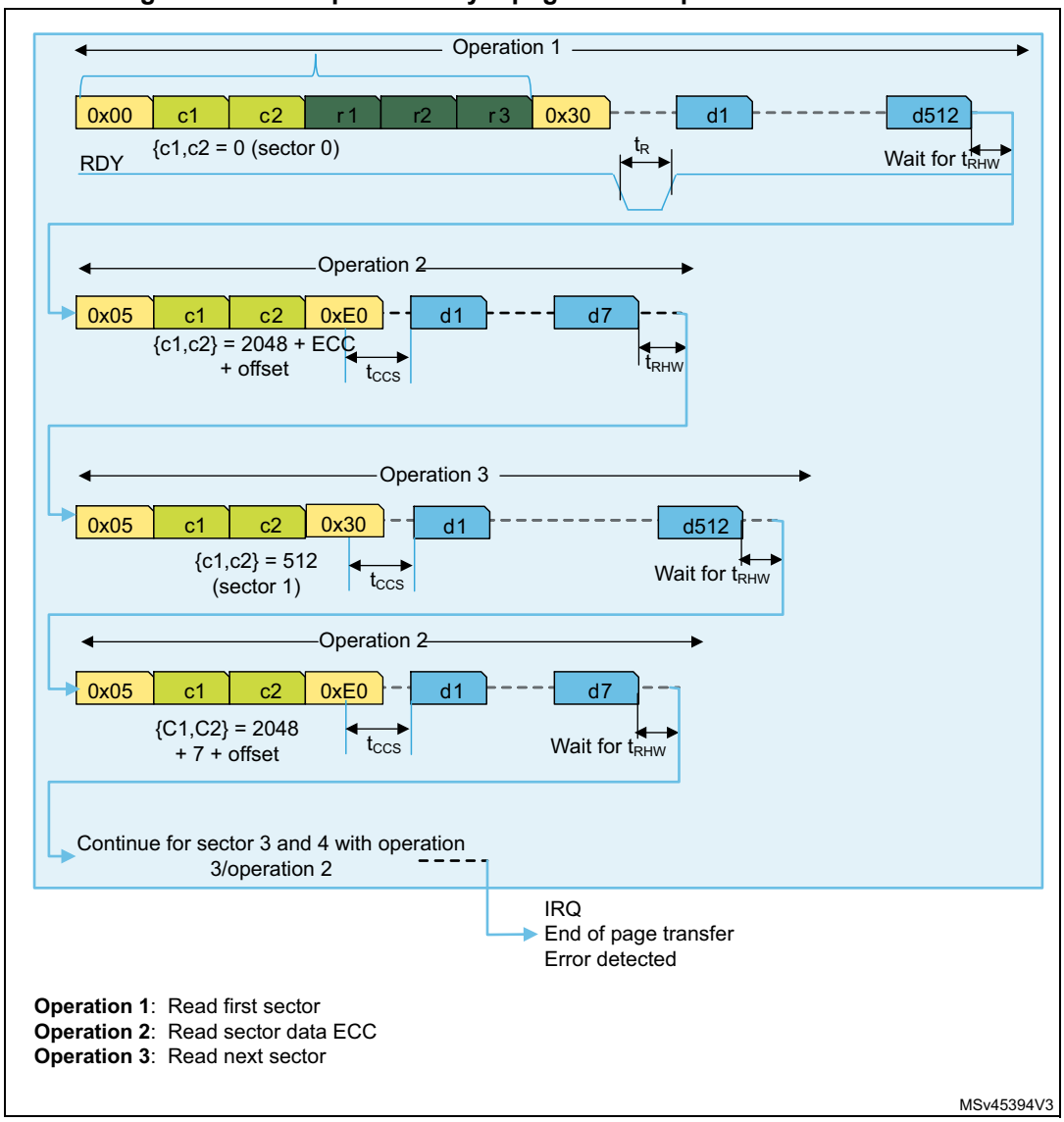
Note: When using the sequencer and DMA channels, the CPU must not read the FMC_BCHPBRx and FMC_BCHDSRx registers before data of all sectors are transferred, that is when TCF flag is set in the FMC_CSQISR register.

Figure 184. Chained-page read command timings



An example of 2 Kbyte page read sequence with BCH is described in *Figure 185*. This example corresponds to four 512-byte sectors and 7 ECC bytes per data 512 bytes.

Figure 185. Example of 2-Kbyte page read sequence with 4-bit BCH



The BCH controller resets the ECC computation whenever a page read (0x00), a page program (0x80), a random read (0x05) or a random program command ((0x85) is issued.

When only part of a page is written/read using the BCH, the command sequencer issues a random read/write command sequence only for the number of sectors to be read/written. This is done by programming the number of sector to be processed.

FMC command sequencer configuration

1. Configure the FMC_CSQCFGR1 register: program the command1 and the command 2 (in case of read access), the number of address cycles to be issued, configure the read or write sequence, and select the timings to be used.
2. Configure the address bytes in FMC_CSQAR1 and FMC_CSQAR2 register.
3. In the FMC_CSQCFGR2 register, configure the change write or change read column commands and the timings to be used for ECC. Program the address offset for the random columns in the FMC_CSQAR2 register. Enable the DMA if it is needed for ECC status register transfer.
4. Configure the number of sectors and the address cycle timings in the FMC_CSQCFGR3 register.
5. Configure the ECC sector size, and select read or write access in the FMC_PCR register.
6. Enable the SQSDTEN bit in the FMC_CSQFCGR2 register. The ECCEN bit in the FMC_PCR register kept at 0.
7. Transfer complete, and transfer error complete or sector transfer complete can be enabled through the FMC_CSQIER register.

27.9.8 NAND flash controller interrupt

An interrupt can be generated on the following events:

- RNB rising edge: an interrupt is generated when a rising edge is detected on RNB NAND flash signal
- RNB falling edge: an interrupt is generated when a falling edge is detected on RNB NAND flash signal
- RNB high-level: an interrupt is generated when a high-level is detected on RNB NAND flash signal.
- Transfer complete: an interrupt is generated when the FMC command sequencer has completed all data transfers for all sectors or when the command sequencer was aborted.
- Sector complete interrupt: an interrupt is generated when the FMC command sequencer has completed all data transfers for one sector.
- Sector error interrupt: an interrupt is generated when the FMC command sequencer has completed a sector transfer with at least one error detected.
- Sector uncorrectable error interrupt: an interrupt is generated when the FMC command sequencer has completed a sector transfer and detected an uncorrectable error.
- Command transfer complete: the command sequencer has completed the transfer of programmed commands and addresses.
- BCH encoder parity bits ready: an interrupt is generated when the BCH encoder has finished processing parity bits.
- BCH decoder syndrome ready: an interrupt is generated when the BCH decoder has finished processing the syndrome and knows if errors have been detected or not.
- BCH decoder error found: an interrupt is generated when the BCH decoder has detected an error.
- BCH decoder error ready: an interrupt is generated when the BCH decoder has finished processing errors, has identified error locations and can then correct them.
- BCH decoder uncorrectable error: an interrupt is generated when the BCH decoder has detected that the errors cannot be corrected (more than four or eight errors according to BCH error correction capability configuration).

The interrupt source can be enabled or disabled separately through the FMC_ IER, FMC_CSQIER and FMC_BCHIER registers.

Table 189. FMC NAND controller Interrupt request

Interrupt event	Event Flag	Enable control bit
Transfer complete	TCF in FMC_CSQISR	TCIE in FMC_CSQIER
Sector complete	SCF in FMC_CSQISR	SCIE in FMC_CSQIER
Sector error	SEF in FMC_CSQISR	SEIE in FMC_CSQIER
Sector uncorrectable error	SUEF in FMC_CSQISR	SUEIE in FMC_CSQIER
Command transfer complete	CMDTCF in FMC_CSQISR	CMDTCIE in FMC_CSQIER
BCH encoder parity bits ready	EPBRF in FMC_BCHISR	EPBRIE in FMC_BCHIER
BCH decoder syndrome ready	DSRF in FMC_BCHISR	DSRIE in FMC_BCHIER
BCH decoder error found	DEFF in FMC_BCHISR	DEFIE in FMC_BCHIER

Table 189. FMC NAND controller Interrupt request (continued)

Interrupt event	Event Flag	Enable control bit
BCH decoder error ready	DERF in FMC_BCHISR	DERIE in FMC_BCHIER
BCH decoder uncorrectable error	DUEF in FMC_BCHISR	DUEIE in FMC_BCHIER

27.9.9 NAND flash controller registers

NAND flash programmable control register (FMC_PCR)

Address offset: 0x80

Reset value: 0x0007 FE08

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	WEN	BCHECC	Res.	Res.	Res.	Res.	ECCSS[2:0]			TAR [3]
						rw	rw					rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TAR[2:0]			TCLR[3:0]				ECCALG	Res.	ECCEN	PWID[1:0]		Res.	PBKEN	PWAITEN	Res.
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw		rw	rw	

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

Bit 25 **WEN:** Write enable

This bit enables read or write access. It must be configured when using the FMC sequencer or when the ECC with BCH code is enabled. It can only be modified when ECCEN bit is reset.

0: Read access enabled

1: Write access enabled

Bit 24 **BCHECC:** BCH error correction capability

0: 4-bit BCH (4-bit error correction and 8-bit error detection per 512 bytes)

1: 8-bit BCH (8-bit error correction and 16-bit error detection per 512 bytes)

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:17 **ECCSS[2:0]:** ECC sector size (used to access spare area)

These bits define the sector size used for the ECC:

000: 256 bytes

001: 512 bytes

010: 1024 bytes

011: 2048 bytes (default)

100: 4096 bytes

101: Reserved.

11X: Reserved.

When the ECC is enabled with BCH code, the sector size must be programmed to 512 bytes.

Bits 16:13 **TAR[3:0]:** ALE to RE delay.

These bits define the time from ALE low to RE low in number of fmc_ker_ck clock cycles:
t_ar ≤ (TAR +1) × $t_{fmc_ker_ck}$
where $t_{fmc_ker_ck}$ is the FMC clock period and *t*_ar is the ALE to RE timing of NAND flash memories.
0000: 1 * fmc_ker_ck cycle
1111: 16 * fmc_ker_ck cycles (default)

Bits 12:9 **TCLR[3:0]:** CLE to RE delay.

These bits define the time from CLE low to RE low in number of fmc_ker_ck clock cycles:
t_clr ≤ (TCLR +1) × $t_{fmc_ker_ck}$
where $t_{fmc_ker_ck}$ is the FMC clock period and *t*_clr is the CLE to RE timing of NAND flash memories.
0000: 1 * fmc_ker_ck cycle
1111: 16 * fmc_ker_ck cycles (default)

Bit 8 **ECCALG:** ECC algorithm

0: Hamming code is selected (default).
1: BCH code is selected.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **ECCEN:** ECC computation logic enable bit

0: ECC logic is disabled and reset (default after reset),
1: ECC logic is enabled.
This bit must be kept reset when using the FMC sequencer

Bits 5:4 **PWID[1:0]:** Data bus width

These bits define the external memory device width.
00: 8 bits (default after reset).
01: 16 bits
1X: Reserved.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **PBKEN:** NAND flash memory region enable bit

This bit enables the memory region. Accessing a disabled memory region causes an AXI bus error.
0: Corresponding memory region is disabled (default after reset)
1: Corresponding memory region is enabled.

Note: The NAND flash memory region must be disabled only when all NAND flash controller commands are complete.

Bit 1 **PWAITEN:** Wait feature enable bit

This bit enables the Wait feature for the NAND flash memory region:
0: disabled (default)
1: enabled

Bit 0 Reserved, must be kept at reset value.

FMC status register (FMC_SR)

Address offset: 0x84

Reset value: 0x0000 0053

This register contains information about the AXI interface isolation status and the NAND write requests status. The FMC has to be disabled before modifying some registers. As requests might be pending, it is necessary to wait till the AXI interface is stable and the core of the block is totally isolated from its AXI interface before reconfiguring the registers.

The PEF and NWEF bits indicate the status of the pipe. If Hamming algorithm is used, the ECC is calculated while data are written to the memory. To read the correct ECC, the software must consequently wait until no write request to the NAND controller are pending, by polling PEF and NWRF bits.

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

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **NWRF:** NAND write request flag

This bit provides the status of the write request issued to the NAND flash memory. When this bit is set, all write requests to the NAND controller are served.

0: NAND flash write requests are pending

1: No NAND flash write requests pending

Bit 5 Reserved, must be kept at reset value.

Bit 4 **PEF:** Pipe Empty Flag

This bit indicates if the pipe contains requests.

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **ISOST[1:0]:** FMC isolation state with respect to the AXI interface

00: FMC is not isolated; AXI transactions are treated by the FMC and its controllers.

01: FMC has been enabled (FMCEN = 1) and waits for the end of pending AXI transactions (which leads to error responses) before going to “non-isolated” state.

10: FMC has been disabled (FMCEN = 0) and waits for the end of pending AXI transactions before going to isolation state.

11: FMC is isolated from its AXI interface. All AXI requests lead to error responses.

FMC common memory space timing register (FMC_PMEM)

Address offset: 0x88

Reset value: 0x0A0A 0A0A

The FMC_PMEM read/write register contains NAND flash memory region timing information. This information is used to access the NAND flash common memory space for command, address write accesses or data read/write accesses.

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

Bits 31:24 **MEMHIZ[7:0]:** Common memory data bus Hi-Z time

These bits define the number of fmc_ker_ck clock cycles (+1) during which the data bus is kept Hi-Z after starting a NAND flash write access to the common memory space. This is only valid for write transactions:

0000 0000: 1 * fmc_ker_ck cycle

0000 0001: 2 * fmc_ker_ck cycles

.....

1111 1111: 256 * fmc_ker_ck cycles

Bits 23:16 **MEMHOLD[7:0]:** Common memory hold time

These bits define the number of fmc_ker_ck clock cycles (+1) during which the address is held (and data for write accesses) after the command is deasserted (NWE, NOE), during NAND flash read or write accesses to the common memory space:

0000 0000: 1 * fmc_ker_ck cycle

0000 0001: 2 * fmc_ker_ck cycles

.....

1111 1111: 256 * fmc_ker_ck cycles

Bits 15:8 **MEMWAIT[7:0]:** Common memory wait time

These bits define the minimum number of fmc_ker_ck clock cycles (+1) to assert the command (NWE, NOE), during NAND flash read or write accesses to the common memory space. The duration of command assertion is extended if the wait signal (NWAIT) is active (low) at the end of the programmed value of fmc_ker_ck:

0000 0000: 1 * fmc_ker_ck cycles (+ wait cycle introduced by deasserting NWAIT)

0000 0001: 2 * fmc_ker_ck cycles (+ wait cycle introduced by deasserting NWAIT)

.....

1111 1111: 256 * fmc_ker_ck cycles (+ wait cycle introduced by deasserting NWAIT)

Bits 7:0 **MEMSET[7:0]:** Common memory setup time

These bits define the number of fmc_ker_ck clock cycles (+1) to set up the address before the command assertion (NWE, NOE), during NAND flash read or write accesses to the common memory space:

0000 0000: 1 * fmc_ker_ck cycles

0000 0001: 2 * fmc_ker_ck cycles

.....

1111 1111: 256 * fmc_ker_ck cycles

FMC attribute memory space timing registers (FMC_PATT)

Address offset: 0x8C

Reset value: 0x0A0A 0A0A

The FMC_PATT read/write register contains NAND flash memory region timing information. It is used for 8-bit accesses to the NAND flash attribute memory space during the last address write access when the timing differs from previous accesses (for Ready/Busy management, refer to *Section 27.9.5: NAND flash prewait function*).

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

Bits 31:24 **ATTHIZ[7:0]:** Attribute memory data bus Hi-Z time

These bits define the number of fmc_ker_ck clock cycles (+1) during which the data bus is kept in Hi-Z after starting a NAND flash write access to the attribute memory space. These bits are only valid for write transactions:
0000 0000: 1 * fmc_ker_ck cycle
0000 0001: 2 * fmc_ker_ck cycle
.....
1111 1111:256 * fmc_ker_ck cycles

Bits 23:16 **ATTHOLD[7:0]:** Attribute memory hold time

These bits define the number of fmc_ker_ck clock cycles (+1) during which the address is held (and data for write access) after the command deassertion (NWE, NOE), during NAND flash read or write accesses to the attribute memory space:
0000 0000: 1 * fmc_ker_ck cycle
0000 0001: 2 * fmc_ker_ck cycle
.....
1111 1111:256 * fmc_ker_ck cycles.

Bits 15:8 **ATTWAIT[7:0]:** Attribute memory wait time

These bits define the minimum number of x fmc_ker_ck (+1) clock cycles to assert the command (NWE, NOE), during NAND flash read or write accesses to the attribute memory space. The duration of command assertion is extended if the wait signal (NWAIT) is active (low) at the end of the programmed value of fmc_ker_ck:
0000 0000: 1 * fmc_ker_ck cycle (+ wait cycle introduced by deassertion of NWAIT)
0000 0001: 2 * fmc_ker_ck cycles (+ wait cycle introduced by deassertion of NWAIT)
.....
1111 1111: 256 * fmc_ker_ck cycles (+ wait cycle introduced by deasserting NWAIT)

Bits 7:0 **ATTSET[7:0]:** Attribute memory setup time

These bits define the number of fmc_ker_ck clock cycles (+1) for address set up before the command assertion (NWE, NOE), during NAND flash read or write accesses to the attribute memory space:
0000 0000: 1 * fmc_ker_ck cycle
0000 0001: 2 * fmc_ker_ck cycle
.....
1111 1111:256 * fmc_ker_ck cycles.

FMC Hamming parity result registers (FMC_HPR)

Address offset: 0x90

Reset value: 0x0000 0000

This register is used during read accesses in conjunction with the FMC sequencer. It contains the current error correction code value computed by the FMC NAND controller Hamming module. When the FMC sequencer reads data from a NAND flash memory page at the correct address, the data read are automatically processed by the Hamming computation module. When X bytes have been read (according to the sector size ECCSS field in the FMC_PCR register), the CPU must read the computed ECC value from the FMC_HECCR register. It then verifies if these computed parity data are the same as the parity value recorded in the spare area and stored in the and the FMC_HPR, to determine whether a page is valid, and to correct it otherwise. The FMC_HPR register must be cleared after being read by setting the ECCEN bit to 0. To compute a new data block, the ECCEN bit must be set.

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

Bits 31:0 **HPR[31:0]:** Hamming parity result

This field contains the parity value computed by the Hamming ECC computation module. *Table 190* describes the contents of these bitfields.

Table 190. HPR relevant bits

ECCSS[2:0]	Sector size in bytes	HPR bits
000	256	HPR[21:0]
001	512	HPR[23:0]
010	1024	HPR[25:0]
011	2048	HPR[27:0]
100	4096	HPR[29:0]
101	8192	HPR[31:0]

FMC Hamming code ECC result register (FMC_HECCR)

Address offset: 0x94

Reset value: 0x0000 0000

This register contain the current error correction code value computed by the FMC NAND controller Hamming module.When the CPU reads/writes data from/to a NAND flash memory page at the correct address (refer to *Section 27.9.6: NAND ECC controller*), the data read/written from/to the NAND flash memory are automatically processed by the Hamming computation module. When X bytes have been read (according to the sector size ECCSS

field in the FMC_PCR register), the CPU must read the computed ECC value from the FMC_HECCR register. It then verifies if these computed parity data are the same as the parity value recorded in the spare area, to determine whether a page is valid, and to correct it otherwise. The FMC_HECCR register must be cleared after being read by setting the ECCEN bit to 0. To compute a new data block, the ECCEN bit must be set.

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

Bits 31:0 **HECC[31:0]:** ECC result

This field contains the value computed by the Hamming ECC computation logic. *Table 191* describes the contents of these bitfields.

Table 191. ECC result relevant bits(1)

ECCSS[2:0]	Sector size in bytes	HECC bits
000	256	HECC[21:0]
001	512	HECC[23:0]
010	1024	HECC[25:0]
011	2048	HECC[27:0]
100	4096	HECC[29:0]
101	8192	HECC[31:0]

1. Only the relevant bits according to the sector size are valid.

FMC NAND interrupt enable register (FMC_IER)

Address offset: 0x180

Reset value: 0x0000 0000

This register configures the interrupt capability generation when the RNB signal is connected on FMC interrupt pin. An interrupt can be generated on the rising edge, falling edge or high level detected on the RNB signal.

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

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

Bit 2 **IFEE:** Interrupt falling edge detection enable bit
0: Interrupt falling edge detection request disabled
1: Interrupt falling edge detection request enabled

Bit 1 **IHLE:** Interrupt high-level detection enable bit
0: Interrupt high-level detection request disabled
1: Interrupt high-level detection request enabled

Bit 0 **IREE:** Interrupt rising edge detection enable bit
0: Interrupt rising edge detection request disabled
1: Interrupt rising edge detection request enabled

FMC interrupt status register (FMC_ISR)

Address offset: 0x184

Reset value: 0x0000 0000

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

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

Bit 2 **IFEF:** Interrupt falling edge flag
The flag is set by hardware and cleared by setting the CFEIF bit.
0: No interrupt falling edge occurred
1: Interrupt falling edge occurred

Bit 1 **IHLF:** Interrupt high-level flag
The flag is set by hardware and cleared by setting the CHLIF bit.
0: No Interrupt high-level occurred
1: Interrupt high-level occurred

Bit 0 **IREF:** Interrupt rising edge flag
The flag is set by hardware and cleared by setting the CREIF bit.
0: No interrupt rising edge occurred
1: Interrupt rising edge occurred

FMC NAND controller interrupt clear register (FMC_ICR)

Address offset: 0x188

Reset value: 0x0000 0000

Writing 1 in one of these bits clears the corresponding flag in the interrupt status register (FMC_ISR).

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

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

Bit 2 **CIFEF:** Clear Interrupt falling edge flag
0: No effect
1: Clears the IFEF flag in the FMC_ISR register

Bit 1 **CIHLF:** Clear Interrupt high-level flag
0: No effect
1: Clears the IHLF flag in the FMC_ISR register

Bit 0 **CIREF:** Clear Interrupt rising edge flag
0: No effect
1:Clear the IREF flag in the FMC_ISR register

FMC NAND command sequencer control register (FMC_CSQCR)

Address offset: 0x200

Reset value: 0x0000 0000

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

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

Bit 0 **CSQSTART**: Command Sequencer Enable
Writing 1 starts the Command sequencer.
Writing 0 has no effect.

Note: Write this bit to 1 to start the sequencer only when the FMC is enabled by setting FMCEN bit of FMC_CFGR to 1 and the NAND flash controller is activated by setting PBKEN bit of FMC_PCR to 1.

FMC NAND command sequencer configuration register 1 (FMC_CSQCFGR1)

Address offset: 0x204

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	CMD2T	CMD1T	CMD2[7:0]							
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMD1[7:0]								Res.	ACYNBR[2:0]			Res.	DMADEN	CMD2EN	Res.
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw		rw	rw	

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

Bit 25 **CMD2T:** Command 2 Sequencer timings

This bit specifies if the Command 2 (CMD2) is issued to the NAND flash memory with the timings defined in FMC_PMEM or FMC_PATT register.

0: CMD2 issued with the timings programmed in FMC_PMEM

1: CMD2 issued with the timings programmed in FMC_PATT

Bit 24 **CMD1T:** Command 1 Sequencer timings

This bit specifies if the Command 1 (CMD1) is issued to the NAND flash memory with the timings programmed in FMC_PMEM or FMC_PATT register.

0: CMD1 issued with the timings programmed in FMC_PMEM

1: CMD1 issued with the timings programmed in FMC_PATT

Bits 23:16 **CMD2[7:0]:** Command 2 sequencer

These bits specify if the Command 2 is issued by the command sequencer for the command second cycle during read operations.This command is only issued when the CMD2EN bit is set.

Example: CMD2[7:0] bits must be programmed to 0x30 to issue the read column command.

Bits 15:8 **CMD1[7:0]:** Command 1 sequencer

These bits specify if the Command 1 is issued by the command sequencer for the first cycle during read or write operations.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ACYNBR[2:0]:** Address Cycle number

These bits define the number of address cycles to be generated by the command sequencer when issuing the command programmed in the registers.

000: No address cycle.

001: 1 address cycle

010: 2 address cycles

011: 3 address cycles

100: 4 address cycles

101: 5 address cycles

Others: Reserved.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **DMADEN:** Command sequencer DMA request data enable

When this bit is set, the command sequencer generates a DMA request to the first DMA channel, to transfer data from/to NAND flash memory.

0: No DMA request transfer

1: A DMA request transfer

Bit 1 **CMD2EN:** Command cycle 2 Enable

This bit specifies that the command cycle 2 has to be generated by the command sequencer.

0: Command cycle 2 not issued.

1: Command cycle 2 (programmed CMD2[7:0]) sent by the command sequencer to the NAND flash memory after the address cycles.

Bit 0 Reserved, must be kept at reset value.

FMC NAND command sequencer configuration register 2 (FMC_CSQCFGR2)

Address offset:0x208

Reset value: 0x0000 0000

This register is used to configure the command sequencer to issue random read/ write commands to read/ write data by sector and automatically read/write data from NAND flash memory at a programmable address offset. This is useful when performing a sector read/write operation followed by an ECC read/write operation in the NAND flash spare area.The command sequencer generates the random commands until all the sectors are read/written.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	RCMD2T	RCMD1T	RCMD2[7:0]							
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RCMD1[7:0]								Res.	Res.	Res.	Res.	Res.	DMASEN	RCMD2EN	SQSDTEN
rw	rw	rw	rw	rw	rw	rw	rw						rw	rw	rw

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

Bit 25 **RCMD2T:** Command 1 sequencer timings

This bit specifies if the CMD1 is issued to the NAND flash memory with the timings programmed in FMC_PMEM or FMC_PATT register.

0: CMD1 issued with the timings programmed in FMC_PMEM

1: CMD1 issued with the timings programmed in FMC_PATT

Bit 24 **RCMD1T:** Command 1 sequencer timings

This bit specifies if the CMD1 is issued to the NAND flash memory with the timings flash memory in FMC_PMEM or FMC_PATT register.

0: CMD1 issued with the timings programmed in FMC_PMEM

1: CMD1 issued with the timings programmed in FMC_PATT

Bits 23:16 **RCMD2[7:0]:** Random Command 2 sequencer

These bits specify that the command 2 is issued by the command sequencer for the second cycle during read operations.This command is only issued when the RCMD2EN bit is set.
Example: the RCMD2 (0xE0) is required to issue the random read column command.

Bits 15:8 **RCMD1[7:0]:** Random Command 1 sequencer

These bits specify that the command 1 value is issued by the command sequencer for the first cycle during read or write accesses.

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **DMASEN:** Command sequencer DMA request decoding status enable

When this bit is set, the command sequencer generates a DMA request to the second DMA channel when valid bits are available in the ECC status register at the end of ECC processing (decoding).
0: No DMA request used for ECC status registers transfer
1: A DMA request used for ECC status registers transfer

Note: This bit must be set only during a page read sequence to offload the saving of the ECC results between sectors by using a second DMA channel.

Bit 1 **RCMD2EN:** Random Command 2 sequencer enable

This bit enables the command 2 to be issued by the command sequencer during read accesses. This command is issued only when the CMD2SQEN bit is set.
0: Command 2 not issued.
1: Command 2 (CMD2SQ[7:0]) issued by the command sequencer to NAND flash memory after the address cycle.

Bit 0 **SQSDTEN:** Sequencer spare data transfer enable

This bit enables the sequencer to access the spare data area after each sector transfer. It also enables the ECC when using the sequencer. ECCEN bit in FMC_PCR register must be reset.
0: ECC disabled and spare data area not accessed by the sequencer
1: ECC enabled and spare data area read or programmed by the sequencer after each sector transfer

FMC NAND sequencer configuration register 3 (FMC_CSQCFGR3)

Address offset: 0x20C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RAC2T	RAC1T	SDT	AC5T	AC4T	AC3T	AC2T	AC1T
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	SNBR[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
		rw	rw	rw	rw	rw	rw								

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **RAC2T:** Random Address cycle 2 sequencer timings

This bit specifies if the random Address cycle 2 (ADDC2) is issued to the NAND flash memory with the timings programmed in FMC_PMEM or in FMC_PATT register.

0: Random ADDC2 issued with the timings programmed in FMC_PMEM

1: Random ADDC2 issued with the timings programmed in FMC_PATT

Bit 22 **RAC1T:** Random Address cycle 1 sequencer timings

This bit specifies if the random Address cycle 1 (ADDC1) is issued to the NAND flash memory with the timings programmed in FMC_PMEM or in FMC_PATT register.

0: Random ADDC1 issued with the timings programmed in FMC_PMEM

1: Random ADDC1 issued with the timings programmed in FMC_PATT

Bit 21 **SDT:** Spare data transfer sequencer timings

This bit specifies if the spare data transfer for read and write operations from/to the NAND flash memory are performed with the timings programmed in FMC_PMEM or FMC_PATT register.

0: Spare data transfer issued with the timings programmed in FMC_PMEM

1: Spare data transfer issued with the timings programmed in FMC_PATT

Bit 20 **AC5T:** Address cycle 5 sequencer timings

This bit defines if the Address cycle 5(ADDC1) is issued to the NAND flash memory with the timings programmed in FMC_PMEM or FMC_PATT register.

0: ADDC5 issued with the timings programmed in FMC_PMEM

1: ADDC5 issued with the timings programmed in FMC_PATT

Bit 19 **AC4T:** Address cycle 4sequencer timings

This bit defines if the Address cycle 4 (ADDC4) is issued to the NAND flash memory with the timings programmed in FMC_PMEM or FMC_PATT register.

0: ADDC4 issued with the timings programmed in FMC_PMEM

1: ADDC4 issued with the timings programmed in FMC_PATT

Bit 18 **AC3T:** Address cycle 3 sequencer timings

This bit defines if the Address cycle 3 (ADDC3) is issued to the NAND flash memory with the timings programmed in FMC_PMEM or FMC_PATT register.

0: ADDC3 issued with the timings programmed in FMC_PMEM

1: ADDC3 issued with the timings programmed in FMC_PATT

Bit 17 **AC2T:** Address cycle 2 sequencer timings

This bit defines if the Address cycle 2 (ADDC2) is issued to the NAND flash memory with the timings programmed in FMC_PMEM or FMC_PATT register.

0: ADDC2 issued with the timings programmed in FMC_PMEM

1: ADDC2 issued with the timings programmed in FMC_PATT

Bit 16 **AC1T:** Address cycle 1 sequencer timings

This bit defines if the Address cycle 1 (ADDC1) is issued to the NAND flash memory with the timings programmed in FMC_PMEM or FMC_PATT register.

0: ADDC1 issued with the timings programmed in FMC_PMEM

1: ADDC1 issued with the timings programmed in FMC_PATT

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

Bits 13:8 **SNBR[5:0]:** Number of sectors to be read/written

These bits define the number of sectors that are processed by the sequencer.
They are used to read/write part of a page or to use the ECC (in particular with the BCH code which requires 512-byte sectors):
000000: 1 sector
000001: 2 sectors
...
111111: 16 sectors
Others: Reserved.

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

FMC NAND command sequencer address register 1 (FMC_CSQAR1)

Address offset: 0x210

Reset value: 0x0000 0000

This register is used to define the value of address cycles 1 to 4 to be issued by the command sequencer.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDC4[7:0]								ADDC3[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDC2[7:0]								ADDC1[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **ADDC4[7:0]:** Address Cycle 4

These bits define the value of address cycle 4 to be issued by the command sequencer during read or write accesses.

Bits 23:16 **ADDC3[7:0]:** Address Cycle 3

These bits define the value of address cycle 2 to be issued by the command sequencer during read or write accesses.

Bits 15:8 **ADDC2[7:0]:** Address Cycle 2

These bits define the value of address cycle 2 to be issued by the command sequencer during read or write accesses.

Bits 7:0 **ADDC1[7:0]:** Address Cycle 1

These bits define the value of address cycle 1 to be issued by the command sequencer during read or write accesses.

FMC NAND command sequencer address register 2 (FMC_CSQAR2)

Address offset: 0x214

Reset value: 0x0000 0000

This register is used to program the fifth address cycle and the address offset in spare area.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SAO[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDC5[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **SAO[15:0]:** Spare Area Address Offset

These bits define the offset of the first ECC byte in the spare area for the program sequence.

Note: The minimum SAQ value is 2 since the first two bytes of the spare area are reserved for Bad block marking and metadata.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 Reserved, must be kept at reset value.

Bits 7:0 **ADDC5[7:0]:** Address Cycle 5

These bits define the value of address cycle 5 to be issued by the command sequencer during read or write accesses.

FMC NAND command sequencer interrupt enable register (FMC_CSQIER)

Address offset: 0x220

Reset value: 0x0000 0000

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

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

Bit 4 **CMDTCIE:** Command Transfer Complete interrupt enable

0: Command Transfer Complete Interrupt disable

1: Command Transfer Complete Interrupt enable

Bit 3 **SUEIE:** Sector Uncorrectable Error interrupt enable

0: Command Transfer Complete Interrupt disable

1: Command Transfer Complete Interrupt enable

Bit 2 **SEIE:** Sector Error interrupt enable
0: Sector Error Interrupt disable
1: Sector Error Interrupt enable

Bit 1 **SCIE:** Sector Complete interrupt enable
0: Sector Complete Interrupt disable
1: Sector Complete Interrupt enable

Bit 0 **TCIE:** Transfer Complete Interrupt enable
0: Transfer Complete Interrupt disable
1: Transfer Complete Interrupt enable

FMC NAND command sequencer interrupt status register (FMC_CSQISR)

Address offset: 0x224

Reset value: 0x0000 0000

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

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

Bit 4 **CMDTCF:** Command Transfer Complete flag
This bit is set when the command sequencer has completed the transfer of programmed commands and addresses.
This bit is cleared by setting CCMDTCF bit. An interrupt can be generated if the CMDTCIE bit is set in the FMC_CSQICR register.
Note: The Sector Transfer Complete interrupt is not needed in sequencer mode.

Bit 3 **SUEF:** Sector Uncorrectable Error flag
This bit is set by hardware and cleared by writing 1 to the CSUEF bit in FMC_CSQICR.
When this bit is set, it indicates that the command sequencer detected an uncorrectable error when decoding a sector. An interrupt can be generated if the SUEIE bit is set in the FMC_CSQICR register.
Note: The SUEF bit is relevant only for BCH code to indicate uncorrectable errors. It must be ignored for Hamming code.

Bit 2 **SEF:** Sector Error flag

This bit is set by hardware and cleared by writing 1 to the CSEF bit in FMC_CSQICR.

This bit is set when the command sequencer has completed the sector data transfer with at least one error detection. An interrupt can be generated if the SEIE bit is set in the FMC_CSQICR register.

Note: The SEF bit is relevant only for BCH code to indicate that errors are present. It must be ignored for Hamming code.

Bit 1 **SCF:** Sector Complete flag

This bit is set by hardware and cleared by writing 1 to the CSCF bit in FMC_CSQICR.

This bit is set when the command sequencer has completed the sector data transfer. An interrupt can be generated if the SCIE bit is set in the FMC_CSQICR register.

Bit 0 **TCF:** Transfer Complete flag

This bit is set by hardware and cleared by writing 1 to the CTCF bit in FMC_CSQICR.

This bit is set when the command sequencer has transferred all the data for all sectors or when the command sequencer was aborted. An interrupt can be generated if the TCIE bit is set in the FMC_CSQICR register.

FMC NAND command sequencer interrupt clear register (FMC_CSQICR)

Address offset: 0x228

Reset value: 0x0000 0000

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

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

Bit 4 **CCMDTCF:** Clear Command Transfer Complete flag

Writing one clears the CMTCF flag in the FMC_CSQISR register

Bit 3 **CSUEF:** Clear Sector uncorrectable Error flag

Writing one clears the SUEF flag in the FMC_CSQISR register

Bit 2 **CSEF:** Clear Sector Error flag

Writing one clears the SEF flag in the FMC_CSQISR register

Bit 1 **CSCF:** Clear Sector Complete flag

Writing one clears the SCF flag in the FMC_CSQISR register

Bit 0 **CTCF**: Clear Transfer Complete flag

Writing one clears the TCF flag in the FMC_CSQISR register

FMC command sequencer error mapping status register (FMC_CSQEMSR)

Address offset: 0x230

Reset value: 0x0000 0000

This register holds a sector error mapping status when the whole transfer is complete.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEM[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **SEM[15:0]:** Sector Error mapping

When SEM[i] bit (0 ≤ i ≤ 15) is set, at least one error has been detected on sector i+1.
As an example, SEM[2] and SEM[1] bits are set when errors are detected in sector 3 and 2.

FMC BCH interrupt enable register (FMC_BCHIER)

Address offset: 0x250

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EPBRIE	DSRIE	DEFIE	DERIE	DUEIE
											rw	rw	rw	rw	rw

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

Bit 4 **EPBRIE:** Decoder Parity Bits Ready Interrupt enable

0: Decoder Parity Bits Ready Interrupt disable
1: Decoder Parity Bits Ready Interrupt enable

Bit 3 **DSRIE:** Decoder Syndrome Ready Interrupt enable

0: Decoder Syndrome Ready Interrupt disable
1: Decoder Syndrome Ready Interrupt enable

Bit 2 **DEFIE:** Decoder Error Found Interrupt enable
0: Decoder Error Found Interrupt disable
1: Decoder Error Found Interrupt enable

Bit 1 **DERIE:** Decoder Error Ready Interrupt enable
0: Decoder Error Ready Interrupt disable
1: Decoder Error Ready Interrupt enable

Bit 0 **DUEIE:** Decoder Uncorrectable Errors Interrupt enable
0: Decoder Uncorrectable Errors Interrupt disable
1: Decoder Uncorrectable Errors Interrupt enable

FMC BCH interrupt and status register (FMC_BCHISR)

Address offset: 0x254

Reset value: 0x0000 0000

This register holds the status of BCH encoder/decoder after processing each sector. When the sequencer is used, this register is automatically cleared and the corresponding interrupt must not be enabled through FMC_BCHIER register.

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

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

Bit 4 **EPBRF:** Encoder Parity Bits Ready flag
This bit is set when the BCH encoder parity bits are available in FMC_BCHPBR1 to 4.

Bit 3 **DSRF:** Decoder Syndrome Ready flag
This bit is set when the syndrome is ready and DEFF is a valid field.

Bit 2 **DEFF:** Decoder Error Found flag
This bit is set by hardware and cleared by writing 1 to CDEFF.
This bit indicates that at least one error has been detected in the sector.

Bit 1 **DERF:** Decoder Error Ready flag
This bit indicates that the decoder has finished searching errors and identifying their location. The number of errors and their location are available in FMC_BCHDSR0 and FMC_BCHDSR1 to 4, respectively.

Bit 0 **DUEF:** Decoder Uncorrectable Errors flag
This field indicates that too many errors have been detected in the sector during the BCH decoding and that the sector is uncorrectable.

FMC BCH interrupt clear register (FMC_BCHICR)

Address offset: 0x258

Reset value: 0x0000 0000

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

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

Bit 4 **CEPBRF:** Clear Encoder Parity Bits Ready flag
Writing 1 clears the EPBRF flag in FMC_BCHISR.

Bit 3 **CDSRF:** Clear Decoder Syndrome Ready flag
Writing 1 clears the DSRF flag in FMC_BCHISR.

Bit 2 **CDEFF:** Clear Decoder Error Found flag
Writing 1 clears the DEFF flag in FMC_BCHISR.

Bit 1 **CDERF:** Clear Decoder Error ready flag
Writing 1 clears the DERF flag in FMC_BCHISR.

Bit 0 **CDUEF:** Clear Decoder Uncorrectable Error flag
Writing 1 clears the DUEF flag in FMC_BCHISR.

FMC BCH parity bits register 1 (FMC_BCHPBR1)

Address offset: 0x260

Reset value: 0x0000 0000

These registers contain the BCH parity bits (BCHPB). For the BCH 4-bit, only BCHPB[51:0] are significant and for the BCH 8-bit BCHPB[103:0] are significant.

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

Bits 31:0 **BCHPB[31:0]:** BCH parity bits

FMC BCH parity bits register 2 (FMC_BCHPBR2)

Address offset: 0x264

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BCHPB[63:48]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BCHPB[47:32]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **BCHPB[63:32]:** BCH parity bits

FMC BCH parity bits register 3 (FMC_BCHPBR3)

Address offset: 0x268

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BCHPB[95:80]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BCHPB[79:64]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **BCHPB[95:64]:** BCH parity bits

FMC BCH parity bits register 4 (FMC_BCHPBR4)

Address offset: 0x26C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BCHPB[103:96]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **BCHPB[103:96]:** BCH parity bits

FMC BCH decoder status register 0 (FMC_BCHDSR0)

Address offset: 0x27C

Reset value: 0x0000 0000

This register contains some fields already available in other registers but that require to be saved when error correction is performed on several sectors at a time (for example a whole NAND flash page). This allows a DMA channel to transfer the content of FMC_BCHDSR0 to 4 to a decoding status buffer.

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

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **DEN[3:0]:** Decoder error number

When DEF bit is set, this field indicates the number of errors detected by the BCH decoder. Following the selected BCH code, if the number of detected errors exceeds the number of correctable errors, this bit is not significant and DUE bit is set.

If the BCH decoder detects a number of errors that exceeds the maximum number of detectable errors, DEN is not significant.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **DEF:** Decoder error found

This field indicates that the decoder has finished searching errors and identifying their location.

The number of errors and their location are available in FMC_BCHDSR0 and FMC_BCHDSR1 to 4, respectively.

Bit 0 **DUE:** Decoder uncorrectable error

This field indicates that too many errors have been detected in the sector during the BCH decoding and that the sector is uncorrectable.

If the BCH decoder detects a number of errors that exceeds the maximum number of detectable errors, DUE is not significant.

FMC BCH decoder status register for memory region 1 (FMC_BCHDSR1)

Address offset: 0x280

Reset value: 0x0000 0000

The maximum error correction capability of the BCH block embedded in the FMC is 8 errors. This register contains the positions of the 1st and 2nd error bits in EBP1 and EPB2 fields, respectively. If less than 8 errors are detected, the useless EPBx fields (x = 1 to 8) are “don’t care”.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	EBP2[12:0]												
			r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	EBP1[12:0]												
			r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:16 **EBP2[12:0]**: Error bit position for error number 2

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:0 **EBP1[12:0]**: Error bit position for error number 1

FMC BCH decoder status register for memory region 2 (FMC_BCHDSR2)

Address offset: 0x284

Reset value: 0x0000 0000

The maximum error correction capability of the BCH block embedded in the FMC is 8 errors. This register contains the positions of the 3rd and 4th error bits in EBP3 and EPB4 fields, respectively. If less than 8 errors are detected, the useless EPBx fields (x = 1 to 8) are don’t care.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	EBP4[12:0]												
			r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	EBP3[12:0]												
			r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:16 **EBP4[12:0]**: Error bit position for error number 4

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:0 **EBP3[12:0]**: Error bit position for error number 3

FMC BCH decoder status register for memory region 3 (FMC_BCHDSR3)

Address offset: 0x288

Reset value: 0x0000 0000

The maximum error correction capability of the BCH block embedded in the FMC is 8 errors. This register contains the positions of the 5th and 6th error bits in EBP5 and EPB6 fields, respectively. If less than 8 errors are detected, the useless EPBx fields (x = 1 to 8) are don't care.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	EBP6[12:0]												
			r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	EBP5[12:0]												
			r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:16 **EBP6[12:0]**: Error bit position for error number 6

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:0 **EBP5[12:0]**: Error bit position for error number 5

FMC BCH decoder status register for memory region 4 (FMC_BCHDSR4)

Address offset: 0x28C

Reset value: 0x0000 0000

The maximum error correction capability of the BCH block embedded in the FMC is 8 errors. This register contains the positions of the 7th and 8th error bits in EBP7 and EPB8 fields, respectively. If less than 8 errors are detected, the useless EPBx fields (x = 1 to 8) are don't care.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	EBP8[12:0]												
			r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	EBP7[12:0]												
			r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:16 **EBP8[12:0]**: Error bit position for error number 8

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:0 **EBP7[12:0]**: Error bit position for error number 7

27.10 SDRAM controller

27.10.1 SDRAM controller main features

The main features of the SDRAM controller are the following:

- Two SDRAM devices with control registers
- 8-bit, 16-bit, 32-bit data bus width with narrow width support using DQM byte masking
- 11- to 13-bit Address Row, 8- to 11-bit Address Column, 2 to 4 internal banks
  Maximum supported device configurations: 4 x 16 M x 32 bits (256 Mbytes), 4 x 16 M x 16 bits (128 Mbytes), 4 x 16 M x 8 bits (64 Mbytes)
- 64-bit AXI interface with word, half-word or byte access
- SDRAM clock ratio programmable to 1, 2 or 3 (fmc_ker_ck, fmc_ker_ck/2 or fmc_ker_ck/3)
- Automatic row and bank boundary crossing management
- Multibank back-to-back access
- Programmable timing parameters
- Automatic refresh operation with programmable refresh rate
- Self-refresh mode
- Power-down mode
- SDRAM power-up initialization by software
- CAS latency of 1,2,3

27.10.2 SDRAM External memory interface signals

Table 192. SDRAM signals

SDRAM signal	I/O type	Description	Alternate function
SDCLK	O	SDRAM clock	
SDCKE[1:0]	O	SDCKE0: SDRAM Bank 1 Clock Enable SDCKE1: SDRAM Bank 2 Clock Enable	
SDNE[1:0]	O	SDNE0: SDRAM Bank 1 chip enable SDNE1: SDRAM Bank 2 chip enable	
A[13:0](1)	O	Address	FMC_A[13:0]
D[31:0]	I/O	Bidirectional data bus	FMC_D[31:0]
BA[1:0]	O	Bank Address	FMC_A[15:14]
NRAS	O	Row Address Strobe	
NCAS	O	Column Address Strobe	
SDNWE	O	Write Enable	
NBL[3:0]	O	Output byte mask for write accesses (memory signal name: DQM[3:0]	FMC_NBL[3:0]

1. A[13:0] address depends on the number of row address bits (NR) and the number of column address bits (NC).
   A[10] is also used as a command parameter.
   A[12] is asserted only when NR = 13.
   A[13] is never asserted.

27.10.3 SDRAM controller functional description

All SDRAM controller outputs (signals, address and data) change on the falling edge of the memory clock (SDCLK).

SDRAM initialization

The initialization sequence is managed by software. If the two devices are used, the initialization sequence must be generated simultaneously to device 1 and device 2 by setting DS1 and DS2 bits in the FMC_SDCMR register:

1. Program the memory device features into the FMC_SDCRx register.The SDRAM clock and RPIPE[1:0] must be programmed in the FMC_SDCR1 register.
2. Program the memory device timing into the FMC_SDTR register.
3. Enable each SDRAM device by setting the corresponding SDEN bit in FMC_SDCRx register.
4. Set the FMCEN bit. The SDRAM clock starts when this step is complete.
5. Wait during the recommended delay period (100 µs). Refer to the SDRAM datasheet for the required delay after power-up.
6. Set the MODE[2:0] bitfield to 0x2 and configure the device select bits (DS1 and DS2) in the FMC_SDCMR register to issue a Precharge all banks (PALL) command.
7. Set the MODE[2:0] bitfield to 0x3 (Auto-refresh) and configure the device select bits (DS1 and DS2). The user application can either generate Auto-refresh commands by burst (NRFS[3:0]) in the FMC_SDCMR register or generate consecutive single Auto-refresh commands. Refer to the SDRAM datasheet for the number of Auto-refresh commands that must be issued during the initialization phase.
8. Perform load mode register: configure the MRD[13:0] bitfield in the FMC_SDCMR register, set the MODE[2:0] bitfield to 0x4, and configure the device select bits (DS1 and DS2) in the FMC_SDCMR register to issue a load mode register command to the memory devices. In particular, the burst length (BL) has to be set.

   If the mode register value is not the same for both SDRAM devices, this step must be repeated twice, once for each device, and the device select bit has to be set accordingly. For mobile SDRAM devices, the MRD[13:0] bitfield is also used to configure the extended mode register while performing the load mode register.
9. Program the refresh rate in the FMC_SDRTR register. The refresh rate corresponds to the delay between refresh cycles. Its value must be adapted depending on the SDRAM device.
10. Set the SDINIT bit for all the enabled devices in the FMC_SDCRx register, to indicate to the hardware that the initialization procedure has been completed for all the enabled devices.

    At the end of this step, the SDRAM device is ready to accept commands. If a system reset occurs during an ongoing SDRAM access, the data bus might still be driven by the SDRAM device.

    Therefore, the SDRAM device must be first initialized again after reset before issuing any new access by the NOR flash/PSRAM/SRAM or NAND flash controller.

    Any AXI request to SDRAM that is occurring before the end of SDRAM initialization returns a bus error and the request is lost.

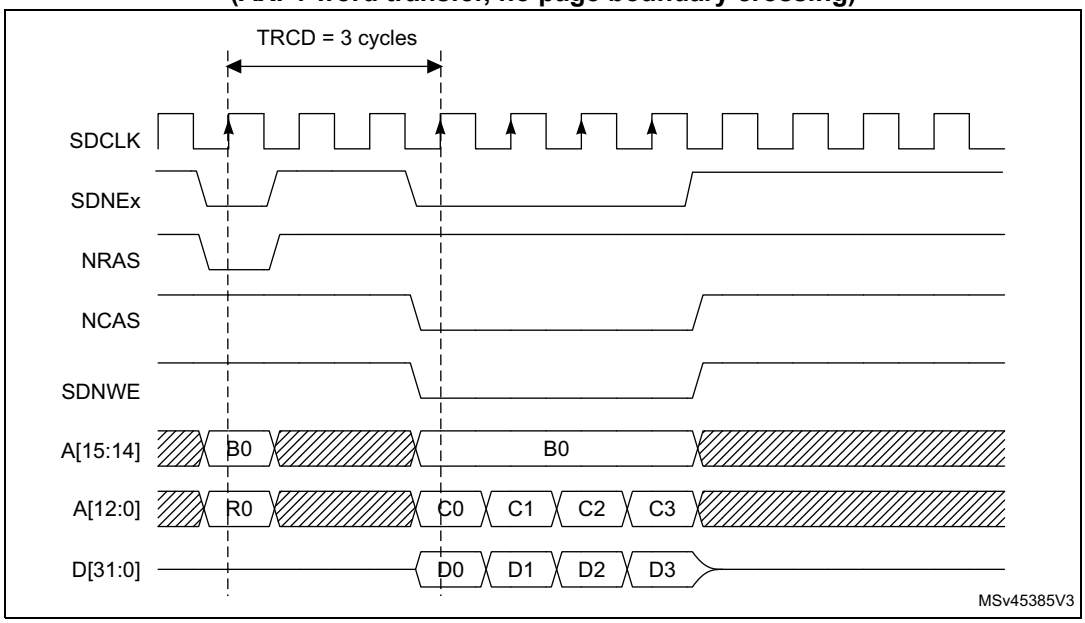
During Initialization, only the commands PALL, AR and LMR are accepted, the other commands are ignored.

SDRAM controller write cycle

The SDRAM controller converts AXI single and burst write transfers into single memory accesses. Consecutive write transfers to the same page or to open pages in different banks can be presented as back-to-back write operations to the SDRAM.

Before performing any write access, the SDRAM bank write protection must be disabled by clearing the WP bit in the FMC_SDCRx register.

Figure 186. SDRAM burst write waveforms (AXI 4-word transfer, no page boundary crossing)



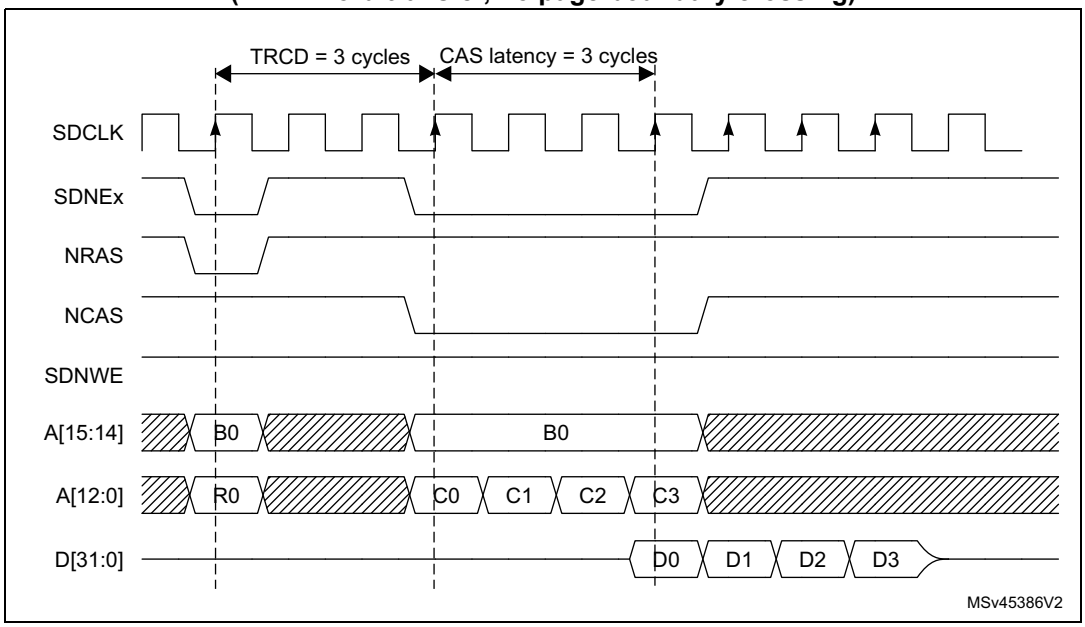
The SDRAM controller always checks the next access.

- If the next access is in the same row or in another active row, the write operation is carried out, and write commands can be back-to-back.
- if the next access targets another row (not active), the SDRAM controller generates a Bank precharge command (PRE), activates the new row and initiates a write command.

SDRAM controller read cycle

The SDRAM controller converts single and burst read transfers into single memory accesses. Consecutive read transfers to the same page or to open pages in different banks can be presented as back-to-back read operations from the SDRAM.

Figure 187. SDRAM burst read waveforms (AXI 4-word transfer, no page boundary crossing)



The address management depends on the next AXI request:

- Next request is performed at sequential address (burst access)

  In this case, the SDRAM controller increments the address.
- Next request is not performed at sequential address
  - If the new read request targets the same row or another active row, the new address is passed to the memory and the master is stalled for the CAS latency period, waiting for the new data from memory.
  - If the new read request does not target an active row, the SDRAM controller generates a Bank precharge command (PRE), activates the new row, and initiates a read command.

Row and bank boundary management

When a read or write access crosses a row boundary, if the next read or write access is sequential and the current access was performed to a row boundary, the SDRAM controller executes the following operations:

1. Precharge of the active row, using the Bank precharge command (PRE)
2. Activation of the new row
3. Start of a read/write command.

At a row boundary, the automatic activation of the next row is supported for all columns and data bus width configurations.

If necessary, the SDRAM controller inserts additional clock cycles between the following commands:

- Between Precharge and Activate commands to match TRP[3:0] parameter (only if the next access is in a different row in the same bank),
- Between Activate and Read commands to match the TRCD[3:0] parameter.

These parameters are defined into the FMC_SDTR register.

Refer to *Figure 188* and *Figure 189* for read and burst write access crossing a row boundary.

Figure 188. SDRAM burst read crossing a row boundary

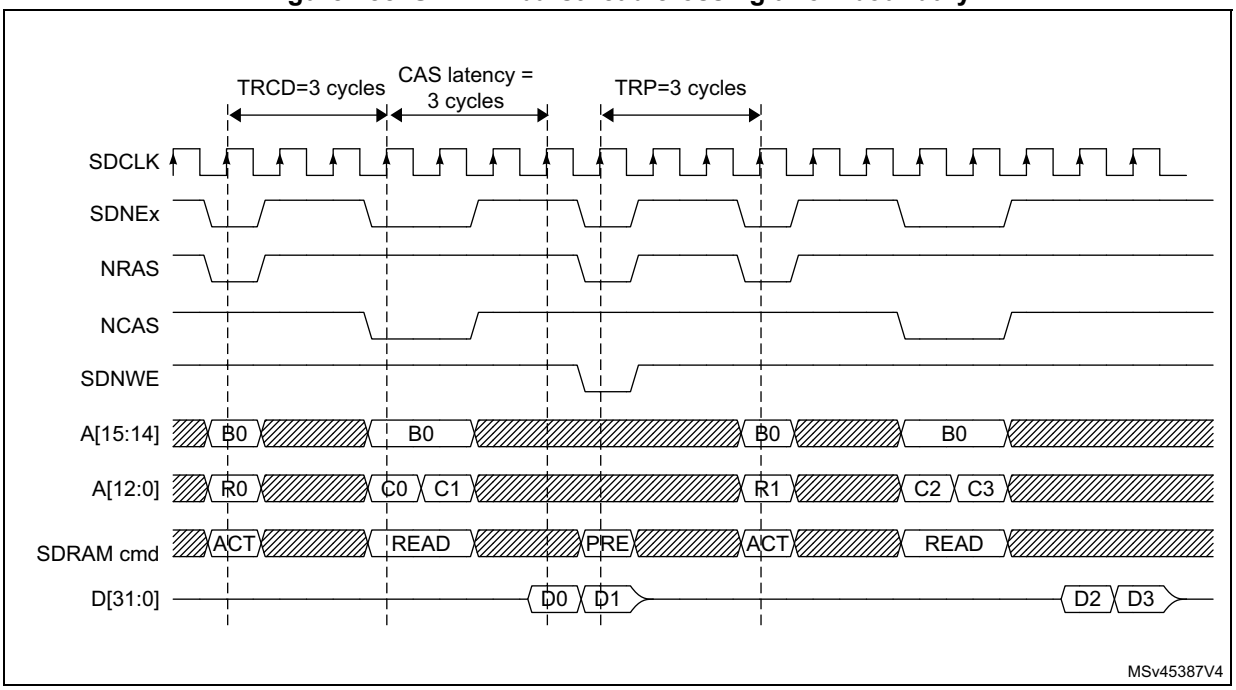
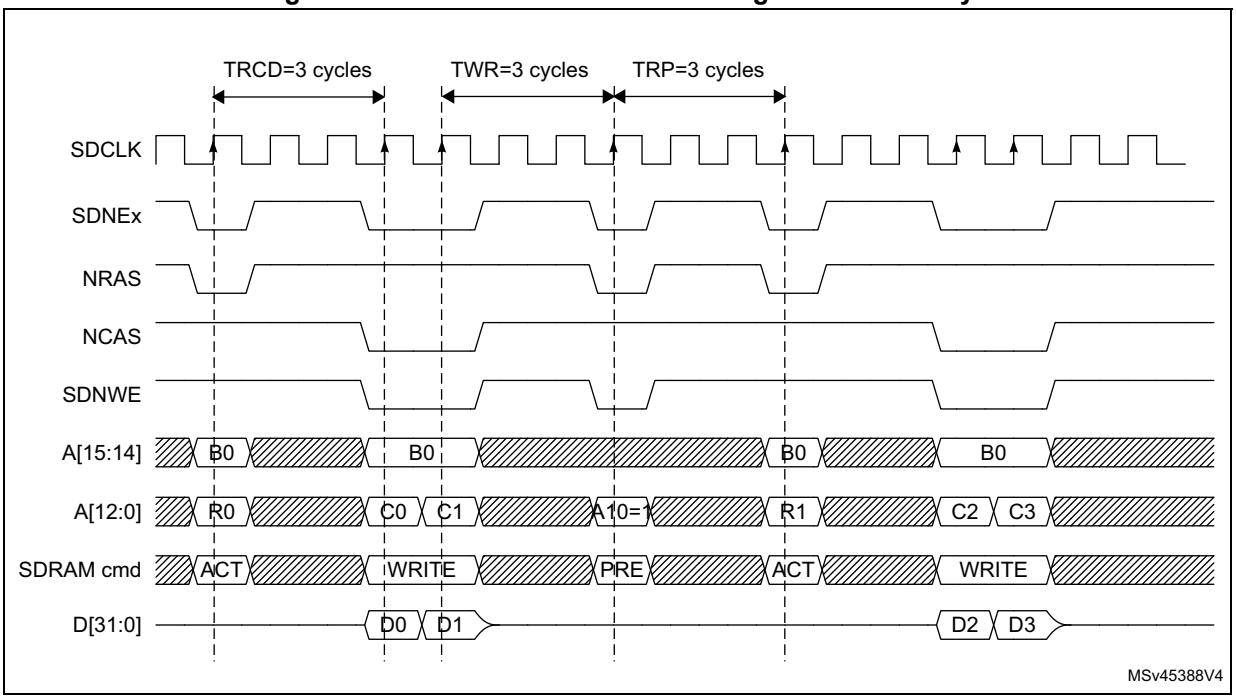


Figure 189. SDRAM burst write crossing a row boundary



An AXI burst operation crossing a row boundary between different banks is automatically split by AXI interconnect because the boundary address is a multiple of 4 Kbytes.

Consecutive read and write operations to the same SDRAM page are back-to-back as shown in *Figure 190* and *Figure 191*.

Figure 190. Read followed by write operation

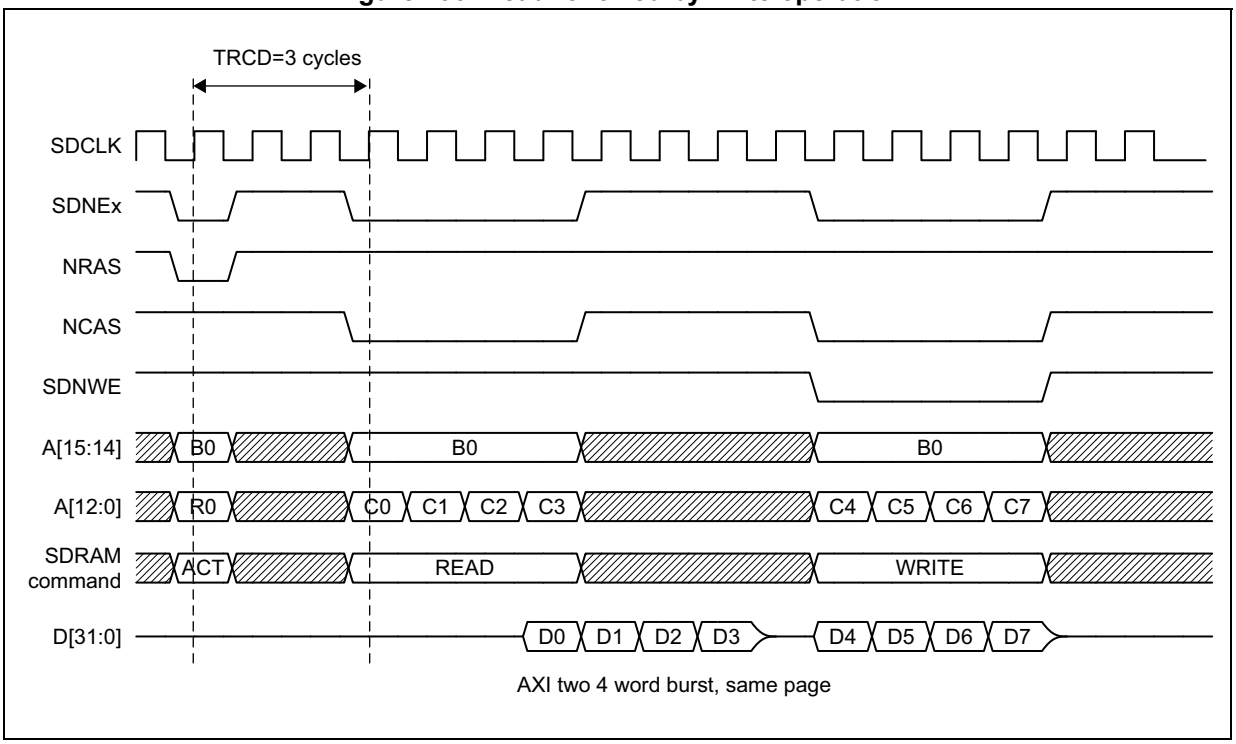
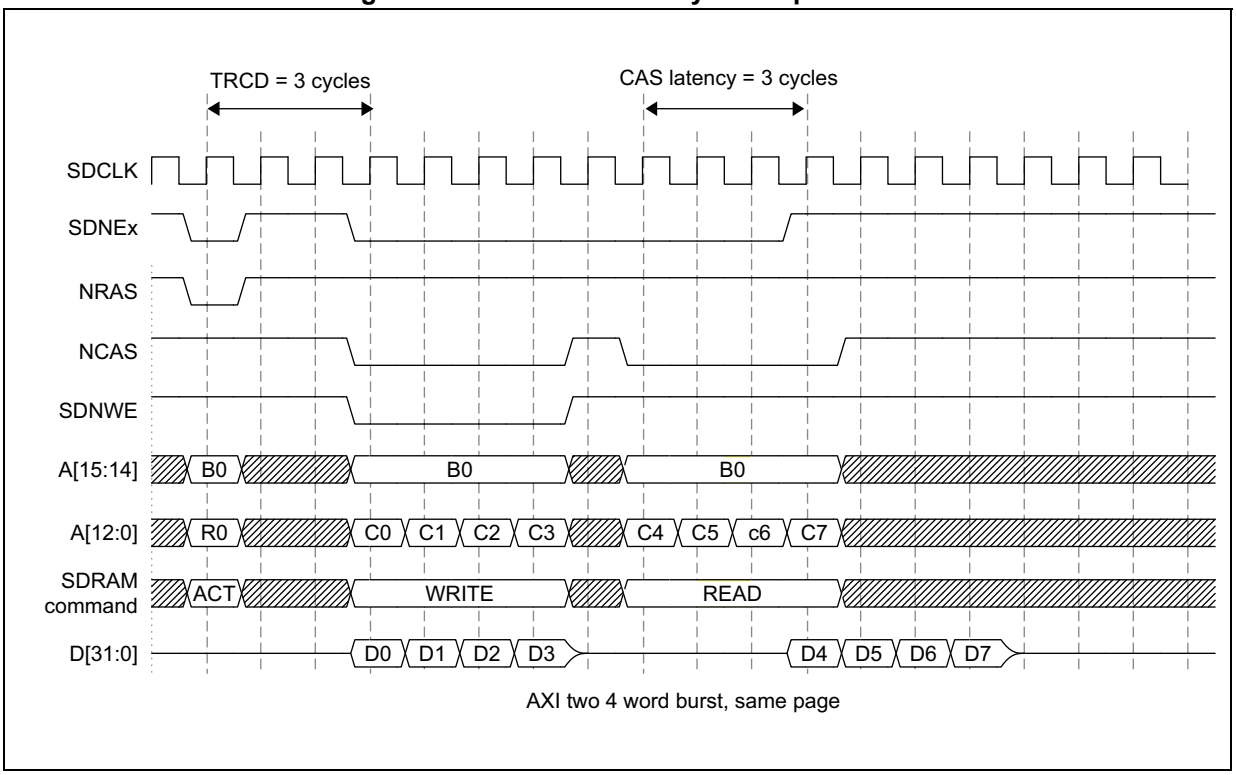


Figure 191. Write followed by read operation



SDRAM controller refresh cycle

The Auto-refresh command is used to refresh the SDRAM device content. The SDRAM controller periodically issues Auto-refresh commands. An internal counter is loaded with the RFSCNT[12:0] value in the register FMC_SDRTR. This value defines the number of memory clock cycles between the refresh cycles (refresh rate). When this counter reaches zero, a refresh command is generated.

The Auto-refresh command is applied on SDRAM device that are enabled (SDEN = 1) and that are not in Self-refresh mode.

If a memory access is in progress, the Auto-refresh request is delayed until the end of the AXI transfer. However, if the memory access and Auto-refresh requests are generated simultaneously, the Auto-refresh takes precedence. In addition, an Auto-refresh request does not suspend consecutive Read or Write commands from an AXI burst. An AXI burst transfer is not interrupted by an Auto-refresh request, except when the Auto-refresh crosses a row or bank boundary. If the memory access occurs during a refresh operation, the request is buffered to be processed when the refresh is complete.

If a new Auto-refresh request occurs while the previous one was not served, the RE (Refresh Error) bit is set in the Status register. An Interrupt is generated if it has been enabled (REIE = 1).

If the SDRAM is not in idle state (not all row are closed), the SDRAM controller generates a PALL (Precharge all banks) command before the Auto-refresh.

If the Auto-refresh command is generated by the FMC_SDCMR register, a PALL command is generated by the SDRAM controller as it is keeping track of the SDRAM opened pages.

27.10.4 Low-power modes

Two low-power modes are available:

- Self-refresh mode

  The Auto-refresh cycles are performed by the SDRAM device itself to retain data without external clocking.
- Power-down mode

  The Auto-refresh cycles are performed by the SDRAM controller.

Self-refresh mode

SDRAM devices can be forced into Self-refresh by software according to the MODE[2:0], DS1, DS2 bits of the FMC_SDCMR register.

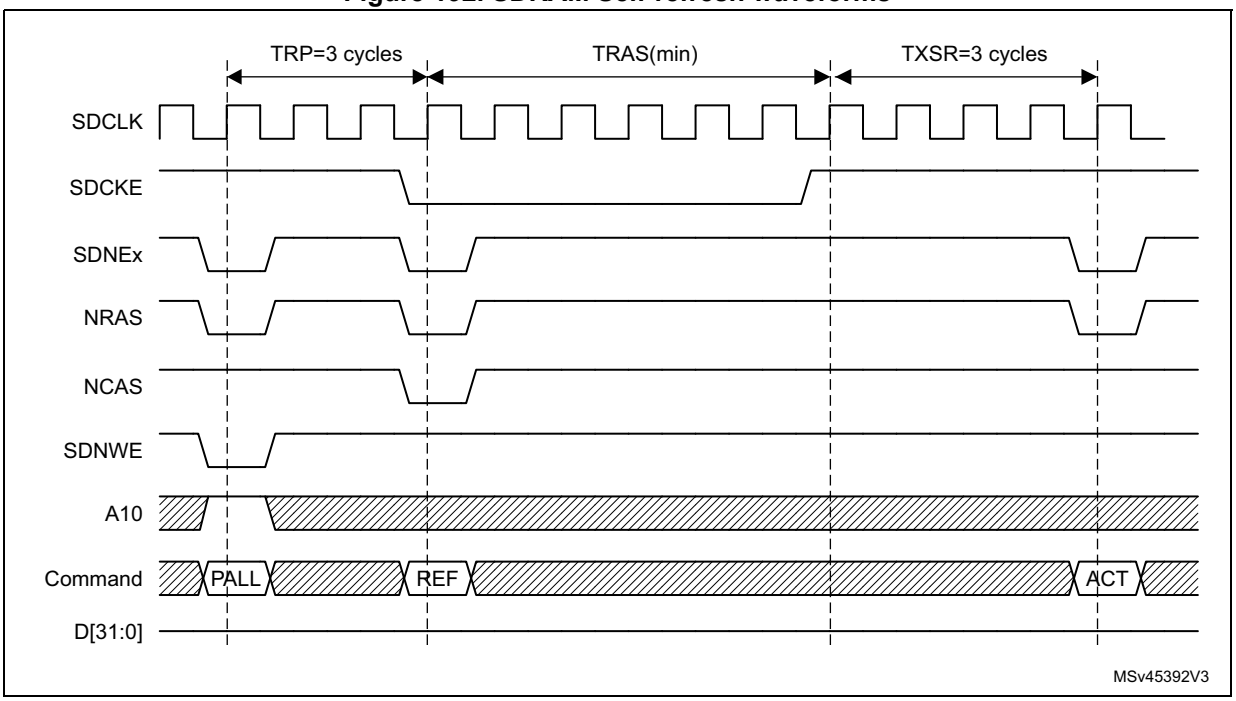
SDRAM devices exit Self-refresh either by a normal mode command from the FMC_SDCMR register (according to DS1 and DS2 bits), or upon the arrival of an AXI request (according to the device enabled through SDEN bit in the FMC_SDCRx).

Note: Exiting Self-refresh by software using FMC_SDCMR is defined on a per-device basis, while the Self-refresh exit upon receiving an AXI transfer is applicable to all devices, irrespectively of the AXI target controller, including NOR flash/PSRAM and NAND flash memories.

Changing the SDRAM frequency is possible only during Self-Refresh: RPIPE[1:0] and SDCLK[1:0] of a device FMC_SDCRx register must not be modified when the device is not in Self-Refresh mode.

Note: During Self-Refresh, any command other than normal mode configured in the FMC_SDCMR register, is ignored by the SDRAM controller.

Figure 192. SDRAM Self-refresh waveforms

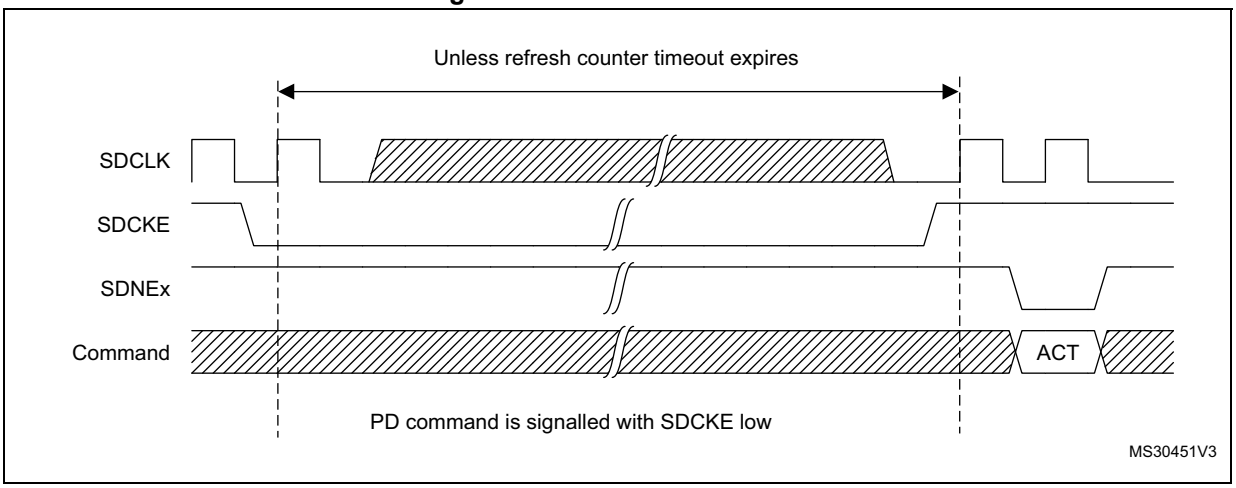


Power-down mode

SDRAM devices can be forced into Power-down by software according to the MODE[2:0], DS1, DS2 bits of the FMC_SDCMR register.
SDRAM devices exit Power-down either by a normal mode command from the FMC_SDCMR register (according to DS1 and DS2 bits) or upon the arrival of an AXI request (according to the device enabled through SDEN bit in the FMC_SDCRx).

Note: During Power-down, any command other than normal mode configured in the FMC_SDCMR register, is ignored by SDRAM controller.

Figure 193. Power-down mode

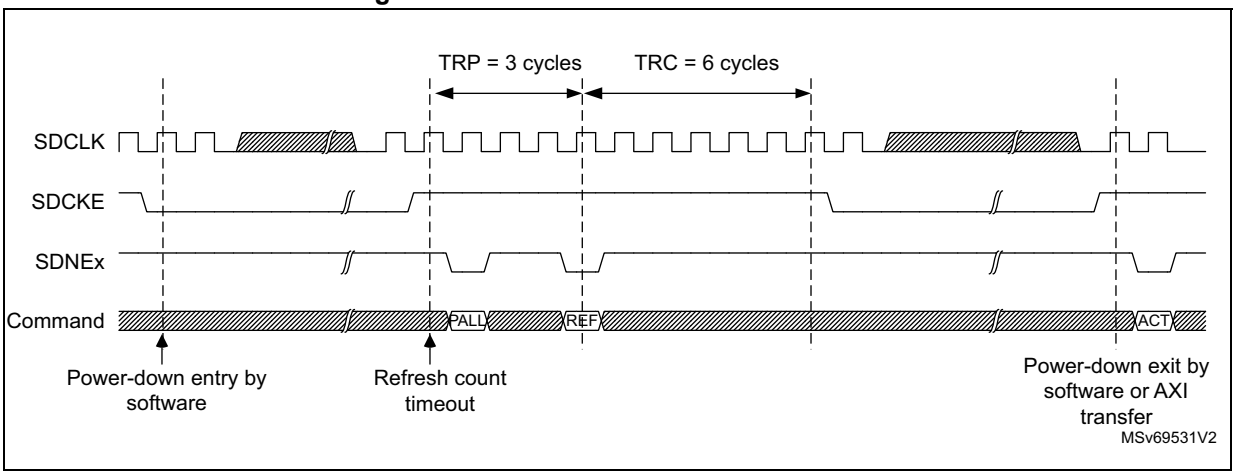


During Power-down mode, all SDRAM device input and output buffers are deactivated except for the SDCKE which remains low.

The SDRAM device cannot remain in Power-down mode longer than the refresh period and cannot perform the Auto-refresh cycles by itself. Therefore, the SDRAM controller carries out the refresh operation by executing the operations below:

1. Exit from Power-down mode and drive the SDCKE high
2. Generate the PALL command only if a row was active during Power-down mode
3. Generate the Auto-refresh command
4. Drive SDCKE low again to return to Power-down mode.

Figure 194. Power-down with Auto-refresh



Clock stop during low-power modes

If all SDRAM devices are simultaneously in low-power mode, the controller stops the SDRAM clock generation. The clock is resumed when at least one device exits from low-power mode.

27.10.5 SDRAM controller registers

SDRAM control registers for SDRAM device 1 (FMC_SDCR1)

Address offset: 0x140

Reset value: 0x0000 02D0

This register contains the control parameters for SDRAM device 1.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RPIPE[1:0]		Res.	SDCLK[1:0]		WP	CAS[1:0]		NB	MWID[1:0]		NR[1:0]		NC[1:0]	
	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 17 **SDINIT:** SDRAM device initialization
1: Initialization is complete and the device is ready to be accessed
0: Initialization is not complete, the AXI accesses are rejected and an AXI slave error is generated.

Bit 16 **SDEN**: SDRAM device enable
1: SDRAM device enabled
0: SDRAM disabled

Bit 15 Reserved, must be kept at reset value.

Bits 14:13 **RPIPE[1:0]:** Read pipe

These bits define the delay, in fmc_ker_ck clock cycles, for reading data after CAS latency, and for both SDRAM devices.

00: No fmc_ker_ck clock cycle delay (default value)

01: One fmc_ker_ck clock cycle delay

10: Two fmc_ker_ck clock cycle delay

11: Reserved.

Bit 12 Reserved, must be kept at reset value.

Bits 11:10 **SDCLK[1:0]**: SDRAM clock configuration

These bits define the SDRAM clock period for both SDRAM devices and allow disabling the clock before changing the frequency.

00: SDCLK clock disabled

01: SDCLK period = 1 * fmc_ker_ck period

10: SDCLK period = 2 * fmc_ker_ck periods

11: SDCLK period = 3 * fmc_ker_ck periods

Note: SDRAM devices must be in Self-refresh mode before a clock frequency change, then all timing parameters are reprogrammed through FMC_SDTR, FMC SDCRx, FMC_SDCMR and load mode register according to CAS latency.

Bit 9 **WP**: Write protection

This bit enables write mode access to the SDRAM bank.

0: Write accesses allowed

1: Write accesses ignored

Bits 8:7 **CAS[1:0]:** CAS Latency

These bits set the SDRAM CAS latency in number of memory clock cycles

00: Reserved.

01: 1 cycle

10: 2 cycles

11: 3 cycles

Note: CAS latency must have the same value for both devices as they operate on the same frequency.

Bit 6 **NB:** Number of banks

This bit sets the number of SDRAM banks.

0: Two banks

1: Four banks

Bits 5:4 **MWID[1:0]:** Memory data bus width.
These bits define the memory device width.
00: 8 bits
01: 16 bits
10: 32 bits
11: Reserved.

Bits 3:2 **NR[1:0]:** Number of row address bits
These bits define the number of bits of a row address.
00: 11 bit
01: 12 bits
10: 13 bits
11: Reserved.

Bits 1:0 **NC[1:0]:** Number of column address bits
These bits define the number of bits of a column address.
00: 8 bits
01: 9 bits
10: 10 bits
11: 11 bits.

SDRAM control registers for SDRAM device 2 (FMC_SDCR2)

Address offset: 0x144

Reset value: 0x0000 02D0

This register contains the control parameters for SDRAM device 2.

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	WP	CAS[1:0]		NB	MWID[1:0]		NR[1:0]		NC[1:0]	
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bit 17 **SDINIT:** SDRAM device initialization
1: Initialization is complete and the device is ready to be accessed
0: Initialization is not complete, the AXI accesses are rejected and an AXI slave error is generated.

Bit 16 **SDEN**: SDRAM device enable
1: SDRAM device enabled
0: SDRAM disabled

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **WP**: Write protection
This bit enables write mode access to the SDRAM bank.
0: Write accesses allowed
1: Write accesses ignored

Bits 8:7 **CAS[1:0]:** CAS Latency
These bits set the SDRAM CAS latency in number of memory clock cycles
00: Reserved.
01: 1 cycle
10: 2 cycles
11: 3 cycles
Note: CAS latency must have the same value for both devices as they operate on the same frequency.

Bit 6 **NB:** Number of banks
This bit sets the number of SDRAM banks.
0: Two banks
1: Four banks

Bits 5:4 **MWID[1:0]:** Memory data bus width.
These bits define the memory device width.
00: 8 bits
01: 16 bits
10: 32 bits
11: Reserved.

Bits 3:2 **NR[1:0]:** Number of row address bits
These bits define the number of bits of a row address.
00: 11 bit
01: 12 bits
10: 13 bits
11: Reserved.

Bits 1:0 **NC[1:0]:** Number of column address bits
These bits define the number of bits of a column address.
00: 8 bits
01: 9 bits
10: 10 bits
11: 11 bits.

SDRAM timing register (FMC_SDTR)

Address offset: 0x148

Reset value: 0x0FFF FFFF

This register contains the timing parameters. It is common to both SDRAM banks.

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

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **TRCD[3:0]:** Row to column delay

This bitfield defines the number of SDRAM clock cycles between an Activate command and a Read or Write command. Its value must meet the SDRAM TRCD timing requirement.

0000: 1 cycle.

0001: 2 cycles

....

1111: 16 cycles

Bits 23:20 **TRP[3:0]:** Row precharge delay

This bitfield defines the number of SDRAM clock cycles between a Precharge command and an Activate or Refresh command. Its value must meet the SDRAM TRP timing requirement.

0000: 1 cycle

0001: 2 cycles

....

1111: 16 cycles

Bits 19:16 **TWR[3:0]:** Recovery delay

This bitfield defines the number of SDRAM clock cycles between a Write command and a Precharge command. Its value must meet the SDRAM TWR timing requirement.

These bits define the delay between a Write and a Precharge command in number of memory clock cycles.

0000: 1 cycle

0001: 2 cycles

....

1111: 16 cycles

Note: The SDRAM write recovery (t_{WR}) can be determined as the highest of the three timings requirements, TWR, TRAS – TRCD and TRC – TRP – TRCD.

Bits 15:12 **TRC[3:0]:** Row cycle delay

This bitfield defines the number of SDRAM clock cycles between two Activate commands or between two Refresh commands. Its value must meet both SDRAM TRC and TRFC timing requirements.

0000: 1 cycle

0001: 2 cycles

....

1111: 16 cycles

Bits 11:8 **TRAS[3:0]:** Self-refresh time

This bitfield defines the number of SDRAM clock cycles between an Activate command and a Precharge command. Its value must meet the SDRAM TRAS timing requirement.

0000: 1 cycle

0001: 2 cycles

....

1111: 16 cycles

Bits 7:4 **TXSR[3:0]:** Exit self-refresh delay

This bitfield defines the number of SDRAM clock cycles between a Self-refresh exit and an Activate command. Its value must meet the SDRAM TXSR timing requirement.

0000: 1 cycle

0001: 2 cycles

....

1111: 16 cycles

Bits 3:0 **TMRD[3:0]:** Load mode register to Activate

This bitfield defines the number of SDRAM clock cycles between a load mode register command and an Activate or Refresh command. Its value must meet the SDRAM TMRD timing requirement.

0000: 1 cycle

0001: 2 cycles

....

1111: 16 cycles

SDRAM command mode register (FMC_SDCMR)

Address offset: 0x150

Reset value: 0x0000 0000

This register contains the command issued when the SDRAM device is accessed. This register is used to initialize the SDRAM device, and to activate the Self-refresh and the Power-down modes. As soon as the MODE[2:0] bitfield is written, the command is issued only to one or to both SDRAM banks according to DS1 and DS2 command bits. This register is the same for both SDRAM banks.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MRD[13:7]						
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MRD[6:0]							NRFS[3:0]				DS1	DS2	MODE[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	w	w	w	w

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:9 **MRD[13:0]:** Mode register definition

The two MSBs, bits 13 and 12, are applied to BA[1:0] to support mode and extended mode register selection by LMR command.

The LSBs, bits 11 to 0, are applied to A[11:0] and define the value programmed by LMR command.

Bits 8:5 **NRFS[3:0]:** Number of Refresh commands

These bits define the number of consecutive Refresh commands.

0000: 1 Refresh cycle

0001: 2 Refresh cycles

....

1110: 15 Refresh cycles

1111: 16 Refresh cycles

Bit 4 **DS1:** Command targeting SDRAM device 1

0: Command not issued to SDRAM device 1

1: Command issued to SDRAM device 1

Bit 3 **DS2:** Command targeting SDRAM device 2

0: Command not issued to SDRAM device 2

1: Command issued to SDRAM device 2

Bits 2:0 **MODE[2:0]:** Command mode

These bits define the command issued to the SDRAM device.

000: Normal mode command (NRM)

010: Precharge all banks command (PALL)

011: Refresh command (REF)

100: Load mode register command (LMR)

101: Self-refresh command (SR)

110: Power-down command (PD)

Others: Reserved, must not be used.

Note: When a command is issued, at least one DSx must be set otherwise the command is ignored.

If two SDRAM devices are used, the Refresh and PALL command must be issued simultaneously to the two devices with DS1 and DS2 bits set.

If only one SDRAM device is used and a command is issued with the associated DSx bit set, the other DSx bit of the unused bank must be kept to 0.

When a device has been put in low-power mode following either a SR or a PD command, only a NRM command can exit it from this mode.

SDRAM refresh timer register (FMC_SDRTR)

Address offset: 0x154

Reset value: 0x0000 0000

This register sets the refresh rate in number of SDCLK clock cycles between the Refresh commands to meet the SDRAM refresh rate.

Example

$$\text{Refresh rate} = 64\ \text{ms} / (8196\text{rows}) = 7.81\mu\text{s}$$

where 64 ms is the SDRAM refresh period.

The refresh rate must be increased to obtain a safe margin.

This 13-bit field is loaded into a timer which is decremented using the SDRAM clock. This timer generates a Refresh command when zero is reached. The RFSCNT[12:0] value must be set at least to 41 SDRAM clock cycles.

As soon as the FMC_SDRTR register is programmed, the timer starts counting. If the value programmed in the register is 0, no refresh is carried out. This register must not be reprogrammed after the initialization procedure to avoid modifying the refresh rate.

Each time a refresh pulse is generated, RFSCNT[12:0] bitfield is reloaded into the counter.

This register is common to SDRAM bank 1 and bank 2.

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

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **REIE:** RES Interrupt Enable

0: Interrupt is disabled

1: An Interrupt is generated if RE = 1

Bits 13:1 **RFSCNT[12:0]:** Refresh Timer Count

This bitfield defines the number of SDRAM clock cycles between Refresh commands. Its value must meet the SDRAM Refresh timing requirement. For example an 8192-cycle refresh every 64 ms requires a 7.8 µs interval between Refresh commands. It is recommended to round down RFSCNT[12:0] to optimize the refresh rate.

Bit 0 **CRE:** Clear Refresh error flag

This bit is used to clear the Refresh Error Flag (RE) in the Status Register.

0: no effect

1: Refresh Error flag is cleared

Note: The programmed RFSCNT[12:0] value must not be equal to the sum of the following timings: TWR+TRP+TRC+TRCD+4 memory clock cycles.

SDRAM status register (FMC_SDSR)

Address offset: 0x158

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMDOK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODES2[1:0]		MODES1[1:0]		RE
r											r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **CMDOK:** Previous command status

This bit defines the status of the previous command.

0: Command not complete

1: Command complete in the SDRAM clock domain

Note: In case of command chaining, CMDOK must be polled between commands, to insure correct command chaining.

Bits 14:5 Reserved, must be kept at reset value.

Bits 4:3 **MODES2[1:0]:** Status mode for SDRAM device 2

This bit defines the status mode of SDRAM device 2.

00: Normal mode, devices are not in Self-refresh or power-down

01: Self-refresh

10: Power-down

Bits 2:1 **MODES1[1:0]:** Status mode for SDRAM device 1

This bit defines the status mode of SDRAM device 1.

00: Normal mode, devices are not in Self-refresh or power-down

01: Self-refresh

10: Power-down

Bit 0 **RE:** Refresh error flag

0: No refresh error has been detected

1: A refresh error has been detected

An interrupt is generated if REIE = 1 and RE = 1

27.11 FMC register map

Table 193. FMC register map

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	FMC_BCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBLSET[1:0]		CSCOUNT1	CSCOUNT0	CBURSTRW	CPSIZE[2:0]			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	Res.	WAITPOL	BURSTEN	Res.	Res.	MWID[1:0]		MTYP[1:0]		MUXEN	MBKEN
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0		1	1			0	1	1	0	1	1

Table 193. FMC register map (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x08	FMC_BCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBLSET[1:0]		CSCOUNT1	CSCOUNT0	CBURSTRW	CPSIZE[2:0]			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	Res.	WAITPOL	BURSTEN	Res.	Res.	MWID[1:0]		MTYP[1:0]		MUXEN	MBKEN
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0		1	1			0	1	0	0	1	0
0x10	FMC_BCR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBLSET[1:0]		CSCOUNT1	CSCOUNT0	CBURSTRW	CPSIZE[2:0]			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	Res.	WAITPOL	BURSTEN	Res.	Res.	MWID[1:0]		MTYP[1:0]		MUXEN	MBKEN
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0		1	1			0	1	0	0	1	0
0x18	FMC_BCR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBLSET[1:0]		CSCOUNT1	CSCOUNT0	CBURSTRW	CPSIZE[2:0]			ASYNCWAIT	EXTMOD	WAITEN	WREN	WAITCFG	Res.	WAITPOL	BURSTEN	Res.	Res.	MWID[1:0]		MTYP[1:0]		MUXEN	MBKEN
	Reset value	0								0	0	0	0	0	0	0	0	0	0	0	0	0		1	1			0	1	0	0	1	0
0x04	FMC_BTR1	DATAHLD		ACCMOD		DATLAT				CLKDIV[3:0]				BUSTURN				DATAST								ADDHLD				ADDSET			
	Reset value	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0C	FMC_BTR2	DATAHLD		ACCMOD		DATLAT				CLKDIV[3:0]				BUSTURN				DATAST								ADDHLD				ADDSET			
	Reset value	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x14	FMC_BTR3	DATAHLD		ACCMOD		DATLAT				CLKDIV[3:0]				BUSTURN				DATAST								ADDHLD				ADDSET			
	Reset value	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x1C	FMC_BTR4	DATAHLD		ACCMOD		DATLAT				CLKDIV[3:0]				BUSTURN				DATAST								ADDHLD				ADDSET			
	Reset value	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x20	FMC_CFGR	FMCEN	Res.	Res.	Res.	Res.	Res.	BMAP[1:0]		Res.	Res.	Res.	CCLKEN	CLKDIV[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0						0	0				0	0	0	0	0															
0x80	FMC_PCR	Res.	Res.	Res.	Res.	Res.	Res.	WEN	BCHECC	Res.	Res.	Res.	Res.	ECCSS[2:0]			TAR[3:0]				TCLR[3:0]				ECCALG	Res.	ECCEN		PWID		Res.	PBKEN	PWAITEN	Res.
	Reset value							0	0					0	1	1	1	1	1	1	1	1	1	1	0		0	0	0			0	0	
0x84	FMC_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NWRF	Res.	PEF	Res.	Res.	ISOST	
	Reset value																										1		1			1	1
0x88	FMC_PMEM	MEMHIZ								MEMHOLD								MEMWAIT								MEMSET							
	Reset value	0	0	0	0	1	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	1	0
0x8C	FMC_PATT	ATTHIZ								ATTHOLD								ATTWAIT								ATTSET							
	Reset value	0	0	0	0	1	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	1	0
0x90	FMC_HPR	HPR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x94	FMC_HECCR	HECC[[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x98	Reserved	Reserved																															

Table 193. FMC register map (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x104	FMC_BWTR1	DATAHLD		ACCMOD		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSTURN				DATAST								ADDHLD				ADDSET			
	Reset value	0	0	0	0									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x10C	FMC_BWTR2	DATAHLD		ACCMOD		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSTURN				DATAST								ADDHLD				ADDSET			
	Reset value	0	0	0	0									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x114	FMC_BWTR3	DATAHLD		ACCMOD		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSTURN				DATAST								ADDHLD				ADDSET			
	Reset value	0	0	0	0									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x11C	FMC_BWTR4	DATAHLD		ACCMOD		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSTURN				DATAST								ADDHLD				ADDSET			
	Reset value	0	0	0	0									1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x140	FMC_SDCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDINIT	SDEN	Res.	RPIPE[1:0]		Res.	SDCLK[1:0]		WP	CAS[1:0]		NB	MWID[1:0]		NR[1:0]		NC[1:0]	
	Reset value															0	0		0	0		0	0	1	0	1	1	0	1	0	0	0	0
0x144	FMC_SDCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDINIT	SDEN	Res.	Res.	Res.	Res.	Res.	Res.	WP	CAS[1:0]		NB	MWID[1:0]		NR[1:0]		NC[1:0]	
	Reset value															0	0							1	0	1	1	0	1	0	0	0	0
0x148	FMC_SDTR	Res.	Res.	Res.	Res.	TRCD[3:0]				TRP[3:0]				TWR[3:0]				TRC[3:0]				TRAS[3:0]				TXSR[3:0]				TMRD[3:0]			
	Reset value					1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x14C	Reserved																																
0x150	FMC_SDCMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MRD[13:0]															NRFS[3:0]				DS1	DS2	MODE[2:0]		
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x154	FMC_SDRTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REIE	RFSCNT[12:0]													CRE
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x158	FMC_SDSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMDOK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODES2[1:0]		MODES1[1:0]		RE
	Reset value																	0											0	0	0	0	0
0x180	FMC_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IFEE	IHLE	IREE
	Reset value																														0	0	0
0x184	FMC_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IFEF	IHLF	IREF
	Reset value																														0	0	0
0x188	FMC_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CIFEF	CIHLF	CIREF
	Reset value																														0	0	0

Table 193. FMC register map (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x200	FMC_CSQCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSQSTART
	Reset value																																0
0x204	FMC_ CSQCFGR1	Res.	Res.	Res.	Res.	Res.	Res.	CMD2T	CMD1T	CMD2								CMD1								ACYNBR			Res.	Res.	DMADEN	CMD2EN	Res.
	Reset value							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	
0x208	FMC_ CSQCFGR2	Res.	Res.	Res.	Res.	Res.	Res.	RCMD2T	RCMD1T	RCMD2								RCMD1								Res.	Res.	Res.	Res.	Res.	DMADEN	RCMD2EN	SQSDTEN
	Reset value							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						0	0	0
0x20C	FMC_ CSQCFGR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RAC2T	RAC1T	SDT	AC5T	AC4T	AC3T	AC2T	AC1T	Res.	Res.	SNBR						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value									0	0	0	0	0	0	0	0			0	0	0	0	0	0								
0x210	FMC_CSQAR1	ADDCA4								ADDC3								ADDC2								ADDC1							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x214	FMC_CSQAR2	SAO																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDC5							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0									0	0	0	0	0	0	0	0
0x220	FMC_CSQIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMDTCIE	SUEIE	SEIE	SCIE	TCIE
	Reset value																												0	0	0	0	0
0x224	FMC_CSQISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMDTCF	SUEF	SEF	SCF	TCF
	Reset value																												0	0	0	0	0
0x228	FMC_CSQICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCMDTCF	CSUEF	CSEF	CSCF	CTCF
	Reset value																												0	0	0	0	0
0x22C	Reserved	Reserved																															
0x230	FMC_CSQEMSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEM[15:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x240	Reserved	Reserved																															
0x250	FMC_BCHIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EPBRIE	DSRIE	DEFIE	DERIE	DUEIE
	Reset value																												0	0	0	0	0
0x254	FMC_BCHISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EPBRF	DSRF	DEFF	DERF	DUEF
	Reset value																												0	0	0	0	0
0x258	FMC_BCHICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CEPBRF	CDSRF	CDEFF	CDERF	CDUEF
	Reset value																												0	0	0	0	0
0x260	FMC_BCHPBR1	BCHPB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x264	FMC_BCHPBR2	BCHPB[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 193. FMC register map (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x268	FMC_BCHPBR3	BCHPB[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x26C	FMC_BCHPBR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BCHPB[103:96]							
	Reset value																									0	0	0	0	0	0	0	0
0x27C	FMC_BCHDSR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEN				Res.	Res.	DEF	DUE
	Reset value																									0	0	0	0			0	0
0x280	FMC_BCHDSR1	Res.	Res.	Res.	EBP2[12:0]													Res.	Res.	Res.	EBP1[12:0]												
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0
0x284	FMC_BCHDSR2	Res.	Res.	Res.	EBP4[12:0]													Res.	Res.	Res.	EBP3[12:0]												
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0
0x288	FMC_BCHDSR3	Res.	Res.	Res.	EBP6[12:0]													Res.	Res.	Res.	EBP5[12:0]												
	Reset value																																
0x28C	FMC_BCHDSR4	Res.	Res.	Res.	EBP8[12:0]													Res.	Res.	Res.	EBP7[12:0]												
	Reset value																																

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

28 Extended-SPI interface (XSPI)

28.1 XSPI introduction

The XSPI supports most external serial memories such as serial PSRAMs, serial NAND and serial NOR flash memories, HyperRAM™ and HyperFlash™ memories, with the following functional modes:

- indirect mode: all the operations are performed using the XSPI registers to preset commands, addresses, data, and transfer parameters.
- automatic status-polling mode: the external memory status register is periodically read and an interrupt can be generated in case of flag setting. This feature is only available in regular-command protocol.
- memory-mapped mode: the external memory is memory mapped and it is seen by the system as if it was an internal memory, supporting both read and write operations.

The XSPI supports the following protocols with associated frame formats:

- the regular-command frame format with the command, address, alternate byte, dummy cycles, and data phase
- the HyperBus™ frame format

28.2 XSPI main features

- Functional modes: indirect, automatic status-polling, and memory-mapped
- Read and write support in memory-mapped mode
- Support for single, dual, quad, and octal communication
- Dual-memory configuration, where 8 bits can be sent/received simultaneously by accessing two quad or two octal memories in parallel
- XSPI mode accessing a single 16-bit memory
- SDR (single-data rate) and DTR (double-transfer rate) support
- Data strobe support
- Fully programmable opcode
- Fully programmable frame format
- Support wrapped-type access to memory in read direction
- HyperBus support
- Integrated FIFO for reception and transmission
- 8-, 16-, and 32-bit data accesses allowed
- DMA protocol support
- DMA channel for indirect mode operations
- Interrupt generation on FIFO threshold, timeout, operation complete, and access error
- AXI acceptance 2: acceptance is offered based on a read transaction followed by a second write or read request (acknowledged on the bus), while the first one is being processed.
- Dual chip select support (NCS1 and NCS2)
- Extended external memory support: if two same size external memories (extmem1 and extmem2) are connected to the same I/O port, contiguously in the memory map and

driven by a single XSPI. This XSPI automatically switches CS to extmem1 or extmem2, according to the address on interconnect side.

28.3 XSPI implementation

Table 194. XSPI implementation

Feature	XSPI1/2/3
HyperBus standard compliant	X
Xccela standard compliant	X
XSPI (JESD251C) standard compliant	X
AMBA® AXI compliant data interface	X
Asynchronous AHB clock versus kernel clock	X
Functional modes: indirect, automatic status-polling, and memory-mapped	X
Dual chip select support (NCS1 and NCS2)	X
Read and write support in memory-mapped mode	X
Dual-quad configuration	X
Dual-octal configuration	X
SDR (single-data rate) and DTR (double-transfer rate)	X
Data strobe (DS, DQS)	X
Fully programmable opcode	X
Fully programmable frame format	X
Integrated FIFO for reception and transmission	X
8, 16, and 32-bit data accesses	X
Interrupt on FIFO threshold, timeout, operation complete, and access error	X
Compliant with dual-XSPI arbiter (communication regulation)	X
Extended CSHT timeout	X
Memory-mapped write	X
Refresh counter	X
High-speed interface	X
GP/HPDMA interface	X
AXI acceptance 2	X
Extended external memory support	X
Automatic prefetch that can be disabled	X
Prefetch disable	X
Prefetch hardware software	X

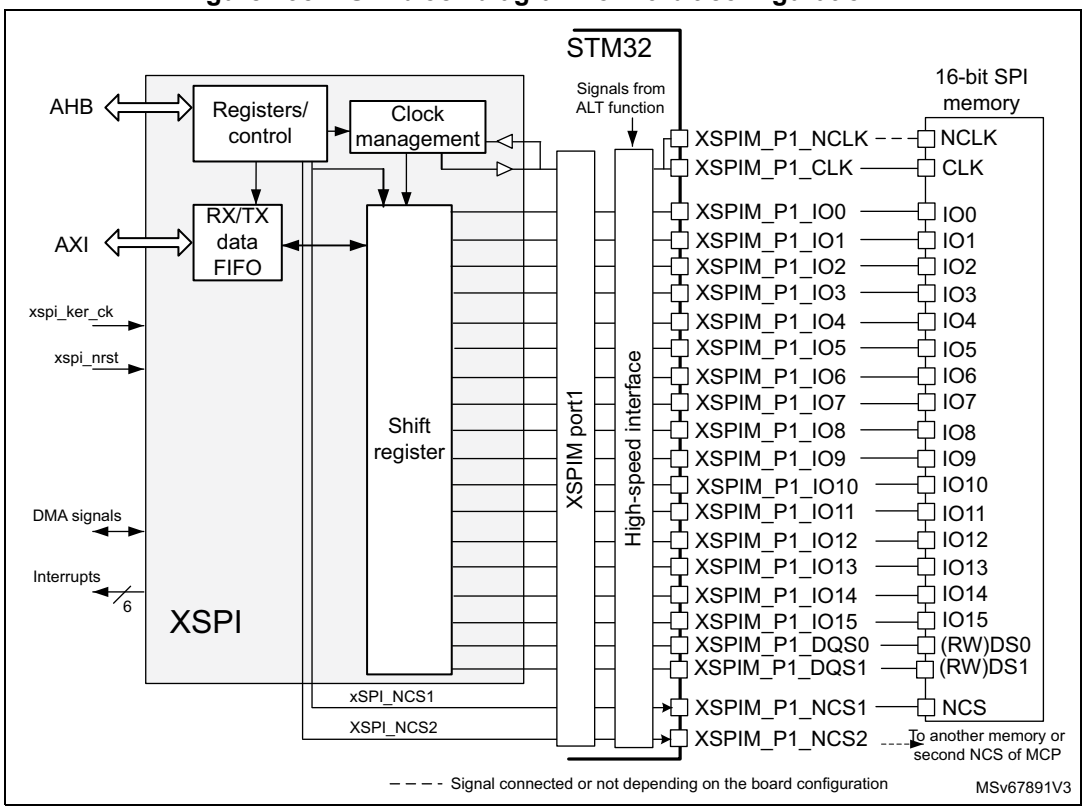
28.4 XSPI functional description

28.4.1 XSPI block diagram

The block diagrams provided below correspond to a single XSPI connected to the I/O ports.

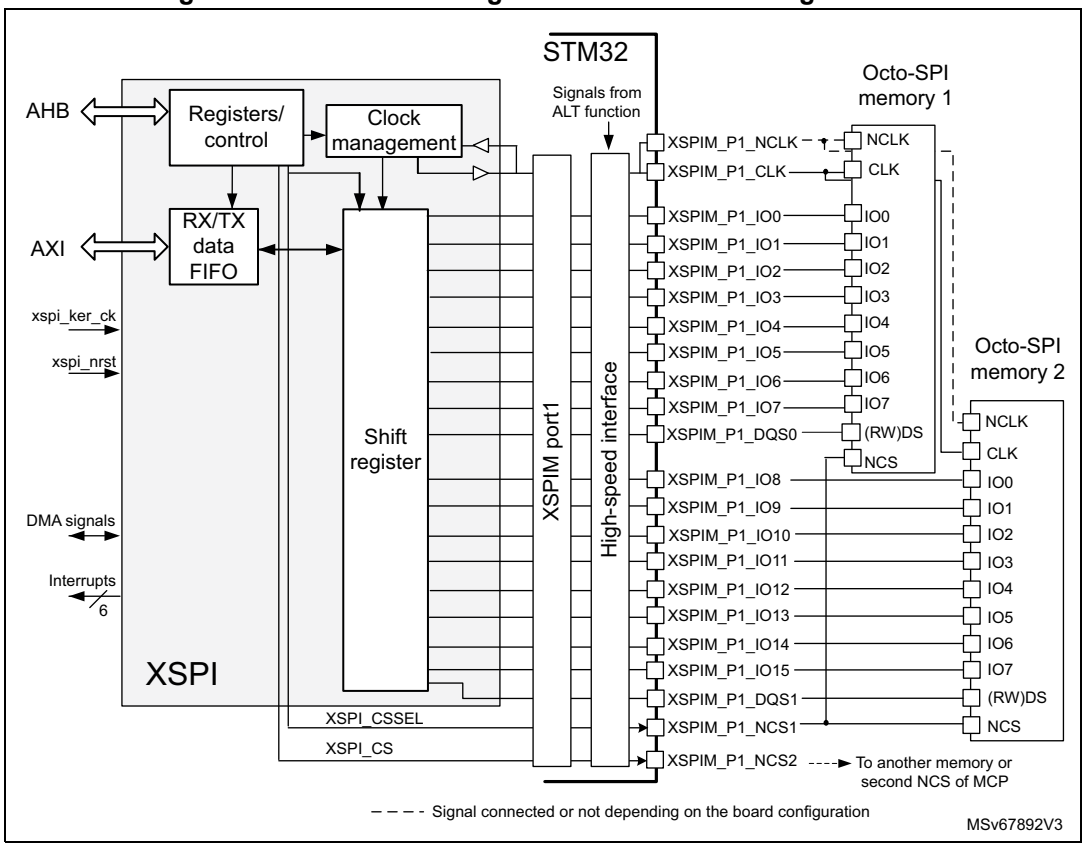
In case several XSPIs are connected to the I/O ports through the XSPIM I/O manager, refer to *Section 29: XSPI I/O manager (XSPIM)* for more details.

Figure 195. XSPI block diagram for 16-bit configuration(1)



1. The configuration depends on the package (see the datasheet for more details).

Figure 196. XSPI block diagram for dual-octal configuration(1)



1. The configuration depends on the package (see the datasheet for more details).

Figure 197. XSPI block diagram for octal configuration

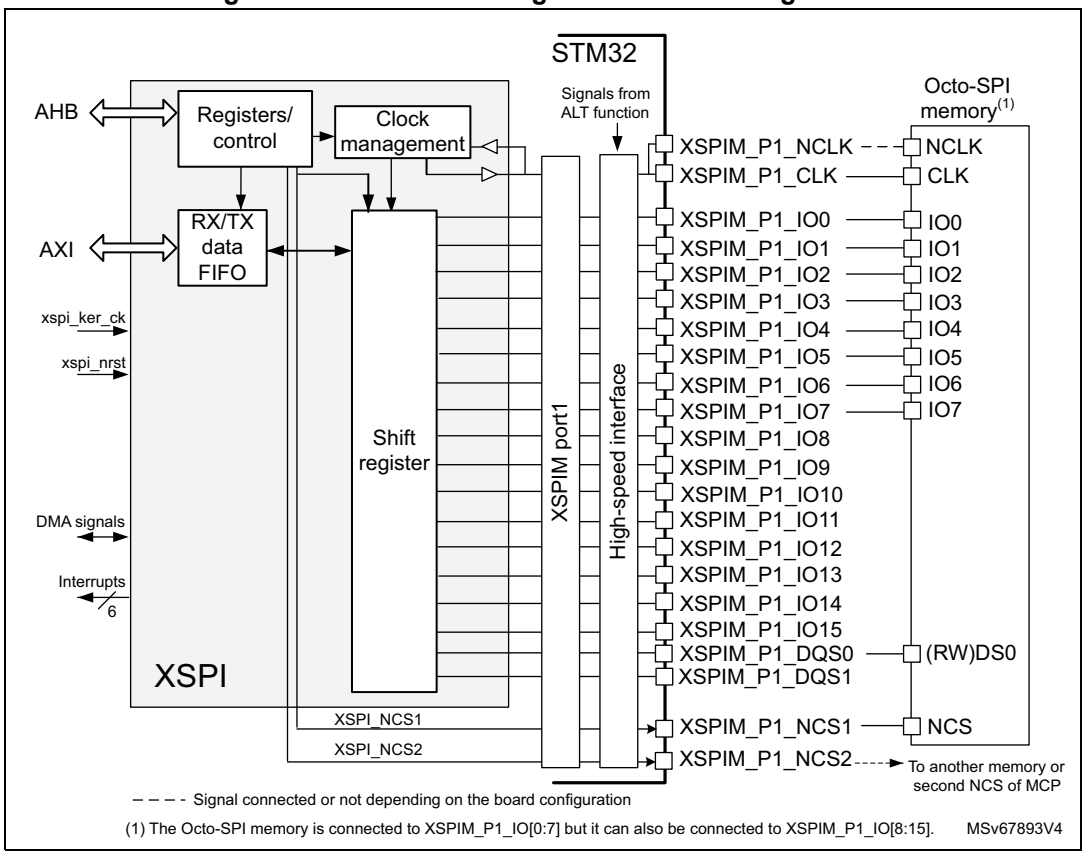


Figure 198. XSPI block diagram in quad configuration

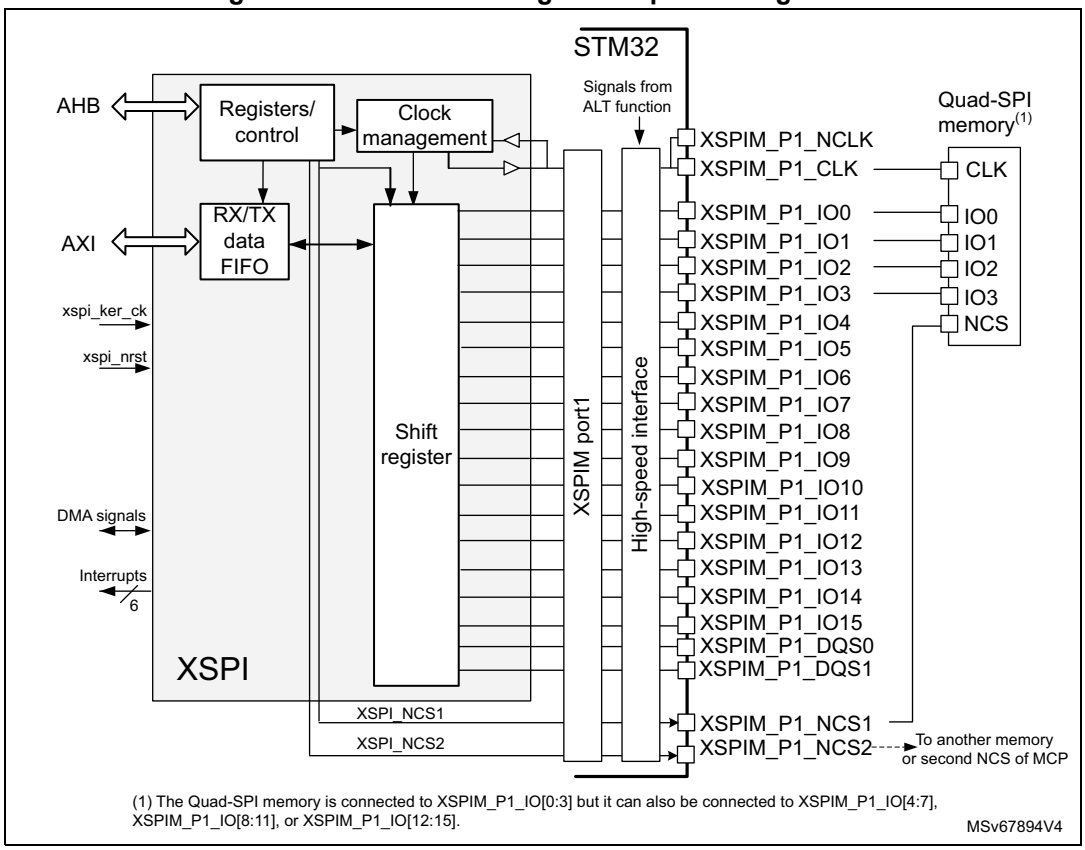
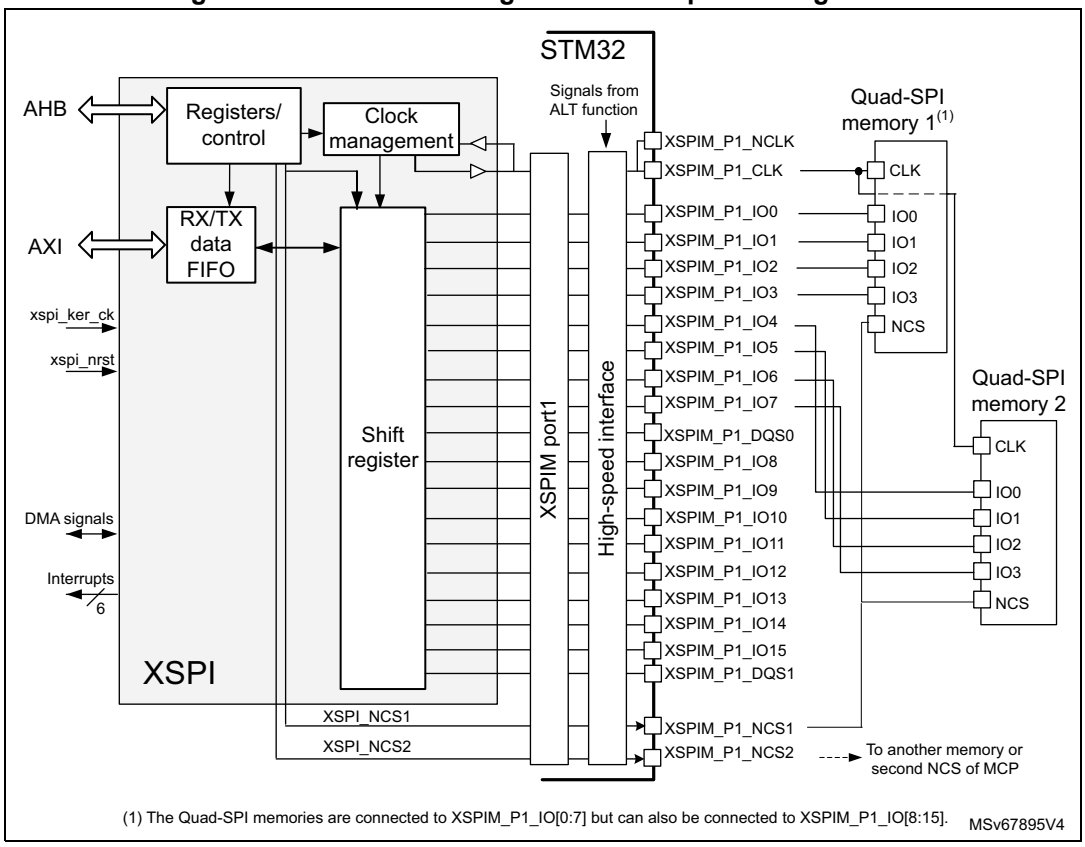


Figure 199. XSPI block diagram for dual-quad configuration



28.4.2 XSPI pins and internal signals

Table 195. XSPI input/output pins(1)

Pin name	Type	Description
XSPIM_Px_NCLK	Output	XSPI inverted clock to support 1.8 V HyperBus protocol.
XSPIM_Px_CLK	Output	XSPI clock
XSPIM_Px_IOn (n = 0 to 15)	Input/output	XSPI data pins
XSPIM_Px_NCS1,2	Output	Chip select for the memory
XSPIM_Px_DQS0,1	Input/output	Data strobe/write mask signal from/to the memory

1. x = 1 to 2.

Caution: Use the same configuration (output speed, HSLV) for all XSPI input/output pins to avoid any data corruption.

Table 196. XSPI internal signals

Signal name	Type	Description
xspi_hclk	Input	XSPI AHB clock
xspi_aclk	Input	XSPI AXI clock

Table 196. XSPI internal signals

Signal name	Type	Description
xspi_ker_ck	Input	XSPI kernel clock
xspi_dma	N/A	DMA request signal
xspi_it	Output	Global interrupt line (see *Table 200* for the multiple sources of interrupt)

28.4.3 Clock constraints

The XSPI AXI clock (xspi_aclk) has to be greater or equal than the XSPI kernel clock (xspi_ker_ck).

28.4.4 XSPI interface to memory modes

The XSPI supports the following protocols:

- regular-command protocol
- HyperBus protocol

The XSPI uses from 6 to 21 signals to interface with a memory, depending on the functional mode:

- NCS: chip-select (in the following sections, unless otherwise specified, NCS applies for NCS1 or NCS2. Only one of these two signals is active at a given moment in time).
- CLK: communication clock
- NCLK: inverted clock used only in the 1.8 V HyperBus protocol
- DQS0, DQS1: data strobe used only in regular-command protocol
- IO[3:0]: data bus LSB
- IO[7:4]:
  - data bus MSB used in dual-quad and octal configurations
  - data bus used as possible remap for quad-SPI mode
- IO[15:8]:
  - data bus MSB used in dual-quad, dual-octal and 16-bit configurations
  - data bus used as possible remap for octal-SPI mode
  - IO[15:12] and IO[11:8] can also be used as possible remap for quad-SPI mode

28.4.5 XSPI regular-command protocol

When in regular-command protocol, the XSPI communicates with the external device using commands. Each command can include the following phases:

- instruction phase
- address phase
- alternate-byte phase
- dummy-cycle phase
- data phase

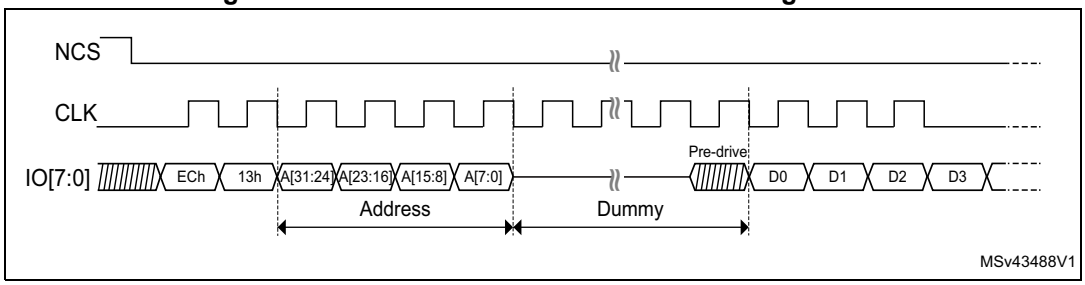
Only the data phase uses 16 bits. Instruction, address, and alternate phases use only the eight LSB of the bus as for octal configuration.

Any of these phases can be configured to be skipped, but single-phase commands supported are only those with instruction phase.

The NCS falls before the start of each command and rises again after each command finishes.

In memory-mapped mode, both read and write operation are supported: as a consequence, some of the configuration registers are duplicated to specify write operations (read operations are configured using regular registers).

Figure 200. SDR read command in 16-bit configuration



1. Data (such as D0, D1, D2) are sent in 16-bit configuration mode over IO[15:0]. Only the command and address are sent over IO[7:0] as for octal mode.

The specific regular-command protocol features are configured through the registers in the 0x0100-0x01FC offset range.

Instruction phase

During this phase, a 1- to 4-byte instruction is sent to the external device specifying the type of operation to be performed. The size of the instruction to be sent is configured in ISIZE[1:0] of XSPI_CCR and the instruction is programmed in INSTRUCTION[31:0] of XSPI_IR.

The instruction phase can optionally send:

- 1 bit at a time (over IO0, SO signal in single-SPI mode)
- 2 bits at a time (over IO0/IO1 in dual-SPI mode)
- 4 bits at a time (over IO0 to IO3 in quad-SPI mode)
- 8 bits at a time (over IO0 to IO7 in octal-SPI mode, or in 16-bit SPI mode)

This can be configured using IMODE[2:0] of XSPI_CCR.

The instruction can be sent in DTR (double-transfer rate) mode on each rising and falling edge of the clock, by setting IDTR in XSPI_CCR.

When IMODE[2:0] = 000 in XSPI_CCR, the instruction phase is skipped, and the command sequence starts with the address phase, if present.

When in memory-mapped mode, the instruction used for the write operation is specified in XSPI_WIR and the instruction format is specified in XSPI_WCCR. The instruction used for the read operation and the instruction format are specified in XSPI_IR and XSPI_CCR.

Address phase

In the address phase, 1 to 4 bytes are sent to the external device, to indicate the address of the operation. The number of address bytes to be sent is configured in ADSIZE[1:0] of XSPI_CCR.

In indirect and automatic status-polling modes, the address bytes to be sent are specified in ADDRESS[31:0] of XSPI_AR. In memory-mapped mode, the address is given directly via the AXI (from any master in the system).

The address phase can send:

- 1 bit at a time (over IO0, SO signal in single-SPI mode)
- 2 bits at a time (over IO0/IO1 in dual-SPI mode)
- 4 bits at a time (over IO0 to IO3 in quad-SPI mode)
- 8 bits at a time (over IO0 to IO7 in octal-SPI mode, or in 16-bit SPI mode)

This can be configured using ADMODE[2:0] of XSPI_CCR.

The address can be sent in DTR mode (on each rising and falling edge of the clock) setting ADDTR in XSPI_CCR.

When ADMODE[2:0] = 000, the address phase is skipped and the command sequence proceeds directly to the next phase, if any.

In memory-mapped mode, the address format for the write operation is specified in XSPI_WCCR. The address format for the read operation is specified in XSPI_CCR.

Warning: Some memory specifications consider that each address corresponds to a 16-bit value. XSPI considers that each address corresponds to an 8-bit value.So the software needs to multiply the address by two when accessing the memory registers.

Alternate-byte phase

In the alternate-bytes phase, 1 to 4 bytes are sent to the external device, generally to control the mode of operation. The number of alternate bytes to be sent is configured in ABSIZE[1:0] of XSPI_CCR. The bytes to be sent are specified in XSPI_ABR.

The alternate-byte phase can send:

- 1 bit at a time (over IO0, SO signal in single-SPI mode)
- 2 bits at a time (over IO0/IO1 in dual-SPI mode)
- 4 bits at a time (over IO0 to IO3 in quad-SPI mode)
- 8 bits at a time (over IO0 to IO7 in octal-SPI mode, or in 16-bit SPI mode)

This can be configured using ABMODE[2:0] of XSPI_CCR.

The alternate bytes can be sent in DTR mode (on each rising and falling edge of the clock) setting ABDTR of XSPI_CCR.

When ABMODE[2:0] = 000, the alternate-bytes phase is skipped and the command sequence proceeds directly to the next phase, if any.

Only a single nibble may need to be sent during the alternate-byte phase rather than a full byte, such as when the dual-SPI mode is used and only two cycles are used for the alternate bytes.

In this case, the firmware can use the quad-SPI mode (ABMODE[2:0] = 011), and send a byte with bits 7 and 3 of ALTERNATE[31:0] set to 1 (keeping the IO3 line high), and bits 6 and 2 set to 0 (keeping the IO2 line low), in XSPI_IR.

The upper two bits of the nibble to be sent are then placed in bits 5:4 of ALTERNATE[31:0], while the lower two bits are placed in bits 1:0. For example, if the nibble 2 (0010) is to be sent over IO0/IO1, then ALTERNATE[31:0] must be set to 0x8A (1000_1010).

In memory-mapped mode, the alternate bytes used for the write operation are specified in XSPI_WABR, and the alternate byte format is specified in XSPI_WCCR. The alternate bytes used for read operation and the alternate byte format are specified in XSPI_ABR and XSPI_CCR.

Dummy-cycle phase (memory latency)

In the dummy-cycle phase, 1 to 31 cycles are given without any data being sent or received, in order to give the external device, the time to prepare for the data phase when higher clock frequencies are used. The number of cycles given during this phase is specified in DCYC[4:0] of XSPI_TCR. In both SDR and DTR modes, the duration is specified as a number of full CLK cycles.

When DCYC[4:0] = 00000, the dummy-cycle phase is skipped, and the command sequence proceeds directly to the data phase, if present.

In order to assure enough “turn-around” time for changing the data signals from the output mode to the input mode, there must be at least one dummy cycle when using the dual-, quad-, octal-, or 16-bit SPI mode, to receive data from the external device.

In memory-mapped mode, the dummy cycles for the write operations are specified in XSPI_WTCR. The dummy cycles for the read operation are specified in XSPI_TCR.

Data phase

During the data phase, any number of bytes can be sent to or received from the external device.

In indirect mode, the number of bytes to be sent/received is specified in XSPI_DLR. In this mode, the data to be sent to the external device must be written to XSPI_DR. In indirect-read mode, the data received from the external device is obtained by reading XSPI_DR.

In automatic status-polling mode, the number of bytes to be received is specified in XSPI_DLR, and the data received from the external device can be obtained by reading XSPI_DR.

In memory-mapped mode, the data read or written, is sent or received directly over the AXI to the Cortex core or to a DMA.

The data phase can send/receive:

- 1 bit at a time (over IO0/IO1 (SO/SI respectively) in single-SPI mode)
- 2 bits at a time (over IO0/IO1 in dual-SPI mode)
- 4 bits at a time (over IO0 to IO3 in quad-SPI mode)
- 8 bits at a time (over IO0 to IO7 in octal-SPI mode)
- 16 bits at a time (over IO0 to IO15 in 16-bit SPI mode)

This can be configured using DMODE[2:0] of XSPI_CCR.

The data can be sent or received in DTR mode (on each rising and falling edge of the clock) setting DDTR of XSPI_CCR.

When DMODE[2:0] = 000, the data phase is skipped, and the command sequence finishes immediately by raising the NCS. This configuration must be used only in indirect-write mode.

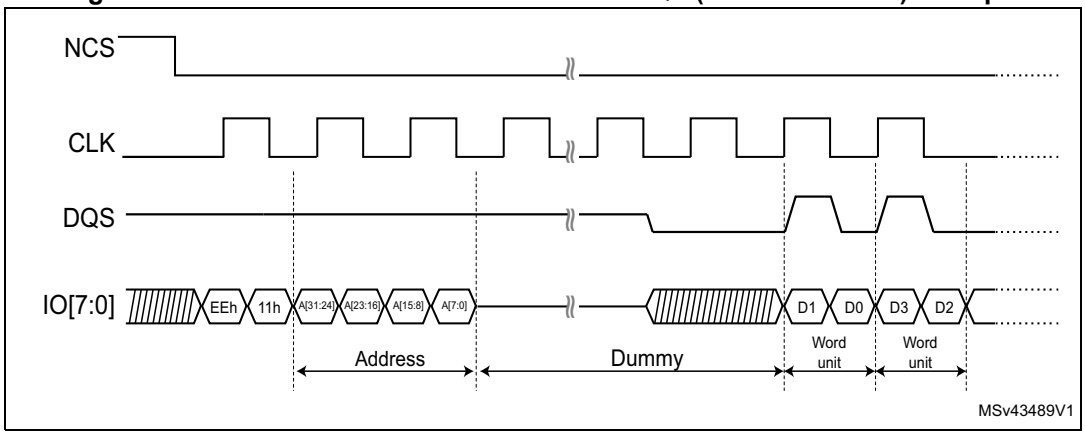
In memory-mapped mode, the data format for the write operation is specified in XSPI_WCCR. The data format for the read operation is specified in XSPI_CCR.

DQS use

The DQS signal can be used for data strobing during the read transactions when the device toggles the DQS aligned with the data.

The DQS management can be enabled by setting DQSE of XSPI_CCR.

Figure 201. DTR read in octal-SPI mode with DQS (Macronix mode) example



28.4.6 XSPI regular-command protocol signal interface

Single-SPI mode

The legacy SPI mode allows just a single bit to be sent/received serially. In this mode, the data is sent to the external device over the SO signal (Single-SPI Output) (whose I/Os are shared with IO0). The data received from the external device arrives via SI (Single-SPI Input) (whose I/Os are shared with IO1).

Compared to the SPI legacy mode, I0/SO and I1/SI are respectively equivalent to MOSI and MISO, having the XSPI generating the clock.

The different phases can each be configured separately to use this single-bit mode by setting to 001 the IMODE, ADMODE, ABMODE, and DMODE fields in XSPI_CCR and XSPI_WCCR.

In each phase configured in single-SPI mode:

- IO0 (SO) is in output mode.
- IO1 (SI) is in input mode (high impedance).
- IO2 is in output mode and forced to 0.
- IO3 is in output mode and forced to 1 (to deactivate the “hold” function).
- IO4 to IO15 are in output mode and forced to 0.

This is the case even for the dummy phase if DMODE[2:0] = 001.

Dual-SPI mode

In dual-SPI mode, two bits are sent/received simultaneously over the IO0/IO1 signals.

The different phases can each be configured separately to use dual-SPI mode by setting to 010 the IMODE, ADMODE, ABMODE, and DMODE fields in XSPI_CCR and XSPI_WCCR.

In each phase configured in dual-SPI mode:

- IO0/IO1 are at high-impedance (input) during the data phase for the read operations, and outputs in all other cases.
- IO2 is in output mode and forced to 0.
- IO3 is in output mode and forced to 1.
- IO4 to IO15 are in output mode and forced to 0.

In the dummy phase, when DMODE[2:0] = 010, IO0 and IO1 are in a high-impedance state during read transactions, and are forced to either high or low levels during write transactions.

Quad-SPI mode

In quad-SPI mode, four bits are sent/received simultaneously over the IO0/IO1/IO2/IO3 signals.

The different phases can each be configured separately to use the quad-SPI mode by setting to 011 the IMODE, ADMODE, ABMODE, and DMODE fields in XSPI_CCR and XSPI_WCCR.

In each phase configured in quad-SPI mode:

- IO0 to IO3 are all at high-impedance (inputs) during the data phase for the read operations, and outputs in all other cases.
- IO4 to IO15 are in output mode and forced to 0.

In the dummy phase, when DMODE[2:0] = 011, IO0 to IO3 are in a high-impedance state during read transactions, and are forced to either high or low levels during write transactions.

Octal-SPI mode

In regular octal-SPI mode, the eight bits are sent/received simultaneously over the IO[0:7] signals.

The different phases can each be configured separately to use the octal-SPI mode by setting to 100 the IMODE, ADMODE, ABMODE, and DMODE fields in XSPI_CCR and XSPI_WCCR.

In each phase that is configured in octal-SPI mode, IO[0:7] are all at high-impedance (input) during the data phase for read operations, and outputs in all other cases.

In the dummy phase, when DMODE[2:0] = 100, IO[0:7] are in a high-impedance state during read transactions, and are forced to either high or low levels during write transactions.

XSPI mode

In XSPI mode, the 16 bits are sent/received simultaneously over the IO[0:15] signals during the data phase.

The following phases must be configured separately to use the XSPI mode:

1. Set to 100 the IMODE, ADMODE, and ABMODE fields (in XSPI_CCR and XSPI_WCCR).
2. Set to 101 the DMODE fields (in XSPI_CCR and XSPI_WCCR).

In each phase that is configured in XSPI mode, IO[0:15] are all at high-impedance (input) during the data phase for read operations, and outputs in all other cases.

In the dummy phase, when DMODE[2:0] = 101, IO[0:15] are in a high-impedance state during read transactions, and are forced to either high or low levels during write transactions.

IO[8:15] are used only in XSPI mode. If none of the phases are configured to use this mode, then the pins corresponding to IO[8:15] can be used for other functions, even while the XSPI is active.

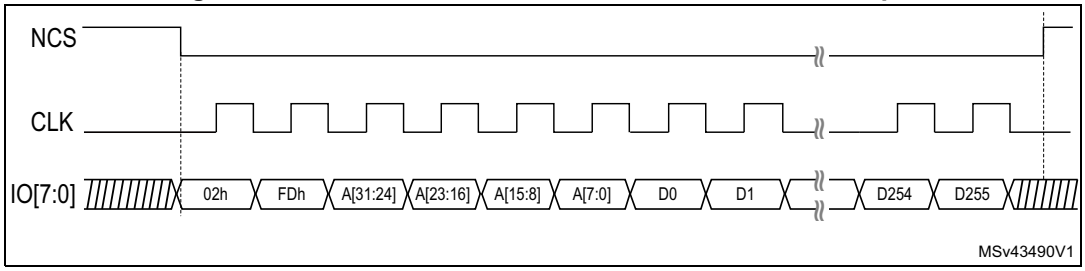
Single-data rate (SDR) mode

By default, all the phases operate in single-data rate (SDR) mode.

In SDR mode, when the XSPI drives the IO0/SO and IO1 to IO15 signals, these signals transition only with the falling edge of CLK.

When receiving data in SDR mode, the XSPI assumes that the external devices also send the data using CLK falling edge. By default (when SSHIFT = 0 in XSPI_TCR), the signals are sampled using the following (rising) edge of CLK.

Figure 202. SDR write command in octal-SPI mode example



Double-transfer rate (DTR) mode

Each of the instruction, address, alternate-byte, and data phases can be configured to operate in DTR mode setting IDTR, ADDTR, ABDTR, and DDTR in XSPI_CCR.

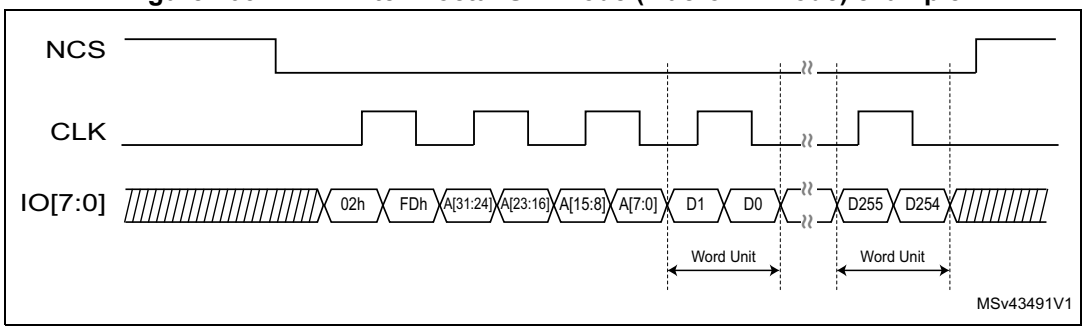
In memory-mapped mode, the DTR mode for each phase of the write operations is specified in XSPI_WCCR. The DTR mode for each phase of the read operations is specified in XSPI_CCR.

In DTR mode, when the XSPI drives the IO0/SO and IO1to IO7 signals in the instruction, address, and alternate-byte phases, a bit is sent or received on each of the falling and rising edges of CLK.

In DTR mode, when the XSPI drives the IO0 to IO15 signals in the data phases, a bit is sent or received on each of the falling and rising edges of CLK.

When receiving data in DTR mode, the XSPI assumes that the external devices also send the data using both CLK rising and falling edges. When DDTR = 1 in XSPI_CCR, the software must clear SSHIFT in XSPI_TCR. Thus, the signals are sampled one half of a CLK cycle later (on the following, opposite edge).

Figure 203. DTR write in octal-SPI mode (Macronix mode) example



Dual-quad configuration

When DMM = 1 in XSPI_CR, the XSPI is in dual-memory configuration: if DMODE = 011, two external quad-SPI devices (device A and device B) are used in order to send/receive eight bits (or 16 bits in DTR mode) every cycle, effectively doubling the throughput.

Each device (A or B) uses the same CLK and NCS signals, but each has separate IO0 to IO3 signals.

The dual-quad configuration can be used in conjunction with the single-SPI, dual-SPI, and quad-SPI modes, as well as with either SDR or DTR mode.

The device size, as specified in DEVSIZE[4:0] of XSPI_DCR1, must reflect the total external device capacity that is the double of the size of one individual component.

If address X is even, then the byte that the XSPI gives for address X is the byte at the address X/2 of device A, and the byte that the XSPI gives for address X + 1 is the byte at the address X/2 of device B. In other words, the bytes at even addresses are all stored in device A and the bytes at odd addresses are all stored in device B.

When reading the status registers of the devices in dual-quad configuration, twice as many bytes must be read compared to the same read in regular-command protocol: if each device gives eight valid bits after the instruction for fetching the status register, then the XSPI must be configured with a data length of 2 bytes (16 bits), and the XSPI receives one byte from each device.

If each device gives a status of 16 bits, then the XSPI must be configured to read 4 bytes to get all the status bits of both devices in dual-quad configuration. The least-significant byte of the result (in the data register) is the least-significant byte of device A status register. The next byte is the least-significant byte of device B status register. Then, the third byte of the data register is the device A second byte. The forth byte is the device B second byte (if devices have 16-bit status registers).

An even number of bytes must always be accessed in dual-quad configuration. For this reason, bit 0 of DL[31:0] in XSPI_DLR is stuck at 1 when DMM = 1.

In dual-quad configuration, the behavior of device A interface signals is basically the same as in normal mode. Device B interface signals have exactly the same waveforms as device A ones during the instruction, address, alternate-byte, and dummy-cycle phases. In other words, each device always receives the same instruction and the same address.

Then, during the data phase, the AIOx and the BIOx buses both transfer data in parallel, but the data that is sent to (or received from) device A is distinct than the one from device B.

Dual-octal configuration

When DMM = 1 in XSPI_CR, the XSPI is in dual-memory configuration: when DMODE = 100, two external octal-SPI devices (device A and device B) are used in order to receive 32 bits in DTR mode every cycle, effectively doubling the throughput as well as the capacity.

Each device (A or B) uses the same CLK and NCS signals, but each has separate IO0 to IO7 signals.

The dual-octal configuration can be used in DTR mode exclusively in conjunction with the single-SPI, dual-SPI, quad-SPI and octal-SPI modes.

The device size, as specified in DEVSIZE[4:0] of XSPI_DCR1, must reflect the total external device capacity that is the double of the size of one individual component.

If address X is even, then the byte that the XSPI gives for address X is the byte at the address X/2 of device A, and the byte that the XSPI gives for address X + 1 is the byte at the address X/2 of device B. In other words, the bytes at even addresses are all stored in device A and the bytes at odd addresses are all stored in device B.

When reading the status registers of the devices in dual-octal DTR mode, twice as many bytes must be read compared to the same read in regular DTR mode: if each device gives twice eight valid bits after the instruction for fetching the status register, then the XSPI must be configured with a data length of 4 bytes. The LSB is the LSB of device A, and the third byte is the LSB of device B.

If each device gives a status of 16 bits, then the XSPI must be configured to read 4 bytes to get all the status bits of both devices in dual-octal DTR mode. In such case, the order of retrieved status bits is as follows:

- first byte: LSB of device A
- second byte: second byte of device A
- third byte: LSB of device B
- fourth byte: second byte of device B

In indirect mode using DTR mode, a number of bytes multiple of four must always be accessed in XSPI mode. For this reason, bit 0 and bit 1 of the DL[31:0] bitfield in XSPI_DLR are stuck at 0 when DMODE[2:0] = 101.

In dual-octal configuration, the behavior of device A interface signals is basically the same as in normal mode. Device B interface signals have exactly the same waveforms as device A ones during the instruction, address, alternate-byte, and dummy-cycles phases. In other words, each device always receives the same instruction and the same address.

Then, during the data phase, the AIOx and the BIOx buses both transfer data in parallel, but the data that is sent to (or received from) device A is distinct than the one from device B.

Note: The variable latency is not supported in dual-octal configuration.

XSPI mode

When DMODE[2:0] = 0b101 in XSPI_CCR, the XSPI is in single 16-bit-memory configuration: when DMODE[2:0] = 0b101 with DMM value ignored in that case. A single

external XSPI device is used in order to send/receive 16 bits (or 32 bits in DTR mode) every cycle, effectively doubling the throughput.

The device provides/receives two separate DQS signals: DQS0 for the eight LSBs and DQS1 for the eight MSBs.

The XSPI mode can be used in conjunction with the single-SPI, dual-SPI, quad-SPI and octal-SPI modes, as well as with either the SDR or the DTR mode.

The device size, as specified in DEVSIZE[4:0] of XSPI_DCR1, must reflect the total external device capacity.

In SDR mode, a number of bytes multiple of two must always be accessed in XSPI mode. For this reason, bit 0 of DL[31:0] in XSPI_DLR is stuck at 0 when DMODE = 101.

In DTR mode, a number of bytes multiple of four must always be accessed in XSPI mode. For this reason, bit 0 and 1 of DL[31:0] in XSPI_DLR are stuck at 0 when DMODE = 101.

28.4.7 HyperBus protocol

The XSPI can communicate with the external device using the HyperBus protocol.

The HyperBus uses 11 to 12 pins in 8-bit data mode, or 19 to 20 pins in 16-bit data mode depending on the operating voltage:

- IO[7:0] as bidirectional data bus for 8-bit data mode and IO[15:0] as bidirectional data bus for 16-bit data mode
- DQS for read and write data strobe and latency insertion
- NCS
- CLK
- NCLK for 1.8 V operations (to support this mode, the device must be powered with 1.8 V)

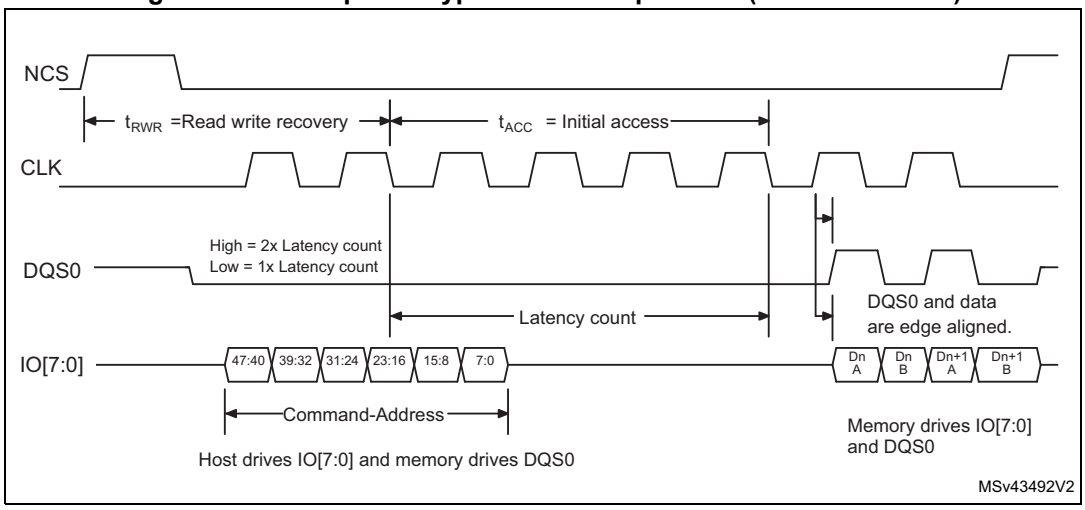
The HyperBus does not require any command specification nor any alternate bytes. As a consequence, a separate register set is used to define the timing of the transaction.

The HyperBus frame is composed of the following phases:

- command/address phase
- data phase

The NCS falls before the start of a transaction and rises again after each transaction finishes.

Figure 204. Example of HyperBus read operation (8-bit data mode)



The specific HyperBus features are configured through the registers in the 0x0200-0x02FC offset range.

Command/address phase

During this initial phase, the XSPI sends 48 bits over IO[7:0] to specify the operations to be performed with the external device.

Table 197. Command/address phase description

CA bit	Bit name	Description
47	R/W#	Identifies the transaction as a read or a write
46	Address space	Indicates if the transaction accesses the memory or the register space
45	Burst type	Indicates if the burst is linear or wrapped
44-16	Row and upper column address	Selects the row and the upper column addresses
15-3	Reserved	-
2-0	Lower column address	Selects the starting 16-bit word within the half page

The address space is configured through the memory type MTYP[2:0] of XSPI_DCR1.

The total size of the device must be considered through DEVSIZE[4:0] of XSPI_DCR1. In case of multi-chip product (MCP), the device size is the sum of all the sizes of all the MCP dies.

Warning: Some memory specifications consider that each address corresponds to a 16-bit value. The XSPI considers that each address corresponds to an 8-bit value. So, the software

needs to multiply the address by two when accessing the memory registers.

Read/write operation with initial latency

The HyperBus read and write operations need to respect two timings:

- t_{RWR}: minimal read/write recovery time for the device (defined by TRWR[7:0] in XSPI_HLCR)
- t_{ACC}: access time for the device (defined in TACC[7:0] of XSPI_HLCR) according to the memory latency

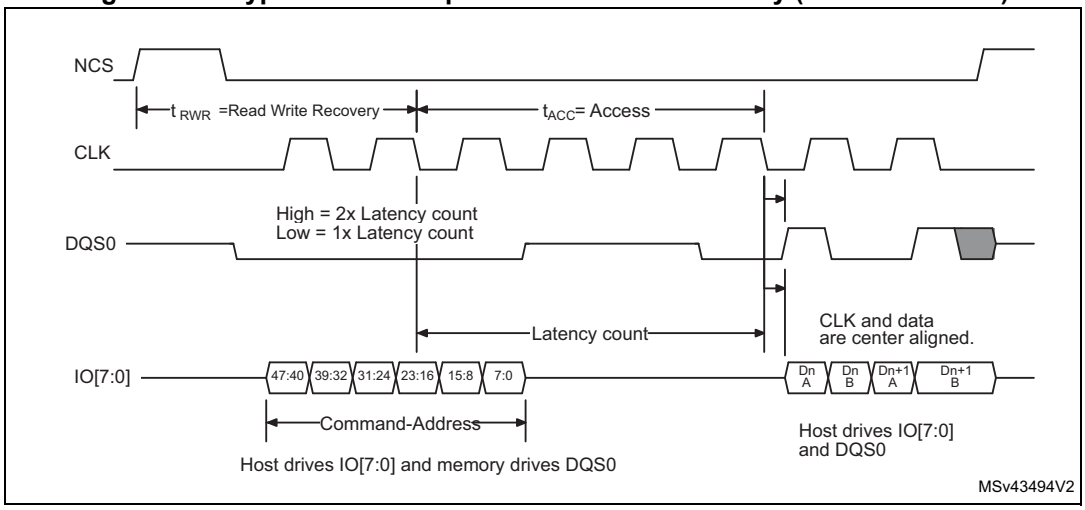
During the read operation, the DQS0/1 is used by the device, in two ways (see *Figure 204*):

- during the command/address phase, to request an additional latency
- during the data phase, for data strobing

During the write operation, the DQS0/1 is used:

- by the device, during the command/address phase, to request an additional latency.
- by the XSPI, during the data phase, for write data masking.

Figure 205. HyperBus write operation with initial latency (8-bit data mode)



Read/write operation with additional latency

If the device needs an additional latency (during refresh period of an SDRAM for example), DQS0/1 must be tied to one during one of the DQS signals, during the command/address phase.

An additional t_{ACC} duration is added by the XSPI to meet the device request.

Figure 206. HyperBus read operation with additional latency (8-bit data mode)

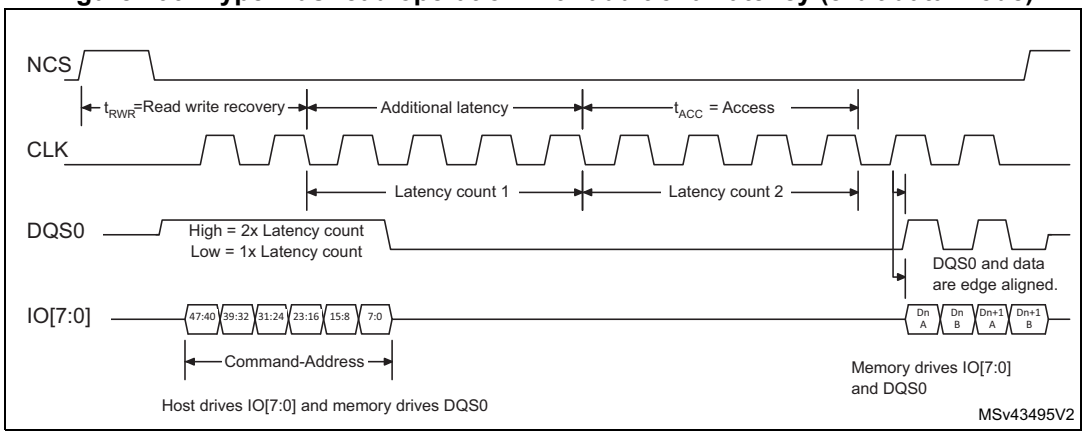
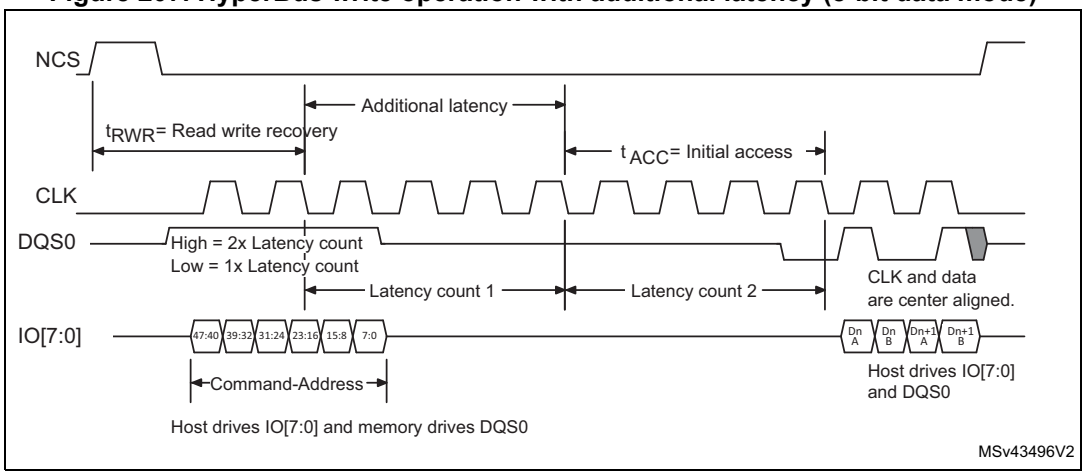


Figure 207. HyperBus write operation with additional latency (8-bit data mode)



Fixed latency mode

Some devices or some applications may not want to operate with a variable latency time as described above.

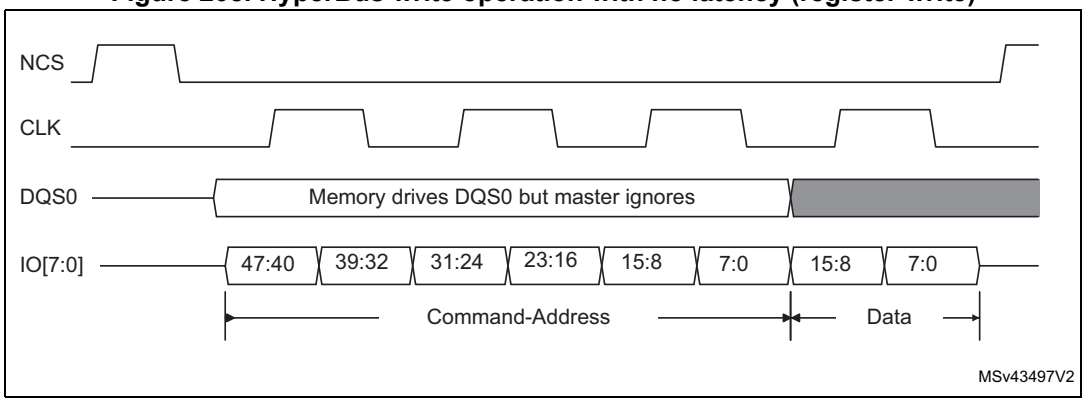
The latency can be forced to 2 x t_{ACC} by setting LM of XSPI_HLCR.

In this XSPI latency mode, the state of the DQS signal is not taken into account by the XSPI and an additional latency is always added leading to a fixed 2 x t_{ACC} latency time.

Write operation with no latency

Some devices can also require a zero latency for the write operations. This write-zero latency can be forced by setting WZL in XSPI_HLCR.

Figure 208. HyperBus write operation with no latency (register write)

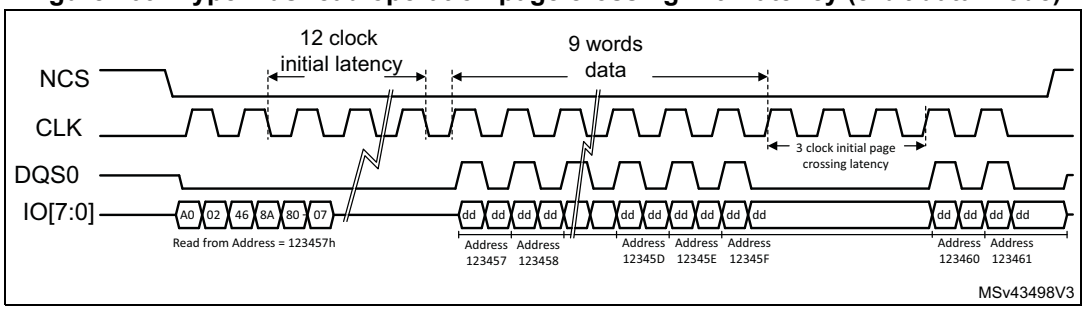


Latency on page-crossing during the read operations

An additional latency can be needed by some devices for the read operation when crossing pages.

The initial latency must be respected for any page access, as a consequence, when the first access is close to the page boundary, a latency is automatically added at the page crossing to respect the t_{ACC} time.

Figure 209. HyperBus read operation page crossing with latency (8-bit data mode)



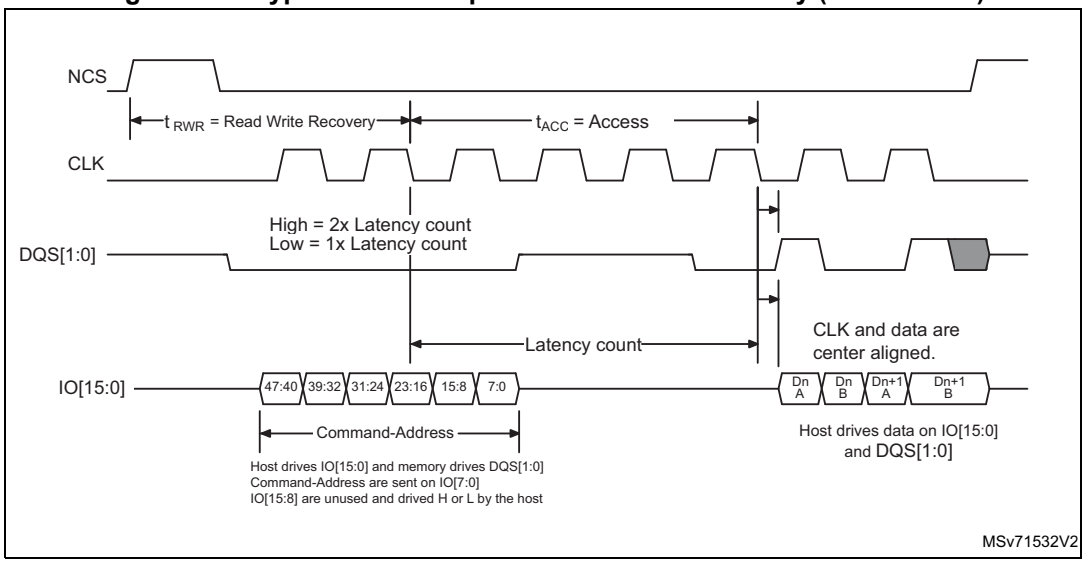
16-bit data transfer using HyperBus

In the HyperBus protocol, the XSPI supports a dual-octal configuration (16-bit data transfers) when DMM = 1 in XSPI_CR. In this configuration, one octal HyperBus memory is connected to IO0-IO7, and another is connected to IO8-IO15, following the same principle as the dual-octal configuration described in the regular command protocol. These memories share all signals except DQS, which are dedicated.

For 16-bit data transfers, DMODE[2:0] must be to equal to 101. Any other value in DMODE[2:0] correspond to 8-bit data transfer (quad-, dual-, and single-bit data transfer do not exist in HyperBus protocol). Command-address phase is always using 8 bits in HyperBus protocol (from IO8-IO15). Only the data is on 16-bit for write or read operations accessing the memory space (from IO0-IO15) as shown for instance in *Figure 210*. For the memory register accesses, the data is on 8-bit (from IO0 to IO7, IO8 to IO15 being not used but driven by the controller) as shown in *Figure 208*.

The behavior of the interface at protocol-level is exactly the same as for HyperBus octal configuration, as described above, except that the variable latency is not supported in dual-octal HyperBus configuration. LM in XSPI_HLCR must be set.

Figure 210. HyperBus write operation with initial latency (16-bit mode)



28.4.8 Specific features

The XSPI supports some specific features, such as:

- wrap support
- NCS boundary and refresh
- communication regulation

Wrap support

The XSPI supports an hybrid wrap as defined by the HyperBus protocol. A hybrid wrap is also supported in the regular-command protocol.

In hybrid wrap, the transaction can continue after the initial wrap with an incremental access.

The wrap size supported by the target memory is configured by WRAPSIZE in XSPI_DCR2.

Wrap is supported only in memory-read direction and only for data size = 8 bytes. Wrapped reads are supported for both HyperBus and regular-command protocols. To enable wrapped-read accesses, the dedicated registers XSPI_WPxxx must be programmed according to the wrapped-read access characteristics. The dedicated XSPI_WPxxx registers apply for both HyperBus and regular-command protocols.

If the target memory is not supporting the hybrid wrap, WRAPSIZE must be set to 0.

Note: Hybrid wrap requires that the nonwrapped registers (XSPI_CCR, XSPI_TCR, XSPI_IR) are set according to the memory configuration to satisfy its correct data prefetch (initiated after the wrap command).

The wrap operation cannot be interrupted by a refresh. The refresh event is only considered after the wrap completion.

NCS boundary and refresh

Two processes can be activated to regulate the XSPI transactions:

- NCS boundary
- refresh

The NCS boundary feature limits a transaction to a boundary of aligned addresses. The size of the address to be aligned with is configured in CSBOUND[4:0] of XSPI_DCR3 and it is equal to $2^{CSBOUND}$.

As an example, if CSBOUND(4:0] = 0x4, the boundary is set to 2^4 = 16 bytes. As a consequence, the NCS is released each time that the LSB address is equal to 0xF, and each time that a new transaction is issued to address the next data.

If CSBOUND[4:0] = 0, the feature is disabled. A minimum value of three is recommended.

The NCS boundary feature cannot be used for flash memory devices in write mode since a command is necessary to program another page of the flash memory.

The refresh feature limits the duration of the transactions to the value programmed in REFRESH[31:0] of XSPI_DCR4. The duration is expressed in number of cycles. This allows an external RAM to perform its internal refresh operation regularly.

The refresh value must be greater than the minimal transaction size in terms of number of cycles including the command, address, alternate/dummy phases.

If NCS boundary and refresh are enabled at the same time, the NCS is released on the first condition met.

Communication regulation

The communication regulation feature is useful in case two or more XSPIs share access to the same I/O port through an I/O manager. This feature limits the maximum length of a transaction to the value programmed in MAXTRAN[7:0] of XSPI_DCR3.

If the number of clock cycles of XSPI1 for example reach the MAXTRAN + 1 value, and if the second interface connected to the I/O manager requests an access, the NCS is released, and a new transaction is issued. If the second interface connected to the I/O manager does not request an access, the transaction is not stopped, and the NCS is not released.

If MAXTRAN[7:0] = 0, no limitation occurs.

The MAXTRAN[7:0] value must be greater than the minimal transaction size in terms of number of cycles including the command, address, alternate, and dummy phases.

Note: The communication regulation feature cannot be used in write mode for the flash memory devices that require extra command to reenable the write operation after the NCS is active again.

If NCS boundary, refresh and communication regulation are enabled at the same time, the NCS is released on the first condition met.

Restarting after an interrupted transfer

When a read or write operation is interrupted by a timeout, the XSPI interface, as soon as possible after getting back the port ownership, re-issues the initial command sequence together with the address following the last address actually accessed before interruption. The transfer initially set goes on and ends seamlessly.

28.4.9 XSPI operating modes introduction

The XSPI has the following operating modes regardless of the low-level protocol used (either regular-command or HyperBus):

- indirect mode (read or write)
- automatic status-polling mode (only in regular-command protocol)
- memory-mapped mode

28.4.10 XSPI indirect mode

In indirect mode, the commands are started by writing to the XSPI registers, and the data is transferred by writing or reading the data register, in a similar way to other communication peripherals.

When FMODE[1:0] = 00 in XSPI_CR, the XSPI is in indirect-write mode: bytes are sent to the external device during the data phase. Data are provided by writing to XSPI_DR.

When FMODE[1:0] = 01, the XSPI is in indirect-read mode: bytes are received from the external device during the data phase. Data are recovered by reading XSPI_DR.

In indirect mode, when the XSPI is configured in DTR mode over eight lanes with DQS disabled, the given starting address and the data length must be even.

Note: The XSPI_AR register must be updated even if the start address is the same as the start address of the previous indirect access.

The number of bytes to be read/written is specified in XSPI_DLR:

- If DL[31:0] = 0xFFFF FFFF, the data length is considered undefined and the XSPI simply continues to transfer data until it reaches the end of the external device (as defined by DEVSIZE). If no bytes are to be transferred, DMODE[2:0] must be set to 0 in XSPI_CCR.
- If DL[31:0] = 0xFFFF FFFFF and DEVSIZE[4:0] = 0x1F (its maximum value indicating at 4-Gbyte device), the transfers continue indefinitely, stopping only after an abort request or after the XSPI is disabled. After the last memory address is read (at address 0xFFFF FFFF), reading continues with address = 0x0000 0000.

When the programmed number of bytes to be transmitted or received is reached, TCF bit is set in XSPI_SR and an interrupt is generated if TCIE = 1 in XSPI_CR. In the case of an undefined number of data, TCF is set when the limit of the external SPI memory is reached, according to the device size defined in XSPI_DCR1.

Triggering the start of a transfer in regular-command protocol

Depending on the XSPI configuration, there are three different ways to trigger the start of a transfer in indirect mode when using regular-command protocol. In general, the start of transfer is triggered as soon as the software gives the last information that is necessary for the command. More specifically in indirect mode, a transfer starts when one of the following sequence of events occurs:

- if no address is necessary (ADMODE[2:0] = 000) and if no data needs to be provided by the software (FMODE[1:0] = 01 or DMODE[2:0] = 000), and at the moment when a write is performed to INSTRUCTION[31:0] in XSPI_IR
- if an address is necessary (when ADMODE[2:0] ≠ 000) and if no data needs to be provided by the software (when FMODE[1:0] = 01 or DMODE[2:0] = 000), and at the moment when a write is performed to ADDRESS[31:0] in XSPI_AR

- if data needs to be provided by the software (when FMODE[1:0] = 00 and DMODE[2:0] ≠ 000), and at the moment when a write is performed to DATA[31:0] in XSPI_DR

A write to XSPI_ABR never triggers the communication start. If alternate bytes are required, they must have been programmed before.

As soon as a command is started, the BUSY bit is automatically set in XSPI_SR.

Triggering the start of a transfer in HyperBus protocol

Depending on the XSPI configuration, there are different ways to trigger the start of a command in indirect mode. In general, it is triggered as soon as the firmware gives the last information that is necessary for the transfer to start, and more specifically, a communication in indirect mode is triggered by one of the following register settings, when it is the last one to be executed:

- when a write is performed to ADDRESS[31:0] (XSPI_AR) with ADMODE[2:0] ≠ 000 in indirect read mode (FMODE[1:0] = 01)
- when a write is performed to DATA[31:0] (XSPI_DR) in indirect-write mode (when FMODE = 00)
- when a (dummy) write is performed to INSTRUCTION[31:0] (XSPI_IR) for indirect read mode (with ADMODE[2:0] = 000 and FMODE = 01)

As soon as a transfer is started, the BUSY bit (XSPI_SR[5]) is automatically set.

FIFO and data management

Data in indirect mode pass through a 64-byte FIFO that is internal to the XSPI. FLEVEL in XSPI_SR indicates how many bytes are currently being held in the FIFO.

In indirect-write mode (FMODE[1:0] = 00), the software adds data to the FIFO when it writes in the XSPI_DR. A word write adds 4 bytes to the FIFO, a half-word write adds 2 bytes, and a byte write adds only 1 byte. If the software adds too many bytes to the FIFO (more than indicated in DL[31:0]), the extra bytes are flushed from the FIFO at the end of the write operation (when TCF is set).

The byte/half-word accesses to the XSPI_DR must be done only to the least significant byte/halfword of the 32-bit register.

FTHRES is used to define a FIFO threshold after which point the FIFO threshold flag, FTF, gets set. In indirect-read mode, FTF is set when the number of valid bytes to be read from the FIFO is above the threshold. FTF is also set if there is any data left in the FIFO after the last byte is read from the external device, regardless of FTHRES setting. In indirect-write mode, the FTF is set when the number of empty bytes in the FIFO is above the threshold.

If FTIE = 1, there is an interrupt when the FTF is set. If DMAEN = 1, a DMA transfer is initiated when the FTF is set. The FTF is cleared by hardware as soon as the threshold condition is no longer true (after enough data has been transferred by the CPU or DMA).

In indirect-read mode, when the FIFO becomes full, the XSPI temporarily stops reading bytes from the external device to avoid an overrun.

The last data read in RX FIFO remains valid as long as there is no request for the next line. This means that, when the application reads several times in a row at the same location, the data is provided from the RX FIFO and not read again from the distant memory.

28.4.11 XSPI automatic status-polling mode

In automatic status-polling mode, the XSPI periodically starts a command to read a defined number of status bytes (up to four). The received bytes can be masked to isolate some status bits and an interrupt can be generated when the selected bits have a defined value.

The automatic status-polling mode must be used only in regular-command protocol. For HyperBus protocol, it is not exploitable since the read status register into the HyperFlash memory must be performed in two steps (a write operation followed by a read operation).

The access to the device begins in the same manner as in indirect-read mode. BUSY in XSPI_SR goes high at this point, and stays high even between the periodic accesses.

The content of MASK[31:0] in XSPI_PSMAR is used to mask the data from the external device in automatic status-polling mode:

- If the MASK[n] = 0, then bit n of the result is masked and not considered.
- If MASK[n] = 1, and the content of bit[n] is the same as MATCH[n] in XSPI_PSMAR, then there is a match for bit n.

If PMM = 0 in XSPI_CR, the AND-match mode is activated: SMF is set in XSPI_SR only when there is a match on all of the unmasked bits.

If PMM = 1 in XSPI_CR, the OR-match mode is activated: SMF gets set if there is a match on any of the unmasked bits.

An interrupt is called when SMF = 1 if SMIE = 1.

If APMS is set in XSPI_CR, the operation stops and BUSY goes to 0 as soon as a match is detected. Otherwise, BUSY stays at 1 and the periodic accesses continue until there is an abort or until the XSPI is disabled (EN = 0).

XSPI_DR contains the latest received status bytes (FIFO deactivated). The content of this register is not affected by the masking used in the matching logic. FTF in XSPI_SR is set as soon as a new reading of the status is complete. FTF is cleared as soon as the data is read.

In automatic status-polling mode, variable latency is not supported. As a consequence, the memory must be configured in fixed latency.

28.4.12 XSPI memory-mapped mode

When configured in memory-mapped mode, the external SPI device is seen as an internal memory.

Note: No more than 256 Mbytes can be addressed even if the external device capacity is larger.

If an access is made to an address outside of the range defined by DEVSIZE[4:0] but still within the 256-Mbyte range, then an AXI error is given. The effect of this error depends on the AXI master that attempted the access:

- If it is the Cortex CPU, a hard-fault interrupt is generated.
- If it is a DMA, a DMA transfer error is generated, and the corresponding DMA channel is automatically disabled.

Byte, half-word, and word access types are all supported.

A support for execute in place (XIP) operation is implemented, where the XSPI continues to load the bytes to the addresses following the most recent access. If subsequent accesses are continuous to the bytes that follow, then these operations end up quickly since their results were prefetched.

By default, the XSPI never stops its prefetch operation, it either keeps the previous read operation active with the NCS maintained low or it relaunches a new transfer, even if no access to the external device occurs for a long time. This prefetch feature can be disabled by setting the NOPREF bit in XSPI_CR. Disabling the prefetch can be beneficial for performance when the transfer requests are short and jumping frequently between non-consecutive addresses.

Since external devices tend to consume more power when the NCS is held low, the application may want to activate the timeout counter (TCEN = 1 in XSPI_CR): the NCS is released after a period defined by TIMEOUT[15:0] in XSPI_LPTR, when x cycles have elapsed without an access since the clock is inactive.

BUSY goes high as soon as the first memory-mapped access occurs. Because of the prefetch operations, BUSY does not fall until there is an abort, or the peripheral is disabled.

It is not recommended to program the flash memory using the memory-mapped writes. The indirect-write mode fulfills this operation.

However, if the application requires the use of the MCE for encryption (check MCE product availability), the memory-mapped write mode may be used to program encrypted data to external flash memory under the following conditions:

- Prefetch must be enabled.
- In block cipher mode, the CPU must write a complete 128-bit data block to prevent the MCE from initiating read-modify-write operations when only a few bytes need to be programmed. This precaution avoids incorrect programming operations. There are no specific constraints to respect if the MCE is used in stream cipher mode.
- Apply the abort sequence to exit memory-mapped mode when the data linked to the page has been written in the external memory buffers. The abort sequence triggers the start of the page programming.
- Switch to the automatic status-polling mode to monitor the completion of the page programing phase.
- Relaunch the write enable command in indirect mode, then switch back to the memory-mapped mode configuration to continue to program additional pages if any.

It is recommended to add a synchronization barrier between the end of the controller registers configuration and the first memory-mapped access to the external memory when the controller is configured in memory-mapped mode.

28.4.13 XSPI configuration introduction

The XSPI configuration is done in three steps:

1. XSPI system configuration
2. XSPI device configuration
3. XSPI mode configuration

28.4.14 XSPI system configuration

The XSPI is configured using XSPI_CR. The user must program:

- the functional mode with FMODE[1:0]
- the automatic status-polling mode behavior if needed with PMM and APMS
- the FIFO level with FTHRES
- DMA use with DMAEN

- the timeout counter use with TCEN
- the dual-memory configuration, if needed, with DMM

In case of an interrupt use, the respective enable bit can also be set during this phase.

If the timeout counter is used, the timeout value is programmed in XSPI_LPTR.

The DMA channel must not be enabled during the XSPI configuration: it must be enabled only when the operation is fully configured, to avoid any unexpected request generation.

The DMA and XSPI must be configured in a coherent manner regarding data length: FTHRES value must reflect the DMA burst size.

28.4.15 XSPI device configuration

The parameters related to the external device targeted are configured through XSPI_DCR1 and XSPI_DCR2.The user must program:

- the device size with DEVSIZE[4:0]
- the chip-select minimum high time with CSHT[5:0]
- the device frequency with PRESCALER[7:0]

DEVSIZE[4:0] defines the size of external memory using the following formula:

$$\text{Number of bytes in the device} = 2^{[\text{DEVSIZE}+1]}$$

where DEVSIZE+1 is the number of address bits required to address the external device. The external device capacity can go up to 4 Gbytes (addressed using 32 bits) in indirect mode, but the addressable space in memory-mapped mode is limited to 256 Mbytes.

If DMM = 1, DEVSIZE[4:0] must reflect the total capacity of the two devices together considering the above formula (DEVSIZE[4:0] value is so equal to one of the two memory capacities).

When the XSPI executes two commands, one immediately after the other, it raises the NCS high between the two commands, at least one CLK cycle by default.

If the external device requires more time between commands, CSHT[5:0] can be used to specify the minimum number of CLK cycles (up to 64) for which the NCS must remain high.

CKMODE indicates the level that the CLK takes between commands (when NCS = 1).

In HyperBus protocol, the device timing (t_{ACC} and t_{RWR}) and the Latency mode must be configured in XSPI_HLCR.

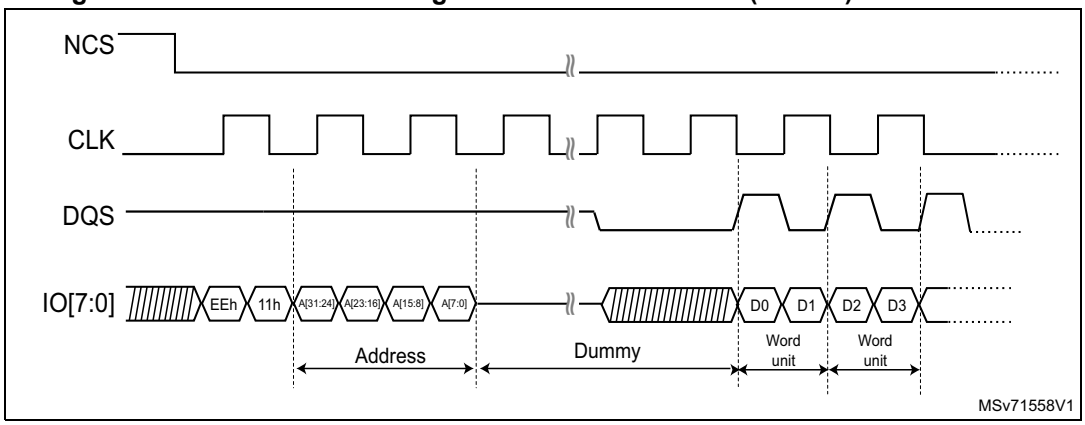
Memory types

External memory providers may have architecture and slight data formatting differences. The bitfield MTYP[2:0] into the XSPI_CR register allows targeting the right controller configuration depending on the associated memory type selected in the application. It is the responsibility of the software developer to align the controller configuration to fit with the targeted memory type.

The memory types are grouped in this way:

- D0/D1 data ordering in octal-SPI data mode (DMODE[2:0] = 100) in DTR mode by configuring MTYP[2:0] = 000. For instance, Micron is using such data ordering. In this configuration, the DQS is sent with a polarity inverted respect to the clock polarity.

Figure 211. D0/D1 data ordering in octal-SPI DTR mode (Micron) - Read access



- D1/D0 data ordering in octal-SPI data mode (DMODE[2:0] = 100) in DTR mode by configuring MTYP[2:0] = 001. For instance, Macronix is using this reverse data ordering in its Octaflash portfolio (this configuration is not adapted to its OctaRAM™ memories). DQS is keeping the same polarity as the clock when reading data from the memory. Refer to *Figure 201: DTR read in octal-SPI mode with DQS (Macronix mode) example*.
- D1/D0 data ordering in octal-SPI data mode (DMODE[2:0] = 100) in DTR mode by configuring MTYP[2:0] = 011 with specific address phase built with row and column to fit with Macronix OctaRAM™ memories requirement (refer to *Table 198: OctaRAM command address bit assignment (based on 64 Mb OctaRAM)*. This is the controller which translates internally the targeted address provided by the software in row/column address formatting to sent to the memory. DQS is keeping the same polarity as the clock when reading data from the memory.

Figure 212. OctaRAM read operation with reverse data ordering D1/D0

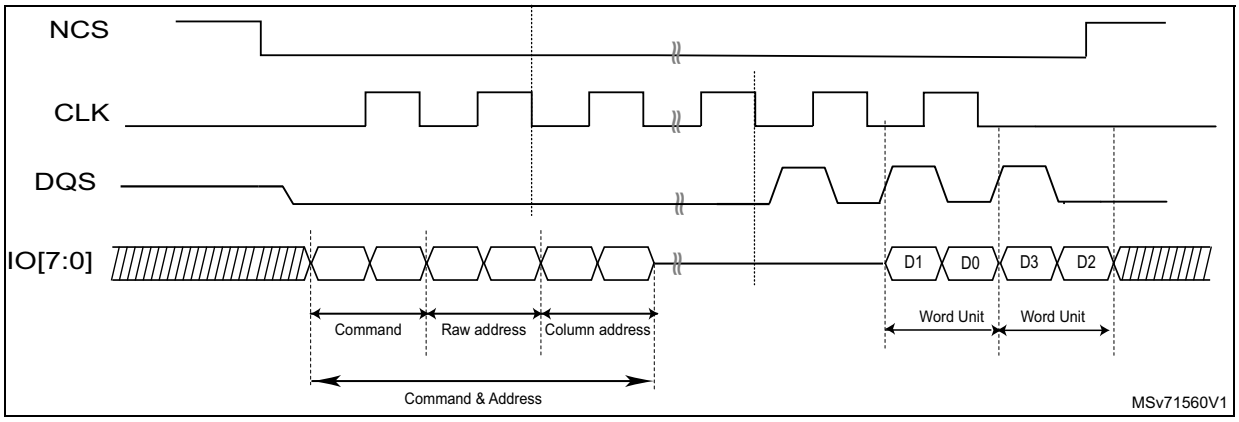


Table 198. OctaRAM command address bit assignment (based on 64 Mb[1] OctaRAM)

Clock	1st clock	2nd clock		3rd clock	
Function	Command	Row address		Column address	
SIO[7]	Command	Reserved	RA7	CA9	Reserved
SIO[6]		Reserved	RA6	CA8	Reserved
SIO[5]		Reserved	RA5	CA7	Reserved
SIO[4]		RA12	RA4	CA6	Reserved
SIO[3]		RA11	RA3	CA5	CA3
SIO[2]		RA10	RA2	CA4	CA2
SIO[1]		RA9	RA1	Reserved	CA1
SIO[0]		RA8	RA0	Reserved	CA0[2]

1. Example of 64 Mb OctaRAM address assignment:
Row Address [RA12:RA0]: 8K. Column address [CA9:CA0]: 1K. 64 Mb density = 8K x 1K x 8 bits
2. Column address A0 must be always 0.

- HyperBus memories need to be selected when targeted by the application. The configuration to set depends on the access type:
  - HyperBus memory mode: The protocol follows the HyperBus specification. MTYP[2:0] = 100 is the configuration to use to access the memory space.
  - HyperBus register mode (addressing register space): the memory-mapped accesses in this mode must be noncacheable, or the indirect read/write modes must be used. The configuration to be set for this particular register space access is MTYP[2:0] = 101.
- The software must configure MTYP[2:0] = 110 when the memory targeted comes from APmemory and DMODE[2:0] = 101 to fit with the memory provider requirements concerning the address formating.
- Standard mode. It is the mode to use whenever the targeted memory is not corresponding to any others configurations described in this section. MTYP[2:0] = 010 for this standard mode.

28.4.16 XSPI regular-command mode configuration

Indirect mode configuration

When FMODE[1:0] = 00, the indirect-write mode is selected and data can be sent to the external device. When FMODE[1:0] = 01, the indirect-read mode is selected and data can be read from the external device.

When the XSPI is used in indirect mode, the frames are constructed in the following way:

1. Specify a number of data bytes to read or write in XSPI_DLR.
2. Specify the frame timing in XSPI_TCR.
3. Specify the frame format in XSPI_CCR.
4. Specify the instruction in XSPI_IR.

5. Specify the optional alternate byte to be sent right after the address phase in XSPI_ABR.
6. Specify the targeted address in XSPI_AR.
7. Enable the DMA channel if needed.
8. Read/write the data from/to the FIFO through XSPI_DR (if no DMA usage).

If neither the address register (XSPI_AR) nor the data register (XSPI_DR) need to be updated for a particular command, then the command sequence starts as soon as XSPI_IR is written. This is the case when both ADMODE[2:0] and DMODE[2:0] equal 000, or if just ADMODE[2:0] = 000 when in indirect-read mode (FMODE[1:0] = 01).

When an address is required (ADMODE[2:0] ≠ 000) and the data register does not need to be written (FMODE[1:0] = 01 or DMODE[2:0] = 000), the command sequence starts as soon as the address is updated with a write to XSPI_AR.

In case of data transmission (FMODE[1:0] = 00 and DMODE[2:0] ≠ 000), the communication start is triggered by a write in the FIFO through XSPI_DR.

Automatic status-polling mode configuration

The automatic status-polling mode is enabled by setting FMODE[1:0] = 10. In this mode, the programmed frame is sent and the data is retrieved periodically.

The maximum amount of data read in each frame is 4 bytes. If more data is requested in XSPI_DLR, it is ignored, and only 4 bytes are read.The periodicity is specified in XSPI_PIR.

Once the status data has been retrieved, the following can be processed:

- Set SMF (an interrupt is generated if enabled).
- Stop automatically the periodic retrieving of the status bytes.

The received value can be masked with the value stored in XSPI_PSMKR, and can be ORed or ANDed with the value stored in XSPI_PSMAR.

In case of a match, SMF is set and an interrupt is generated if enabled. The XSPI can be automatically stopped if AMPS is set. In any case, the latest retrieved value is available in XSPI_DR.

When the XSPI is used in automatic status-polling mode, the frames are constructed in the following way:

1. Specify the input mask in XSPI_PSMKR.
2. Specify the comparison value in XSPI_PSMAR.
3. Specify the read period in XSPI_PIR.
4. Specify a number of data bytes to read in XSPI_DLR.
5. Specify the frame timing in XSPI_TCR.
6. Specify the frame format in XSPI_CCR.
7. Specify the instruction in XSPI_IR.
8. Specify the optional alternate byte to be sent right after the address phase in XSPI_ABR.
9. Specify the optional targeted address in XSPI_AR.

If the address register (XSPI_AR) does not need to be updated for a particular command, then the command sequence starts as soon as XSPI_CCR is written. This is the case when ADMODE[2:0] = 000.

When an address is required (ADMODE[2:0] ≠ 000), the command sequence starts as soon as the address is updated with a write to XSPI_AR.

Memory-mapped mode configuration

In memory-mapped mode, the external device is seen as an internal memory but with some latency during accesses. Read and write operations are allowed to the external device in this mode.

It is not recommended to program the flash memory using memory-mapped writes, as the internal flags for erase or programming status have to be polled. The indirect-write mode fulfills this operation, possibly in conjunction with the automatic status-polling mode.

Memory-mapped mode is entered by setting FMODE[1:0] = 11 in XSPI_CR.

The programmed instruction and frame are sent when an AXI master accesses the memory mapped space.

The FIFO is used as a prefetch buffer to anticipate any linear reads. Any access to XSPI_DR in this mode returns zero.

The data length register (XSPI_DLR) has no meaning in memory-mapped mode.

When the XSPI is used in memory-mapped mode, the frames are constructed in the following way:

1. Specify the frame timing in XSPI_TCR for read operation.
2. Specify the frame format in XSPI_CCR for read operation.
3. Specify the instruction in XSPI_IR.
4. Specify the optional alternate byte to be sent right after the address phase in XSPI_ABR for read operation.
5. Specify the frame timing in XSPI_WTCR for write operation.
6. Specify the frame format in XSPI_WCCR for write operation.
7. Specify the instruction in XSPI_WIR.
8. Specify the optional alternate byte to be sent right after the address phase in XSPI_WABR for write operation.

All the configuration operations must be completed (ensured by checking BUSY = 0) before the first access to the memory area: any register write operation when BUSY = 1 have no effect and is not signaled with an error response. On the first access, the XSPI becomes busy, and no further configuration is allowed. Then, the only way to get BUSY low is to clear the ENABLE bit or to abort by setting the ABORT bit.

Memory-mapped write operations are not supported without write strobe.

XSPI delayed data sampling when no DQS is used

By default, when no DQS is used, the XSPI samples the data driven by the external device one half of a CLK cycle after the external device drives the signal.

In case of any external signal delays, it may be useful to sample the data later. Using SSHIFT in XSPI_TCR, the sampling of the data can be shifted by half of a CLK cycle.

The firmware must clear SSHIFT when the data phase is configured in DTR mode (DDTR = 1).

XSPI delayed data sampling when DQS is used

When external DQS is used as a sampling clock, it is shifted precisely by one quarter of the SPI clock cycle, for all frequencies above freq_min, to compensate the data propagation delay in the “high-speed interface” when the product embeds one.

28.4.17 XSPI HyperBus protocol configuration

Indirect mode configuration (HyperBus)

When FMODE[1:0] = 00, the indirect-write mode is selected and data can be sent to the external device. When FMODE[1:0] = 01, the indirect-read mode is selected where data can be read from the external device. ADMODE must be configured with a value different from 000 (for instance ADMODE = 100).

When the XSPI is used in indirect mode, the frames are constructed in the following way:

1. Specify a number of data bytes to read or write in XSPI_DLR.
2. Specify the targeted address in XSPI_AR.
3. Enable the DMA channel if needed.
4. Read/write the data from/to the FIFO through XSPI_DR (if no DMA usage).

In indirect-read mode, the command sequence starts as soon as the address is updated with a write to XSPI_AR.

In indirect-write mode, the communication start is triggered by a write in the FIFO through XSPI_DR.

Memory-mapped mode configuration (HyperBus)

In memory-mapped mode, the external device is seen as an internal memory but with some latency during the accesses. Read and write operations are allowed to the external device in this mode.

It is not recommended to program the flash memory using the memory-mapped writes: the indirect-write mode fulfills this operation.

The memory-mapped mode is entered by setting FMODE[1:0] = 11. The programmed instruction and frame is sent when an AXI master is accessing the memory mapped space.

The FIFO is used as a prefetch buffer to anticipate any linear reads. Any access to XSPI_DR in this mode returns zero.

The data length register (XSPI_DLR) has no meaning in memory-mapped mode.

All the configuration operation must be completed before the first access to the memory area. On the first access, the XSPI becomes busy, and no configuration is allowed. Then, the only way to get BUSY low is to clear the ENABLE bit or to abort by setting the ABORT bit.

28.4.18 XSPI error management

The following errors set the TEF flag in XSPI_SR and generates an interrupt if enabled (TEIE = 1 in XSPI_CR):

- in indirect or automatic status-polling mode, when a wrong address has been programmed in XSPI_AR (according to the device size defined by DEVSIZE[4:0]).

- in indirect mode, if the address plus the data length exceed the device size. TEF is set as soon as the access is triggered.

In memory-mapped mode, the XSPI generates an AXI slave error in the following situations:

- The memory-mapped mode is disabled and an AXI read or write request occurs.
- Read or write address exceeds the size of the external memory.
- Abort is received while a read or write burst is ongoing when the abort condition is present at the moment of transfer phases preceding the data phase (In incremental or wrap burst type modes). To avoid the error, refer to *Section 28.4.20: XSPI BUSY and ABORT*.
- The XSPI is disabled while a read or write burst is requested.
- A write wrap burst is received.
- A write request is received while DQSE = 0 in XSPI_WCCR in octal DTR mode, in dual-memory configuration, in hyperbus mode or 16-bit mode.
- A read or write request is received while DMODE[2:0] = 000 (no data phase), except when MTYP[2:0] is HyperBus.
- Illegal access size when wrap read burst. This means AXI Bus transfer size (ARSIZE) is different from 8 bytes (only for memory-mapped mode).
- Bit DMM is set in XSPI_CR (dual-memory configuration) and 16-bit mode is set.

28.4.19 XSPI high-speed interface and calibration

To reach higher frequencies, a dedicated high-speed interface is inserted between the XSPI (or the I/O manager in case the product embeds one), and the I/O pads.

The following is valid for all data bus sizes 1, 2, 4, 8 or 16 bits.

The high-speed interface block embeds resynchronization registers that are clocked by a delayed clock created from a DLL (delay locked loop) also located in the high-speed interface. The high-speed interface features are controlled by registers located in the XSPI.

The purpose of resynchronization is primary to shift data or data strobe by one quarter of controller bus clock period, with a correct timing accuracy. DLL must be calibrated versus this clock period.

The calibration process is automatically enabled when one of the three conditions below is met:

- The XSPI exits reset state.
- A value is written in PRESCALER[7:0] of XSPI_DCR2.
- A value is written in XSPI_CCR.

The calibration process starts when the two following conditions are **both** met:

- The calibration has been enabled by one of the three conditions above.
- An action that sets BUSY = 1 is performed. For example the first transfer to memory after calibration is enabled. When the calibration is completed, BUSY returns to 0.

In case a periodic recalibration is needed (for example to take in account possible variations in temperature or power supply on a long duration), this recalibration must be triggered by writing periodically in PRESCALER[7:0] of XSPI_DCR2, while BUSY = 0.

Once the calibration is completed, the value of the SPI bus clock period, expressed in number of unitary delay, is available to user in COARSE[5:0] and FINE[6:0] of XSPI_CALFCR.

After auto-calibration, XSPI_CALSOR and XSPI_CALSIR are automatically loaded with the same value that corresponds to the delay for a quarter cycle.

The automatic calibration is not executed if at least one of both registers XSPI_CALSOR/XSPI_CALSIR is written between the last write operation on XSPI_DCR2 or XSPI_CCR and the next transfer start. In such case, the calibration values must be set by the software code into XSPI_CALSOR and XSPI_CALSIR registers. It may improve the flash programming time without relaunching the automatic calibration processing between two programming pages.

When the memory is not supporting DQS (DQSE = 0), the automatic calibration is not used in reception. The DLL Master is used instead for delaying the feedback clock (XSPI_CALMR). This delay needs to be adjusted by the application itself, using a software sequence that determines which delay is optimal to guarantee the correct read operations.

When the clock is divided in DTR transmission mode, the quarter cycle delays on DQS/data are not inserted by the DLLs themselves, but by internal flops design scheme. In SDR transmission mode, the DLLs are not used and this, whatever the clock prescaler value.

In case of DTR mode and prescaler bypassed (PRESCALER[7:0] = 0), the kernel clock provided to interface must have a 50 % duty-cycle.

When using the high-speed-interface, the system clock (AXI clock) must be at least as fast as the SPI clock.

28.4.20 XSPI BUSY and ABORT

Once the XSPI starts an operation with the external device, BUSY is automatically set in XSPI_SR.

In indirect mode, BUSY is reset once the XSPI has completed the requested command sequence and the FIFO is empty.

In automatic status-polling mode, BUSY goes low only after the last periodic access is complete, due to a match when APMS = 1 or due to an abort.

After the first access in memory-mapped mode, BUSY goes low only on an abort.

Any operation can be aborted by setting ABORT in XSPI_CR. Once the abort is completed, BUSY and ABORT are automatically reset, and the FIFO is flushed.

Before setting ABORT, the software must ensure that all the current transactions are finished using the synchronization barriers. When DMA is enabled to handle the data read or write operations in XSPI_DR, it is recommended to disable the DMA channel before aborting the XSPI.

Note: Some devices may misbehave if a write operation to a status register is aborted.

28.4.21 XSPI reconfiguration or deactivation

Before any XSPI reconfiguration, the software must ensure that all the transactions are completed:

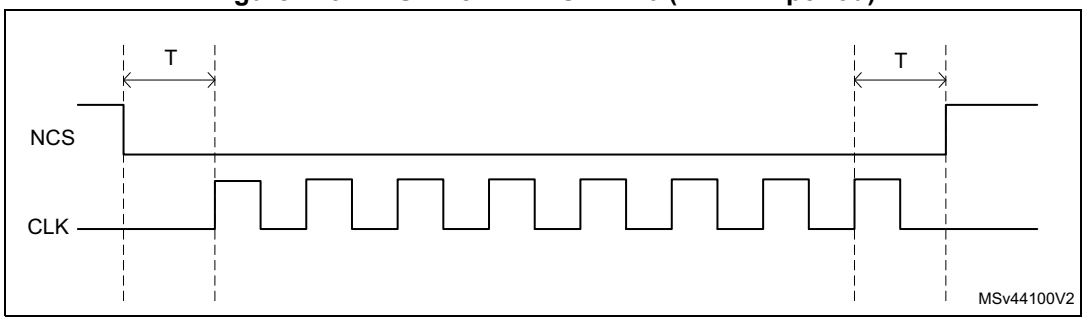
- After a memory-mapped write, the software must perform a dummy read followed by a synchronization barrier, then an abort.
- After a memory-mapped read, the software must perform a synchronization barrier then an abort.

28.4.22 NCS behavior

By default, NCS is high, deselecting the external device. NCS falls before an operation begins and rises as soon as it finishes.

When CKMODE = 0 (clock mode 0: CLK stays low when no operation is in progress), NCS falls one CLK cycle before an operation first rising CLK edge, and NCS rises one CLK cycle after the operation final rising CLK edge (see the figure below).

Figure 213. NCS when CKMODE = 0 (T = CLK period)



Note: Clock mode 3 is not supported.

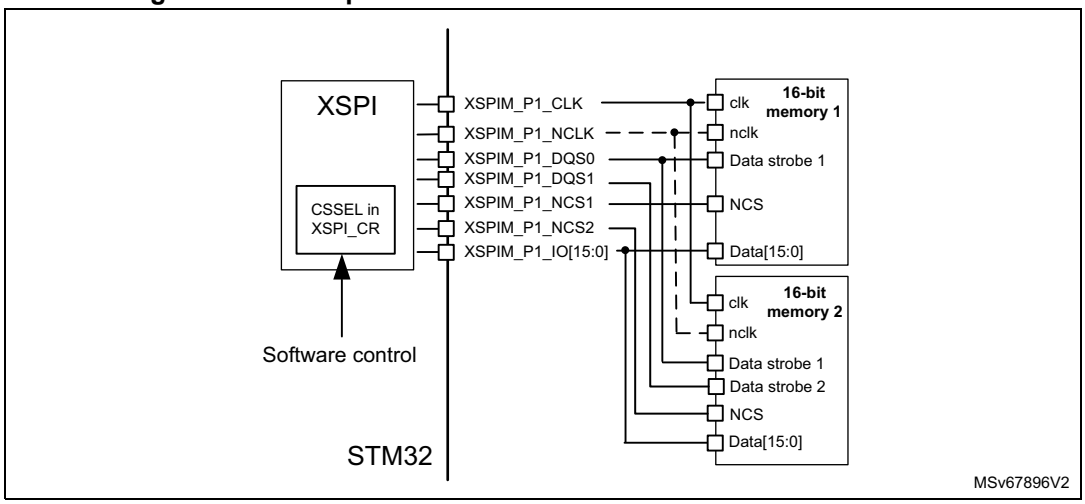
In dual-memory configuration, the NCS2 behaves exactly the same as the NCS1. The quad-SPI/octal-SPI memory 2 can then use the NCS1, and the NCS2 pin can be used for other functions.

28.4.23 Software control of two external memories

One XSPI instance is able to control the generation of the two chip-select signals (NCS1 and NCS2). This allows two external memories to be connected to the same I/O port driven by the XSPI.

The XSPI generates one NCS signal and the user defines how the NCS is directed to NCS1 or NCS2, by setting the CSSEL bit in XSPI_CR. Intended operation is semistatic commutation, such as starting an application reading an external flash memory and then, for the rest of the application, use an external SRAM connected to the same I/O bus.
From a memory-map point of view, these two memories are located in the same address range.

Figure 214. Example of software control of two external memories



28.4.24 Hardware-controlled extended memory support

One XSPI instance is able to control autonomously the generation of the two chip-select signals (NCS1 and NCS2). This feature is activated by setting EXTENDMEM in XSPI_DCR1 and allows two external memories to be connected to the same I/O port driven by the XSPI.

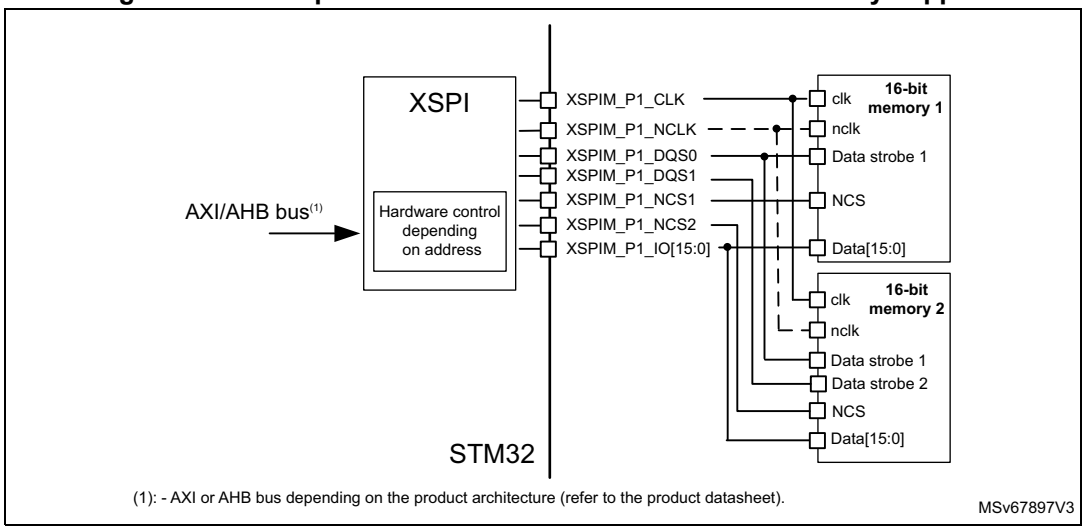
The aim is to extend by a factor two, the available external memory size. Both memories must be contiguous in the memory map. Both external memories must be of the same type and size. The size of one external memory must be set in the DEVSIZE bitfield in XSPI_CR.

- When the address presented on the interconnect bus in memory-mapped mode or in XSPI_AR register in indirect mode is in the range [XSPI_base_address : XSPI_base_address + $2^{DEVSIZE}$ - 1], NCS1 is generated.
- When the address presented on the interconnect bus in memory-mapped mode or in XSPI_AR register in indirect mode is in the range [XSPI_base_address + $2^{DEVSIZE}$:XSPI_base_address + 2 * $2^{DEVSIZE}$-1], NCS2 is generated.

Note: If the access crosses the middle boundary of the external memory, CS changes automatically (no write into XSPI_AR in indirect mode, or new memory access in memory-mapped mode is needed).

The EXTENDMEM feature (set in bit 21 of XSPI*_DCR1) and the Dual-memory mode (set in bit 6 of* XSPI*_CR) are exclusive. It is not possible to set both EXTENDMEM to 1 and DMM to 1 as this configuration is not supported.*

Figure 215. Example of hardware-controlled extended memory support



28.5 Address alignment and data number

The following table summarizes the effect of the address alignment and programmed data number depending on the use case.

Table 199. Address alignment cases

Memory type	Transaction type	Constraint on address(1)	Impact if constraint on address not respected	Constraint on number of bytes(1)	Impact if constraint on bytes not respected
Single, dual, quad flash or SRAM (DMM = 0)	IND(2) read	None	None	None	None
	MM(3) read				
	IND write				
	MM write				
Single, dual, quad flash or SRAM (DMM = 1)	IND read	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM read	None	None	None	None
	IND write	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM write	Even	Slave error	Even	Last byte is lost.
Octal flash in SDR mode	IND read	None	None	None	None
	MM read				
	IND write				
	MM write				
Octal memory in DTR mode without WDM(6), or 16-bit memory in SDR mode without WDM	IND read	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM read	None	None	None	None
	IND write	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM write	Even	Slave error	Even	Last byte is lost.

Table 199. Address alignment cases (continued)

Memory type	Transaction type	Constraint on address(1)	Impact if constraint on address not respected	Constraint on number of bytes(1)	Impact if constraint on bytes not respected
16-bit or dual-octal memory in DTR mode without WDM	IND read 32-bit	Aligned	ADDR[1:0] is assumed to be 00.(4)	N × 4	DLR[1:0] is assumed to be 11.(5)
	MM read	None	None	None	None
	IND write	Aligned	ADDR[1:0] is assumed to be 00.(4)	N × 4	DLR[1:0] is assumed to be 11.(5)
	MM write	Aligned	Slave error	N × 4	Last byte is lost.
Octal flash or RAM in DTR mode with WDM	IND read	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM read	None	None	None	None
	IND write(7)				
	MM write				
HyperBus	IND read	Even	ADDR[0] is set to 0.(4)	Even	DLR[0] is set to 1.(5)
	MM read	None	None	None	None
	IND write(7)				
	MM write				
16-bit data bus memory or dual-octal memory with WDM	IND read 32-bit	Aligned	ADDR[1:0] is assumed to be 00.(4)	N × 4	DLR[1:0] is assumed to be 11.(5)
	MM read	None	None	None	None
	IND write(8)				
	MM write				
16-bit HyperBus	IND read 32-bit	Aligned	ADDR[1:0] is assumed to be 00.(4)	N × 4	DLR[1:0] is assumed to be 11.(5)
	MM read	None	None	None	None
	IND write(9)				
	MM write				

1. To be respected by the software.
2. IND = indirect mode.
3. MM = memory-mapped mode.
4. Extra data at transfer start.
5. Extra data at transfer end.
6. WDM = write data mask.
7. If the FTHRES bitfield is set to the maximum value with DLR value greater than the data burst length, and if the DMA is enabled or the interrupt based on FIFO THRESHOLD Flag is enabled (FTF), the address must be modulo 2 aligned in DTR mode when the initiator (DMA, CPU, ...) is writing the data with a burst length equal to the FIFO size.
8. If the FTHRES bitfield is set to the maximum value with DLR value greater than the data burst length, and if the DMA is enabled or the interrupt based on FIFO THRESHOLD Flag is enabled (FTF), the address must be modulo 4 aligned in DTR mode or modulo 2 in SDR mode when the initiator (DMA, CPU, ...) is writing the data with a burst length equal to the FIFO size.
9. If the FTHRES bitfield is set to the maximum value with DLR value greater than the data burst length, and if the DMA is enabled or the interrupt based on FIFO THRESHOLD Flag is enabled (FTF), the address must be modulo 4 aligned in DTR mode when the initiator (DMA, CPU, ...) is writing the data with a burst length equal to the FIFO size.

28.6 XSPI interrupts

An interrupt can be produced on the following events:

- timeout
- status match
- FIFO threshold
- transfer complete
- transfer error

Separate interrupt enable bits are available to provide more flexibility.

Table 200. XSPI interrupt requests

Interrupt event	Event flag	Enable control bit
Timeout	TOF	TOIE
Status match	SMF	SMIE
FIFO threshold	FTF	FTIE
Transfer complete	TCF	TCIE
Transfer error	TEF	TEIE

28.7 XSPI registers

28.7.1 XSPI control register (XSPI_CR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSEL[1:0]		FMODE[1:0]		Res.	NOPREF_AXI	NOPREF	CSSEL	PMM	APMS	Res.	TOIE	SMIE	FTIE	TCIE	TEIE
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	FTHRES[5:0]						Res.	DMM	Res.	Res.	TCEN	DMAEN	ABORT	EN
		rw	rw	rw	rw	rw	rw		rw			rw	rw	rw	rw

Bits 31:30 **MSEL[1:0]**: Flash select

These bits select the memory to be addressed in single-, dual-, quad-, or octal-SPI mode in single-memory configuration (when DMM = 0).
- when in quad-SPI mode:
00: Data exchanged over IO[3:0]
01: Data exchanged over IO[7:4]
10: Data exchanged over IO[11:8]
11: Data exchanged over IO[15:12]
- when in octal-SPI mode or dual-quad configuration:
0x: Data exchanged over IO[7:0]
1x: Data exchanged over IO[15:8]
These bits are ignored when in dual-octal configuration (data on 8 bits and DMM = 1) or 16-bit configuration (data exchanged over IO[15:0]).

Note: This bitfield can be modified only when BUSY = 0.

Bits 29:28 **FMODE[1:0]**: Functional mode

This bitfield defines the XSPI functional mode of operation.
00: Indirect-write mode
01: Indirect-read mode
10: Automatic status-polling mode (relevant in regular-command protocol only)
11: Memory-mapped mode
If DMAEN = 1 already, then the DMA controller for the corresponding channel must be disabled before changing the FMODE[1:0] value.

Note: This bitfield can be modified only when BUSY = 0.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **NOPREF_AXI**: No prefetch for signaled AXI transactions

This bit disables the automatic prefetch in the external memory when the corresponding AXI transaction is signaled as not-prefetchable.
0: Prefetch is enabled, whatever the AXI prefetch transaction request.
1: Prefetch is disabled when the AXI transaction is signaled as not-prefetchable.
This bit has no effect when NOPREF = 1.

Bit 25 **NOPREF**: No prefetch data

This bit disables the automatic prefetch in the external memory.
0: Automatic prefetch enabled
1: Automatic prefetch disabled

Bit 24 **CSSEL**: chip select selection

This bit indicates if the XSPI must activate NCS1 or NCS2.
0: NCS1 active
1: NCS2 active

Note: This bit can be modified only when BUSY = 0.

Bit 23 **PMM**: Polling match mode

This bit indicates which method must be used to determine a match during the automatic status-polling mode.
0: AND-match mode, SMF is set if all the unmasked bits received from the device match the corresponding bits in the match register.
1: OR-match mode, SMF is set if any of the unmasked bits received from the device matches its corresponding bit in the match register.

Note: This bit can be modified only when BUSY = 0.

Bit 22 **APMS**: Automatic status-polling mode stop
This bit determines if the automatic status-polling mode is stopped after a match.
0: Automatic status-polling mode is stopped only by abort or by disabling the XSPI.
1: Automatic status-polling mode stops as soon as there is a match.
Note: This bit can be modified only when BUSY = 0.

Bit 21 Reserved, must be kept at reset value.

Bit 20 **TOIE**: Timeout interrupt enable
This bit enables the timeout interrupt.
0: Interrupt disabled
1: Interrupt enabled

Bit 19 **SMIE**: Status match interrupt enable
This bit enables the status match interrupt.
0: Interrupt disabled
1: Interrupt enabled

Bit 18 **FTIE**: FIFO threshold interrupt enable
This bit enables the FIFO threshold interrupt.
0: Interrupt disabled
1: Interrupt enabled

Bit 17 **TCIE**: Transfer complete interrupt enable
This bit enables the transfer complete interrupt.
0: Interrupt disabled
1: Interrupt enabled

Bit 16 **TEIE**: Transfer error interrupt enable
This bit enables the transfer error interrupt.
0: Interrupt disabled
1: Interrupt enabled

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

Bits 13:8 **FTHRES[5:0]**: FIFO threshold level
This bitfield defines, in indirect mode, the threshold number of bytes in the FIFO that causes the FIFO threshold flag FTF in XSPI_SR, to be set.
000000: FTF is set if there are one or more free bytes available to be written to in the FIFO in indirect-write mode, or if there are one or more valid bytes can be read from the FIFO in indirect-read mode.
000001: FTF is set if there are two or more free bytes available to be written to in the FIFO in indirect-write mode, or if there are two or more valid bytes can be read from the FIFO in indirect-read mode.
...
111111: FTF is set if there are 64 free bytes available to be written to in the FIFO in indirect-write mode, or if there are 64 valid bytes can be read from the FIFO in indirect-read mode.
Note: If DMAEN = 1, the DMA controller for the corresponding channel must be disabled before changing the FTHRES[5:0] value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **DMM**: Dual-memory configuration

This bit activates the dual-memory configuration, where two external devices are used simultaneously to double the throughput and the capacity.

0: Dual-memory configuration disabled

1: Dual-memory configuration enabled

Note: This bit can be modified only when BUSY = 0.

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **TCEN**: Timeout counter enable

This bit is valid only when the memory-mapped mode (FMODE[1:0] = 11) is selected. This bit enables the timeout counter.

0: The timeout counter is disabled, and thus the chip-select (NCS) remains active indefinitely after an access in memory-mapped mode.

1: The timeout counter is enabled, and thus the chip-select is released in the memory-mapped mode after TIMEOUT[15:0] cycles of external device inactivity.

Note: This bit can be modified only when BUSY = 0.

Bit 2 **DMAEN**: DMA enable

In indirect mode, the DMA can be used to input or output data via XSPI_DR. DMA transfers are initiated when FTF is set.

0: DMA disabled for indirect mode

1: DMA enabled for indirect mode

Note: Resetting the DMAEN bit while a DMA transfer is ongoing, breaks the handshake with the DMA. Do not write this bit during DMA operation.

Bit 1 **ABORT**: Abort request

This bit aborts the ongoing command sequence. It is automatically reset once the abort is completed. This bit stops the current transfer.

0: No abort requested

1: Abort requested

Note: This bit is always read as 0.

Bit 0 **EN**: Enable

This bit enables the XSPI.

0: XSPI disabled

1: XSPI enabled

Note: The DMA request can be aborted without having received the ACK in case this EN bit is cleared during the operation. In case this bit is set to 0 during a DMA transfer, the REQ signal to DMA returns to inactive state without waiting for the ACK signal from DMA to be active.

28.7.2 XSPI device configuration register 1 (XSPI_DCR1)

Address offset: 0x008

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:24 **MTYP[2:0]**: Memory type

This bitfield indicates the type of memory to be supported.

000: Micron mode, D0/D1 ordering in DTR 8-data-bit mode. Regular-command protocol in single-, dual-, quad-, and octal-SPI modes. In this mode, DQS signal polarity is inverted with respect to the memory clock signal. This is the default value and care must be taken to change MTYP[2:0] for memories different from Micron.

001: Macronix mode, D1/D0 ordering in DTR 8-data-bit mode. Regular-command protocol in single-, dual-, quad-, and octal-SPI modes.

010: Standard mode

011: Macronix RAM mode, D1/D0 ordering in DTR 8-data-bit mode. Regular-command protocol in single-, dual-, quad-, and octal-SPI modes with dedicated address mapping (address is built with row and column to fit with Macronix requirements).

100: HyperBus memory mode, the protocol follows the HyperBus specification.

101: HyperBus register mode, addressing register space. The memory-mapped accesses in this mode must be noncacheable, or indirect read/write modes must be used.

110: APmemory mode. If DMODE = 101, there is a special hardware operation on address word from the bit 10 and above to fit the provider requirement (shift operation on the left from the address bit 10, keeping this last at 0).

Others: Reserved

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **EXTENDMEM**: Extended memory support

This bit is only relevant when one XSPI drives two same size external memories located in contiguous places in the memory map. In this case, DEVSIZE[4:0] represents the size of each external memory.

When this bit is set, the XSPI automatically controls which of NCS1 or NCS2 is active.

0: NCS1 and NCS2 values depend from CSSEL bit in XSPI_CR (software controlled).

1: NCS1 and NCS2 values depend from the address of transfer (hardware controlled).

Bits 20:16 **DEVSIZE[4:0]**: Device size

This bitfield defines the size of the external device using the following formula:
Number of bytes in device = $2^{[DEVSIZE+1]}$.
DEVSIZE + 1 is effectively the number of address bits required to address the external device. The device capacity can be up to 4 Gbytes (addressed using 32-bits) in indirect mode, but the addressable space in memory-mapped mode is limited to 256 Mbytes.
In regular-command protocol, if DMM = 1 or EXTENDMEM = 1, DEVSIZE[4:0] must reflect the total capacity of the two devices together considering the above formula (DEVSIZE[4:0] value is so equal to one of the two memory capacities).

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

Bits 13:8 **CSHT[5:0]**: Chip-select high time

CSHT + 1 defines the minimum number of CLK cycles where the chip-select (NCS) must remain high between commands issued to the external device.
0x0: NCS stays high for at least 1 cycle between external device commands.
0x1: NCS stays high for at least 2 cycles between external device commands.
...
0x3F: NCS stays high for at least 64 cycles between external device commands.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **FRCK**: Free running clock

This bit configures the free running clock.
0: CLK is not free running.
1: CLK is free running (always provided).

Note: Once the bit has been set to 1, the BSY bit is set, the NCS signal remains set to 1 and the clock is sent. The only way to stop the clock is to perform an abort (which clears the BSY bit). Then, the FRCK bit must be cleared by software to allow controller transactions to take place with the external memory.

Bit 0 **CKMODE**: Clock mode 0

This bit is read-only and is read as zero.

28.7.3 XSPI device configuration register 2 (XSPI_DCR2)

Address offset: 0x00C

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **WRAPSIZE[2:0]**: Wrap size

This bitfield indicates the wrap size to which the memory is configured. For memories, which have a separate command for wrapped instructions, this bitfield indicates the wrap-size associated with the command held in XSPI_WPIR.

000: Wrapped reads are not supported by the memory.

010: External memory supports wrap size of 16 bytes.

011: External memory supports wrap size of 32 bytes.

100: External memory supports wrap size of 64 bytes.

101: External memory supports wrap size of 128 bytes.

Others: Reserved

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **PRESCALER[7:0]**: Clock prescaler

This bitfield defines the scaler factor for generating the CLK based on the kernel clock (value + 1).

0: $F_{CLK} = F_{KERNEL}$, kernel clock used directly as XSPI CLK (prescaler bypassed). In this case, if the DTR mode is used, it is mandatory to provide to the XSPI a kernel clock that has 50% duty-cycle.

1: $F_{CLK} = F_{KERNEL}/2$

2: $F_{CLK} = F_{KERNEL}/3$

...

255: $F_{CLK} = F_{KERNEL}/256$

For odd clock division factors, the CLK duty cycle is not 50%. The clock signal remains low one cycle longer than it stays high.

Writing this bitfield automatically starts a new calibration of high-speed interface DLL at the start of next transfer, except in case XSPI_CALSOR or XSPI_CALSIR have been written in the meantime. BUSY stays high during the whole calibration execution.

28.7.4 XSPI device configuration register 3 (XSPI_DCR3)

Address offset: 0x010

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **CSBOUND[4:0]**: NCS boundary

This bitfield enables the transaction boundary feature. When active, a minimum value of three is recommended.

The NCS is released on each boundary of $2^{CSBOUND}$ bytes.

0: NCS boundary disabled

Others: NCS boundary set to $2^{CSBOUND}$ bytes

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **MAXTRAN[7:0]**: Maximum transfer

This bitfield enables the communication regulation feature.
The NCS is released every MAXTRAN+1 clock cycles when the other XSPI request the access to the bus.
0: Maximum communication disabled
Others: Maximum communication is set to (MAXTRAN + 1) bytes.

28.7.5 XSPI device configuration register 4 (XSPI_DCR4)

Address offset: 0x014

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:0 **REFRESH[31:0]**: Refresh rate

This bitfield enables the refresh rate feature. The NCS is released every REFRESH + 1 clock cycles for writes, and REFRESH + 4 clock cycles for reads. These two values can be extended with few clock cycles when refresh occurs during a byte transmission in single-, dual-, or quad-SPI mode, because the byte transmission must be completed.
0: Refresh disabled
Others: Maximum communication length is set to REFRESH + 1 clock cycles.

28.7.6 XSPI status register (XSPI_SR)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	FLEVEL[6:0]							Res.	Res.	BUSY	TOF	SMF	FTF	TCF	TEF
	r	r	r	r	r	r	r			r	r	r	r	r	r

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:8 **FLEVEL[6:0]**: FIFO level

This bitfield gives the number of valid bytes that are being held in the FIFO.FLEVEL = 0 when the FIFO is empty, and 64 when it is full. In automatic-status polling mode, FLEVEL is zero.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **BUSY**: Busy

This bit is set when an operation is ongoing. It is cleared automatically when the operation with the external device is finished and the FIFO is empty.

Bit 4 **TOF**: Timeout flag

This bit is set when timeout occurs. It is cleared by writing 1 to CTOF.

Bit 3 **SMF**: Status match flag

This bit is set in automatic status-polling mode when the unmasked received data matches the corresponding bits in the match register (XSPI_PSMAR).
It is cleared by writing 1 to CSMF.

Bit 2 **FTF**: FIFO threshold flag

In indirect mode, this bit is set when the FIFO threshold has been reached, or if there is any data left in the FIFO after the reads from the external device are complete.
It is cleared automatically as soon as the threshold condition is no longer true.
In automatic status-polling mode this bit is set every time the status register is read, and the bit is cleared when the data register is read.

Bit 1 **TCF**: Transfer complete flag

This bit is set in indirect mode when the programmed number of data has been transferred or in any mode when the transfer has been aborted.It is cleared by writing 1 to CTCF.

Bit 0 **TEF**: Transfer error flag

This bit is set in indirect mode when an invalid address is being accessed in indirect mode.
It is cleared by writing 1 to CTEF.

28.7.7 XSPI flag clear register (XSPI_FCR)

Address offset: 0x024

Reset value: 0x0000 0000

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

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

Bit 4 **CTOF**: Clear timeout flag

Writing 1 clears the TOF flag in the XSPI_SR register.

Bit 3 **CSMF**: Clear status match flag

Writing 1 clears the SMF flag in the XSPI_SR register.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CTCF**: Clear transfer complete flag

Writing 1 clears the TCF flag in the XSPI_SR register.

Bit 0 **CTEF**: Clear transfer error flag

Writing 1 clears the TEF flag in the XSPI_SR register.

28.7.8 XSPI data length register (XSPI_DLR)

Address offset: 0x040

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:0 **DL[31:0]**: Data length

Number of data to be retrieved (value + 1) in indirect and automatic status-polling modes. A value not greater than three (indicating 4 bytes) must be used for automatic status-polling mode.

All 1's in indirect mode means undefined length, where XSPI continues until the end of the memory, as defined by DEVSIZE.

0x0000_0000: 1 byte is to be transferred.

0x0000_0001: 2 bytes are to be transferred.

0x0000_0002: 3 bytes are to be transferred.

0x0000_0003: 4 bytes are to be transferred.

...

0xFFFF_FFFD: 4,294,967,294 (4G-2) bytes are to be transferred.

0xFFFF_FFFE: 4,294,967,295 (4G-1) bytes are to be transferred.

0xFFFF_FFFF: undefined length; all bytes, until the end of the external device, (as defined by DEVSIZE) are to be transferred. Continue reading indefinitely if DEVSIZE = 0x1F.

DL[0] is stuck at 1 in dual-memory configuration (DMM = 1) even when 0 is written to this bit, thus assuring that each access transfers an even number of bytes.

This bitfield has no effect when in memory-mapped mode.

28.7.9 XSPI address register (XSPI_AR)

Address offset: 0x048

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0 and FMODE ≠ 11.

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

Bits 31:0 **ADDRESS[31:0]**: Address

Address to be sent to the external device. In HyperBus protocol, this bitfield must be even as this protocol is 16-bit word oriented. In dual-memory configuration, AR[0] is forced to 0.

Caution: Some memory specifications consider that each address corresponds to a 16-bit value. XSPI considers that each address corresponds to an 8-bit value. So the software needs to multiply the address by two when accessing the memory registers.

28.7.10 XSPI data register (XSPI_DR)

Address offset: 0x050

Reset value: 0x0000 0000

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

Bits 31:0 **DATA[31:0]**: Data

Data to be sent/received to/from the external SPI device

In indirect-write mode, data written to this register is stored on the FIFO before it is sent to the external device during the data phase. If the FIFO is too full, a write operation is stalled until the FIFO has enough space to accept the amount of data being written.

In indirect-read mode, reading this register gives (via the FIFO) the data that was received from the external device. If the FIFO does not have as many bytes as requested by the read operation and if BUSY = 1, the read operation is stalled until enough data is present or until the transfer is complete, whichever happens first.

In automatic status-polling mode, this register contains the last data read from the external device (without masking).

Word, half-word, and byte accesses to this register are supported. In indirect-write mode, a byte write adds 1 byte to the FIFO, a half-word write 2 bytes, and a word write 4 bytes.

Similarly, in indirect-read mode, a byte read removes 1 byte from the FIFO, a halfword read 2 bytes, and a word read 4 bytes. Accesses in indirect mode must be aligned to the bottom of this register: A byte read must read DATA[7:0] and a half-word read must read DATA[15:0].

28.7.11 XSPI polling status mask register (XSPI_PSMKR)

Address offset: 0x080

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:0 **MASK[31:0]**: Status mask

Mask to be applied to the status bytes received in automatic status-polling mode

For bit n:

0: Bit n of the data received in automatic status-polling mode is masked and its value is not considered in the matching logic.

1: Bit n of the data received in automatic status-polling mode is unmasked and its value is considered in the matching logic.

28.7.12 XSPI polling status match register (XSPI_PSMAR)

Address offset: 0x088

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:0 **MATCH[31:0]**: Status match

Value to be compared with the masked status register to get a match

28.7.13 XSPI polling interval register (XSPI_PIR)

Address offset: 0x090

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **INTERVAL[15:0]**: Polling interval

Number of CLK cycles between a read during the automatic status-polling phases

28.7.14 XSPI communication configuration register (XSPI_CCR)

Address offset: 0x100

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	DQSE	Res.	DDTR	DMODE[2:0]			Res.	Res.	ABSIZE[1:0]		ABDTR	ABMODE[2:0]		
		rw		rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ADSIZE[1:0]		ADDTR	ADMODE[2:0]			Res.	Res.	ISIZE[1:0]		IDTR	IMODE[2:0]		
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **DQSE**: DQS enable

This bit enables the data strobe management.

0: DQS disabled

1: DQS enabled

Bit 28 Reserved, must be kept at reset value.

Bit 27 **DDTR**: Data double transfer rate

This bit sets the DTR mode for the data phase.

0: DTR mode disabled for the data phase

1: DTR mode enabled for the data phase

Bits 26:24 **DMODE[2:0]**: Data mode
This bitfield defines the data phase mode of operation.
000: No data
001: Data on a single line
010: Data on two lines
011: Data on four lines
100: Data on eight lines
101: Data on 16 lines
Others: Reserved

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **ABSIZE[1:0]**: Alternate-byte size
This bitfield defines the alternate-byte size.
00: 8-bit alternate bytes
01: 16-bit alternate bytes
10: 24-bit alternate bytes
11: 32-bit alternate bytes

Bit 19 **ABDTR**: Alternate-byte double transfer rate
This bit sets the DTR mode for the alternate-byte phase.
0: DTR mode disabled for the alternate-byte phase
1: DTR mode enabled for the alternate-byte phase

Bits 18:16 **ABMODE[2:0]**: Alternate-byte mode
This bitfield defines the alternate-byte phase mode of operation.
000: No alternate bytes
001: Alternate bytes on a single line
010: Alternate bytes on two lines
011: Alternate bytes on four lines
100: Alternate bytes on eight lines
Others: Reserved

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

Bits 13:12 **ADSIZE[1:0]**: Address size
This bitfield defines the address size.
00: 8-bit address
01: 16-bit address
10: 24-bit address
11: 32-bit address

Bit 11 **ADDTR**: Address double transfer rate
This bit sets the DTR mode for the address phase.
0: DTR mode disabled for the address phase
1: DTR mode enabled for the address phase

Bits 10:8 **ADMODE[2:0]**: Address mode
This bitfield defines the address phase mode of operation.
000: No address
001: Address on a single line
010: Address on two lines
011: Address on four lines
100: Address on eight lines
Others: Reserved

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **ISIZE[1:0]**: Instruction size
This bitfield defines the instruction size.
00: 8-bit instruction
01: 16-bit instruction
10: 24-bit instruction
11: 32-bit instruction

Bit 3 **IDTR**: Instruction double transfer rate
This bit sets the DTR mode for the instruction phase.
0: DTR mode disabled for the instruction phase
1: DTR mode enabled for the instruction phase

Bits 2:0 **IMODE[2:0]**: Instruction mode
This bitfield defines the instruction phase mode of operation.
000: No instruction
001: Instruction on a single line
010: Instruction on two lines
011: Instruction on four lines
100: Instruction on eight lines
Others: Reserved

28.7.15 XSPI timing configuration register (XSPI_TCR)

Address offset: 0x108

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

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

Bit 31 Reserved, must be kept at reset value.

Bit 30 **SSHIFT**: Sample shift
By default, the XSPI samples data 1/2 of a CLK cycle after the data is driven by the external device.
This bit allows the data to be sampled later in order to consider the external signal delays.
0: No shift
1: 1/2 cycle shift
The software must ensure that SSHIFT = 0 when the data phase is configured in DTR mode (when DDTR = 1.)

Bits 29:5 Reserved, must be kept at reset value.

Bits 4:0 **DCYC[4:0]**: Number of dummy cycles
This bitfield defines the duration of the dummy phase according to the memory latency.
In both SDR and DTR modes, it specifies a number of CLK cycles (0-31).

28.7.16 XSPI instruction register (XSPI_IR)

Address offset: 0x110

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:0 **INSTRUCTION[31:0]**: Instruction
Instruction to be sent to the external SPI device

28.7.17 XSPI alternate bytes register (XSPI_ABR)

Address offset: 0x120

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:0 **ALTERNATE[31:0]**: Alternate bytes
Optional data to be sent to the external SPI device right after the address.

28.7.18 XSPI low-power timeout register (XSPI_LPTR)

Address offset: 0x130

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TIMEOUT[15:0]**: Timeout period

After each access in memory-mapped mode, the XSPI prefetches the subsequent bytes and hold them in the FIFO.

This bitfield indicates how many CLK cycles the XSPI waits after the clock becomes inactive and until it raises the NCS, putting the external device in a lower-consumption state.

28.7.19 XSPI wrap communication configuration register (XSPI_WPCCR)

Address offset: 0x140

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	DQSE	Res.	DDTR	DMODE[2:0]			Res.	Res.	ABSIZE[1:0]		ABDTR	ABMODE[2:0]		
		rw		rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ADSIZE[1:0]		ADDTR	ADMODE[2:0]			Res.	Res.	ISIZE[1:0]		IDTR	IMODE[2:0]		
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **DQSE**: DQS enable

This bit enables the data strobe management.

0: DQS disabled

1: DQS enabled

Bit 28 Reserved, must be kept at reset value.

Bit 27 **DDTR**: Data double transfer rate

This bit sets the DTR mode for the data phase.

0: DTR mode disabled for the data phase

1: DTR mode enabled for the data phase

Bits 26:24 **DMODE[2:0]**: Data mode

This bitfield defines the data phase mode of operation.

000: No data

001: Data on a single line

010: Data on two lines

011: Data on four lines

100: Data on eight lines

101: Data on 16 lines

Others: Reserved

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **ABSIZE[1:0]**: Alternate-byte size

This bitfield defines the alternate-byte size.

00: 8-bit alternate bytes

01: 16-bit alternate bytes

10: 24-bit alternate bytes

11: 32-bit alternate bytes

Bit 19 **ABDTR**: Alternate-byte double transfer rate
This bit sets the DTR mode for the alternate-byte phase.
0: DTR mode disabled for the alternate-byte phase
1: DTR mode enabled for the alternate-byte phase

Bits 18:16 **ABMODE[2:0]**: Alternate-byte mode
This bitfield defines the alternate byte phase mode of operation.
000: No alternate bytes
001: Alternate bytes on a single line
010: Alternate bytes on two lines
011: Alternate bytes on four lines
100: Alternate bytes on eight lines
Others: Reserved

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

Bits 13:12 **ADSIZE[1:0]**: Address size
This bitfield defines the address size.
00: 8-bit address
01: 16-bit address
10: 24-bit address
11: 32-bit address

Bit 11 **ADDTR**: Address double transfer rate
This bit sets the DTR mode for the address phase.
0: DTR mode disabled for the address phase
1: DTR mode enabled for the address phase

Bits 10:8 **ADMODE[2:0]**: Address mode
This bitfield defines the address phase mode of operation.
000: No address
001: Address on a single line
010: Address on two lines
011: Address on four lines
100: Address on eight lines
Others: Reserved

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **ISIZE[1:0]**: Instruction size
This bitfield defines the instruction size.
00: 8-bit instruction
01: 16-bit instruction
10: 24-bit instruction
11: 32-bit instruction

Bit 3 **IDTR**: Instruction double transfer rate
This bit sets the DTR mode for the instruction phase.
0: DTR mode disabled for the instruction phase
1: DTR mode enabled for the instruction phase

Bits 2:0 **IMODE[2:0]**: Instruction mode
This bitfield defines the instruction phase mode of operation.
000: No instruction
001: Instruction on a single line
010: Instruction on two lines
011: Instruction on four lines
100: Instruction on eight lines
Others: Reserved

28.7.20 XSPI wrap timing configuration register (XSPI_WPTCR)

Address offset: 0x148

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

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

Bit 31 Reserved, must be kept at reset value.

Bit 30 **SSHIFT**: Sample shift
By default, the XSPI samples data 1/2 of a CLK cycle after the data is driven by the external device.
This bit allows the data to be sampled later in order to consider the external signal delays.
0: No shift
1: 1/2 cycle shift
The software must ensure that SSHIFT = 0 when the data phase is configured in DTR mode (when DDTR = 1).

Bits 29:5 Reserved, must be kept at reset value.

Bits 4:0 **DCYC[4:0]**: Number of dummy cycles
This bitfield defines the duration of the dummy phase according to the memory latency.
In both SDR and DTR modes, it specifies a number of CLK cycles (0-31).

28.7.21 XSPI wrap instruction register (XSPI_WPIR)

Address offset: 0x150

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:0 **INSTRUCTION[31:0]**: Instruction
Instruction to be sent to the external SPI device

28.7.22 XSPI wrap alternate byte register (XSPI_WPABR)

Address offset: 0x160

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:0 **ALTERNATE[31:0]**: Alternate bytes
Optional data to be sent to the external SPI device right after the address

28.7.23 XSPI write communication configuration register (XSPI_WCCR)

Address offset: 0x180

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0. Its content has a meaning only when requesting write operations in memory-mapped mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	DQSE	Res.	DDTR	DMODE[2:0]			Res.	Res.	ABSIZE[1:0]		ABDTR	ABMODE[2:0]		
		rw		rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ADSIZE[1:0]		ADDTR	ADMODE[2:0]			Res.	Res.	ISIZE[1:0]		IDTR	IMODE[2:0]		
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **DQSE**: DQS enable
This bit enables the data strobe management.
0: DQS disabled
1: DQS enabled

Bit 28 Reserved, must be kept at reset value.

Bit 27 **DDTR**: Data double transfer rate
This bit sets the DTR mode for the data phase.
0: DTR mode disabled for the data phase
1: DTR mode enabled for the data phase

Bits 26:24 **DMODE[2:0]**: Data mode
This bitfield defines the data phase mode of operation.
000: No data
001: Data on a single line
010: Data on two lines
011: Data on four lines
100: Data on eight lines
101: Data on 16 lines
Others: Reserved

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **ABSIZE[1:0]**: Alternate-byte size
This bitfield defines the alternate-byte size:
00: 8-bit alternate bytes
01: 16-bit alternate bytes
10: 24-bit alternate bytes
11: 32-bit alternate bytes

Bit 19 **ABDTR**: Alternate-byte double-transfer rate
This bit sets the DTR mode for the alternate-byte phase.
0: DTR mode disabled for the alternate-byte phase
1: DTR mode enabled for the alternate-byte phase

Bits 18:16 **ABMODE[2:0]**: Alternate-byte mode
This bitfield defines the alternate-byte phase mode of operation.
000: No alternate bytes
001: Alternate bytes on a single line
010: Alternate bytes on two lines
011: Alternate bytes on four lines
100: Alternate bytes on eight lines
Others: Reserved

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

Bits 13:12 **ADSIZE[1:0]**: Address size
This bitfield defines the address size.
00: 8-bit address
01: 16-bit address
10: 24-bit address
11: 32-bit address

Bit 11 **ADDTR**: Address double transfer rate
This bit sets the DTR mode for the address phase.
0: DTR mode disabled for the address phase
1: DTR mode enabled for the address phase

Bits 10:8 **ADMODE[2:0]**: Address mode
This bitfield defines the address phase mode of operation.
000: No address
001: Address on a single line
010: Address on two lines
011: Address on four lines
100: Address on eight lines
Others: Reserved

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **ISIZE[1:0]**: Instruction size
This bitfield defines the instruction size:
00: 8-bit instruction
01: 16-bit instruction
10: 24-bit instruction
11: 32-bit instruction

Bit 3 **IDTR**: Instruction double transfer rate
This bit sets the DTR mode for the instruction phase.
0: DTR mode disabled for the instruction phase
1: DTR mode enabled for the instruction phase

Bits 2:0 **IMODE[2:0]**: Instruction mode
This bitfield defines the instruction phase mode of operation.
000: No instruction
001: Instruction on a single line
010: Instruction on two lines
011: Instruction on four lines
100: Instruction on eight lines
Others: Reserved

28.7.24 XSPI write timing configuration register (XSPI_WTCR)

Address offset: 0x188

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0. Its content has a meaning only when requesting write operations in memory-mapped mode.

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

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

Bits 4:0 **DCYC[4:0]**: Number of dummy cycles
This bitfield defines the duration of the dummy phase according to the memory latency. In both SDR and DTR modes, it specifies a number of CLK cycles (0-31).

28.7.25 XSPI write instruction register (XSPI_WIR)

Address offset: 0x190

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0. Its content has a meaning only when requesting write operations in memory-mapped mode.

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

Bits 31:0 **INSTRUCTION[31:0]**: Instruction
Instruction to be sent to the external SPI device

28.7.26 XSPI write alternate byte register (XSPI_WABR)

Address offset: 0x1A0

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0. Its content has a meaning only when requesting write operations in memory-mapped mode.

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

Bits 31:0 **ALTERNATE[31:0]**: Alternate bytes
Optional data to be sent to the external SPI device right after the address

28.7.27 XSPI HyperBus latency configuration register (XSPI_HLCR)

Address offset: 0x200

Reset value: 0x0000 0000

This register can be modified only when BUSY = 0.

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

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **TRWR[7:0]:** Read-write minimum recovery time

Device Read-to-write/write-to-read minimum recovery time expressed in number of communication clock cycles

Bits 15:8 **TACC[7:0]**: Access time

Device access time according to the memory latency, expressed in number of communication clock cycles

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

Bit 1 **WZL:** Write zero latency

This bit enables zero latency on write operations.

0: Latency on write accesses

1: No latency on write accesses

Bit 0 **LM:** Latency mode

This bit selects the latency mode.

0: Variable initial latency

1: Fixed latency

Note: This bit must be set when using the dual-octal HyperBus configuration.

28.7.28 XSPI full-cycle calibration configuration (XSPI_CALFCR)

Address offset: 0x210

Reset value: 0x0000 0000

This read-only register gives the calibration code needed by the DLL master so that its delay is equivalent to a full memory-clock cycle. The value of this register is updated every time that auto-calibration finishes.

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

Bit 31 **CALMAX:** Max value
This bit gets set when the memory-clock period is outside the range of DLL master, in which case XSPI_CALFCR and XSPI_CALSR are updated with the values for the maximum delay.

Bits 30:21 Reserved, must be kept at reset value.

Bits 20:16 **COARSE[4:0]**: Coarse calibration
The delay unitary value for this bitfield depends on product technology (see the datasheet).

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:0 **FINE[6:0]**: Fine calibration
The delay unitary value for this bitfield depends on product technology (see the datasheet).

28.7.29 XSPI DLL master calibration configuration (XSPI_CALMR)

Address offset: 0x218

Reset value: 0x0000 0000

The DLL Master is used for delaying the feedback clock when reading without DQS. The delay of the master DLL is determined by the value in this register.

This register can always be read by software and can be modified only when BUSY = 0.

This register never gets updated automatically by hardware.

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

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **COARSE[4:0]**: Coarse calibration
The delay unitary value for this bitfield depends on product technology (see the datasheet).

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:0 **FINE[6:0]**: Fine calibration
The delay unitary value for this bitfield depends on product technology (see the datasheet).

28.7.30 XSPI DLL slave output calibration configuration (XSPI_CALSOR)

Address offset: 0x220

Reset value: 0x0000 0000

The DLL output slave is used to delay the output data in DDR mode for write operations. The delay of the output slave DLL is determined by the value in this register.

This register is updated automatically by hardware at the end of calibration (at the same moment that XSPI_CALFCR is updated).

If this register is written after the last write to XSPI_DCR2 or XSPI_CCR, then auto-calibration is not executed on the next transfer (auto-calibration is not performed for XSPI_CALSIR as well).

This register can always be read by software and can be modified only when BUSY = 0.

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

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **COARSE[4:0]**: Coarse calibration
The delay unitary value for this bitfield depends on product technology.

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:0 **FINE[6:0]**: Fine calibration
The delay unitary value for this bitfield depends on product technology.

28.7.31 XSPI DLL slave input calibration configuration (XSPI_CALSIR)

Address offset: 0x228

Reset value: 0x0000 0000

The DLL input slave is used to delay the DQS input for sampling the data when DQS is enabled for read operations. The delay of the input slave DLL is determined by the value in this register.

This register is updated automatically by hardware at the end of calibration (at the same moment that XSPI_CALFCR is updated).

If this register is written after the last write to XSPI_DCR2 or XSPI_CCR, then auto-calibration is not executed on the next transfer (auto-calibration is not performed for XSPI_CALSOR as well).

This register can always be read by software and can be modified only when BUSY = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COARSE[4:0]				
											rw	rw	rw	rw	rw

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

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **COARSE[4:0]**: Coarse calibration
The delay unitary value for this bitfield depends on product technology.

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:0 **FINE[6:0]**: Fine calibration
The delay unitary value for this bitfield depends on product technology.

28.7.32 XSPI register map

Table 201. XSPI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	XSPI_CR	MSEL[1:0]		FMDOE[1:0]		Res.	NOPREF_AXI	NOPREF	CSSEL	PMM	APMS	Res.	TOIE	SMIE	FTIE	TCIE	TEIE	Res.	Res.	FTHRES[5:0]						Res.	DMM	Res.	Res.	TCEN	DMAEN	ABORT	EN
	Reset value	0	0	0	0		0	0	0	0	0		0	0	0	0	0			0	0	0	0	0	0		0			0	0	0	0
0x0004	Reserved	Reserved																															
0x008	XSPI_DCR1	Res.	Res.	Res.	Res.	Res.	MTYP [2:0]			Res.	Res.	Res. EXTENDMEM	DEVSIZE[4:0]					Res.	Res.	CSHT[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	FRCK	CKMODE
	Reset value						0	0	0			0	0	0	0	0	0			0	0	0	0	0	0							0	0

Table 201. XSPI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00C	XSPI_DCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRAPSIZE[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[7:0]							
	Reset value														0	0	0									0	0	0	0	0	0	0	0
0x010	XSPI_DCR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSBOUND[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MAXTRAN[7:0]							
	Reset value												0	0	0	0	0									0	0	0	0	0	0	0	0
0x014	XSPI_DCR4	REFRESH[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018-0x01C	Reserved	Reserved																															
0x020	XSPI_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLEVEL[6:0]							Res.	Res.	BUSY	TOF	SMF	FTF	TCF	TEF
	Reset value																		0	0	0	0	0	0	0			0	0	0	0	0	0
0x024	XSPI_FCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTOF	CSMF	Res.	CTCF	CTEF
	Reset value																												0	0		0	0
0x028-0x03C	Reserved	Reserved																															
0x040	XSPI_DLR	DL[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	Reserved	Reserved																															
0x048	XSPI_AR	ADDRESS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	Reserved	Reserved																															
0x050	XSPI_DR	DATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054-0x07C	Reserved	Reserved																															
0x080	XSPI_PSMKR	MASK[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084	Reserved	Reserved																															
0x088	XSPI_PSMAR	MATCH[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08C	Reserved	Reserved																															
0x090	XSPI_PIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTERVAL[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x094-0x0FC	Reserved	Reserved																															

Table 201. XSPI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x100	XSPI_CCR	Res.	Res.	DQSE	Res.	DDTR	DMODE [2:0]			Res.	Res.	ABSIZE [1:0]		ABDTR	ABMODE [2:0]			Res.	Res.	ADSIZE [1:0]		ADDTR	ADMODE [2:0]			Res.	Res.	ISIZE[1:0]		IDTR	IMODE[2:0]		
	Reset value			0		0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0
0x104	Reserved	Reserved																															
0x108	XSPI_TCR	Res.	SSHIFT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCYC[4:0]				
	Reset value		0																										0	0	0	0	0
0x10C	Reserved	Reserved																															
0x110	XSPI_IR	INSTRUCTION[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x114-0x11C	Reserved	Reserved																															
0x120	XSPI_ABR	ALTERNATE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x124-0x12C	Reserved	Reserved																															
0x130	XSPI_LPTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIMEOUT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x134-0x13C	Reserved	Reserved																															
0x140	XSPI_WPCCR	Res.	Res.	DQSE	Res.	DDTR	DMODE [2:0]			Res.	Res.	ABSIZE [1:0]		ABDTR	ABMODE [2:0]			Res.	Res.	ADSIZE [1:0]		ADDTR	ADMODE [2:0]			Res.	Res.	ISIZE [1:0]		IDTR	IMODE [2:0]		
	Reset value			0		0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0
0x144	Reserved	Reserved																															
0x148	XSPI_WPTCR	Res.	SSHIFT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCYC[4:0]				
	Reset value		0																										0	0	0	0	0
0x14C	Reserved	Reserved																															
0x150	XSPI_WPIR	INSTRUCTION[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x154-0x15C	Reserved	Reserved																															
0x160	XSPI_WPABR	ALTERNATE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x164-0x17C	Reserved	Reserved																															
0x180	XSPI_WCCR	Res.	Res.	DQSE	Res.	DDTR	DMODE [2:0]			Res.	Res.	ABSIZE [1:0]		ABDTR	ABMODE [2:0]			Res.	Res.	ADSIZE [1:0]		ADDTR	ADMODE [2:0]			Res.	Res.	ISIZE [1:0]		IDTR	IMODE [2:0]		
	Reset value			0		0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0

Table 201. XSPI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x184	Reserved	Reserved																															
0x188	XSPI_WTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCYC[4:0]				
	Reset value																												0	0	0	0	0
0x18C	Reserved	Reserved																															
0x190	XSPI_WIR	INSTRUCTION[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x194-0x19C	Reserved	Reserved																															
0x1A0	XSPI_WABR	ALTERNATE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1A4-0x1FC	Reserved	Reserved																															
0x200	XSPI_HLCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRWR[7:0]								TACC[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	WZL	LM
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							0	0
0x204-0x20C	Reserved	Reserved																															
0x210	XSPI_CALFCR	CALMAX	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COARSE[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FINE[6:0]						
	Reset value	0											0	0	0	0	0										0	0	0	0	0	0	0
0x214	Reserved	Reserved																															
0x218	XSPI_CALMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COARSE[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FINE[6:0]						
	Reset value												0	0	0	0	0										0	0	0	0	0	0	0
0x22C	Reserved	Reserved																															
0x220	XSPI_CALSOR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COARSE[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FINE[6:0]						
	Reset value												0	0	0	0	0										0	0	0	0	0	0	0
0x224	Reserved	Reserved																															
0x228	XSPI_CALSIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COARSE[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FINE[6:0]						
	Reset value												0	0	0	0	0										0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

29 XSPI I/O manager (XSPIM)

29.1 XSPIM introduction

The XSPI I/O manager is a low-level interface, enabling an efficient XSPI pin assignment with a full I/O matrix (before alternate function map), and multiplex of single/dual/quad/octal/16-bit SPI interfaces over the same bus.

29.2 XSPIM main features

- Supports up to two single/dual/quad/octal/16-bit SPI interfaces
- Supports up to two ports for pin assignment
- Supports high-speed interfaces
- Manages up to three XSPIs

29.3 XSPIM implementation

Table 202 describes the XSPIM implementation.

Table 202. XSPIM implementation

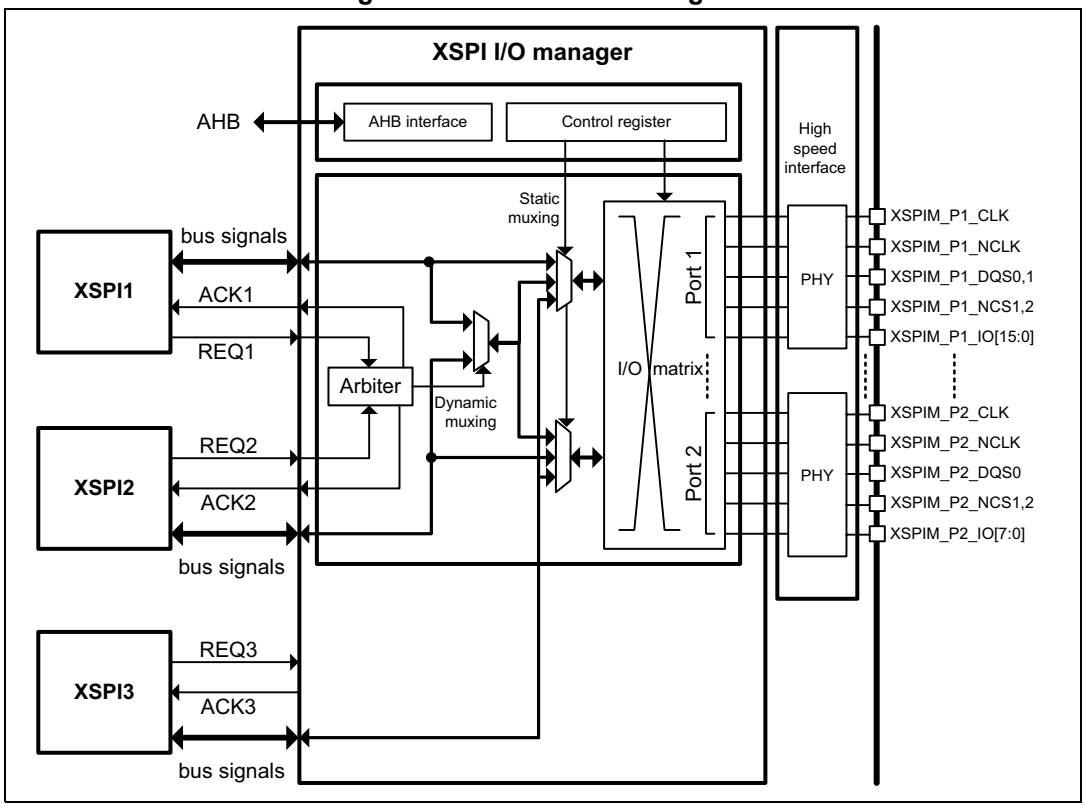
XSPI feature	Available on the devices
Supports up to two single/dual/quad interfaces	X
Fully I/O multiplexing capability	X
Supports time-multiplexed mode	X
Supports high-speed interface	X
Chip select selection if XSPI provides dual chip select	X
Supports 16-bit data interface and dual-octal mode	X
Supports up to three XSPIs	X

29.4 XSPIM functional description

29.4.1 XSPIM block diagram

The block diagram of the XSPI I/O manager is shown in *Figure 216*.

Figure 216. XSPIM block diagram



29.4.2 XSPIM input/output pins

Table 203. XSPIM input/output pins

Pin name(1)	Signal type	Description
XSPIM_Px_NCLK	Output	XSPI inverted clock to support 1.8 V HyperBus protocol. The inversion must be activated only in this case, setting bit INVCLK in the advanced I/O configuration section.
XSPIM_Px_CLK		XSPI clock
XSPIM_Px_IOn (n = 0 to 15)	Input/output	XSPI data pins
XSPIM_Px_NCS1,2	Output	Chip select for the memory
XSPIM_Px_DQS0,1	Input/output	Data strobe/write mask signals from/to the memory

1. x = 1 to 2.

29.4.3 XSPIM matrix

The XSPI I/O manager matrix allows the user to set a premapping of functions:

- XSPI1 mapped to Port 1, with XSPI2 mapped to Port 2 (direct mode)
- XSPI1 mapped to Port 2, with XSPI2 mapped to Port 1 (swapped mode)
- XSPI1 and XSPI2 both mapped to Port 1, with arbitration (multiplexed mode), and XSPI3 mapped to Port 2
- XSPI1 and XSPI2 both mapped to Port 2, with arbitration (multiplexed mode), and XSPI3 mapped to Port 1

Note: There is no possibility to use mixed combinations of signals (like NCS of XSPI1 with data of XSPI2).

When several I/O pins have the same configuration and are enabled at the same time, the result can be unpredictable.

In the default out-of-reset configuration, all the XSPI1 and XSPI2 signals are mapped, respectively, on Port 1 and on Port 2.

The configuration can be changed only when all XSPIs are disabled.

29.4.4 XSPIM multiplexed mode

When this mode is set, the XSPIs are time-multiplexed over the same bus. They get the ownership of the bus (in turn) through a request/acknowledge protocol with REQ/ACK signals.

The time-multiplexing is enabled by setting the MUXEN bit in the configuration register XSPIM_CR.

The fairness counter (MAXTRAN) of each XSPI can be used to manage the maximum duration for which a given XSPI takes the bus: this feature ensures the maximum bus access latency for the other XSPI(s). When one XSPI releases the bus, a round-robin arbitration phase occurs: when another XSPI requests the bus, it gets it.

When the multiplexed mode is enabled, either the fairness counter or the refresh timeout counter of both XSPI interfaces must be activated.

Each XSPI delivers two NCS signals (NCS1, NCS2). In multiplexed modes, the active NCS signal is the one of the XSPI owning the bus. Only XSPIn_IOs, XSPIn_DQS and XSPIn_CLK / XSPIn_NCLK are multiplexed.

When the multiplexed mode is used, only clock mode 0 is supported on the XSPIs.

Due to arbitration and bus sharing, the auto polling interval time of the XSPI, when used, can be increased.

Minimum switching duration

The minimum number of cycles needed to switch from an XSPI to another can be configured. This guarantees a latency between the falling edge of the REQ signal of the active XSPI (the active one releases the bus), and the rising edge of the ACK signal to the requesting XSPI (the bus is granted to the requesting one).

The duration is defined by the REQ2ACK_TIME field in the XSPIM_CR register.

29.5 Use cases description

The following table summarizes the use cases corresponding to IO manager modes.

Table 204. Use cases

IO manager mode	MUXEN	Mode	Reference
Direct (XSPI1 to Port1, XSPI2 to Port2, XSPI3 not used)	0	0	*Section 29.5.1* *Section 29.5.2*
Swapped (XSPI1 to Port2, XSPI2 to Port1, XSPI3 not used)	0	1	*Section 29.5.4*
Multiplexed to Port1 (XSPI1 and XSPI2 to Port1, XSPI3 to Port2)	1	0	*Section 29.5.5*
Multiplexed to Port2 (XSPI1 and XSPI2 to Port2, XSPI3 to Port1).	1	1	*Section 29.5.6*
XSPI1 and 2 drive a single external memory connected to Port1 or Port2, and XSPI3 drives one external memory connected to the other port.	1	0 or 1, depending upon the port connected to the memory.	*Section 29.5.7*
XSPI1 or 2 drives two external memories connected to Port1 or Port2, and XSPI3 is not used.	0	0 or 1, depending upon the port connected to the memories. The selected XSPI instance hits the corresponding memory thanks to the NCS selection.	*Section 29.5.8*

29.5.1 XSPIs direct octal mode

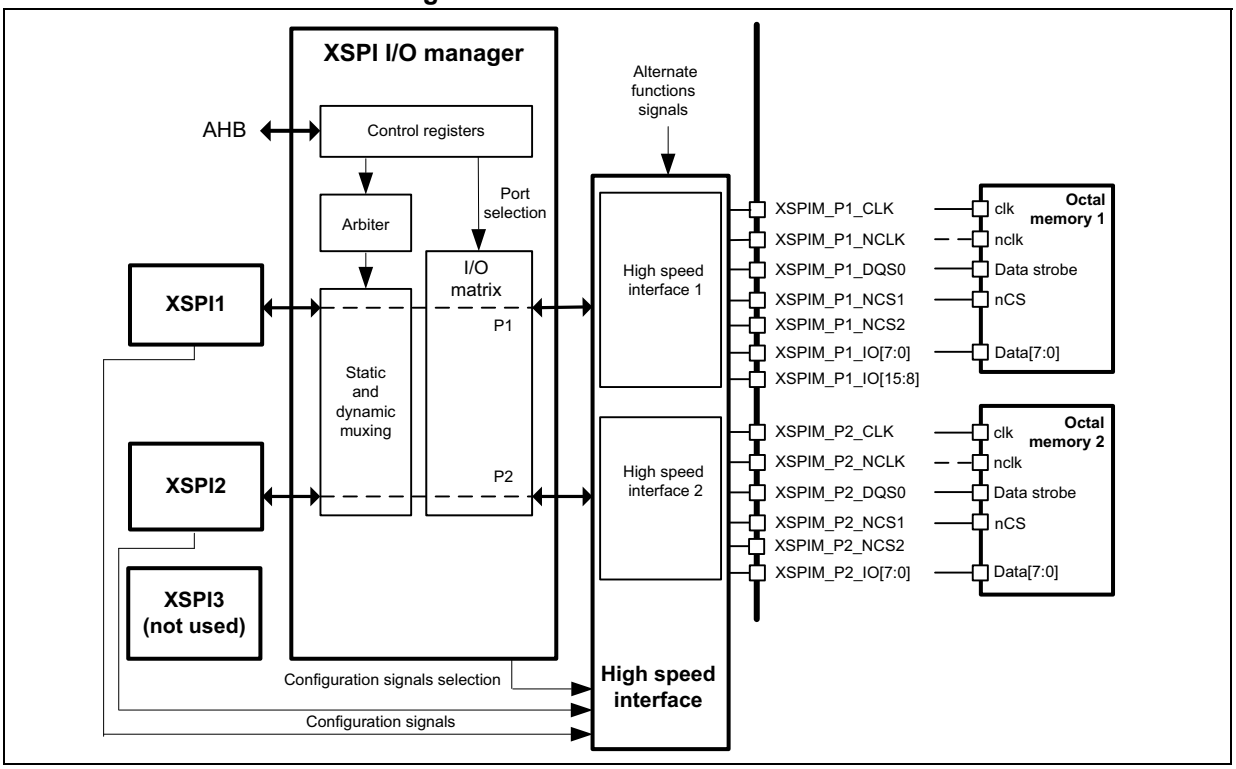
This is the baseline mode: each XSPI independently drives the corresponding port and high-speed interface. There is no multiplexing.

The configuration selection signal selects the signals used for each high speed interface. These signals come from the XSPI_CR and XSPI_CALxxx register outputs.

Alternate function signals can be routed on NCS2 pins (they are not used for the XSPI bus), nclk clocks are requested in case of 1.8 V HyperBus memory type.

This mode is selected by default through bits XSPIM_CR[1:0].

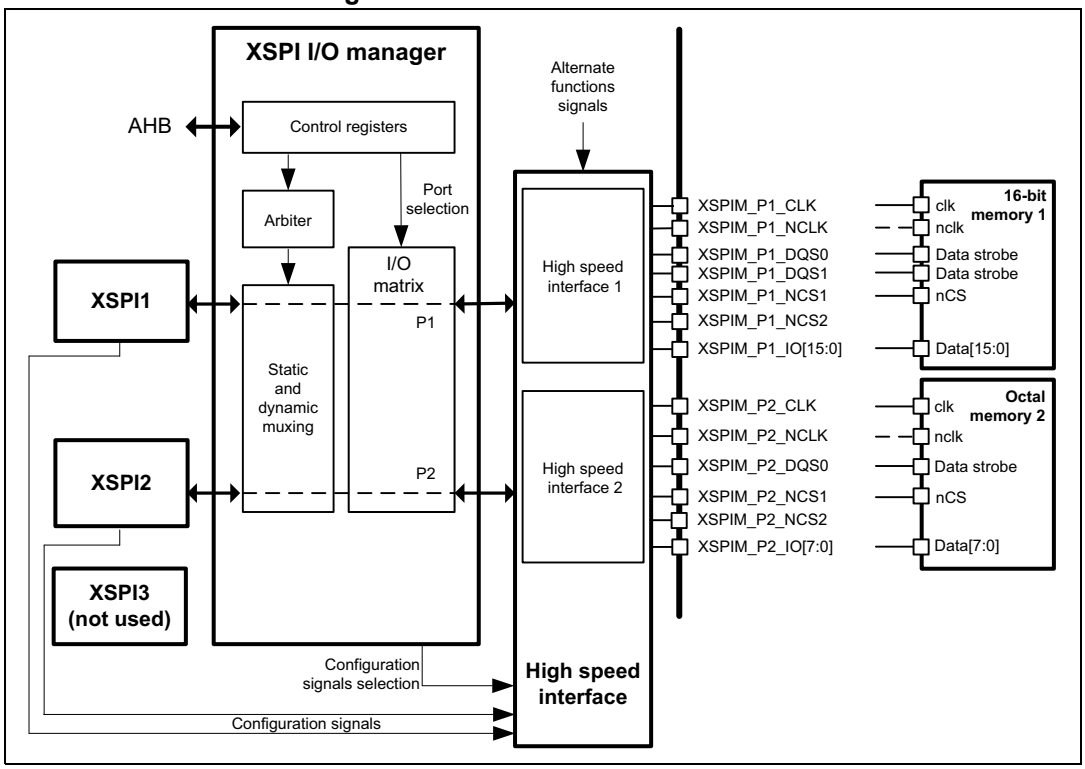
Figure 217. XSPI direct octal mode



29.5.2 XSPI direct 16-bit mode

This is the baseline mode, completely similar to XSPI octal direct mode, where each XSPI directly drives the corresponding port (XSPI1 connected to Port 1, XSPI2 connected to Port 2). Two DQS (1 per byte) are needed for each 16-bit port, when available.

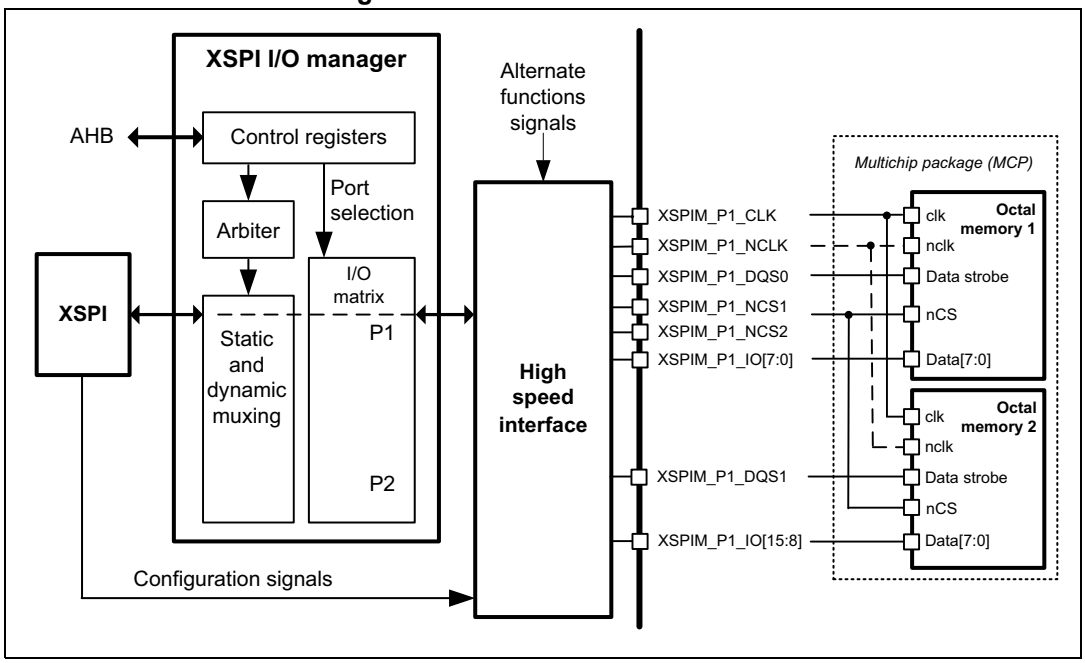
Figure 218. XSPI direct 16-bit mode



29.5.3 XSPI dual-octal mode

This mode is equivalent to XSPI direct mode, but, for simplicity, only one XSPI interface is represented. Two external octal memories are accessed in parallel, similarly to the dual-quad mode implemented with octal interfaces.

Figure 219. XSPI dual-octal mode



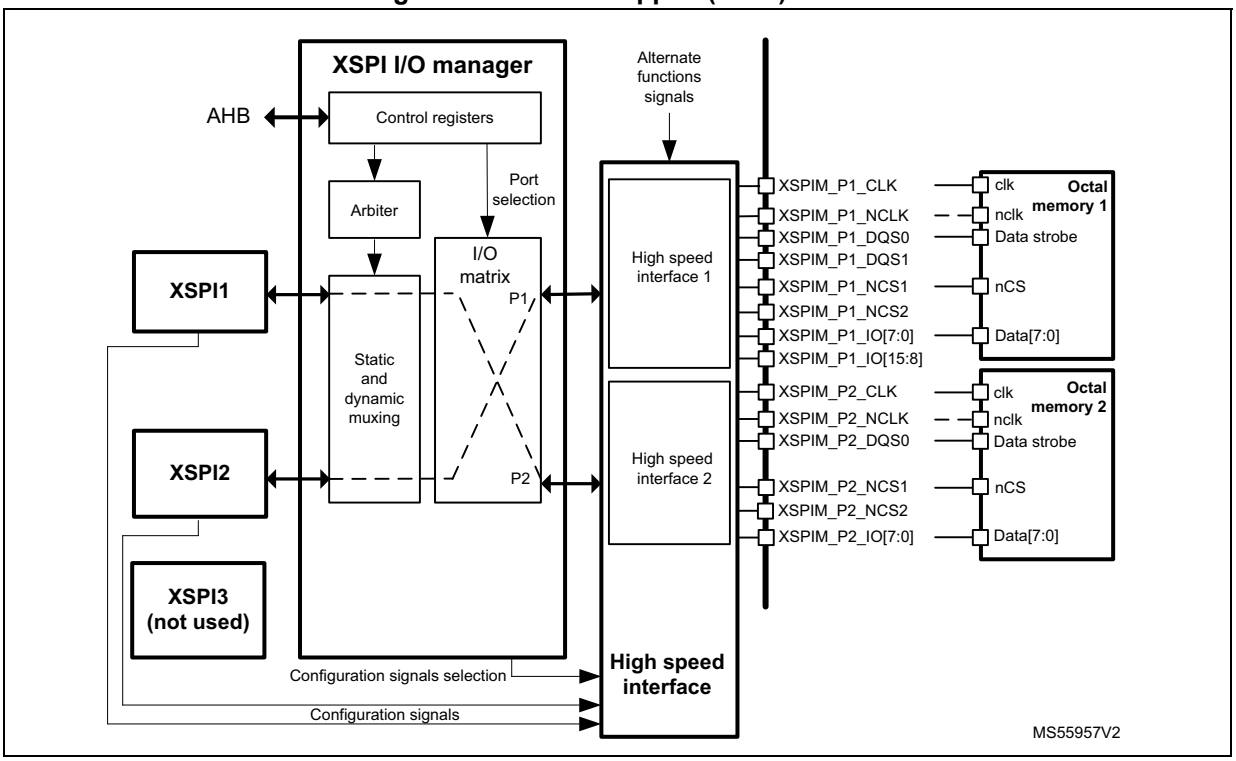
29.5.4 XSPI swapped mode

This mode is similar to XSPI direct mode, but the ports are swapped, to help I/O mapping. The mode is selected by bits XSPIM_CR[1:0].

If XSPIM.CR[0] or XSPIM.CR[1] is toggled in application, XSPI DCR2 or XSPI.CCR must be written, so that the calibration of the high speed interface is launched according to the new XSPI using it.

Note: In swapped mode, the XSPI2 can be configured in 16-bit mode, and the XSPI1 can be configured in octal mode, to connect an external 16-bit memory on Port 1, and to connect in a concurrent way an octal external memory connected to Port 2 of the I/O manager.

Figure 220. XSPI swapped (octal) mode



29.5.5 Two XSPIs multiplexed mode to Port 1 accessing two external memories, and the third XSPI accessing Port 2

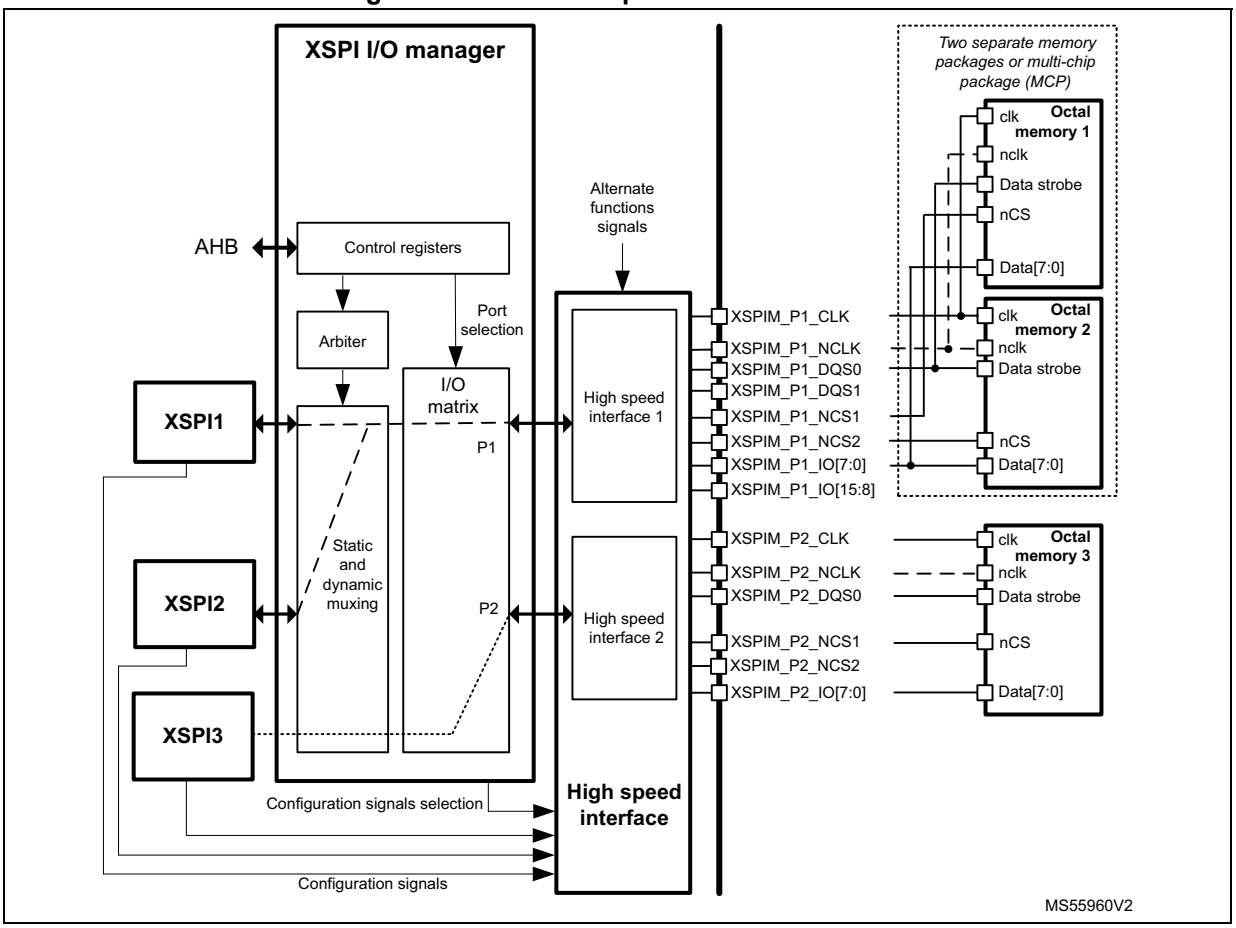
If only one output port is used to access two memories, alternate function signals can be routed to pins corresponding to the other port. This mode is selected by bits XSPIM_CR[1:0].

The arbiter in the IO manager selects XSPI1 or XSPI2 to own the octal-SPI bus according to the existing transfer requests and status of the two MAXTRAN fairness counters.

The external memories can be on separate chips, or embedded in a single multichip package. In this case, each memory requests a dedicated Chip select, selected according to the CSSEL control of the XSPI currently owning the bus.

The configuration signals of the high speed interface toggle when the corresponding XSPI is selected, because the timing characteristics of the two memories can be different.

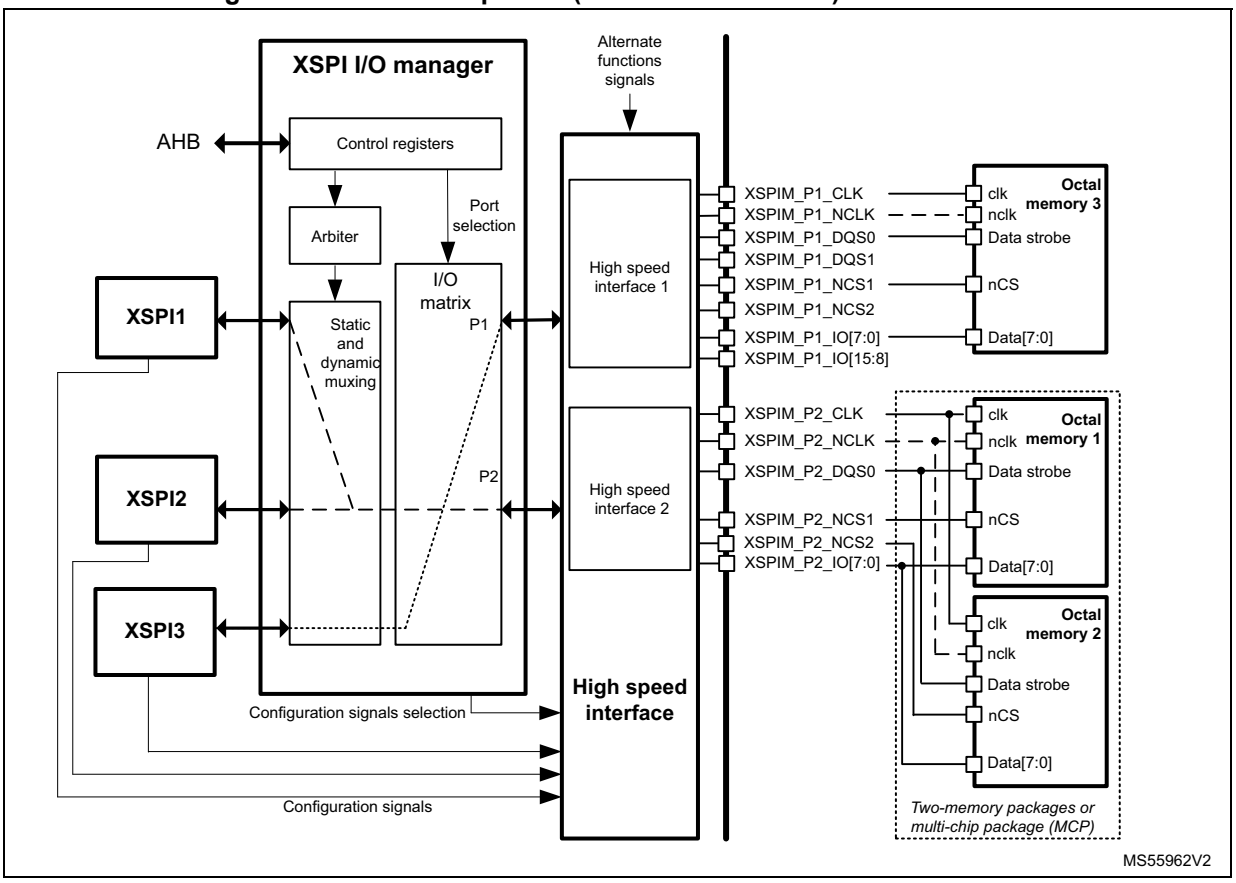
Figure 221. XSPI multiplexed mode to Port1



29.5.6 Two XSPIs multiplexed mode to Port 2 accessing two external memories, and the third XSPI accessing Port 1

This is similar to XSPI multiplexed mode to Port 1, but the target is Port 2. This mode is selected by bits XSPIM_CR[2:1].

Figure 222. XSPI multiplexed (octal and dual-octal) mode to Port 2

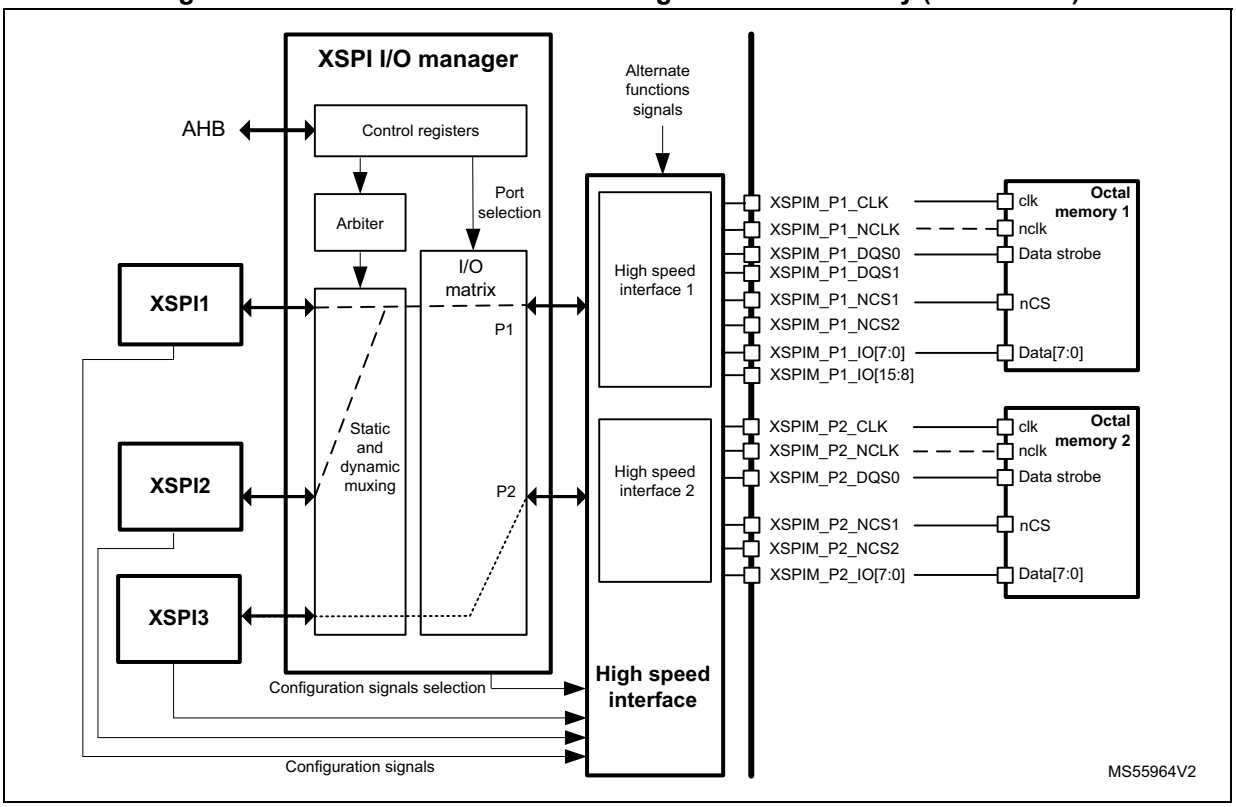


Note: It is possible to connect a 16-bit memory on Port 1, whereas Port 2 uses multiplexing mode on two octal memories.

29.5.7 XSPI1 and XSPI2 drive a single external memory

This is similar to XSPIs multiplexed mode to Port 1, but in this case only one external memory is present. Both CSSEL controls must target the same NCS (NCS1 in *Figure 223*).

Figure 223. XSPI1 and XSPI2 drive a single external memory (octal mode)



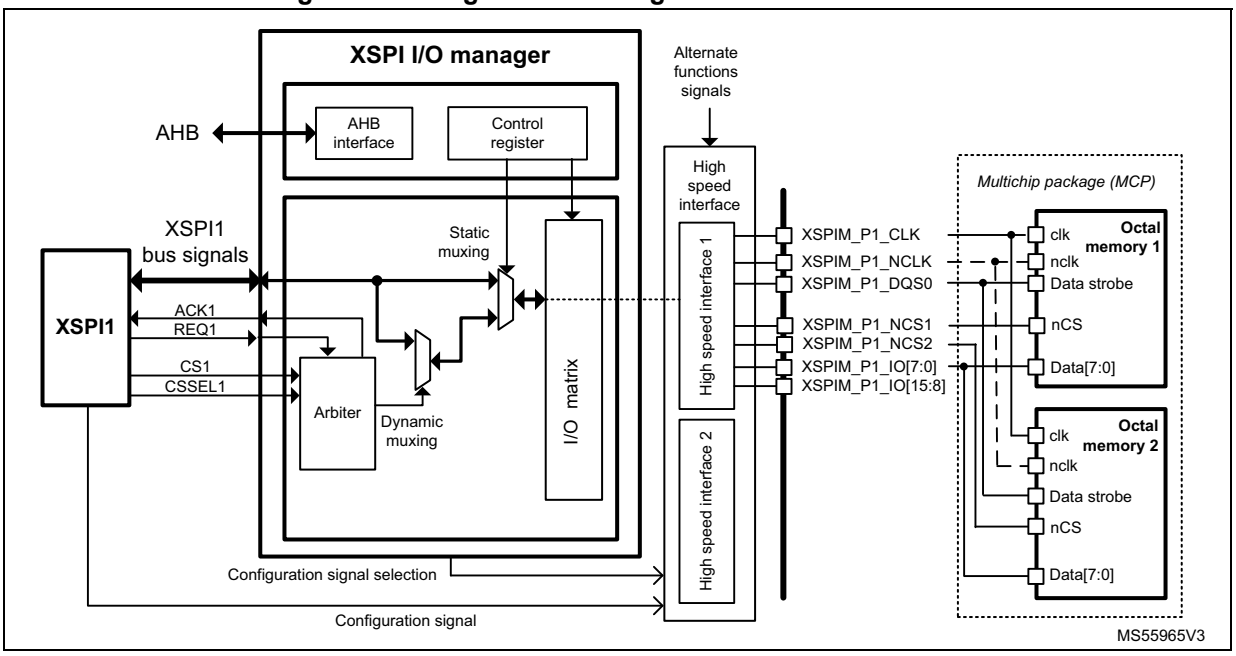
This use case is valid even when Port 2 is used instead of Port 1, and also possible with XSPI1 and XSPI2 interfacing a 16-bit memory (connecting XSPIM_P1_DQS1 to the memory).

29.5.8 A single XSPI drives two external memories

This is a subset of direct mode and corresponds to semi-dynamic switching. There are two target memories, but the arbiter is not used. The software must set the XSPI1 chip selector (bit 24 of the XSPI_CR register) each time the target memory changes.

The baseline use case boots on flash (memory1) and then switches to SRAM (memory2) for the rest of the application.

Figure 224. Single XSPI driving two external memories



This use case shows a configuration using XSPI1 on Port 1. It is also valid if the instance is XSPI2 instead of XSPI1, but with MODE = 1. The use case is also valid when Port 2 is used instead of Port 1, by configuring MODE bit according to the instance (XSPI1 or XSPI2) driving the two external memories.

29.6 XSPIM registers

29.6.1 XSPIM control register (XSPIM_CR)

Address offset: 0x0000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REQ2ACK_TIME[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSSEL_OVR_O2	CSSEL_OVR_O1	CSSEL_OVR_EN	Res.	Res.	MODE	MUXEN
									rw	rw	rw			rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **REQ2ACK_TIME[7:0]**: REQ to ACK time

In Multiplexed mode (MUXEN = 1), this field defines the time between two transactions. The value is the number of XSPI clock cycles - 1

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **CSSEL_OVR_O2**: Chip select selector override setting for XSPI2

0: XSPI2 can only use NCS1 (accesses using NCS2 are ignored)

1: XSPI2 can only use NCS2 (accesses using NCS1 are ignored)

Bit 5 **CSSEL_OVR_O1**: Chip select selector override setting for XSPI1

0: XSPI1 can only use NCS1 (accesses using NCS2 are ignored)

1: XSPI1 can only use NCS2 (accesses using NCS1 are ignored)

Bit 4 **CSSEL_OVR_EN**: Chip select selector override enable

0: CSSEL_OVR_O1 and CSSEL_OVR_O2 bit values are ignored, the chip select signals from OCTOSPIs or XSPIs are transmitted unconditionally

1: CSSEL_OVR_O1 and CSSEL_OVR_O2 bit values are taken into account

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **MODE**: XSPI multiplexing mode

0: if MUXEN = 0 direct mode, if MUXEN = 1 arbitration mode to output Port 1

1: if MUXEN = 0 swapped mode, if MUXEN = 1 arbitration mode to output Port 2

Bit 0 **MUXEN**: Multiplexed mode enable

This bit enables the multiplexing of the two XSPIs.

0: No multiplexing, hence no arbitration

1: XSPI1 and XSPI2 are multiplexed over the same bus

29.6.2 XSPIM register map

Table 205. XSPIM register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	**XSPIM _CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REQ2ACK_TIME[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSSEL_OVR_O2	CSSEL_OVR_O1	CSSEL_OVR_EN	Res.	Res.	MODE	MUXEN
	Reset value									0	0	0	0	0	0	0	0										0	0	0			0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

30 Secure digital input/output MultiMediaCard interface (SDMMC)

30.1 SDMMC main features

The SD/SDIO, embedded MultiMediaCard (*e*•MMC) host interface (SDMMC) provides an interface between the AHB bus and SD memory cards, SDIO cards and *e*•MMC devices.

The MultiMediaCard system specifications are available through the MultiMediaCard Association website at www.jedec.org, published by the MMCA technical committee.

SD memory card and SD I/O card system specifications are available through the SD card Association website at www.sdcard.org.

The SDMMC features include the following:

- Compliance with *Embedded MultiMediaCard System Specification Version 5.1*. Card support for three different databus modes: 1-bit (default), 4-bit and 8-bit. (HS200 SDMMC_CK speed limited to maximum allowed I/O speed)(HS400 is not supported).
- Full compatibility with previous versions of MultiMediaCards (backward compatibility).
- Full compliance with *SD memory card specifications version 6.0*. (SDR104 SDMMC_CK speed limited to maximum allowed I/O speed, SPI mode and UHS-II mode not supported).
- Full compliance with *SDIO card specification version 4.0.* Card support for two different databus modes: 1-bit (default) and 4-bit. (SDR104 SDMMC_CK speed limited to maximum allowed I/O speed, SPI mode and UHS-II mode not supported).
- Data transfer up to 208 Mbyte/s for the 8-bit mode. (depending maximum allowed I/O speed).
- Data and command output enable signals to control external bidirectional drivers.
- IDMA linked list support

The MultiMediaCard/SD bus connects cards to the host.

The current version of the SDMMC supports only one SD/SDIO/*e*•MMC card at any one time and a stack of *e*•MMC.

30.2 SDMMC implementation

Table 206. SDMMC features

SDMMC modes/features(1)	SDMMC1	SDMMC2
Variable delay	X	X
SDMMC_CKIN	X	-
SDMMC_CDIR, SDMMC_D0DIR	X	-
SDMMC_D123DIR	X	-

1. X = supported.

30.3 SDMMC bus topology

Communication over the bus is based on command/response and data transfers.

The basic transaction on the SD/SDIO/*e*•MMC bus is the command/response transaction. These types of bus transaction transfer their information directly within the command or response structure. In addition, some operations have a data token.

Data transfers are done in the following ways:

- Block mode: data block(s) with block size 2^N bytes with N in the range 0-14
- SDIO multibyte mode: single data block with block size range 1-512 bytes
- *e*•MMC Stream mode: continuous data stream

Data transfers to/from *e*•MMC cards are done in data blocks or streams.

Figure 225. SDMMC "no response" and "no data" operations

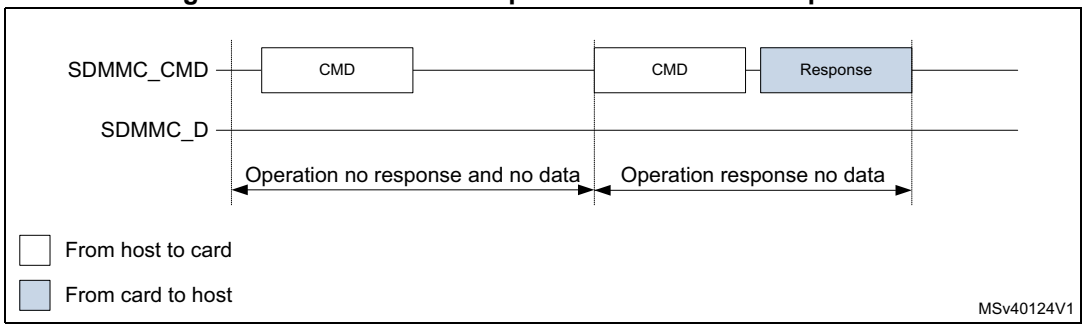
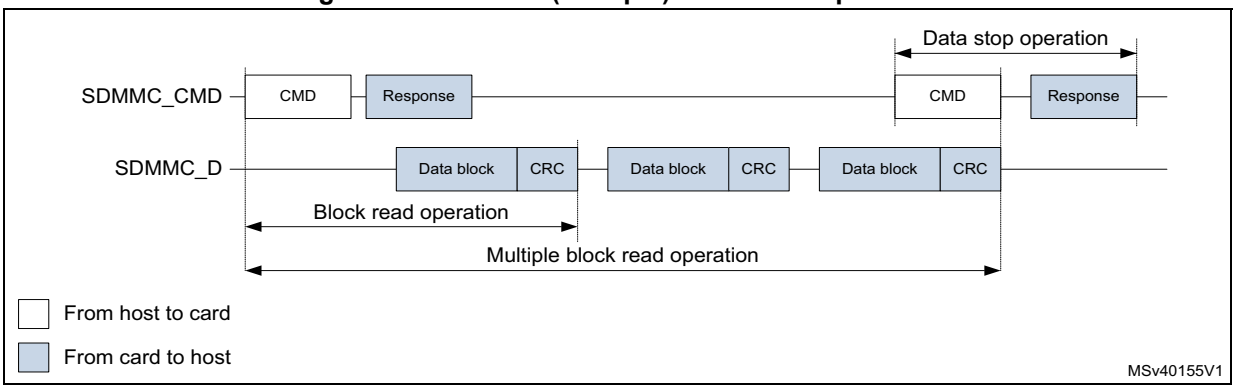
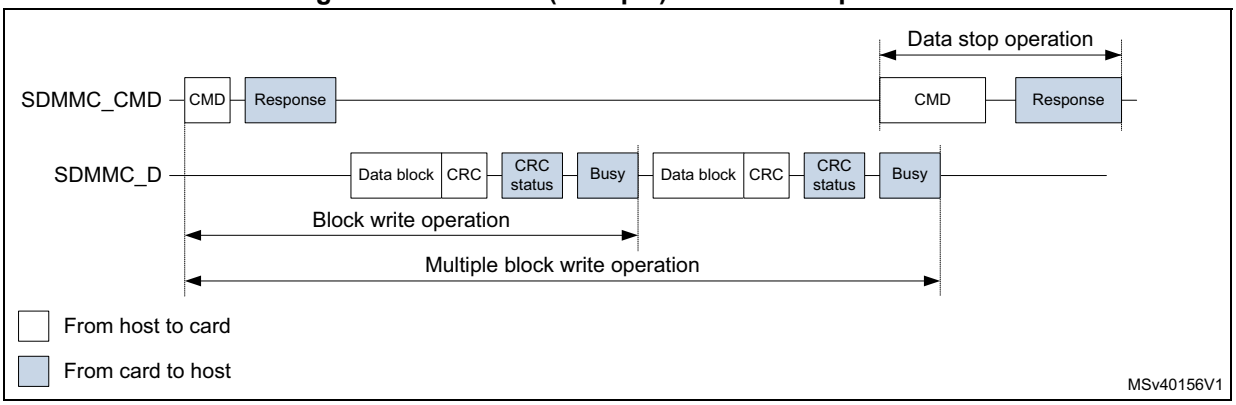


Figure 226. SDMMC (multiple) block read operation



Note: The Stop Transmission command is not required at the end of a e•MMC multiple block read with predefined block count.

Figure 227. SDMMC (multiple) block write operation



Note: The Stop Transmission command is not required at the end of an e•MMC multiple block write with predefined block count.

The SDMMC does not send any data as long as the Busy signal is asserted (SDMMC_D0 pulled low).

Figure 228. SDMMC (sequential) stream read operation

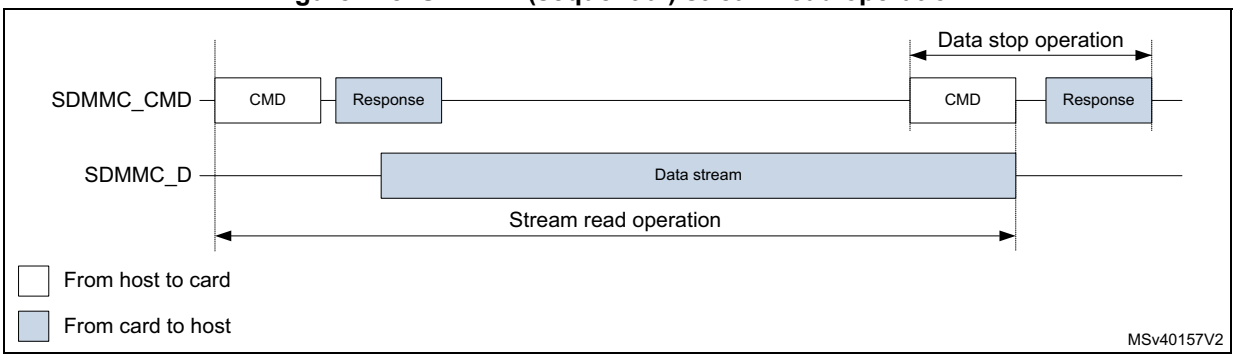
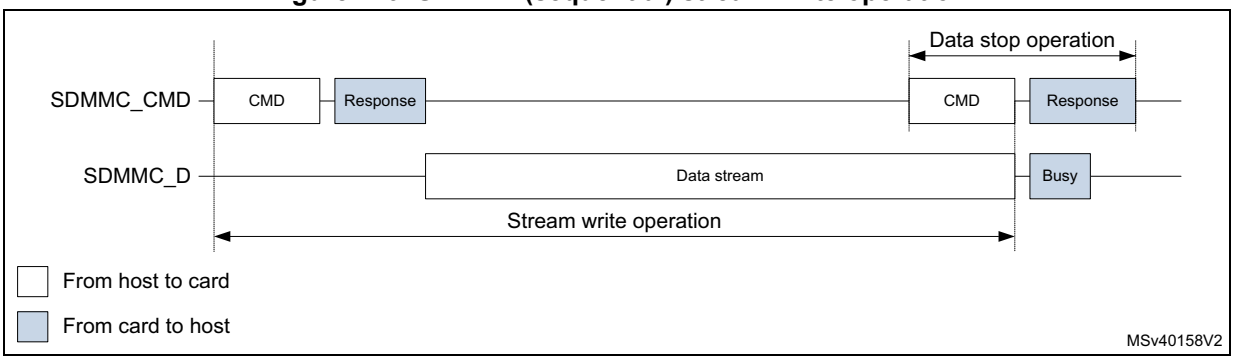


Figure 229. SDMMC (sequential) stream write operation



Stream data transfer operates only in a 1-bit wide bit bus configuration on SDMMC_D0 in single data rate modes (DS, HS, and SDR).

30.4 SDMMC operation modes

Table 207. SDMMC operation modes SD and SDIO

SDIO bus speed modes(1)(2)	Max bus speed (3) [Mbyte/s]	Max clock frequency [MHz](4)	Signal voltage [V]
DS (default speed)	12.5	25	3.3
HS (high speed)	25	50	3.3
SDR12	12.5	25	1.8
SDR25	25	50	1.8
DDR50	50	50	1.8
SDR50	50	100	1.8
SDR104	104	208	1.8

1. SDR single data rate signaling.
2. DDR double data rate signaling (data is sampled on both SDMMC_CK clock edges).
3. SDIO bus speed with 4-bit bus width.
4. Maximum frequency depending on maximum allowed I/O speed.

SDR104 mode requires variable delay support using sampling point tuning. The use of variable delay is optional for SDR50 mode.

Table 208. SDMMC operation modes *e*•MMC

e•MMC bus speed modes (1)(2)	Max bus speed (3) [Mbyte/s]	Max clock frequency [MHz](4)	Signal voltage [V](5)
Legacy compatible	26	26	3/1.8/1.2V
High speed SDR	52	52	3/1.8/1.2V
High speed DDR	104	52	3/1.8/1.2V
High speed HS200	200	200	1.8/1.2V

1. SDR single data rate signaling.
2. DDR double data rate signaling. (data is sampled on both SDMMC_CK clock edges).
3. *e*•MMC bus speed with 8-bit bus width.
4. Maximum frequency depending on maximum allowed I/O speed.
5. Supported signal voltage level depends on I/O port characteristics, refer to device datasheet.

HS200 mode requires variable delay support using sampling point tuning.

30.5 SDMMC functional description

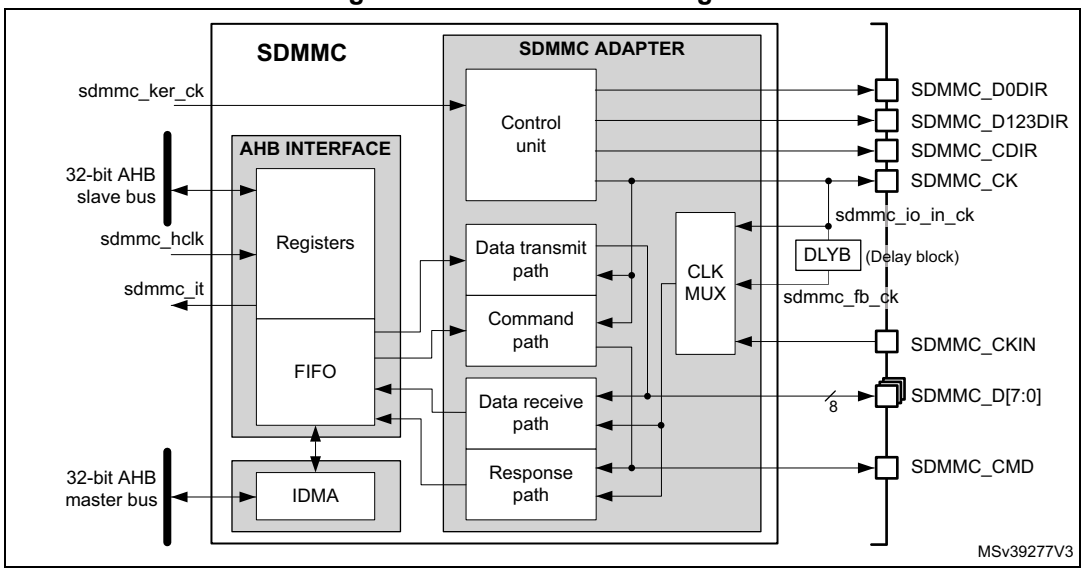
The SDMMC consists of four parts:

- The AHB slave interface accesses the SDMMC adapter registers, and generates interrupt signals and IDMA control signals.
- The SDMMC adapter block provides all functions specific to the *e*•MMC/SD/SD I/O card such as the clock generation unit, command and data transfer.
- The internal DMA (IDMA) block with its AHB master interface.
- A delay block (DLYB) taking care of the receive data sample clock alignment. The delay block is NOT part of the SDMMC. A delay block is mandatory when supporting SDR104 or HS200.

30.5.1 SDMMC block diagram

Figure 230 shows the SDMMC block diagram.

Figure 230. SDMMC block diagram



30.5.2 SDMMC pins and internal signals

Table 209 lists the SDMMC internal input/output signals, *Table 210* the SDMMC pins (alternate functions).

Table 209. SDMMC internal input/output signals

Signal name	Signal type	Description
sdmmc_ker_ck	Digital input	SDMMC kernel clock
sdmmc_hclk	Digital input	AHB clock
sdmmc_it	Digital output	SDMMC global interrupt

Table 209. SDMMC internal input/output signals (continued)

Signal name	Signal type	Description
sdmmc_io_in_ck	Digital input	SD/SDIO/*e*•MMC card feedback clock. This signal is internally connected to the SDMMC_CK pin (for DS and HS modes).
sdmmc_fb_ck	Digital input	SD/SDIO/*e*•MMC card tuned feedback clock after DLYB delay block (for SDR50, DDR50, SDR104, HS200)

Table 210. SDMMC pins

Pin name	Pin type	Description
SDMMC_CK	Digital output	Clock to SD/SDIO/*e*•MMC card
SDMMC_CKIN	Digital input	Clock feedback from an external driver for SD/SDIO/*e*•MMC card. (for SDR12, SDR25, SDR50, DDR50)
SDMMC_CMD	Digital input/output	SD/SDIO/*e*•MMC card bidirectional command/response signal.
SDMMC_CDIR	Digital output	SD/SDIO/*e*•MMC card I/O direction indication for the SDMMC_CMD signal.
SDMMC_D[7:0]	Digital input/output	SD/SDIO/*e*•MMC card bidirectional data lines.
SDMMC_D0DIR	Digital output	SD/SDIO/*e*•MMC card I/O direction indication for the SDMMC_D0 data line.
SDMMC_D123DIR	Digital output	SD/SDIO/*e*•MMC card I/O direction indication for the data lines SDMMC_D[3:1].

30.5.3 General description

The **SDMMC_D[7:0]** lines have different operating modes:

- By default, SDMMC_D0 line is used for data transfer. After initialization, the host can change the databus width.
- For an *e*•MMC, 1-bit (SDMMC_D0), 4-bit (SDMMC_D[3:0]) or 8-bit (SDMMC_D[7:0]) data bus widths can be used.
- For an SD or an SDIO card, 1-bit (SDMMC_D0) or 4-bit (SDMMC_D[3:0]) can be used. All data lines operate in push-pull mode.

To allow the connection of an external driver (a voltage switch transceiver), the direction of data flow on the data lines is indicated with I/O direction signals. The **SDMMC_D0DIR** signal indicates the I/O direction for the SDMMC_D0 data line, the **SDMMC_D123DIR** for the SDMMC_D[3:1] data lines.

SDMMC_CMD only operates in push-pull mode:

To allow the connection of an external driver (a voltage switch transceiver), the direction of data flow on the SDMMC_CMD line is indicated with the I/O direction signal **SDMMC_CDIR**.

SDMMC_CK clock to the card originates from **sdmmc_ker_ck**:

- When the sdmmc_ker_ck clock has 50 % duty cycle, it can be used even in bypass mode (CLKDIV = 0).
- When the sdmmc_ker_ck duty cycle is not 50 %, the CLKDIV must be used to divide it by 2 or more (CLKDIV > 0).
- The phase relation between the SDMMC_CMD / SDMMC_D[7:0] outputs and the SDMMC_CK can be selected through the NEGEDGE bit. The phase relation depends on the CLKDIV, NEGEDGE, and DDR settings. See *Figure 231*.

Figure 231. SDMMC Command and data phase relation

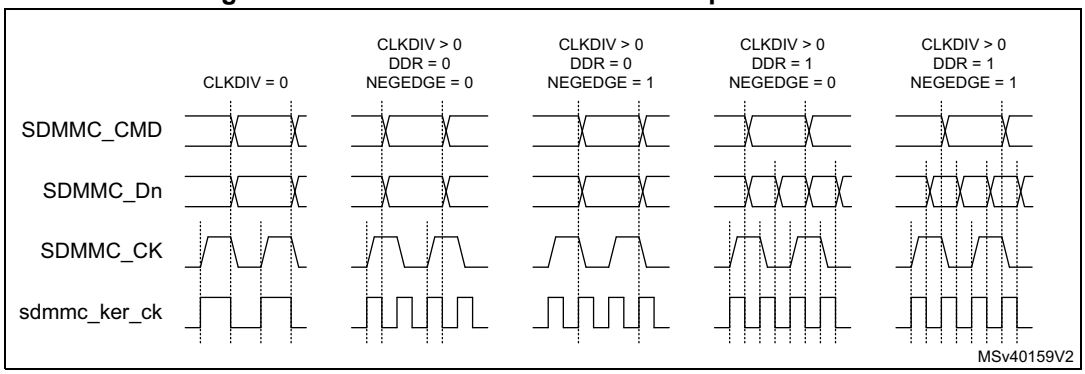


Table 211. SDMMC Command and data phase selection

CLKDIV	DDR	NEGEDGE	SDMMC_CK	Command out	Data out
0	x	x	= sdmmc_ker_ck	Generated on sdmmc_ker_ck falling edge	
>0	0	0	Generated on sdmmc_ker_ck rising edge	Generated on sdmmc_ker_ck falling edge succeeding the SDMMC_CK rising edge.	
		1		Generated on the same sdmmc_ker_ck rising edge that generates the SDMMC_CK falling edge.	
	1	0		Generated on sdmmc_ker_ck falling edge succeeding the SDMMC_CK rising edge.	Generated on sdmmc_ker_ck falling edge succeeding a SDMMC_CK edge.
		1		Generated on the same sdmmc_ker_ck rising edge that generates the SDMMC_CK falling edge.	

By default, the **sdmmc_io_in_ck** feedback clock input is selected for sampling incoming data in the SDMMC receive path. It is derived from the SDMMC_CK pin.

For tuning the phase of the sampling clock to accommodate the receive data timing, the DLYB delay block available on the device can be connected between **sdmmc_io_in_ck** signal (DLYB input dlyb_in_ck) and **sdmmc_fb_ck** clock input of SDMMC (DLYB output dlyb_out_ck). Selecting the **sdmmc_fb_ck** clock input in the receive path then enables using the phase-tuned sampling clock for the incoming data. This is required for SDMMC to support the SDR104 and HS200 operating mode and optional for SDR50 and DDR50 modes.

When using an external driver (a voltage switch transceiver), the SDMMC_CKIN feedback clock input can be selected to sample the receive data.

For an SD/SDIO/*e*•MMC card, the clock frequency can vary between 0 and 208 MHz (limited by maximum I/O speed).

Depending on the selected bus mode (SDR or DDR), one bit or two bits are transferred on SDMMC_D[7:0] lines with each clock cycle. The SDMMC_CMD line transfers only one bit per clock cycle.

30.5.4 SDMMC adapter

The SDMMC adapter (see *Figure 230: SDMMC block diagram*) is a multimedia/secure digital memory card bus master that provides an interface to a MultiMediaCard stack or to a secure digital memory card. It consists of the following subunits:

- Control unit
- Data transmit path
- Command path
- Data receive path
- Response path
- Receive data path clock multiplexer
- Delay block (DLYB), external to the SDMMC
- Adapter register block
- Data FIFO
- Internal DMA (IDMA)

Note: The adapter registers and FIFO use the AHB clock domain (sdmmc_hclk). The control unit, command path and data transmit path use the SDMMC adapter clock domain (sdmmc_ker_ck). The response path and data receive path use the SDMMC adapter feedback clock domain from the sdmmc_io_in_ck, or SDMMC_CKIN, or from the sdmmc_fb_ck generated by DLYB.

The DLYB delay block on the device can be used in conjunction with the SDMMC adapter, to tune the phase of the sampling clock for incoming data in SDMMC receive mode. It is required for the SDMMC to support the SDR104 and HS200 operating mode and optional for SDR50 and DDR50 modes.

Adapter register block

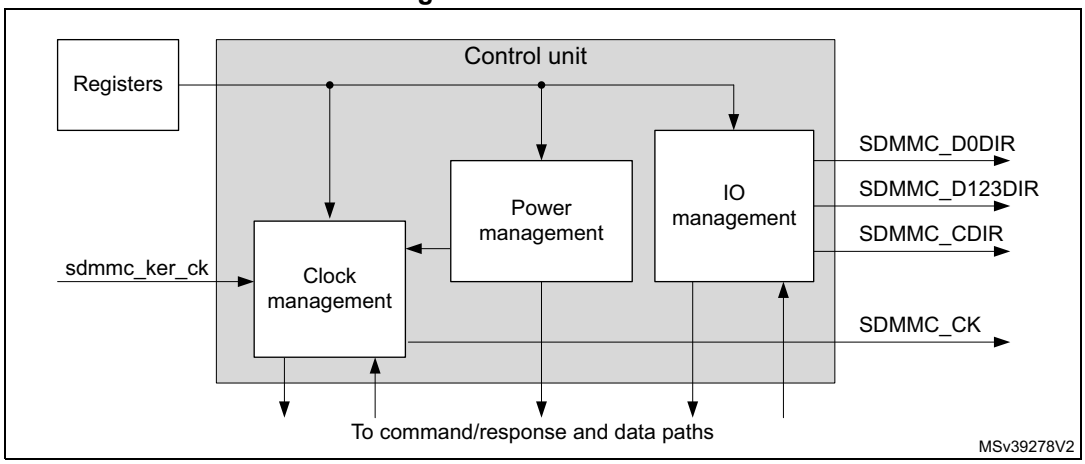
The adapter register block contains all system control registers, the SDMMC command and response registers and the data FIFO.

This block also generates the signals from the corresponding bit location in the SDMMC Clear register that clear the static flags in the SDMMC adapter.

Control unit

The control unit illustrated in *Figure 232*, contains the power management functions, the SDMMC_CK clock management with divider, and the I/O direction management.

Figure 232. Control unit



The power management subunit disables the card bus output signals during the power-off and power-up phases.

There are three power phases:

- power-off
- power-up
- power-on

The clock management subunit uses the sdmmc_ker_ck to generate the SDMMC_CK and provides the division control. It also takes care of stopping the SDMMC_CK for flow control, for example.

The clock outputs are inactive:

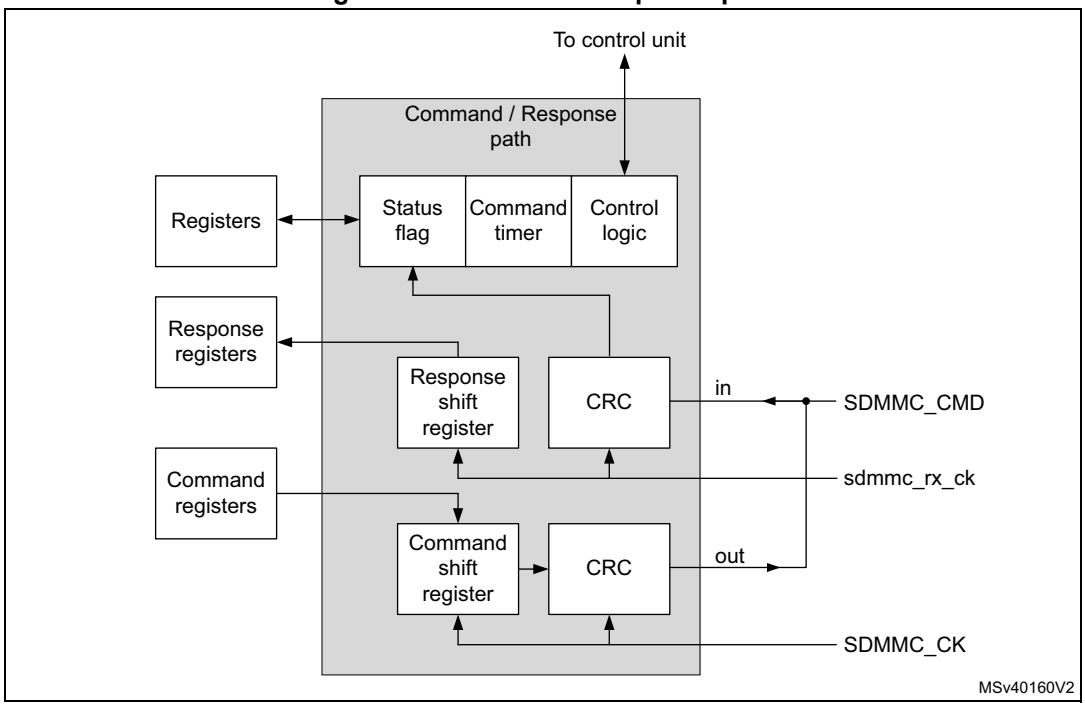
- after reset
- during the power-off or power-up phases
- if the power saving mode (register bit PWRSAV) is enabled and the card bus is in the Idle state for eight clock periods. The clock is stopped eight cycles after both the command/response CPSM and data path DPSM subunits have enter the Idle phase. The clock is restarted when the command/response CPSM or data path DPSM is activated (enabled).

The I/O management subunit takes care of the SDMMC_Dn and SDMMC_CMD I/O direction signals, which controls the external voltage transceiver.

Command/response path

The command/response path subunit transfers commands and responses on the SDMMC_CMD line. The command path is clocked on the SDMMC_CK and sends commands to the card,. The response path is clocked on the sdmmc_rx_ck and receives responses from the card.

Figure 233. Command/response path

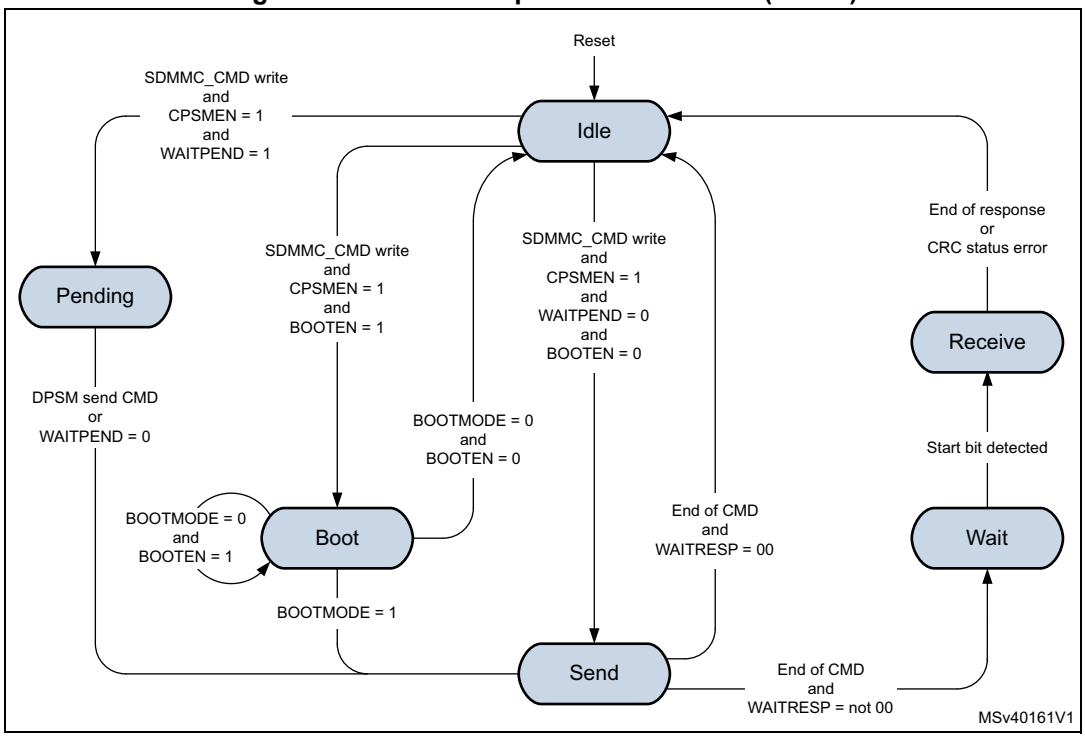


Command/response path state machine (CPSM):

- When the command register is written to and the enable bit is set, command transfer starts. When the command has been sent the CRC is appended and the command path state machine (CPSM) sets the status flags and:
  - if a response is not required enters the Idle state.
  - If a response is required, it waits for the response.
- When the response is received,
  - for a response with CRC, the received CRC code and the internally generated code are compared, and the appropriate status flag is set according the result.
  - for a response without CRC, no CRC is checked, and the appropriate status flag is not set.

When ever the CPSM is active (not in the Idle state), the CPSMACT bit is set.

Figure 234. Command path state machine (CPSM)



- **Idle**: The command path is inactive. When the command control register is written and the enable bit (CPSMEN) is set, the CPSM activates the SDMMC_CK clock (when stopped due to power save PWRSAV bit) and moves
  - to the Send state when WAITPEND = 0 and BOOTEN = 0.
  - to the Pending state when WAITPEND = 1.
  - to the Boot state when BOOTEN = 1.
- **Send**: The command is sent and the CRC is appended.
  - When CMDTRANS bit is set or when BOOTEN bit is set and BOOTMODE is alternative boot, and the DTDIR = receive, the CPSM DataEnable signal is issued to the DPSM at the end of the command.
  - When the CMDTRANS bit is set and the CMDSUSPEND bit is 0 the interrupt period is terminated at the end of the command.
  - When CMDSTOP bit is set the CPSM Abort signal is issued to the DPSM at the end of the command.
  - If no response is expected (WAITRESP = 00) the CPSM moves to the Idle state and the CMDSENT flag is set. When BOOTMODE = 1 and BOOTEN = 0 the CMDSENT flag is delayed 56 cycles after the command end bit, otherwise the

CMDSENT flag is generated immediately after the command end bit.
The RESPCMDR and RESPxR registers are not modified.

– If a command response is expected (WAITRESP = not 00) the CPSM moves to the Wait state and start the response timeout.

- **Wait**: The command path waits for a response.
  – When WAITINT bit is 0 the command timer starts running and the CPSM waits for a start bit.
  a) If a start bit is detected before the timeout the CPSM moves to the Receive state.
  b) If the timeout is reached before the CPSM detect a response start bit, the timeout flag (CTIMEOUT) is set and the CPSM moves to the Idle state.
  The RESPCMDR and RESPxR registers are not modified.
  – When WAITINT bit is 1, the timer is disabled and the CPSM waits for an interrupt request (response start bit) from one of the cards.
  a) When a start bit is detected the CPSM moves to the Receive state.
  b) When writing WAITINT to 0 (interrupt mode abort), the host sends a response by its self and on detecting the start bit the CPSM move to the Receive state.
- **Receive**: The command response is received. Depending the response mode bits WAITRESP in the command control register, the response can be either short or long, with CRC or without CRC. The received CRC code when present is verified against the internally generated CRC code.
  – When the CMDSUSPEND bit is set and the SDIO Response bit BS = 0 (response bit [39]), the interrupt period is started after the response.
  When the CMDSUSPEND bit is cleared, or the CMDSUSPEND bit is 1 and the SDIO Response bit BS = 1 (response bit [39]), there is no interrupt period started.
  – When the CMDTRANS bit is set and the CMDSUSPEND bit is set and the SDIO Response bit DF= 1 (response bit [32]) the interrupt period is terminated after the response.
  – When the CRC status passes or no CRC is present the CMDREND flag is set, the CPSM moves to the Idle state.
  The RESPCMDR and RESPxR registers are updated with received response.
  - When BOOTMODE = 1 and BOOTEN = 0 the CMDREND flag is delayed 56 cycles after the response end bit, otherwise the CMDREND flag is generated immediately after the response end bit.
  - When CMDTRANS bit is set and the DTDIR = transmit, the CPSM DataEnable signal is issued to the DPSM at the end of the command response.
  – When the CRC status fails the CCRCFAIL flag is set and the CPSM moves to the Idle state.
  The RESPCMDR and RESPxR registers are updated with received response.
- **Pending**: According the pending WAITPEND bit in the command register, the CPSM enters the pending state.
  – When DATALENGTH ≤ 5 bytes the CPSM moves to the Sent state and generates the DataEnable signal to start the data transfer aligned with the CMD12 Stop Transmission command.
  – When DATALENGTH > 5 bytes, the CPSM DataEnable signal is issued to the DPSM to start the data transfer. The CPSM waits for a send CMD signal from the

DPSM before moving to the Send state. This enables, for example, the CMD12 Stop Transmission command to be sent aligned with the data.

– When writing WAITPEND to 0, the CPSM moves to the Send state.

- **Boot**: If the BOOTEN bit is set in the command register, the CPSM enters the Boot state, and when:
  – BOOTMODE = 0 the SDMMC_CMD line is driven low and when CMDTRANS bit is set and the DTDIR = receive, the CPSM DataEnable signal is issued to the DPSM. This enables normal boot operation. This state is left at the end of the boot procedure by clearing the register bit BOOTEN, which cause the SDMMC_CMD line to be driven high and the CPSM Abort signal is issued to the DPSM, before moving to the Idle state. The CMDSENT flag is generated 56 cycles after SDMMC_CMD line is high.
  – BOOTMODE = 1, move to the Send state. This enables sending of the CMD0 (boot). Clearing BOOTEN has no effect.

Note: The CPSM remains in the Idle state for at least eight SDMMC_CK periods to meet the N_{CC} and N_{RC} timing constraints. N_{CC} is the minimum delay between two host commands, and N_{RC} is the minimum delay between the host command and the card response.

The response timeout has a fixed value of 64 SDMMC_CK clock periods.

A command is a token that starts an operation. Commands are sent from the host to either a single card (addressed command) or all connected cards (broadcast command are available for *e*•MMC V3.31 or previous). Commands are transferred serially on the SDMMC_CMD line. All commands have a fixed length of 48 bits. The general format for a command token for SD-Memory cards, SDIO cards, and *e*•MMC cards is shown in *Table 212.*

The command token data is taken from two registers, one containing a 32-bit argument and the other containing the 6-bit command index (six bits sent to a card).

Table 212. Command token format

Bit position	Width	Value	Description
47	1	0	Start bit
46	1	1	Transmission bit
[45:40]	6	x	Command index
[39:8]	32	x	Argument
[7:1]	7	x	CRC7
0	1	1	End bit

Next to the command data there are command type (WAITRESP) bits controlling the command path state machine (CPSM). These bits also determine whether the command requires a response, and whether the response is short (48 bit) or long (136 bits) long, and if a CRC is present or not.

A response is a token that is sent from an addressed card or synchronously from all connected cards to the host as an answer to a previous received command. All responses are sent via the command line SDMMC_CMD. The response transmission always starts with the left bit of the bit string corresponding to the response code word. The code length depends on the response type. Response tokens R1, R2, R3, R4, R5, and R6 have various

coding schemes, depending on their content. The general formats for the response tokens for SD-Memory cards, SDIO cards, and *e*•MMC cards are shown in *Table 213*, *Table 214* and *Table 215*.

A response always starts with a start bit (always 0), followed by the bit indicating the direction of transmission (card = 0). A value denoted by x in the tables below indicates a variable entry. Most responses, except some, are protected by a CRC. Every command code word is terminated by the end bit (always 1).

The response token data is stored in five registers, four containing the 32-bits card status, OCR register, argument or 127-bits CID or CSD register including internal CRC, and one register containing the 6-bits command index.

Table 213. Short response with CRC token format

Bit position	Width	Value	Description
47	1	0	Start bit
46	1	0	Transmission bit
[45:40]	6	x	Command index (or reserved 111111)
[39:8]	32	x	Argument
[7:1]	7	x	CRC7
0	1	1	End bit

Table 214. Short response without CRC token format

Bit position	Width	Value	Description
47	1	0	Start bit
46	1	0	Transmission bit
[45:40]	6	x	Command index (or reserved 111111)
[39:8]	32	x	Argument
[7:1]	7	1111111	(reserved 1111111)
0	1	1	End bit

Table 215. Long response with CRC token format

Bit position	Width	Value	Description
135	1	0	Start bit
134	1	0	Transmission bit
[133:128]	6	111111	Reserved
[127:1]	127:8	x	CID or CSD slices
	7:1	x	CRC7 (included in CID or CSD)
0	1	1	End bit

The command/response path operates in a half-duplex mode, so that either commands can be sent or responses can be received. If the CPSM is not in the Send state, the

SDMMC_CMD output is in the Hi-Z state. Data sent on SDMMC_CMD are synchronous with the SDMMC_CK according the NEGEDGE register bit see *Figure 231*.

The command and short response with CRC, the CRC generator calculates the CRC checksum for all 40 bits before the CRC code. This includes the start bit, transmission bit, command index, and command argument (or card status).

For the long response the CRC checksum is calculated only over the 120 bits of R2 CID or CSD. Note that the start bit, transmission bit and the six reserved bits are not used in the CRC calculation.

The CRC checksum is a 7-bit value:

CRC[6:0] = remainder [(M(x) * x^7) / G(x)]

G(x) = $x^7 + x^3 + 1$

M(x) = (first bit) * x^n + (second bit) * x^{n-1}+... + (last bit before CRC) * x^0

Where n = 39 or 119.

The CPSM can send a number of specific commands to handle various operating modes when CPSMEN is set, see *Table 216*.

Table 216. Specific Commands overview

VSWITCH	BOOTEN	BOOTMOD	CMDTRAN	WAITPEND	CMDSTOP	WAITINT	Description
1	x	x	x	x	x	x	Start voltage switch sequence
0	1	x	x	x	x	x	Start normal boot
0	1	1	x	x	x	x	Start alternative boot
0	0	1	x	x	x	x	Stop alternative boot.
0	0	0	1	x	x	x	Send command with associated data transfer.
0	0	0	0	1	1	x	*e*•MMC stream data transfer, command (STOP_TRANSMISSION) pending until end of data transfer.
0	0	0	0	1	0	x	*e*•MMC stream data transfer, command different from (STOP_TRANSMISSION) pending until end of data transfer.
0	0	0	0	0	1	x	Send command (STOP_TRANSMISSION), stopping any ongoing data transmission.
0	0	0	0	0	0	1	Enter *e*•MMC wait interrupt (Wait-IRQ) mode.
0	0	0	0	0	0	0	Any other none specific command

The command/response path implements the status flags and associated clear bits shown in *Table 217*:

Table 217. Command path status flags

Flag	Description
CMDSENT	Set at the end of the command without response (CPSM moves from Send to Idle).
CMDREND	Set at the end of the command response when the CRC is OK (CPSM moves from Receive to Idle).
CCRCFAIL	Set at the end of the command response when the CRC is FAIL (CPSM moves from Receive to Idle).
CTIMEOUT	Set after the command when no response start bit received before the timeout (CPSM moves from Wait to Idle).
CKSTOP	Set after the voltage switch (VSWITCHEN = 1) command response when the CRC is OK and the SDMMC_CK is stopped (no impact on CPSM).
VSWEND	Set after the voltage switch (VSWITCH = 1) timeout of 5 ms + 1 ms (no impact on CPSM).
CPSMACT	Command transfer in progress (CPSM not in Idle state).

The command path error handling is shown in *Table 218*:

Table 218. Command path error handling

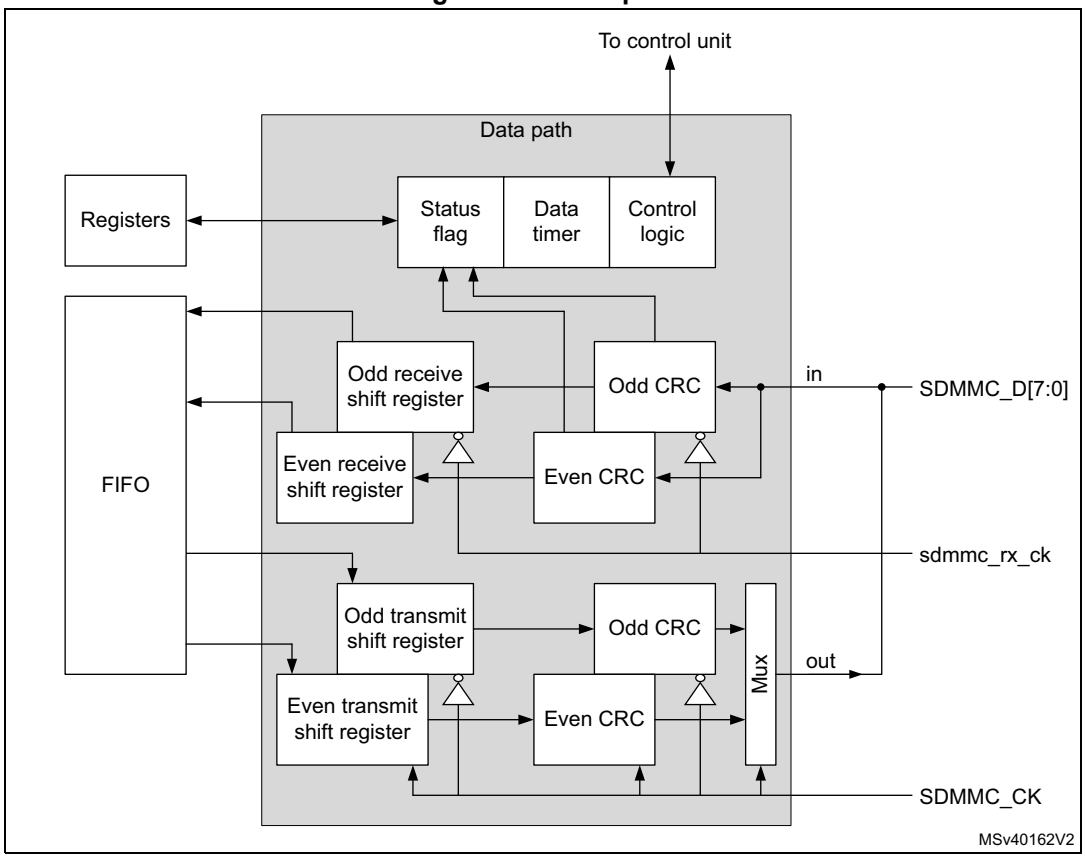
Error	CPSM state	Cause	Card action	Host action	CPSM action
Timeout	Wait	No start bit in time	Unknown	Reset or cycle power card(1)	Move to Idle
CRC status	Receive	Negative status	Command ignored	Resend command(1)	Move to Idle
		Transmission error	Command accepted	Resend command(1)	

1. When CMDTRANS is set, also a stop_transmission command must be send to move the DPSM to Idle.

Data path

The data path subunit transfers data on the SDMMC_D[7:0] lines to and from cards. The data transmit path is clocked on the SDMMC_CK and sends data to the card. The data receive path is clocked on the sdmmc_rx_ck and receives data from the card. *Figure 235* shows the data path block diagram.

Figure 235. Data path



The card data bus width can be programmed in the clock control register bits WIDBUS. The supported data bus width modes are:

- If the wide bus mode is not enabled, only one bit is transferred over SDMMC_D0.
- If the 4-bit wide bus mode is enabled, data is transferred at four bits over SDMMC_D[3:0].
- If the 8-bit wide bus mode is enabled, data is transferred at eight bits over SDMMC_D[7:0].

Next to the data bus width the data sampling mode can be programmed in the clock control register bit DDR. The supported data sampling modes are:

- Single data rate signaling (SDR), data is clocked on the rising edge of the clock.
- Double data rate signaling (DDR), data is clocked on the both edges of the clock. DDR mode is only supported in wide bus mode (4-bit wide and 8-bit wide).

Note: The data sampling mode only applies to the SDMMC_D[7:0] lines. (not applicable to the SDMMC_CMD line.)

In DDR mode, data is sampled on both edges of the SDMMC_CK according the following rules, see also *Figure 236* and *Figure 237*:

- On the rising edge of the clock odd bytes are sampled.
- On the falling edge of the clock even bytes are sampled.
- Data payload size is always a multiple of 2 bytes.
- Two CRC16 are computed per data line
  - Odd bits CRC16 clocked on the falling edge of the clock.
  - Even bits CRC16 clocked on the rising edge of the clock.
- Start, end bits and idle conditions are full cycle.
- CRC status / boot acknowledgment and busy signaling are full cycle and are only sampled on the rising edge of the clock.

In DDR mode, the SDMMC_CK clock division must be ≥ 2.

Figure 236. DDR mode data packet clocking

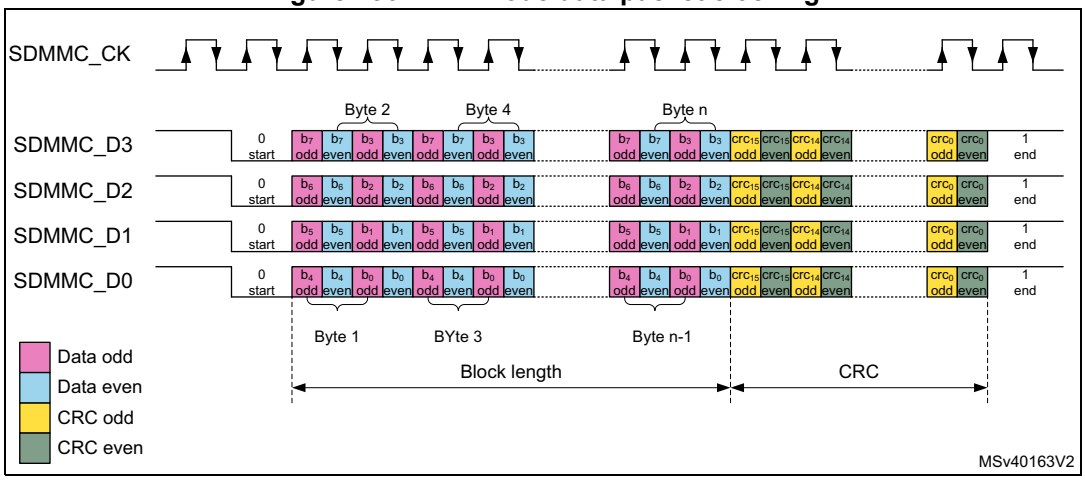
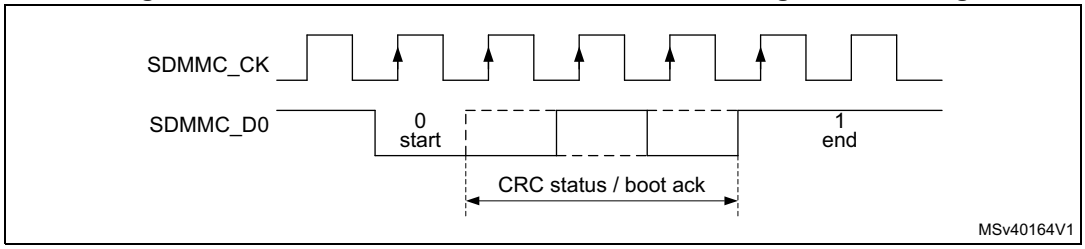


Figure 237. DDR mode CRC status / boot acknowledgment clocking



Data path state machine (DPSM)

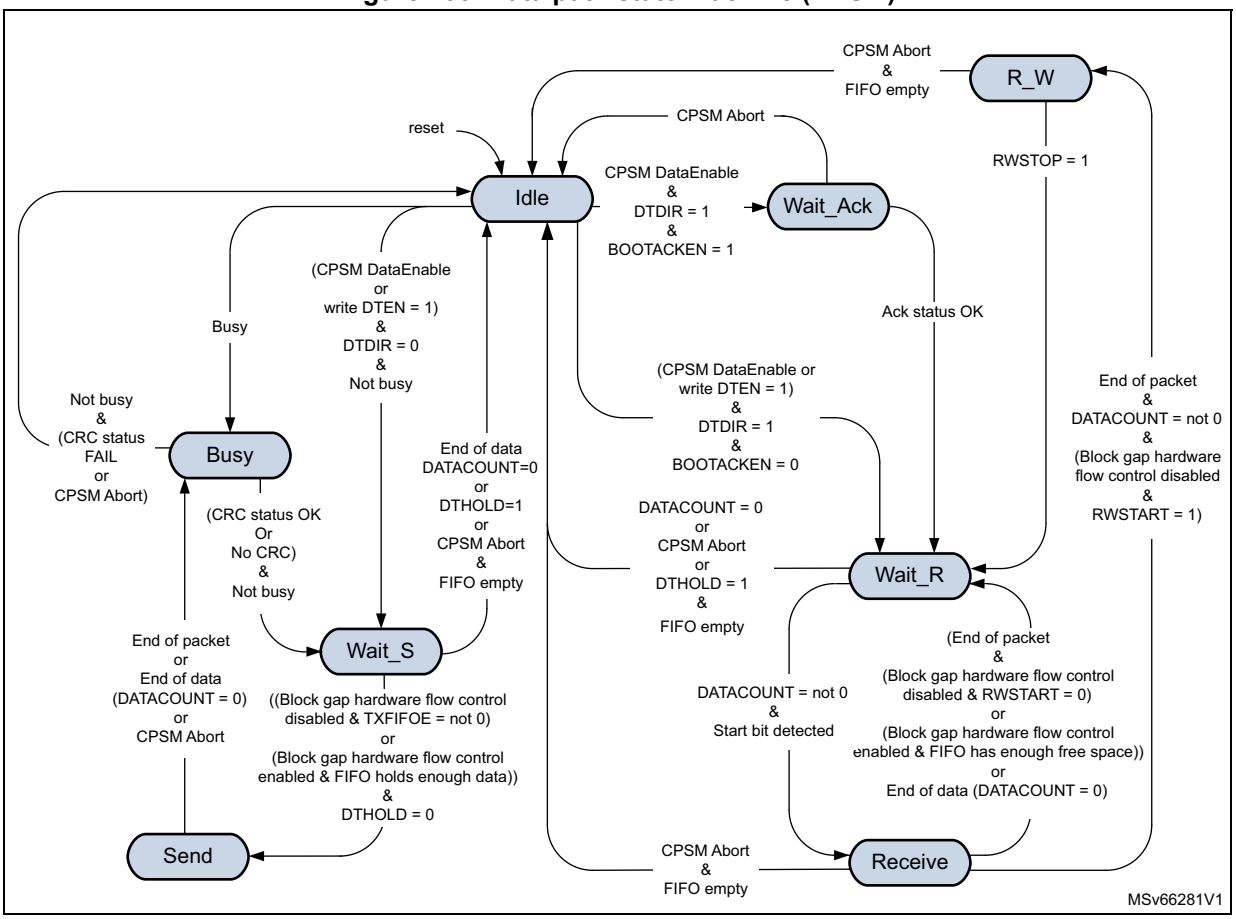
Depending on the transfer direction (send or receive), the data path state machine (DPSM) moves to the Wait_S or Wait_R state when it is enabled:

- Send: the DPSM moves to the Wait_S state. If there is data in the transmit FIFO, the DPSM moves to the Send state, and the data path subunit starts sending data to a card.
- Receive: the DPSM moves to the Wait_R state and waits for a start bit. When it receives a start bit, the DPSM moves to the Receive state, and the data path subunit starts receiving data from a card.

For boot operation with acknowledgment the DPSM moves to the Wait_Ack state and waits for the boot acknowledgment before moving to the Wait_R state.

The DPSM operates at SDMMC_CK. The DPSM has the following states, as shown in *Figure 238*. When ever the DPSM is active (not in the Idle state), the DPSMACT bit is set.

Figure 238. Data path state machine (DPSM)



- **Idle** state: the data path is inactive, and the SDMMC_D[7:0] outputs are according the PWRCTRL setting. The DPSM is activated either by sending a command with CMDTRANS bit set or by setting the DTEN bit, or by detecting Busy on SDMMC_D0 (that is, after a command with R1b response).

  When not busy, the DPSM activates the SDMMC_CK clock (when stopped due to power save PWRSAV bit), loads the data counter with a new (DATALENGTH) value and:

  – When the data direction bit (DTDIR) indicates send, moves to the Wait_S.

  – When the data direction bit (DTDIR) indicates receive, moves to the
  - Wait_R when BOOTACKEN register bit is clear.
  - Wait_Ack when BOOTACKEN register bit is set and start the acknowledgment timeout.

  When busy the DPSM keeps the SDMMC_CK clock active and move to the Busy state.

Note: DTEN must not be used to start data transfer with SD, SDIO and e•MMC cards.

- **Wait_Ack** state: the data path waits for the boot acknowledgment token.
  - The DPSM moves to the Wait_R state if it receives an error free acknowledgment before a timeout.
  - When a pattern different from the acknowledgment is received an acknowledgment status error is generated, and the ack fail status flag (ACKFAIL) is set. The DPSM stays in Wait_Ack.
  - If it reaches a timeout (ACKTIME) before it detects a start bit, it sets the timeout status flag (ACKTIMEOUT). The DPSM stays in Wait_Ack.
  - When the CPSM Abort signal is set it moves to the Idle state and sets the DABORT flag.
- **Wait_R** state: the data path, if the data counter is not zero and data is not hold, waits for a start bit on SDMMC_D[n:0]. If the data counter is zero or data is hold, wait for the FIFO to be empty.
  - In block mode, if a start bit is received before a timeout the DPSM moves to the Receive state and loads the data block counter with DBLOCKSIZE.
  - In SDIO multibyte mode, if a start bit is received before a timeout the DPSM moves to the Receive state and loads the data block counter with DATALENGTH.
  - In stream mode, if a start bit is received before a timeout the DPSM moves to the Receive state and loads the data counter with DATALENGTH.
  - if the data counter (DATACOUNT) equals zero (end of data) the DPSM moves to the Idle state when the receive FIFO is empty and the DATAEND flag is set.
  - If it reaches a timeout (DATATIME) before it detects a start bit, it sets the timeout status flag (DTIMEOUT) and the DPSM stays in the Wait_R state.
  - If the CPSM Abort signal is set:
    If DATACOUNT > 0, the DPSM moves to the Idle state when the FIFO is empty and when IDMAEN = 0 reset with FIFORST, and sets the DABORT flag.
    If DATACOUNT is zero normal operation is continued, there is no DABORT flag since the transfer has completed normally.
  - if the DTHOLD bit is set:
    - When DATACOUNT > 0, the DPSM moves to the Idle state when the receive FIFO is empty and when IDMAEN = 0 reset with FIFORST, and issues the DHOLD flag. When holding the timeout is disabled. When an CPSM Abort signal is received during holding, the transfer is aborted.
    - When DATACOUNT = 0, the transfer is completed normally and there is no DHOLD flag.
  - When DPSM has been started with DTEN, after an error (DTIMEOUT) the DPSM moves to the Idle state when the FIFO is empty and when IDMAEN = 0 reset with FIFORST.

- **R_W** state: the data path Read Wait the bus.
  - The DPSM moves to the Wait_R state when the Read Wait stop bit (RWSTOP) is set, and start the receive timeout.
  - If the CPSM Abort signal is set, wait for the FIFO to be empty and when IDMAEN = 0 reset with FIFORST, then moves to the Idle state and sets the DABORT flag.
- **Receive** state: the data path receives serial data from a card. Pack the data in bytes and written it to the data FIFO. Depending on the transfer mode selected in the data control register (DTMODE), the data transfer mode can be either block or stream:
  - In block mode, when the data block size (DBLOCKSIZE) number of data bytes are received, the DPSM waits until it receives the CRC code.
  - In SDIO multibyte mode, when the data block size (DATALENGTH) number of data bytes are received, the DPSM waits until it receives the CRC code.

  a) If the received CRC code matches the internally generated CRC code, the DPSM moves to the
  - R_W state when RWSTART = 1 and DATACOUNT > zero, the DBCKEND flag is set.
  - Wait_R state otherwise, either immediately when block gap hardware flow control is disabled, or after enough free area is available in the FIFO when block gap hardware flow control is enabled.

  b) If the received CRC code fails the internally generated CRC code any further data reception is prevented.
  - When not all data has been received (DATACOUNT > 0), the CRC fail status flag (DCRCFAIL) is set and the DPSM stays in the Receive state.
  - When all data has been received (DATACOUNT = 0), wait for the FIFO to be empty after which the CRC fail status flag (DCRCFAIL) is set and the DPSM moves to the Idle state.
  - In stream mode, the DPSM receives data while the data counter DATACOUNT > 0. When the counter is zero, the remaining data in the shift register is written to the data FIFO, and the DPSM moves to the Wait_R state.
  - When a FIFO overrun error occurs, the DPSM sets the FIFO overrun error flag (RXOVERR) and any further data reception is prevented. The DPSM stays in the Receive state.
  - When an CPSM Abort signal is received:
  - If the CPSM Abort signal is received before the 2 last bits of the data with DATACOUNT = 0, the transfer is aborted. The remaining data in the shift register is written to the data FIFO, wait for the FIFO to be empty and when IDMAEN = 0 reset with FIFORST, then the DPSM moves to the Idle state and the DABORT flag is set.
  - If the CPSM Abort signal is received during or after the 2 last bits of the transfer with DATACOUNT = 0, the transfer is completed normally. The DPSM stays in the Receive state no DABORT flag is generated.
  - When DPSM has been started with DTEN, after an error (DCRCFAIL when DATACOUNT > 0, or RXOVERR) the DPSM moves to the Idle state when the FIFO is empty and when IDMAEN = 0 reset with FIFORST.
- **Wait_S** state: the data path waits for data to be available from the FIFO.
  - If the data counter DATACOUNT > 0,
  - When block gap hardware flow control is disabled, waits until the data FIFO empty flag (TXFIFOE) is de-asserted and DTHOLD is not set, and moves to the

Send state.
- When block gap hardware flow control is enabled, waits until the data FIFO hold enough data and DTHOLD is not set, and moves to the Send state.

– If the data counter (DATACOUNT) = 0 the DPSM moves to the Idle state.
- When DTHOLD is disabled, the DATAEND flag is set.
- When DTHOLD is enabled, the DHOLD flag is set.

– When DTHOLD is set and the DATACOUNT > 0
- When IDMA is enabled, the DBCKEND flag is set and subsequently the FIFO is flushed, furthermore the DPSM moves to the Idle state and the DHOLD flag is set.
- When IDMA is disabled the DBCKEND flag is set. Wait for the FIFO to be reset by software with FIFORST, then DPSM moves to the Idle state and issues the DHOLD flag.

– When DTHOLD is set and DATACOUNT = 0 the transfer is completed normally.

– When receiving the CPSM Abort signal
- If the CPSM Abort signal is received before the 2 last bits of the data with DATACOUNT = 0, the transfer is aborted, wait for the FIFO to be empty and when IDMAEN = 0 reset with FIFORST, then the DPSM moves to the Idle state and sets the DABORT flag.
- If the CPSM Abort signal is received during or after the 2 last bits of the transfer with DATACOUNT = 0, normal operation is continued, there is no DABORT flag since the transfer has completed normally.

Note: The DPSM remains in the Wait_S state for at least two clock periods to meet the N_{WR} timing requirements, where N_{WR} is the number of clock cycles between the reception of the card response and the start of the data transfer from the host.

- **Send** state: the DPSM starts sending data to a card. Depending on the transfer mode bit in the data control register, the data transfer mode can be either block, SDIO multibyte or stream:
  – In block mode, when the data block size (DBLOCKSIZE) number of data bytes are send, the DPSM sends an internally generated CRC code and end bit, and moves to the Busy state and start the transmit timeout.
  – In SDIO multibyte mode, when the data block size (DATALENGTH) number of data bytes are send, the DPSM sends an internally generated CRC code and end bit, and moves to the Busy state and start the transmit timeout.
  – In stream mode, the DPSM sends data to a card while the data counter DATACOUNT > 0. When the data counter reaches zero moves to the Busy state and start the transmit timeout.
  Before sending the last stream byte according to DATACOUNT, the DPSM issues a trigger on the send CMD signal. This signal is used by the CPSM to sent any pending command (CMD12 Stop Transmission command).
  – If a FIFO underrun error occurs, the DPSM sets the FIFO underrun error flag (TXUNDERR). The DPSM stays in the Send state.
  – When receiving the CPSM Abort signal
  - If the CPSM Abort signal is received before the 2 last bits of the transfer with DATACOUNT = 0, the transfer is aborted. The DPSM sends a last data bit followed by an end bit. The FIFO is disabled/flushed, and the DPSM moves to the Busy state to wait for not busy before setting the DABORT flag.
  - If the CPSM Abort signal is received during or after the 2 last bits of the transfer

with DATACOUNT = 0, the transfer is completed normally, there is no DABORT flag.

- **Busy** state: the DPSM waits for the CRC status token when expected, and wait for a not busy signal:
  - If a CRC status token is expected and indicate “non-erroneous transmission” or when there is no CRC expected:
    - it moves to the Wait_S state when SDMMC_D0 is not low (the card is not busy).
    - When the card is busy SDMMC_D0 is low it remains in the Busy state.
  - If a CRC status token is expected and indicates “erroneous transmission”.
    - When not all data has been send (DATACOUNT > 0). The DPSM waits for not busy after which the CRC fail status flag (DCRCFAIL) is set. The FIFO is disabled/flushed and the DPSM stays in the Busy state.
    - When all data has been send (DATACOUNT = 0). The DPSM waits for not busy after which the CRC fail status flag (DCRCFAIL) is set and the DPSM moves to the Idle state.
  - If a CRC status (Ncrc) timeout occurs while the DPSM is in the Busy state, it sets the data timeout flag (DTIMEOUT) and stays in the Busy state.
  - If a busy timeout occurs while the DPSM is in the Busy state, it sets the data timeout flag (DTIMEOUT) and stays in the Busy state.
  - When receiving the CPSM Abort signal in the Busy state:
    - If the CPSM Abort signal is received before the 2 last bits of the CRC response with DATACOUNT > 0, the data transfer is aborted. The DPSM waits for not busy and the FIFO to be disabled/flushed before moving to the Idle state and the DABORT flag is set.
    - If the CPSM Abort signal is received during or after the 2 last bits of the CRC response when DATACOUNT = 0 or when no CRC is expected and DATACOUNT = 0 and there has been no DTIMEOUT error, the DPSM stays in the Busy state no DABORT flag is generated, since the transfer may completed normally.
    - If the CPSM Abort signal is received when a DTIMEOUT error has occurred the DPSM waits for not busy and the FIFO to be disabled/flushed before moving to the Idle state and the DABORT flag is set.
  - When entering the Busy state due to an abort in the Send state, the DPSM waits for not busy before moving to the Idle state and the DABORT flag is set.
  - When DPSM has been started with DTEN, after an error (DCRCFAIL when DATACOUNT > 0, or DTIMEOUT) the DPSM moves to the Idle state when the FIFO is reset.
  - When the DPSM has been started due to Busy on SDMMC_D0, waits for not busy after which the Busy end status flag (BUSYD0END) is set and the DPSM moves to the Idle state.

The data timer (DATATIME) is enabled when the DPSM is in the Wait_R or Busy state 2 cycles after the data block end bit, or data read command end bit, or R1b response, and generates the data timeout error (DTIMEOUT):

- When transmitting data, the timeout occurs
  - when a CRC status is expected and no start bit is received withing 8 SDMMC_CK cycles, the DTIMEOUT flag is set.
  - when the Busy state takes longer than the programmed timeout period., the DTIMEOUT flag is set.
- When receiving data, the timeout occurs
  - when there is still data to be received DATACOUNT > 0 and no start bit is received before the programmed timeout period, the DTIMEOUT flag is set.
- After a R1b response, the timeout occurs
  - when the Busy state takes longer than the programmed timeout period., the DTIMEOUT flag is set.

When DATATIME = 0:

- In receive the start bit must be present 2 cycles after the data block end bit or data read command end bit.
- In transmit busy is timed out 2 cycles after the CRC token end bit or stream data end bit.
- After a R1b response busy is timed out 2 cycles after the response end bit.

Data can be transferred from the card to the host (transmit, send) or vice versa (receive). Data are transferred via the SDMMC_Dn data lines, they are stored in a FIFO.

Table 219. Data token format

Description	Start bit	Data(1)	CRC16	End bit	DTMODE
Block data	0	(DBLOCKSIZE, DATALENGTH)	Yes	1	00
SDIO multibyte	0	(DATALENGTH)	Yes	1	01
e•MMC stream	0	(DATALENGTH)	No	1	10

1. The total amount of data to transfer is given by DATALENGTH. Where for Block data the amount of data in each block is given by DBLOCKSIZE.

The data token format is selected with register bits DTMODE according.

The data path implements the status flags and associated clear bits shown in *Table 220*:

Table 220. Data path status flags and clear bits

Flag		Description
DATAEND	TX	Set at the end of the complete data transfer when the CRC is OK and busy has finished and both DTHOLD = 0 and DATACOUNT = 0. (DPSM moves from Wait_S to Idle)
	RX	Set at the end of the complete data transfer when the CRC is OK and all data has been read, (DATACOUNT = 0 and FIFO is empty). (DPSM moves from Wait_R to Idle)
	Boot	

Table 220. Data path status flags and clear bits (continued)

Flag		Description
DCRCFAIL	TX	Set at the end of the CRC when FAIL and busy has finished. (DPSM stay in Busy when there is still data to send and wait for CPSM Abort) (DPSM moves from Busy to Idle when all data has been sent) or DPSM has been started with DTEN
	RX	Set at the end of the CRC when FAIL and FIFO is empty. (DPSM stays in Receive when there is still data to be received and wait for CPSM Abort) (DPSM moves from Receive to Idle when all data has been received or DPSM has been started with DTEN)
	Boot	
ACKFAIL	Boot	Set at the end of the boot acknowledgment when fail. (DPSM stays in Wait_Ack and wait for CPSM Abort)
DTIMEOUT	CMD R1b	Set after the command response no end of busy received before the timeout. (DPSM stays in Busy and wait for CPSM Abort)
	TX	Set when no CRC token start bit received within Ncrc, or no end of busy received before the timeout. (DPSM stays in Busy and wait for CPSM Abort) (When DPSM has been started with DTEN move to Idle) Note: The DCRCFAIL flag may also be set when CRC failed before the busy timeout.
	RX	Set when no start bit received before the timeout. (DPSM stays in Wait_R and wait for CPSM Abort) (When DPSM has been started with DTEN move to Idle)
	Boot	
ACKTIMEOUT	Boot	Set when no start bit received before the timeout. (DPSM stays in Wait_Ack and wait for CPSM Abort)
DBCKEND	TX	When DTHOLD = 1 and IDMAEN = 0: Set at the end of data block transfer when the CRC is OK and busy has finished, when data transfer is not complete (DATACOUNT >0). (DPSM moves from Busy to Wait_S)
	RX	When RWSTART = 1: Set at the end of data block transfer when the CRC is OK, when data transfer is not complete (DATACOUNT > 0). (DPSM moves from Receive to R_W)
	Boot	
DHOLD	TX	When DTHOLD = 1: Set at the end of data block transfer when the CRC is OK and busy has finished. (DPSM moves from Wait_S to Idle)
	RX	When DTHOLD = 1: Set at the end of data block transfer when the CRC is OK and all data has been read (FIFO is empty), when data transfer is not complete (DATACOUNT >0). (DPSM moves from Wait_R to Idle)
DABORT	CMD R1b	When CPSM Abort event has been sent by the CPSM and busy has finished. (DPSM moves from Busy to Idle)
	TX	When CPSM Abort event has been sent by the CPSM before the 2 last bits of the transfer. (DPSM moves from any state to Idle)
	RX	
	Boot	
BUSYD0END	CMD R1b	Set after the command response when end of busy before the timeout. (DPSM moves from Busy to Idle)
DPSMACT		Data transfer in progress. (DPSM not in Idle state)

The data path error handling is shown in *Table 221*:

Table 221. Data path error handling

Error	DPSM state	Cause	Card action	Host action	DPSM action
Timeout	Wait_Ack	No Ack in time	unknown	Card cycle power	Stay in Wait_Ack (reset the SDMMC with the RCC.SDMMCxRST register bit)
	Wait_R	No start bit in time	unknown	Stop data reception Send stop transmission command	On CPSM Abort move to Idle
			unknown	Stop boot procedure	
	Busy	Busy too long (due to data transfer)	unknown	Stop data reception Send stop transmission command	
		Busy too long (due to R1b)	unknown	Send reset command	
CRC	Receive	transmission error	Send further data	Stop data reception Send stop transmission command	On CPSM Abort move to Idle
CRC status	Busy	Negative status	Ignore further data	Stop data transmission Send stop transmission command	On CPSM Abort move to Idle
		transmission error	wait for further data		
Ack status	Wait_Ack	transmission error	Send boot data	Stop boot procedure	On CPSM Abort move to Idle
Overrun	Receive	FIFO full	Send further data	Stop data reception Send stop transmission command	On CPSM Abort move to Idle
Underrun	Send	FIFO empty	Receive further data	Stop data transmission Send stop transmission command	On CPSM Abort move to Idle

Data FIFO

The data FIFO (first-in-first-out) subunit contains the transmit and receive data buffer. A single FIFO is used for either transmit or receive as selected by the DTDIR bit. The FIFO contain a 32-bit wide, 256-word deep data buffer and control logic. Because the data FIFO operates in the AHB clock domain (sdmmc_hclk), all signals from the subunits in the SDMMC clock domain (SDMMC_CK/sdmmc_rx_ck) are resynchronized.

The FIFO can be in one of the following states:

- The transmit FIFO refers to the transmit logic and data buffer when sending data out to the card. (DTDIR = 0)
- The receive FIFO refers to the receive logic and data buffer when receiving data in from the card. (DTDIR = 1)

The end of a correctly completed SDMMC data transfer from the FIFO is indicated by the DATAEND flags driven by the data path subunit. Any incorrect (aborted) SDMMC data transfer from the FIFO is indicated by one of the error flags (DCRCFAIL, DTIMEOUT, DABORT) driven by the data path subunit, or one of the FIFO error flags (TXUNDERR, RXOVERR) driven by the FIFO control.

The data FIFO can be accessed in the following ways, see *Table 222*.

Table 222. Data FIFO access

Data FIFO access	IDMAEN
From firmware via AHB slave interface	0
From IDMA via AHB master interface	1

Transmit FIFO:

Data can be written to the transmit FIFO when the DPSM has been activated (DPSMACT = 1).

When IDMAEN = 1 the FIFO is fully handled by the IDMA.

When IDMAEN = 0 the FIFO is controlled by firmware via the AHB slave interface. The transmit FIFO is accessible via sequential addresses. The transmit FIFO contains a data output register that holds the data word pointed to by the read pointer. When the data path subunit has loaded its shift register, it increments the read pointer and drives new data out. The transmit FIFO is handled in the following way:

1. Write the data length into DATALENGTH and the block length in DBLOCKSIZE.
   – For block data transfer (DTMODE = 0), DATALENGTH must be an integer multiple of DBLOCKSIZE.
2. Enable hardware flow control HWFC_EN.
   If bus speed SDR50, SDR104, HS200 or DDR50 is used FIFO threshold can be used to provide flow control at the receive block gap.
3. Set the SDMMC in transmit mode (DTDIR = 0).
   – Configures the FIFO in transmit mode.
4. Enable the data transfer
   – either by sending a command from the CPSM with the CMDTRANS bit set
   – or by setting DTEN bit
5. When (DPSMACT = 1) write data to the FIFO.
   – When BUSSPEED = 0: The DPSM stays in the Wait_S state until FIFO is full (TXFIFOF = 1), or the number indicated by DATALENGTH.
   - The SDMMC keeps sending data as long as FIFO is not empty, hardware flow control during data transfer is used to prevent FIFO underrun.
   – When BUSSPEED = 1 and the FIFO threshold is zero: The DPSM reads the transmit data from the FIFO.

- The SDMMC keeps transmitting data as long as FIFO is not empty, hardware flow control during the data transfer is used to prevent FIFO underrun.

– When BUSSPEED = 1 and the FIFO threshold is not zero: The DPSM reads the transmit data from the FIFO.
- The SDMMC only starts sending a new block when there is enough data in the FIFO according the threshold. Block gap hardware flow control is used to prevent FIFO underrun.

6. Write data to the FIFO.
   – When the FIFO is handled by software, wait until the FIFO is half empty (TXFIFOHE flag), write data to the FIFO until FIFO is full (TXFIFOF = 1), or last data has been written.
   – When the FIFO is handled by the IDMA, the IDMA transfers the FIFO date.
7. When last data has been written wait for end of data (DATAEND flag)
   – SDMMC has completely sent all data and the DPSM is disabled (DPSMACT = 0).

In case of a data transfer error or transfer hold when IDMAEN = 0, firmware must stop writing to the FIFO and flush and reset the FIFO with the FIFORST register bit.

The transmit FIFO status flags are listed in *Table 223*.

Table 223. Transmit FIFO status flags

Flag	Description
TXFIFOF	Set to high when all transmit FIFO words contain valid data.
TXFIFOE	Set to high when the transmit FIFO does not contain valid data.
TXFIFOHE	Set to high when half or more transmit FIFO words are empty.
TXUNDERR	Set to high when an underrun error occurs. This flag is cleared by writing to the SDMMC Clear register.

Receive FIFO:

Data can be read from the receive FIFO when the DPSM is activated (DPSMACT = 1).

When IDMAEN = 1 the FIFO is fully handled by the IDMA.

When IDMAEN = 0 the FIFO is controlled by firmware via the AHB slave interface.When the data path subunit receives a word of data, it drives the data on the write databus. The write pointer is incremented after the write operation completes. On the read side, the contents of the FIFO word pointed to by the current value of the read pointer is driven onto the read databus. The receive FIFO is accessible via sequential addresses.

The receive FIFO is handled in the following way:

1. Write the data length into DATALENGTH and the block length in DBLOCKSIZE.
   – For block data transfer (DTMODE = 0), DATALENGTH must be an integer multiple of DBLOCKSIZE.
2. Enable hardware flow control HWFC_EN.
   – If bus speed SDR50, SDR104, HS200 or DDR50 is used FIFO threshold can be used to provide flow control at the receive block gap.
3. Set the SDMMC in receive mode (DTDIR = 1).
   – Configures the FIFO in receive mode.
4. Enable the DPSM transfer
   – either by sending a command from the CPSM with the CMDTRANS bit set
   – or by setting DTEN bit.
5. When (DPSMACT = 1) the FIFO is ready to receive data.
   – When BUSSPEED = 0: The DPSM writes the received data to the FIFO.
   - The SDMMC keeps receiving data as long as FIFO is not full, hardware flow control during the data transfer is used to prevent FIFO overrun.
   – When BUSSPEED = 1 and the FIFO threshold is zero: The DPSM writes the received data to the FIFO.
   - The SDMMC keeps receiving data as long as FIFO is not full, hardware flow control during the data transfer is used to prevent FIFO overrun.
   – When BUSSPEED = 1 and the FIFO threshold is not zero: The DPSM writes the received data to the FIFO.
   - The SDMMC only starts receiving a new block when there is enough free space in the FIFO according the threshold. Block gap hardware flow control is used to prevent FIFO overrun.
6. Read data from the FIFO.
   – When the FIFO is handled by software, wait until the FIFO is half full (RXFIFOHF flag), read data from the FIFO until FIFO is empty (RXFIFOE = 1).
   - When last data has been received, read data from the FIFO until FIFO is empty (DATAEND = 1).
   – When the FIFO is handled by the IDMA, the IDMA transfers the FIFO date.
7. SDMMC has completely received all data and the DPSM is disabled (DPSMACT = 0).

In case of a data transfer hold when IDMAEN = 0, the firmware must read the remaining data until the FIFO is empty and reset the FIFO with the FIFORST register bit. This causes the DPSM to go to the Idle state (DPSMACT = 0).

In case of a data transfer error when IDMAEN = 0, the firmware must stop reading the FIFO and flush and reset the FIFO with the FIFORST register bit. This causes the DPSM to go to the Idle state (DPSMACT = 0).

The receive FIFO status flags are listed in *Table 224*.

Table 224. Receive FIFO status flags

Flag	Description
RXFIFOF	Set to high when all receive FIFO words contain valid data
RXFIFOE	Set to high when the receive FIFO does not contain valid data.
RXFIFOHF	Set to high when half or more receive FIFO words contain valid data.
RXOVERR	Set to high when an overrun error occurs. This flag is cleared by writing to the SDMMC Clear register.

CLKMUX unit

The CLKMUX selects the source for clock sdmmc_rx_ck to be used with the received data and command response. The receive data clock source can be selected by the clock control register bit SELCLKRX, between:

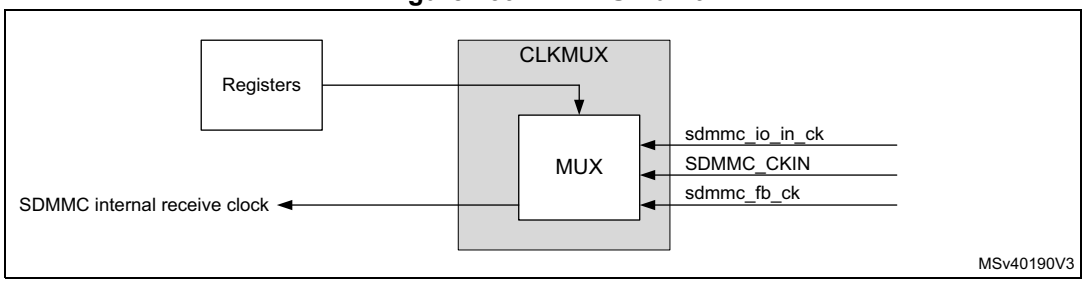
- sdmmc_io_in_ck bus master main feedback clock.
- SDMMC_CKIN external bus feedback clock.
- sdmmc_fb_ck bus tuned feedback clock.

The sdmmc_io_in_ck is selected when there is no external driver, with DS and HS.

The SDMMC_CKIN is selected when there is an external driver with SDR12, SDR25, SDR50 and DDR50.

The sdmmc_fb_ck clock input must be selected when the DLYB block on the device is used with SDR104, HS200 and optionally with SDR50 and DDR50 modes.

Figure 239. CLKMUX unit



The sdmmc_rx_ck source must be changed when the CPSM and DPSM are in the Idle state.

30.5.5 SDMMC AHB slave interface

The AHB slave interface generates the interrupt requests, and accesses the SDMMC adapter registers and the data FIFO. It consists of a data path, register decoder, and interrupt logic.

SDMMC FIFO

The FIFO access is restricted to word access only:

- In transmit FIFO mode
  - Data are written to the FIFO in words (32-bits) until all data according DATALENGTH has been transfered. When the DATALENGTH is not an integer

multiple of 4, the last remaining data (1, 2 or 3 bytes) are written with a word transfer.

- In receive FIFO mode
  - Data are read from the FIFO in words (32-bits) until all data according DATALENGTH has been transfered. When the DATALENGTH is not an integer multiple of 4, the last remaining data (1, 2 or 3 bytes) are read with a word transfer padded with 0 value bytes.

When accessing the FIFO with half word or byte accesses an AHB bus fault is generated.

SDMMC interrupts

The interrupt logic generates an interrupt request signal that is asserted when at least one of the unmasked status flags is active. A mask register is provided to allow selection of the conditions that generate an interrupt. A status flag generates the interrupt request if a corresponding mask flag is set. Some status flags require an implicit clear in the clear register.

30.5.6 SDMMC AHB master interface

The AHB master interface is used to transfer the data between a memory and the FIFO using the SDMMC IDMA.

SDMMC IDMA

Direct memory access (DMA) is used to provide high-speed transfer between the SDMMC FIFO and the memory. The AHB master optimizes the bandwidth of the system bus. The SDMMC internal DMA (IDMA) provides one channel to be used either for transmit or receive.

The IDMA is enabled by the IDMAEN bit and supports burst transfers of 16 beats.

- In transmit burst transfer mode:
  - Data are fetched in burst from memory whenever the FIFO is empty for the number of burst transfers, until all data according DATALENGTH has been transfered. When the DATALENGTH is not an integer multiple of the burst size the remaining, smaller then burst size data is transfered using single transfer mode. When the DATALENGTH is not an integer multiple of 4, the last remaining data (1, 2 or 3 bytes) are fetched with a word transfer.
- In receive burst transfer mode:
  - Data are stored in burst in to memory whenever the FIFO contains the number of burst transfers, until all data according DATALENGTH has been transfered. When the DATALENGTH is not an integer multiple of the burst transfer the remaining, smaller then burst size data, is transfered using single transfer mode. When the DATALENGTH is not an integer multiple of 4, the last remaining data (1, 2 or 3 bytes) are stored with halfword and or byte transfers.

In addition the IDMA provides the following channel configurations selected by bit IDMABMODE:

- single buffered channel
- linked list channel

Single buffered channel

In single buffer configuration the data at the memory side is accessed in a linear matter starting from the base address IDMABASE. When the IDMA has finished transferring all data the and the DPSM has completed the transfer the DATAEND flag is set.

Linked list channel

In linked list configuration, IDMAMODE = 1, the data at the memory side is subsequently accessed from linked buffers, located at base address IDMABASE. The size of the memory buffers is defined by IDMABSIZE. The buffer size must be an integer multiple of the burst size. The bit ULA is used to indicate if a new linked list buffer configuration has to be loaded from the linked list table. A new linked list configuration is loaded when the ULA bit for the current linked list item is set.

The first linked list item configuration is programmed by firmware directly in the SDMMC registers.

When the IDMA has finished transferring all the data of one linked list buffer, according IDMABSIZE, and when the linked list item ULA bit is set, the IDMA loads the new linked list item from the linked list table, and continues transferring data from the next linked list buffer. When the IDMA has finished transferring all data, according IDMABSIZE and ULA, and the DPSM has completed the transfer, according DATALENGHT, the DATAEND flag is set.

In the following cases, the linked list provides more buffer space than the data to transfer which means the current linked list buffer data has not completely be transfered:

- the ULA bit is set, and all SDMMC data according DATALENGTH has been transfered (DATAEND flag)
- a transfer error (DCRCFAIL when DATACOUNT > 0, RXOVERR, TXUNDERR) occurs
- a transfer is hold (DTHOLD)

In all above cases, the IDMA linked list is stopped and the FIFO is flushed/reset. Before starting or restarting a new SDMMC transfer, the software must initialize a new linked list with correct IDMABASE and IDMABSIZE.

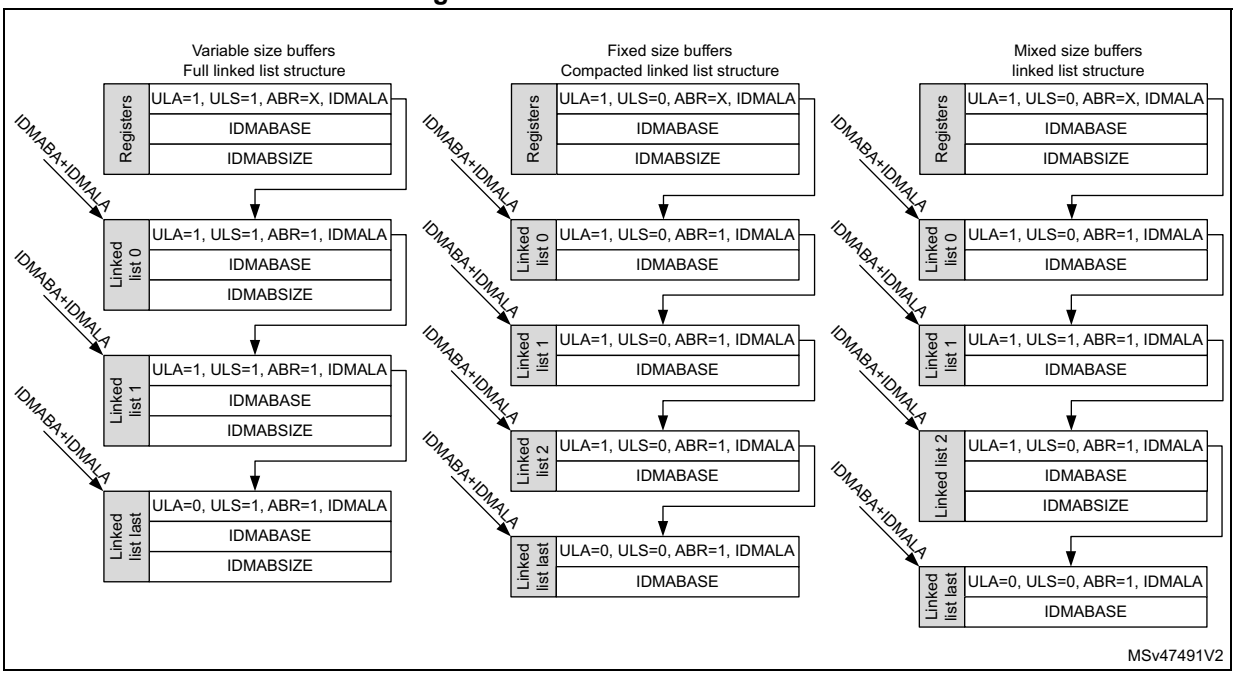
When a IDMA transfer error occurs (see *Section : IDMA transfer error management*) or when the linked list does not provide sufficient buffer space:

- the linked list ends with ULA = 0 and all last linked list buffer data has been transfered, and not all SDMMC data according DATALENGTH has been transfered. The SDMMC transfer is stopped and an IDMA transfer error is generated (see *Section : IDMA transfer error management*).

For a given linked list item, the base address is given by the linked list base IDMABA register value plus the linked list offset IDMALA register value.

The content of each linked list item can be specified by the ULS bit, which makes possible to optionally load the IDMABSIZE, resulting in a 3-word linked list structure. When the IDMABSIZE is not to be loaded (fixed size buffers) a compacted reduced 2-word linked list structure can be used containing only the IDMABASER and the IDMALAR values.

Figure 240. Linked list structures



There is no restriction on mixing both linked list item structures in a single list, this enables the IDMABSIZE to be updated only when needed.

Whenever a linked list buffer has been transfered and the current buffer ULA = 1, an end-of-linked-list-buffer-transfer-complete interrupt (IDMABTC) may be generated (if interrupt is enabled).

Linked list acknowledgment

In the case where software dynamically updates the linked list, during the SDMMC transfer, the availability of a new linked list buffer can be acknowledged by the acknowledge buffer ready (ABR) bit.

When ABR acknowledges that the new linked list buffer is ready, the IDMA continues transferring data from the new linked list buffer.

When ABR indicates that the new linked list buffer is not ready, an IDMA transfer error is generated (see *Section : IDMA transfer error management*). Depending when the IDMA transfer error occurs, it normally causes the generation of an TXUNDERR or RXOVERR error. When a linked list buffer is not acknowledged in time the SDMMC transfer is stopped.

The ABR information is “don't care” when starting the linked list from software programmed register information. The first linked list buffer must be ready to be used before starting the SDMMC transfer.

IDMA transfer error management

An IDMA transfer error can occur:

- When reading or writing a reserved address space (for data or linked list information).
- When there is no more linked list buffer space to store received SDMMC data.
- When all linked list buffer data has been transfered and still more SDMMC data needs to be sent.
- When the availability of a linked list buffer is not acknowledged.

On an IDMA transfer error subsequent IDMA transfers are disabled and an IDMATE flag is set and hardware flow control is disabled. Depending when the IDMA transfer error occurs, it normally causes the generation of a TXUNDERR or RXOVERR error.

The behavior of the IDMATE flag depend on when the IDMA transfer error occurs during the SDMMC transfer:

- An IDMA transfer error is detected before any SDMMC transfer error (TXUNDERR, RXOVERR, DCRCFAIL, or DTIMEOUT):
  - The IDMATE flag is set at the same time as the SDMMC transfer error flag.
  - The TXUNDERR, RXOVERR, DCRCFAIL, or DTIMEOUT interrupt is generated.
- An IDMA transfer error is detected during a STOP_TRANSNMISSION command:
  - The IDMATE flag is set at the same time as the DABORT flag.
  - The DABORT interrupt is generated.
- An IDMA transfer error is detected at the end of the SDMMC transfer (DHOLD, or DATAEND).
  - The IDMATE flag is set at the end of the SDMMC transfer.
  - A SDMMC transfer end interrupt is generated and a DHOLD or DATAEND flag is set.

The IDMATE is generated on an other SDMMC transfer interrupt (TXUNDERR. RXOVERR, DCRCFAIL, DTIMEOUT, DABORT, DHOLD, or DATAEND).

30.5.7 AHB and SDMMC_CK clock relation

The AHB must at least have between 1.72x and 3x more bandwidth than the SDMMC bus bandwidth i.e. for SDR50 4-bit mode (50 Mbyte/s) the minimum sdmmc_hclk frequency is 21.43 MHz (85.72 Mbyte/s).

Table 225. AHB and SDMMC_CK clock frequency relation

SDMMC bus mode	SDMMC bus width	Maximum SDMMC_CK [MHz]	Minimum AHB clock [MHz]
e•MMC DS	8	26	13
e•MMC HS	8	52	26
e•MMC DDR52	8	52	78
e•MMC HS200	8	200	100
SD DS / SDR12	4	25	5.36
SD HS / SDR25	4	50	10.72
SD DDR50	4	50	25

Table 225. AHB and SDMMC_CK clock frequency relation (continued)

SDMMC bus mode	SDMMC bus width	Maximum SDMMC_CK [MHz]	Minimum AHB clock [MHz]
SD SDR50	4	100	21.43
SD SDR104	4	208	44.58

Warning: When minimum HCLK frequency is not respected, overrun or underrun errors may occur even when hardware flow control is enabled.

30.6 Card functional description

30.6.1 SD I/O mode

The following features are SDMMC specific operations:

- SDIO interrupts
- SDIO suspend/resume operation (write and read suspend)
- SDIO Read Wait operation by stopping the clock
- SDIO Read Wait operation by SDMMC_D2 signaling

Table 226. SDIO special operation control

Operation mode	SDIOEN	RWMOD	RWSTOP	RWSTART	DTDIR
Interrupt detection	1	X	X	X	X
Suspend/Resume operation	X	X	X	X	X
Read Wait SDMMC_CK clock stop (START)	X	1	0	1	1
Read Wait SDMMC_CK clock stop (STOP)	X	1	1	1	1
Read Wait SDMMC_D2 signaling (START)	X	0	0	1	1
Read Wait SDMMC_D2 signaling (STOP)	X	0	1	1	1

SD I/O interrupts

To allow the SD I/O card to interrupt the host, an interrupt function is available on pin 8 (shared with SDMMC_D1 in 4-bit mode) on the SD interface. The use of the interrupt is optional for each card or function within a card. The SD I/O interrupt is level-sensitive, which means that the interrupt line must be held active (low) until it is either recognized and acted upon by the host or deasserted due to the end of the interrupt period. After the host has serviced the interrupt, the interrupt status bit is cleared via an I/O write to the appropriate bit in the SD I/O card internal registers. The interrupt output of all SD I/O cards is active low and the application must provide external pull-up resistors on all data lines (SDMMC_D[3:0]).

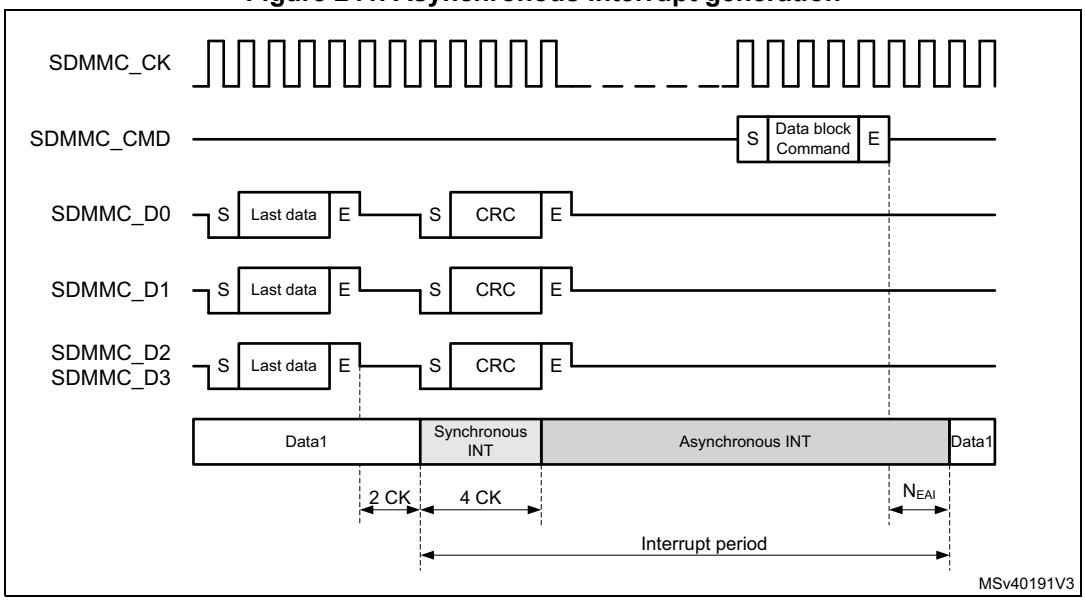
In SD 1-bit mode pin 8 is dedicated to the interrupt function (IRQ), and there are no timing constraints on interrupts.

In SD 4-bit mode the host samples the level of pin 8 (SDMMC_D1/IRQ) into the interrupt detector only during the interrupt period. At all other times, the host interrupt ignores this value. The interrupt period begins when interrupts are enabled at the card and SDIOEN bit is set see register settings in *Table 226*.

In 4-bit mode the card can generate a synchronous or asynchronous interrupt as indicated by the card CCCR register SAI and EAI bits.

- Synchronous interrupt, require the SDMMC_CK to be active.
- Asynchronous interrupt, can be generated when the SDMMC_CK is stopped, 4 cycles after the start of the card interrupt period following the last data block.

Figure 241. Asynchronous interrupt generation



The timing of the interrupt period is depending on the bus speed mode.

In DS, HS, SDR12, and SDR25 mode, selected by register bit BUSSPEED, the interrupt period is synchronous to the SD clock.

- The interrupt period ends at the next clock from the end bit of a command that transfers data block(s) (Command sent with the CMDTRANS bit is set), or when the DTEN bit is set.
- The interrupt period resumes 2 SDMMC_CK after the completion of the data block.
- At the data block gap the interrupt period is limited to 2 SDMMC_CK cycles.

Note: DTEN must not be used to start data transfer with SD and e•MMC cards.

Figure 242. Synchronous interrupt period data read

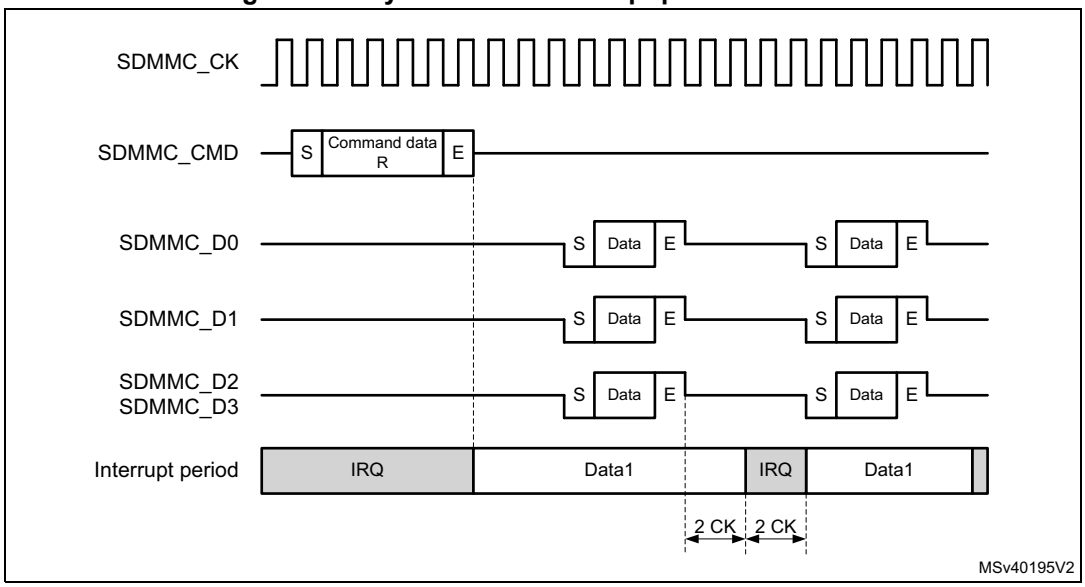
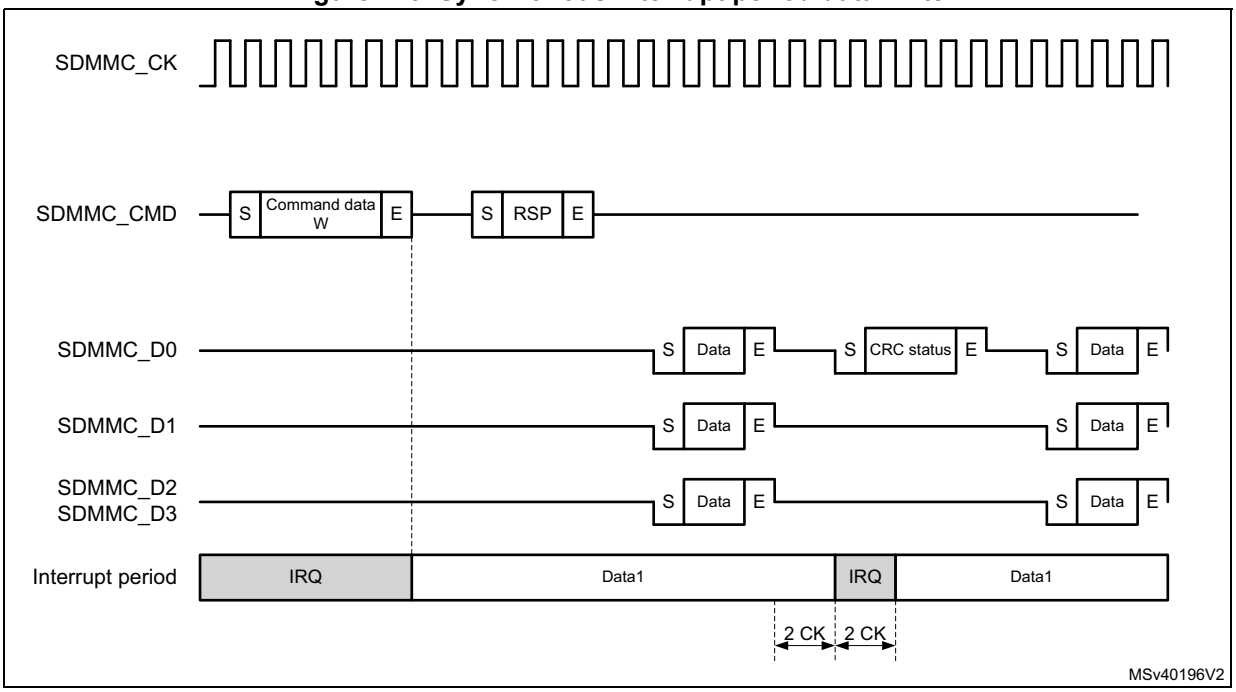


Figure 243. Synchronous interrupt period data write



In SDR50, SDR104, and DDR50, selected by register bit BUSSPEED, due to propagation delay from the card to host, the interrupt period is asynchronous.

- The card interrupt period ends after 0 to 2 SDMMC_CK cycles after the end bit of a command that transfers data block(s) (Command sent with the CMDTRANS bit is set), or when the DTEN bit is set. At the host the interrupt period ends after the end bit of a

command that transfers data block(s). A card interrupt issued in the 1 to 2 cycles after the command end bit are not detected by the host during this interrupt period.

- The card interrupt period resumes 2 to 4 SDMMC_CK after the completion of the last data block. The host resumes the interrupt period always 2 cycles after the last data block.
- There is NO interrupt period at the data block gap.

Note: DTEN must not be used to start data transfer with SD and e•MMC cards.

Figure 244. Asynchronous interrupt period data read

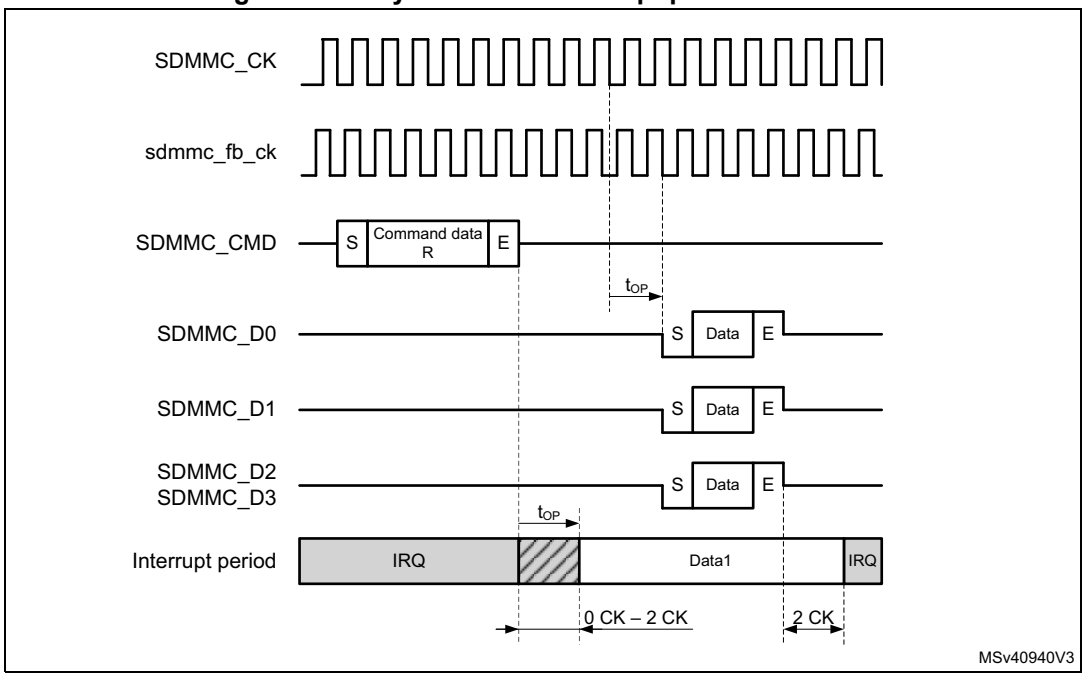
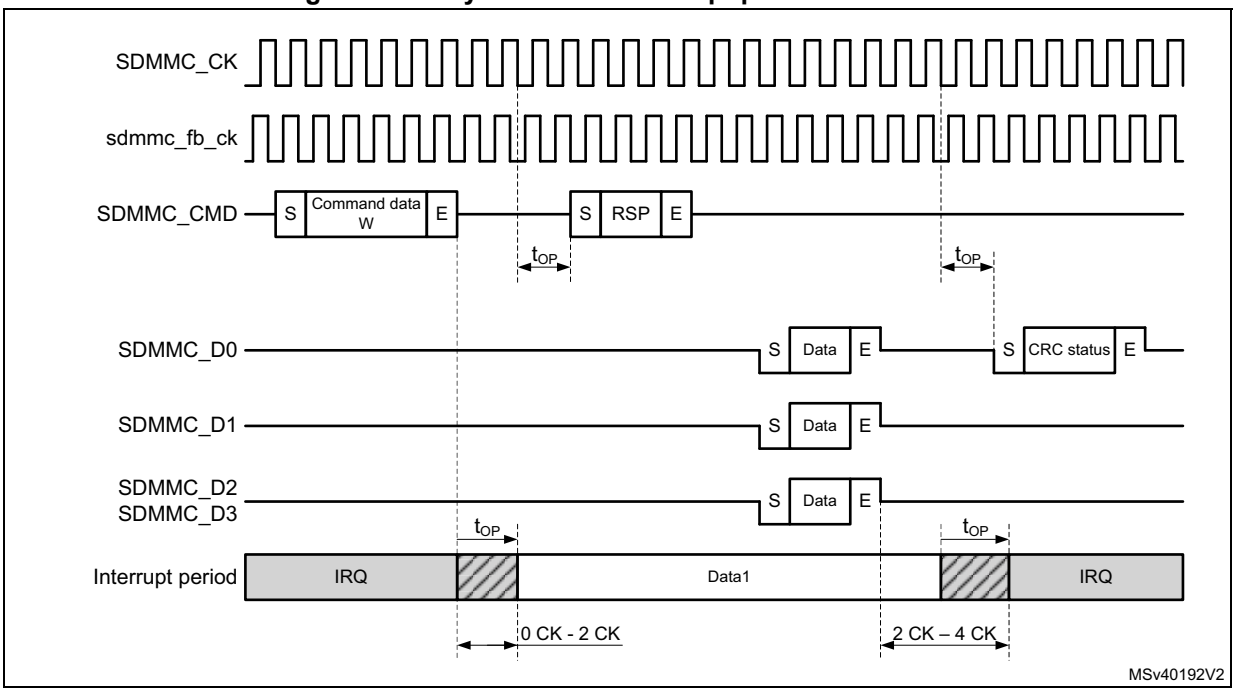


Figure 245. Asynchronous interrupt period data write



When transferring Open-ended multiple block data and using DTMODE “block data transfer ending with STOP_TRANSMISSION command”, the SDMMC masks the interrupt period after the last data block until the end of the CMD12 STOP_TRANSMISSION command.

The interrupt period is applicable for both memory and I/O operations.

In 4-bit mode interrupts can be differentiated from other signaling according *Table 227*.

Table 227. 4-bit mode Start, interrupt, and CRC-status Signaling detection

SDMMC data line	Start	Interrupt	CRC-status
SDMMC_D0	0	1 or CRC-status	0
SDMMC_D1	0	0	X
SDMMC_D2	0	1 or Read Wait	X
SDMMC_D3	0	1	X

SD I/O suspend and resume

This function is NOT supported in SDIO version 4.00 or later.

Within a multifunction SD I/O or a card with both I/O and memory functions, there are multiple devices (I/O and memory) that share access to the *e*•MMC/SD bus. To share access to the host among multiple devices, SD I/O and combo cards optionally implement the concept of suspend/resume. When a card supports suspend/resume, the host can temporarily halt (suspend) a data transfer operation to one function or memory to free the bus for a higher-priority transfer to a different function or memory. After this higher-priority transfer is complete, the original transfer is restarted (resume) where it left off.

To perform the suspend/resume operation on the bus, the host performs the following steps:

1. Determines the function currently using the SDMMC_D[3:0] line(s).
2. Requests the lower-priority or slower transaction to suspend.
3. Waits for the transaction suspension to complete.
4. Begins the higher-priority transaction.
5. Waits for the completion of the higher priority transaction.
6. Restores the suspended transaction.

The card receiving a suspend command responds with its current bus status. Only when the bus has been suspended by the card the bus status indicates suspension completed.

There are different suspend cases conditions:

- Suspend request accepted prior to the start of data transfer.
- Suspend request not accepted, (due to data being transfered at the same time), the host keeps checking the request until it is accepted (data transfer has suspended).
- Suspend request during write busy.
- Suspend request with write multiple.
- Suspend request during Read Wait.

For the host to know if the bus has been released it must check the status of the suspend request, suspension completed.

When the bus status of the suspend request response indicates suspension completed, the card has released the bus. At this time the state of the suspended operation must be saved where after an other operation can start.

The suspend command must be sent with the CMDSUSPEND bit set. This makes possible to start the interrupt period after the suspend command response when the bus is suspended (response bit BS = 0).

The hardware does not save the number of remaining data to be transfered when resuming the suspended operation. It is up to firmware to determine the data that has been transferred and resume with the correct remaining number of data bytes.

While receiving data from the card, the SDMMC can suspend the read operation after the read data block end (DPSM in Wait_R). After receiving the suspend acknowledgment response from the card the following steps must be taken by firmware:

1. The normal receive process must be stopped by setting DTHOLD bit.
   a) The remaining number of data bytes in the FIFO must be read until the receive FIFO is empty (RXFIFOE flag is set), and when IDMAEN = 0 the FIFO must be reset with FIFORST.
2. The confirmation that all data has been read from the FIFO, and that the suspend is completed is indicated by the DHOLD flag.
   a) The remaining number of data bytes (multiple of data blocks) still to be read when resuming the operation must be determined from the remaining number of bytes indicated by the DATACOUNT.

Note: When a DTIMEOUT flag occurs during the suspend procedure, this must be ignored.

To resume receiving data from the card, the following steps must be taken by firmware:

1. The remaining number of data bytes (multiple of data blocks) must be programmed in DATALENGTH.
2. The DPSM must be configured to receive data in the DTDIR bit.
3. The resume command must be sent from the CPSM, with the CMDTRANS bit set and the CMDSUSPEND bit set, which ends the interrupt period when data transfer is resumed (response bit DF = 1) and enabled the DPSM, after which the card resumes sending data.

While sending data to the card, the SDMMC can suspend the write operation after the write data block CRC status end (DPSM in Busy). Before sending the suspend command to the card the following steps must be taken by firmware:

1. Enable DHOLD flag (and DBCKEND flag when IDMAEN = 0)
2. The DPSM must be prevented from start sending a new data block by setting DTHOLD.
3. When IDMAEN = 0: When receiving the DBCKEND flag the data transfer is stopped. Firmware can stop filling the FIFO, after which the FIFO must be reset with FIFORST. Any bytes still in the FIFO need to be rewritten when resuming the operation.
4. When receiving the DHOLD flag the data transfer is stopped. The remaining number of data bytes still to be written when resuming must be determined from the remaining number of bytes indicated by the DATACOUNT.
5. To suspend the card the suspend command must be sent by the CPSM with the CMDSUSPEND bit set. This makes possible to start the interrupt period after the suspend command response when the bus is suspended (response bit BS = 0).

To resume sending data to the card, the following steps must be taken by firmware:

1. The remaining number of data bytes must be programmed in DATALENGTH.
2. The DPSM must be configured for transmission with DTDIR set and enabled by having the CPSM send the resume command with the CMDTRANS bit set and the CMDSUSPEND bit set. This ends the interrupt period and start the data transfer. The DPSM either goes to the Wait_S state when SDMMC_D0 does not signal busy, or goes to the Busy state when busy is signaled.
3. When IDMAEN = 1: The IDMA needs to be reprogrammed for the remaining bytes to be transfered.
4. When IDMAEN = 0: Firmware must start filling the FIFO with the remaining data.

SD I/O Read Wait

There are two methods to pause the data transfer during the block gap:

1. Stopping the SDMMC_CK.
2. Using Read Wait signaling on SDMMC_D2.

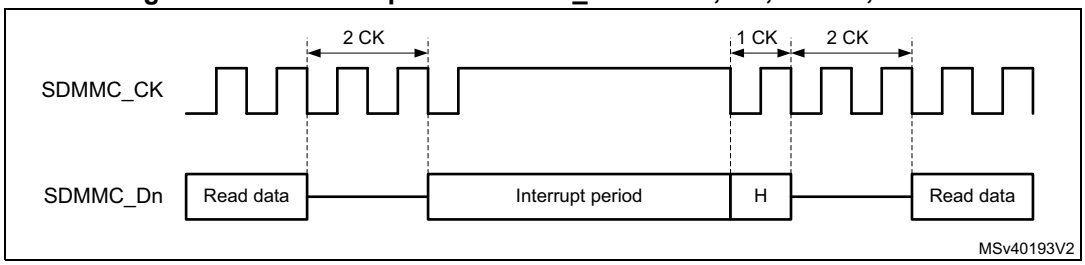
The SDMMC can perform a Read Wait with register settings according *Table 226*.

Read Wait is not available when block gap hardware flow control is enabled. (see *Section 30.7.2: Block gap hardware flow control*)

Depending the SDMMC operation mode (DS, HS, SDR12, SDR25) or (SDR50, SDR104, DDR) each method has a different characteristic.

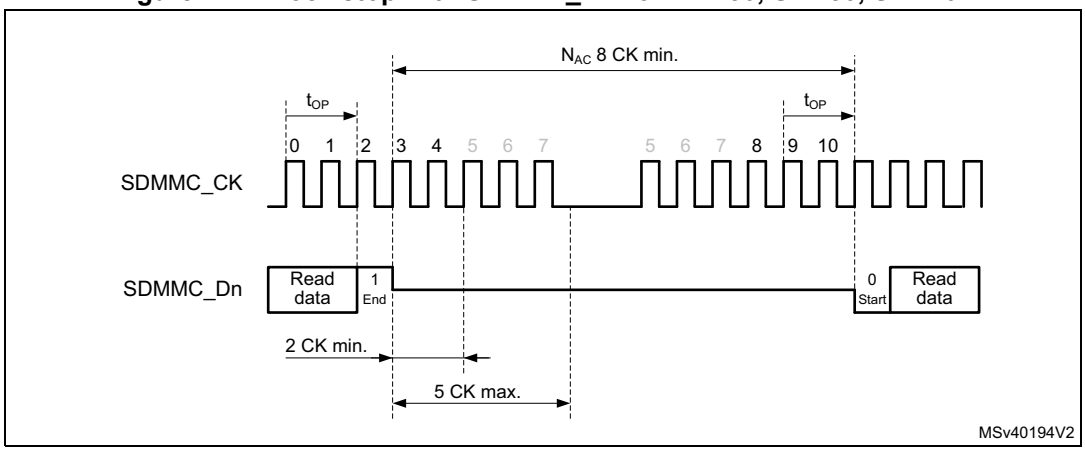
The timing for pause read operation by stopping the SDMMC_CK for DS, HS, SDR12, and SDR25, the SDMMC_CK may be stopped 2 SDMMC_CK cycles after the end bit. When ready the host resumes by restarting clock (see *Figure 246*).

Figure 246. Clock stop with SDMMC_CK for DS, HS, SDR12, SDR25



The timing for pause read operation by stopping the SDMMC_CK for SDR50, SDR104, and DDR50, the SDMMC_CK may be stopped minimum 2 SDMMC_CK cycles and maximum 5 SDMMC_CK cycles, after the end bit. When ready the host resumes by restarting clock, see *Figure 247*. (In DDR50 mode the SDMMC_CK must only be stopped after the falling edge, when the clock line is low.)

Figure 247. Clock stop with SDMMC_CK for DDR50, SDR50, SDR104



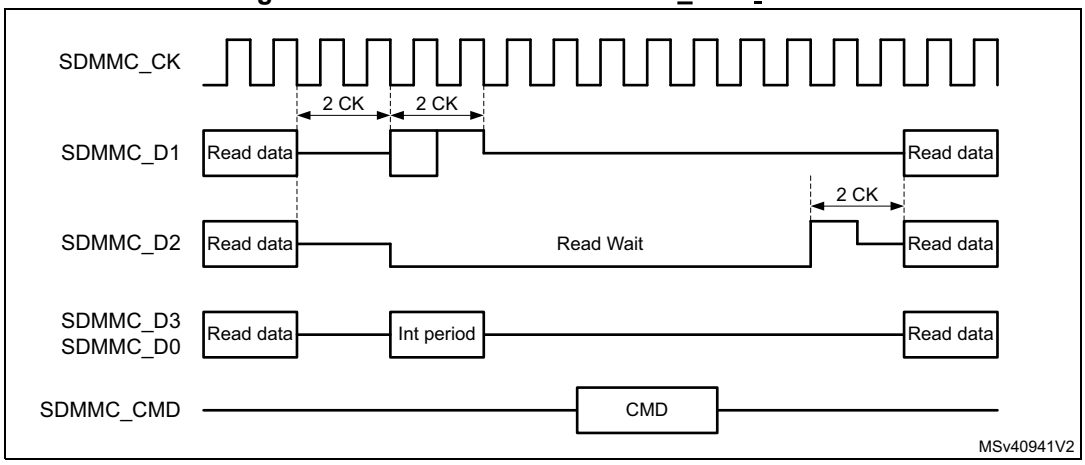
In Read Wait SDMMC_CK clock stopping, when RWSTART is set, the DSPM stops the clock after the end bit of the current received data block CRC. The clock start again after writing 1 to the RWSTOP bit, where after the DPSM waits for a start bit from the card.

As SDMMC_CK is stopped, no command can be issued to the card. During a Read Wait interval, the SDMMC can still detect SDIO interrupts on SDMMC_D1.

The optional Read Wait signaling on SDMMC_D2 (RW) operation is defined only for the SD 1-bit and 4-bit modes. The Read Wait operation enables the host to signal a card that is reading multiple registers (IO_RW_EXTENDED, CMD53) to temporarily stall the data transfer while allowing the host to send commands to any function within the SD I/O device. To determine when a card supports the Read Wait protocol, the host must test capability bits in the internal card registers.

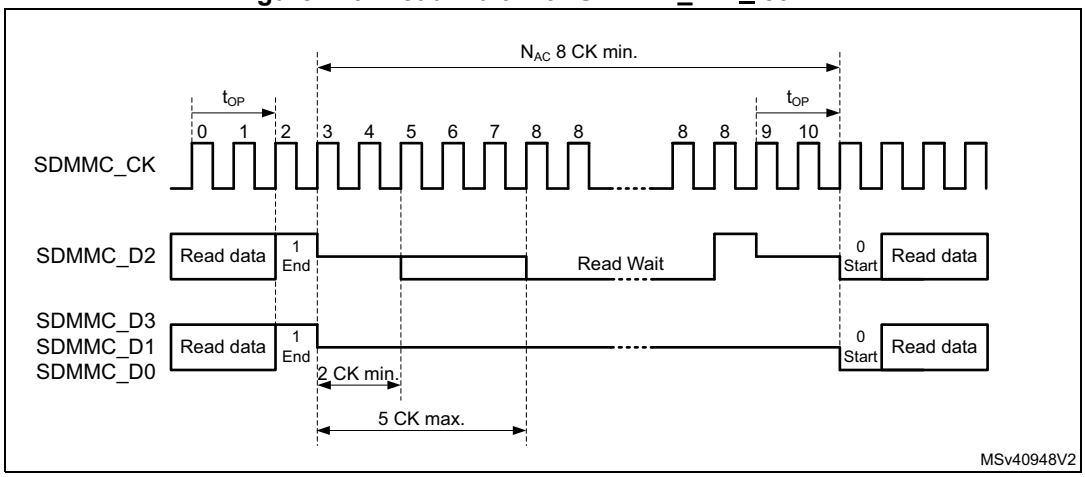
The timing for Read Wait with a SDMMC_CK less then 50MHz (DS, HS, SDR12, SDR25) is based on the interrupt period generated by the card on SDMMC_D1. The host by asserting SDMMC_D2 low during the interrupt period requests the card to enter Read Wait. To exit Read Wait the host must raise SDMMC_D2 high during one SDMMC_CK cycles before making it Hi-Z, see *Figure 248*.

Figure 248. Read Wait with SDMMC_CK < 50 MHz



For SDR50, SDR104 with a SDMMC_CK more than 50MHz, and DDR50, the card treats the Read Wait request on SDMMC_D2 as an asynchronous event. The host by asserting SDMMC_D2 low after minimum 2 SDMMC_CK cycles and maximum 5 SDMMC_CK cycles, request the card to enter Read Wait. To exit Read Wait the host must raise SDMMC_D2 high during one SDMMC_CK cycles before making it Hi-Z. The host must raise SDMMC_D2 on the SDMMC_CK clock (see *Figure 249*).

Figure 249. Read Wait with SDMMC_CK ≥ 50 MHz



In Read Wait SDMMC_D2 signaling, when RWSTART is set, the DPSM drives SDMMC_D2 after the end bit of the current received data block CRC. The Read Wait signaling on SDMMC_D2 is removed when writing 1 to the RWSTOP bit. The DPSM remains in R_W state for two more SDMMC_CK clock cycles to drive SDMMC_D2 to 1 for one clock cycle (in accordance with SDIO specification), where after the DPSM waits for a start bit from the card.

During the Read Wait signaling on SDMMC_D2 commands can be issued to the card. During the Read Wait interval, the SDMMC can detect SDIO interrupts on SDMMC_D1.

30.6.2 CMD12 send timing

CMD12 is used to stop/abort the data transfer, the card data transmission is terminated two clock cycles after the end bit of the Stop Transmission command.

Table 228. CMD12 use cases

Data operation	Stop Transmission command CMD12 Description
SDMMC stream write	The data transfer is stopped/aborted by sending the Stop Transmission command.
SDMMC open ended multiple block write	The data transfer is stopped/aborted by sending the Stop Transmission command. If the card detects an error, the host must abort the operation by sending the Stop Transmission command.
SDMMC block write with predefined block count	The Stop Transmission command is not required at the end of this type of multiple block write. (sending the Stop Transmission command after the card has received the last block is regarded as an illegal command.) If the card detects an error, the host must abort the operation by sending the Stop Transmission command.
SDMMC stream read	The data transfer is stopped/aborted by sending the Stop Transmission command.
SDMMC open ended multiple block read	The data transfer is stopped/aborted by sending the Stop Transmission command. If the card detects an error, the host must abort the operation by sending the Stop Transmission command.
SDMMC block read with predefined block count	The Stop Transmission command is not required at the end of this type of multiple block read. (sending the Stop Transmission command after the card has transmitted the last block is regarded as an illegal command.) Transaction can be aborted by sending the Stop Transmission command. If the card detects an error, the host must abort the operation by sending the Stop Transmission command.

All data write and read commands can be aborted any time by a Stop Transmission command CMD12. The following data abort procedure applies during an ongoing data transfer:

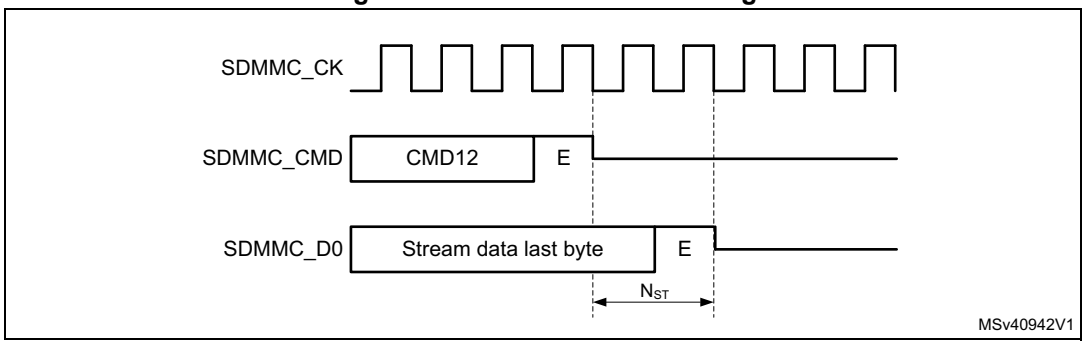
1. Load CMD12 Stop Transmission command in registers and set the CMDSTOP bit.
   – This causes the CPSM Abort signal to be generated when the command is sent to the DPSM.
2. Configure the CPSM to send a command immediately (clear WAITPEND bit).
   – The card, when sending data, stops data transfer 2 cycles after the Stop Transmission command end bit.
   The card when no data is being sent, does not start sending any new data.
   – The host, when sending data, sends one last data bit followed by an end bit after the Stop Transmission command end bit.
   The host when not sending data, does not start sending any new data.
3. When IDMAEN = 0, the FIFO need to be reset with FIFORST.
   – When writing data to the card. On the CMDREND flag, firmware must stop writing data to the FIFO. Subsequently the FIFO must be reset with FIFORST, this flushes the FIFO.
   – When reading data from the card. On the CMDREND flag, firmware must read the remaining data from the FIFO. Subsequently the FIFO must be reset with FIFORST.
4. When IDMAEN = 1, hardware takes care of the FIFO.
   – When writing data to the card. On the CPSM Abort signal, hardware stops the IDMA and subsequently the FIFO is flushed.
   – When reading data from the card. On the CPSM Abort signal, hardware instructs the IDMA to transfer the remaining data from the FIFO to RAM.
5. When the FIFO is empty/reset the DABORT flag is generated.

Stream operation and CMD12

To stop the stream transfer after the last byte to be transfered, the CMD12 end bit timing must be sent aligned with the data stream end of last byte. The following write stream data procedure applies:

1. Initialize the stream data in the DPSM, DTMODE = MCC stream data transfer.
2. Send the WRITE_DATA_STREAM command from the CPSM with CMDTRANS = 1.
3. Preload CMD12 in command registers, with the CMDSTOP bit set.
4. Configure the CPSM to send a command only after a wait pending (WAITPEND = 1) end of last data (according DATALENGTH).
5. Enabling the CPSM to send the STOP_TRANSMISSION command, the stream data end bit and command end bit are aligned.
   – When DATALENGTH > 5 bytes, Command CMD12 is waited in the CPSM to be aligned with the data transfer end bit.
   – When DATALENGHT < 5 bytes, Command CMD12 is started before and the DPSM remains in the Wait_S state to align the data transfer end with the CMD12 end bit.
6. The write stream data can be aborted any time by clearing the WAITPEND bit. This causes the Preloaded CMD12 to be sent immediately and stop the write data stream.

Figure 250. CMD12 stream timing



To stop the read stream transfer after the last byte, the CMD12 end bit timing must occur after the last data stream byte. The following read stream data procedure applies:

1. Wait for all data to be received by the DPSM and read from the FIFO (DATAEND flag).
   – The DPSM does not receive more data than indicated by DATALENGTH, even if the card is sending more data.
2. Send CMD12 by the CPSM.
   – CMD12 stops the card sending data.

Note: The SDMMC does not receive any more data from the card when DATACOUNT = 0, even when the card continues sending data.

Block operation and CMD12

To stop block transfer at the end of the data, the CMD12 end bit must be sent after the last block end bit.

When writing data to the card the CMD12 end bit must be sent after the write data block CRC token end bit. This requires the CMD12 sending to be tied to the data block transmission timing. To stop an Open-ended Multiple block write, the following procedure applies:

1. Before starting the data transfer, set DTMODE to “block data transfer ending with STOP_TRANSMISSION command”.
2. Wait for all data to be sent by the DPSM and the CRC token to be received, (DATAEND flag).
   – The DPSM does not send more data than indicated by DATALENGTH.
3. Send CMD12 by the CPSM.
   – CMD12 sets the card to Idle mode.

When reading data from the card the CMD12 end bit must be sent earliest at the same time as the card read data block last data bit. This requires the CMD12 sending to be tied to the data block reception timing. The following stop Open-ended Multiple block read data block procedure applies:

1. Before starting the data transfer, set DTMODE to “block data transfer ending with STOP_TRANSMISSION command”.
2. Wait for all data to be received by the DPSM and read from the FIFO (DATAEND flag).
   – The DPSM does not receive more data than indicated by DATALENGTH, even if the card is sending more data.
3. Send CMD12 with CMDSTOP bit set by the CPSM.
   – CMD12 stops the Card sending more data and set the card to Idle mode. Any ongoing block transfer is aborted by the Card.

Note: The SDMMC does not receive any more data from the card when DATACOUNT = 0, even when the card continues sending data.

30.6.3 Sleep (CMD5)

The *e*•MMC card may be switched between a Sleep state and a Standby state by CMD5. In the Sleep state the power consumption of the card is minimized and the Vcc power supply may be switched off.

The CMD5 (SLEEP) is used to initiate the state transition from Standby state to Sleep state. The card indicates Busy, pulling down SDMMC_D0, during the transition phase. The Sleep state is reached when the card stops pulling down the SDMMC_DO line.

To set the card into Sleep state the following procedure applies:

1. Enable interrupt on BUSYD0END.
2. Send CMD5 (SLEEP).
3. On BUSYD0END interrupt, card is in Sleep state.
4. Vcc power supply can be switched off.

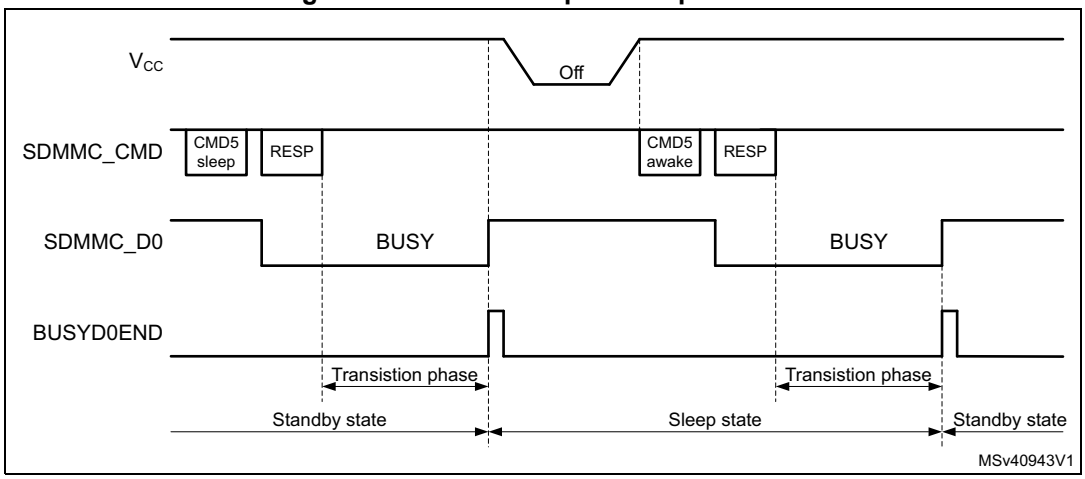
The CMD5 (AWAKE) is used to initiate the state transition from Sleep state to Standby state. The card indicates Busy, pulling down SDMMC_D0, during the transition phase. The Standby state is reached when the card stops pulling down the SDMMC_DO line.

To set the card into Sleep state the following procedure applies:

1. Switch on Vcc power supply and wait unit minimum operating level is reached.
2. Enable interrupt on BUSYD0END.
3. Send CMD5 (AWAKE).
4. On BUSYD0END interrupt card is in Standby state.

The Vcc power supply can be switched off only after the Sleep state has been reached. The Vcc supply must be reinstalled before CMD5 (AWAKE) is sent.

Figure 251. CMD5 Sleep Awake procedure



30.6.4 Interrupt mode (Wait-IRQ)

The host and card enter and exit interrupt mode (Wait-IRQ) simultaneously. In interrupt mode there is no data transfer. The only message allowed is an interrupt service request response from the card or the host. For the interrupt mode to work correctly the SDMMC_CK frequency must be set in accordance with the achievable SDMMC_CMD data rate in Open Drain mode, which depend on the capacitive load and pull-up resistor. The CLKDIV must be set >1, and the SETCLKRX must select either the sdmmc_io_in_ck or SDMMC_CLKin source.

The host must ensure that the card is in Standby state before issuing the CMD40 (GO_IRQ_STATE). While waiting for an interrupt response the SDMMC_CK clock signal must be kept active.

A card in interrupt mode (IRQ state):

- is waiting for an internal card interrupt event. Once the event occurs, the card starts to send the interrupt service request response. The response is sent in open-drain mode.
- while waiting for the internal card interrupt event, the card also monitors the SDMMC_CMD line for a start bit. Upon detection of a start bit the card aborts the interrupt mode and switch to Standby state.

The host in interrupt mode (CPSM Wait state waiting for interrupt):

- is waiting for a card interrupt service request response (start bit).
- while waiting for a card interrupt service request response the host may abort the interrupt mode (by clearing the WAITINT register bit), which causes the host to send a interrupt service request response R5 with RCA = 0x0000 in open-drain mode.

When sending the interrupt service request response, the sender bit-wise monitors the SDMMC_CMD bit stream. The sender whose interrupt service request response bit does not correspond to the bit on the SDMMC_CMD line stops sending. In the case of multiple senders only one successfully sends its full interrupt service request response. If the host sends simultaneously, it loses sending after the transmission bit.

To handle the interrupt mode, the following procedure applies:

1. Set the SDMMC_CK frequency in accordance with the achievable SDMMC_CMD data rate in Open-drain mode, CLKDIV must be set >1, and SETCLKRX must select the sdmmc_io_in_ck.
2. Load CMD40 (GO_IRQ_STATE) in the command registers.
3. Enable wait for interrupt by setting WAITINT register bit.
4. Configure the CPSM to send a command immediately.
   – This causes the CMD40 to be sent and the CPSM to be halted in the Wait state, waiting for a interrupt service request response.
5. To exit the wait for interrupt state (CPSM Wait state):
   – Upon the detection of an interrupt service request response start bit the CPSM moves to the Receive state where the response is received. The complete reception of the response is indicated by the CMDREND or the command CRC error flags.
   – To abort the interrupt mode the host clears the WAITINT register bit, which causes the host to send an interrupt service request response by itself. This moves the CPSM to the Receive state. The complete reception of the response is indicated by the CMDREND or the command CRC error flags.

Note: On a simultaneous send interrupt service request response start bit collision the host loses the bus access after the transmission bit.

30.6.5 Boot operation

In boot operation mode the host can read boot data from the card by either one of the two boot operation functions:

- Normal boot. (keeping CMD line low)
- Alternative boot (sending CMD0 with argument 0xFFFFFFFA)

The boot data can be read according the following configuration options, depending on card register settings:

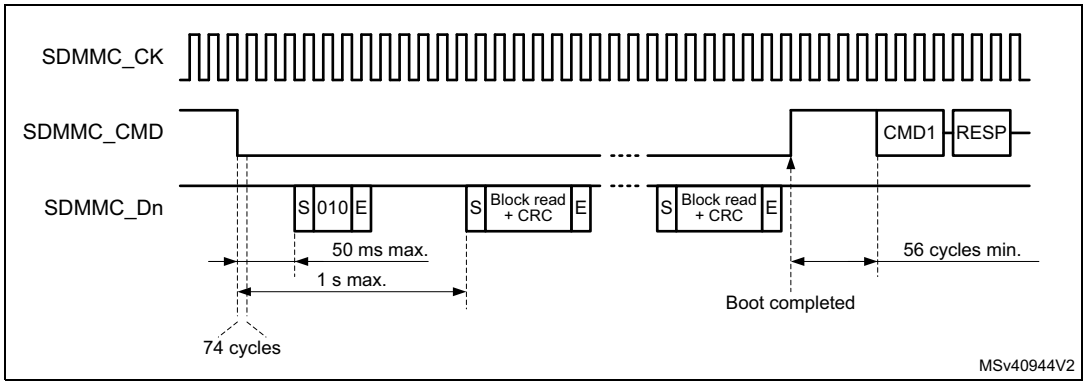
- The partition from which boot data is read (EXT_CSD Byte[179])
- The boot data size (EXT_CSD Byte[226])
- The bus configuration during boot (EXT_CSD Byte[177])
- Receiving boot acknowledgment from the card. (EXT_CSD Byte[179])

If boot acknowledgment is enabled the card send pattern 010 on SDMMC_D0 within 50ms after boot mode has been requested by either CMD line going low or after CMD0 with argument 0xFFFFFFFA. A boot acknowledgment timeout (ACKTIMEOUT) and acknowledgment status (ACKFAIL) is provided.

Normal boot operation

If the SDMMC_CMD line is held low for at least 74 clock cycles after card power-up or reset, before the first command is issued, the card recognizes that boot mode is being initiated. Within 1 second after the CMD line goes low, the card starts to sent the first boot code data on the SDMMC_Dn line(s). The host must keep the SDMMC_CMD line low until after all boot data has been read. The host can terminate boot mode by pulling the SDMMC_CMD line high.

Figure 252. Normal boot mode operation



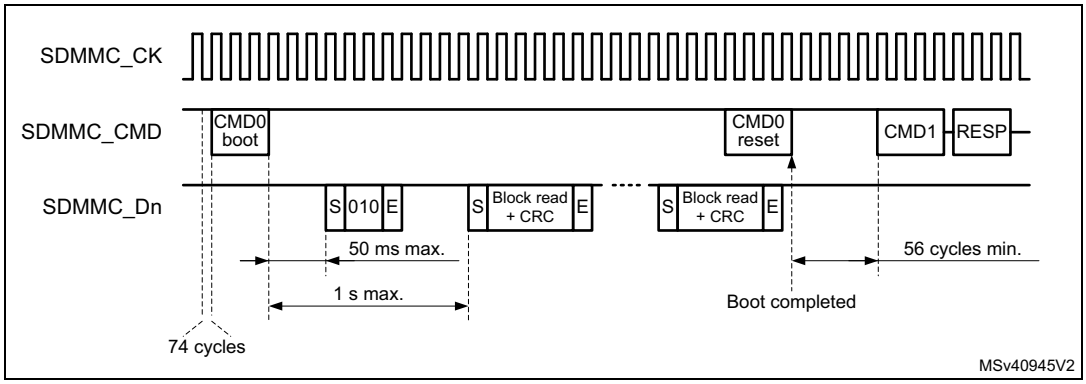
To perform the normal boot procedure the following steps needed:

1. Reset the card.
2. if a boot acknowledgment is requested enable the BOOTACKEN and set the ACKTIME and enable the ACKFAIL and ACKTIMEOUT interrupt.
3. enable the data reception by setting the DPSM in receive mode (DTDIR) and the number of data bytes to be received in DATALENGTH.
4. Enable the DTIMEOUT, DATAEND, and CMDSENT interrupts for end of boot command confirmation.
5. Select the normal boot operation mode in BOOTMODE, and enable boot in BOOTEN. The boot procedure is started by enabling the CPSM with CPSMEN.This causes:
   - the SDMMC_CMD to be driven low. (BOOTMODE = normal boot).
   - the ACK timeout to start.
   - DPSM to be enabled.
6. The incorrect reception of the boot acknowledgment can be detected with ACKFAIL flag or ACKTIMEOUT flag when enabled.
   - when an incorrect boot acknowledgment is received the ACKFAIL flag occurs.
   - when the boot acknowledgment is not received in time the ACKTIMEOUT flag occurs.
7. when all boot data has been received the DATAEND flag occurs.
   - when data CRC fails the DCRCFAIL flag is also generated.
   - when the data timeout occurs the DTIMEOUT flag is also generated.
8. When last data has been received, read data from the FIFO until FIFO is empty after which end of data DATAEND flag is generated.
   - SDMMC has completely received all data and the DPSM is disabled.
9. The boot procedure is terminated by firmware clearing BOOTEN, which causes the SDMMC_CMD line to go high. The CMDSENT flag is generated 56 cycles later to indicate that a new command can be sent.
   - If the boot procedure is aborted by firmware before all data has been received the CPSM Abort signal stops data reception and disables the DPSM which triggers an DABORT flag when enabled.
10. The CMDSENT flag signals the end of the boot procedure and the card is ready to receive a new command.

Alternative boot operation

After card power-up or reset, if the host send CMD0 with the argument 0xFFFFFFFA after 74 clock cycles before CMD0 is issued, the card recognizes that boot mode is being initiated. Within 1 second after the CMD0 with argument 0xFFFFFFFA has been sent, the card starts to send the first boot code data on the SDMMC_Dn line(s). The master terminates boot operation by sending CMD0 (Reset).

Figure 253. Alternative boot mode operation



To perform the alternative boot procedure the following steps needed:

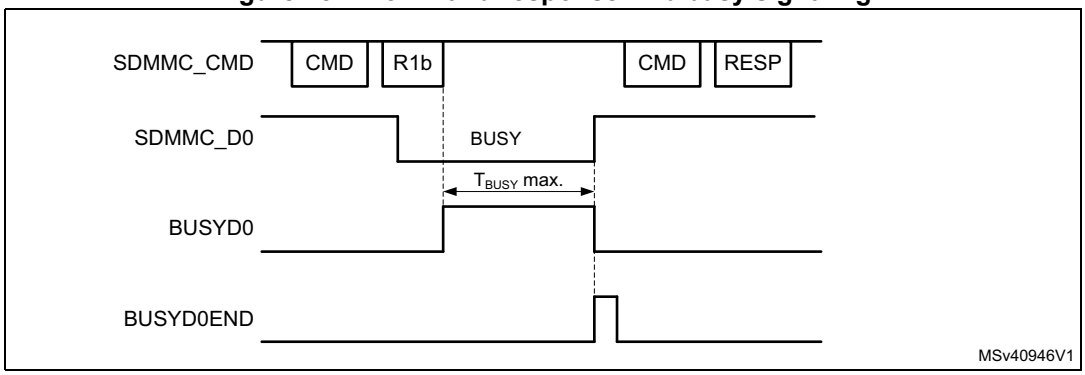
1. Move the SDMMC to power-off state, and reset the card.
2. Move the SDMMC to power-on state. This guarantees the 74 SCDMMC_CK cycles to be clocked before any command.
3. if a boot acknowledgment is requested enable the BOOTACKEN and set the ACKTIME and enable the ACKTIMEOUT flag.
4. enable the data reception by setting the DPSM in receive mode (DTDIR) and the number of data to be received in DATALENGTH. Enable the DTIMEOUT and DATAEND flags.
5. Select the alternative boot operation mode in BOOTMODE, load the CMD0 with the 0xFFFFFFFA argument in the command registers. Enable CMDSENT flag for end of boot command confirmation, and enable boot in BOOTEN. The boot procedure is started by enabling the CPSM with CPSMEN. This causes:
   - the loaded command and argument to be sent out. (BOOTMODE = alternative boot).
   - the ACK timeout to start.
   - DPSM to be enabled.
6. When the command has been sent the CMDSENT flag is generated, at which time the BOOTEN bit must be cleared.
7. the reception of the boot acknowledgment can be detected with ACKFAIL flag when enabled.
   - when the boot acknowledgment is not received in time the ACKTIMEOUT flag occurs.
8. when all boot data has been received the DATAEND flag occurs.
   - when data CRC fails the DCRCFAIL flag is also generated.
   - when the data timeout occurs the DTIMEOUT flag is also generated.

9. When last data has been received, read data from the FIFO until FIFO is empty after which end of data DATAEND flag is generated.
   – SDMMC has completely received all data and the DPSM is disabled.
10. The BOOTEN bit must be cleared, before terminating the boot procedure by sending CMD0 (Reset) with BOOTMODE = alternative boot. This causes the CMDSENT flag to occur 56 cycles after the Command.
   – if the boot procedure is aborted by firmware before all data has been received the CPSM Abort signal stops the data transfer and disable the DPSM which triggers an DABORT flag when enabled.
11. The CMDSENT flag signals the end of the boot procedure and the card is ready to receive a new command. When the RESET command has been sent successfully, the BOOTMODE control bit has to be cleared to terminate the boot operation.

30.6.6 Response R1b handling

When sending commands which have a R1b response the busy signaling is reflected in the BUSYD0 register bit and the release of busy with the BUSYD0END flag. The SDMMC_D0 line is sampled at the end of the R1b response and signaled in the BUSYD0 register bit. The BUSYD0 register bit is reset to not busy when the SDMMC_D0 line release busy, at the same time the BUSYD0END flag is generated.

Figure 254. Command response R1b busy signaling



The expected maximum busy time must be set in the DATATIME register before sending the command. When enabled, the DTIMEOUT flag is set when after the R1b response busy stays active longer then the programmed time.

To detect the SDMMC_D0 busy signaling when sending a Command with R1b response the following procedure applies:

- Enable CMDREND flag.
- Send Command through CPSM.
- On the CMDREND flag check the BUSYD0 register bit.
  – If BUSYD0 signals not busy, signal busy release to firmware
  – If BUSYD0 signals busy, wait for BUSYD0END flag
- On BUSYD0END flag signal busy released to firmware.
- On DTIMEOUT flag busy is active longer then programmed time.

30.6.7 Reset and card cycle power

Reset

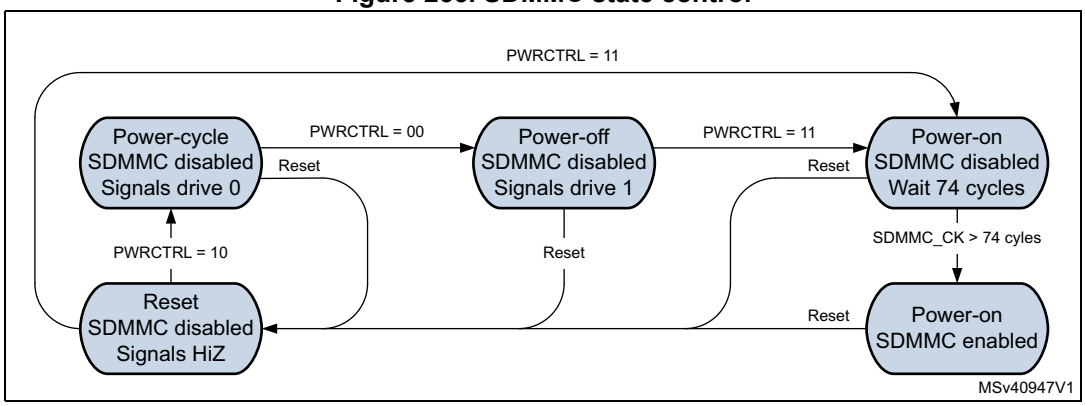
Following reset the SDMMC is in the reset state. In this state the SDMMC is disabled and no command nor data can be transfered. The SDMMC_D[7:0], and SDMMC_CMD are in HiZ and the SDMMC_CK is driven low.

Before moving to the power-on state the SDMMC must be configured.

In the power-on state the SDMMC_CK clock is running. First 74 SDMMC_CK cycles are clocked after which the SDMMC is enabled and command and data can be transfered.

The SDMMC states are controlled by Firmware with the PWRCTL register bits according *Figure 255.*.

Figure 255. SDMMC state control

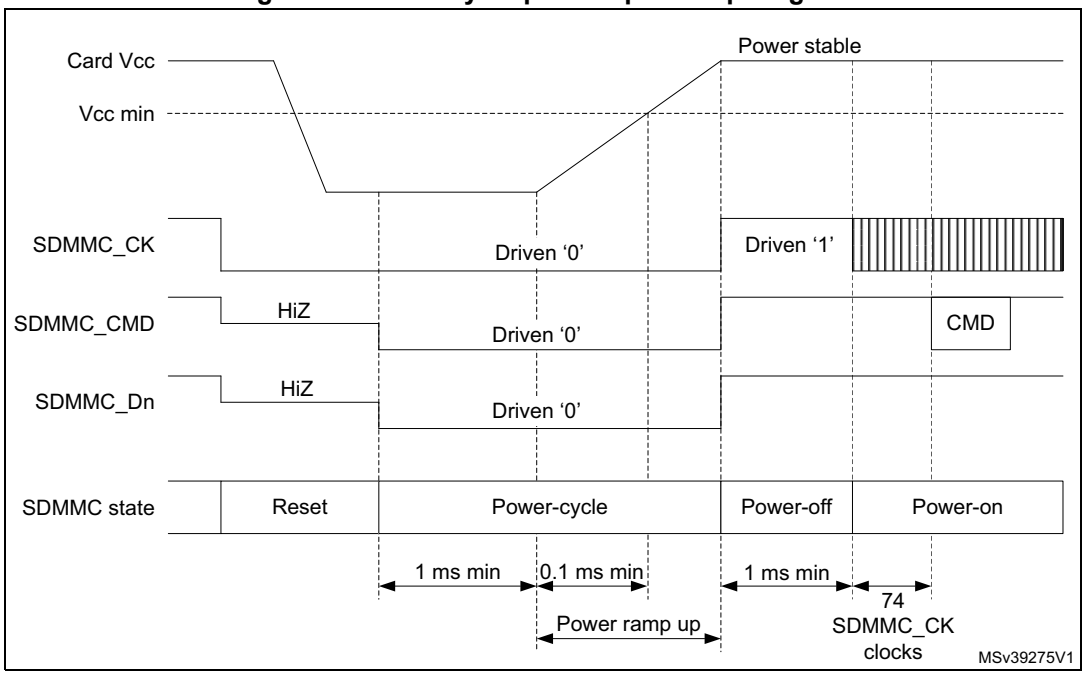


Card cycle power

To perform a card cycle power the following procedure applies:

1. Reset the SDMMC with the RCC.SDMMCxRST register bit. This resets the SDMMC to the reset state and the CPSM and DPSM to the Idle state.
2. Disable the Vcc power to the card.
3. Set the SDMMC in power-cycle state. This makes that the SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are driven low, to prevent the card from being supplied through the signal lines.
4. After minimum 1 ms enable the Vcc power to the card.
5. After the power ramp period set the SDMMC to the power-off state for minimum 1 ms. The SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are set to drive 1.
6. After the 1 ms delay set the SDMMC to power-on state in which the SDMMC_CK clock is enabled.
7. After 74 SDMMC_CK cycles the first command can be sent to the card.

Figure 256. Card cycle power / power up diagram



30.7 Hardware flow control

There are two hardware flow control mechanisms:

1. Hardware flow control during data transfer:
   - Only to be used with cycle-aligned data transfers.
2. Block gap hardware flow control:
   - Only used in mode with bus speed SDR50, SDR104, HS200 and DDR50.
   - Must be used with variable delay.

30.7.1 Hardware flow control during data transfer

The hardware flow control during data transfer functionality is used to avoid FIFO underrun (TX mode) and overrun (RX mode) errors.

The behavior is to stop SDMMC_CK during data transfer and freeze the SDMMC state machines. The data transfer is stalled when the FIFO is unable to transmit or receive data. The data transfer remains stalled until the transmit FIFO is half full or all data according DATALENGHT has been stored, or until the receive FIFO is half empty. Only state machines clocked by SDMMC_CK are frozen, the AHB interfaces are still alive. The FIFO can thus be filled or emptied even if flow control is activated.

On an IDMA linked list transfer error, the hardware flow control is disabled. As a consequence, depending on when the IDMA linked list transfer error occurs, an underrun or overrun error may also occur (see *Section : IDMA transfer error management*).

To enable hardware flow control during data transfer, the HWFC_EN register bit must be set to 1. After reset hardware flow control is disabled.

Hardware flow control during data transfer must only be used when the SDMMC_Dn data is cycle-aligned with the SDMMC_CK. Whenever the sdmmc_fb_ck from the DLYB delay block is used, i.e in the case of SDR104 mode with a t_{OP} and Dt_{OP} delay > 1 cycle, hardware flow control during data transfer can not be used and block gap hardware flow control must be used instead.

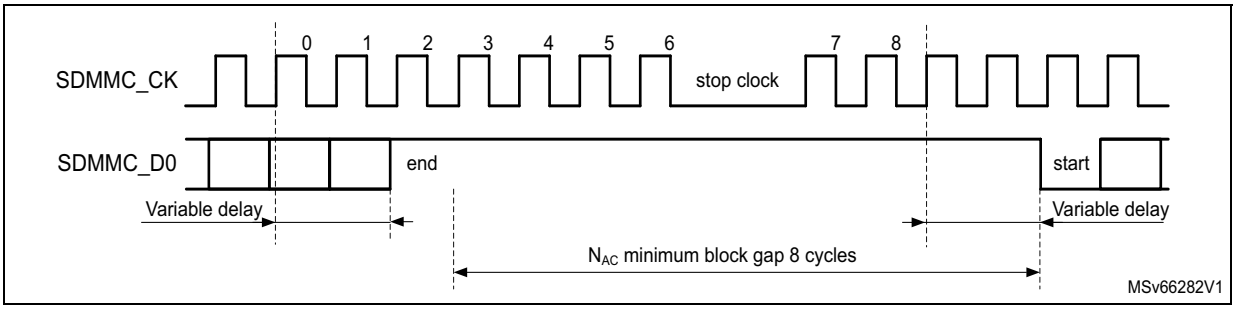
30.7.2 Block gap hardware flow control

The block gap hardware flow control functionality is used to avoid FIFO underrun (TX mode) and overrun (RX mode) errors in SDR50, SDR104, HS200 and DDR50 mode (BUSSPEED = 1).

When transmitting, the behavior is to delay the start of a new write block and freeze the SDMMC state machines. The start of a new write block is delayed until the FIFO holds enough data according to the threshold. Only the state machines clocked by SDMMC_CK are frozen, the AHB interfaces are still alive. The FIFO can thus be filled even if flow control is activated.

When receiving, the behavior is to stop SDMMC_CK during the read block gap and freeze the SDMMC state machines. The start of a new block transfer is stalled, due to stopped clock, until the FIFO has enough free space according to the threshold. Only the state machines clocked by SDMMC_CK are frozen, the AHB interfaces are still alive. The FIFO can thus be emptied even if flow control is activated.

Figure 257. Read block gap hardware flow timing



Warning: When variable delay is used and block gap hardware flow control is enabled, sending commands during multiple block read may disturb the command response reception (CMD12 is not impacted by this).

Table 229. Data block gap hardware flow control behavior

Condition (HWFC_EN = 1 and BUSSPEED = 1)	Description
Data block size ≤ FIFO threshold	New block transfer is only started when the FIFO has enough data (TX mode) / free space (RX mode) up to or more than the THR threshold. *Note: Data flow control is guaranteed. A block transfer is only stared when enough data for at least one block is available / free in the FIFO*
Data block size > FIFO threshold	New block transfer is only started when the FIFO has enough data (TX mode) / free space (RX mode) up to or more than the THR threshold. *Note: Data flow control is not guaranteed. A block transfer is started even when not enough data for one block is available / free in the FIFO. The threshold can be used to trade off transfer latency / throughput versus FIFO underrun / overrun risk.*

On an IDMA linked list transfer error, the hardware flow control is disabled. As a consequence, depending on when the IDMA linked list transfer error occurs, an underrun or overrun error may also occur (see *Section : IDMA transfer error management*).

The block gap hardware flow control is only enabled when HWFC_EN is set to 1, BUSSPEED = 1 and the THR FIFO threshold is different than 0. When BUSSPEED = 0 or FIFO threshold is set to 0, the hardware flow control is used during the data transfer. After reset, the hardware flow control is disabled.

Table 230. Hardware flow control selection

<table>
<tr><th>HWFC_EN</th><th>THR</th><th>BUSSPEED</th><th>Description</th></tr>
<tr><td>0</td><td>X</td><td rowspan="2">X</td><td>Hardware flow control disabled</td></tr>
<tr><td rowspan="3">1</td><td>0</td><td rowspan="2">Hardware flow control during data transfer enabled</td></tr>
<tr><td rowspan="2">not 0</td><td>0</td></tr>
<tr><td>1</td><td>Block gap hardware flow control enabled</td></tr>
</table>

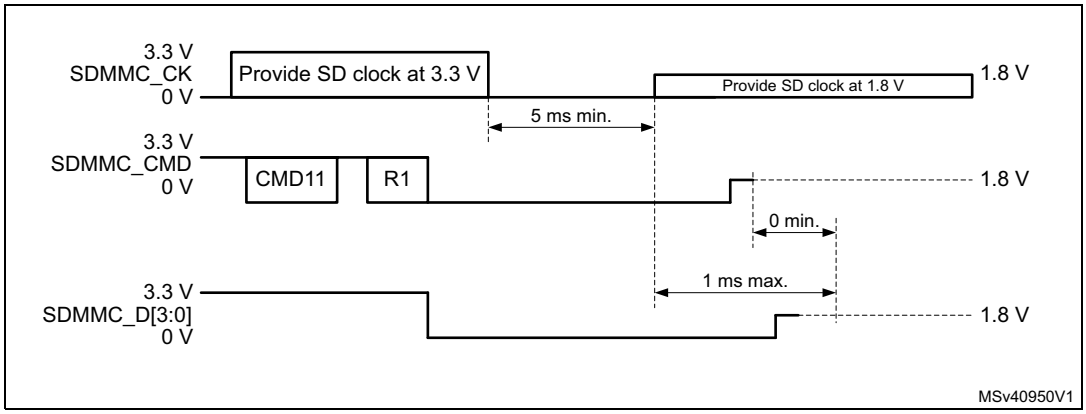
Block gap hardware flow control must be used when the SDMMC_Dn receive data are not cycle-aligned with the SDMMC_CK. This is whenever the sdmmc_fb_ck from the DLYB delay block is used, for example in the case of SDR104 receive mode with a t_{OP} and Dt_{OP} delay > 1 cycle.

30.8 Ultra-high-speed phase I (UHS-I) voltage switch

UHS-I mode (SDR12, SDR25, SDR50, SDR104, and DDR50) requires the support for 1.8 V signaling. After power up the card starts in 3.3V mode. CMD11 invokes the voltage switch

sequence to the 1.8V mode. When the voltage sequence is completed successfully the card enters UHS-I mode with default SDR12 and card input and output timings are changed.

Figure 258. CMD11 signal voltage switch sequence



To perform the signal voltage switch sequence the following steps are needed:

1. Before starting the Voltage Switch procedure, the SDMMC_CK frequency must be set in the range 100 kHz - 400 kHz.
2. The host starts the Voltage Switch procedure by setting the VSWITCHEN bit before sending the CMD11.
3. The card returns an R1 response.
   - if the response CRC is pass, the Voltage Switch procedure continues the host does no longer drive the CMD and SDMMC_D[3:0] signals until completion of the voltage switch sequence. Some cycles after the response the SDMMC_CK is stopped and the CKSTOP flag is set.
   - if the response CRC is fail (CCRCFAIL flag) or no response is received before the timeout (CTIMEOUT flag), the Voltage Switch procedure is stopped.
4. The card drives CMD and SDMMC_D[3:0] to low at the next clock after the R1 response.
5. The host, after having received the R1 response, may monitor the SDMMC_D0 line using the BUSYD0 register bit. The SDMMC_D0 line is sampled two SDMMC_CK clock cycles after the Response. The Firmware may read the BUSYD0 register bit following the CKSTOP flag.
   - When the BUSYD0 is detected low the host firmware switches the Voltage regulator to 1.8V, after which it instructs the SDMMC to start the timing critical section of the Voltage Switch sequence by setting register bit VSWITCH. The hardware continues to stop the SDMMC_CK by holding it low for at least 5 ms.
   - When the BUSYD0 is detected high the host aborts the Voltage Switch sequence and cycle power the card.
6. The card after detecting SDMMC_CK low begins switching signaling voltage to 1.8 V.
7. The host SDMMC hardware after at least 5 ms restarts the SDMMC_CK.
8. The card within 1 ms from detecting SDMMC_CK transition drives CMD and DAT[3:0] high for at least 1 SDMMC_CK cycle and then stop driving CMD and DAT[3:0].
9. The host SDMMC hardware, 1 ms after the SDMMC_CK has been restarted, the SDMMC_D0 is sampled into BUSYD0 and the VSWEND flag is set.

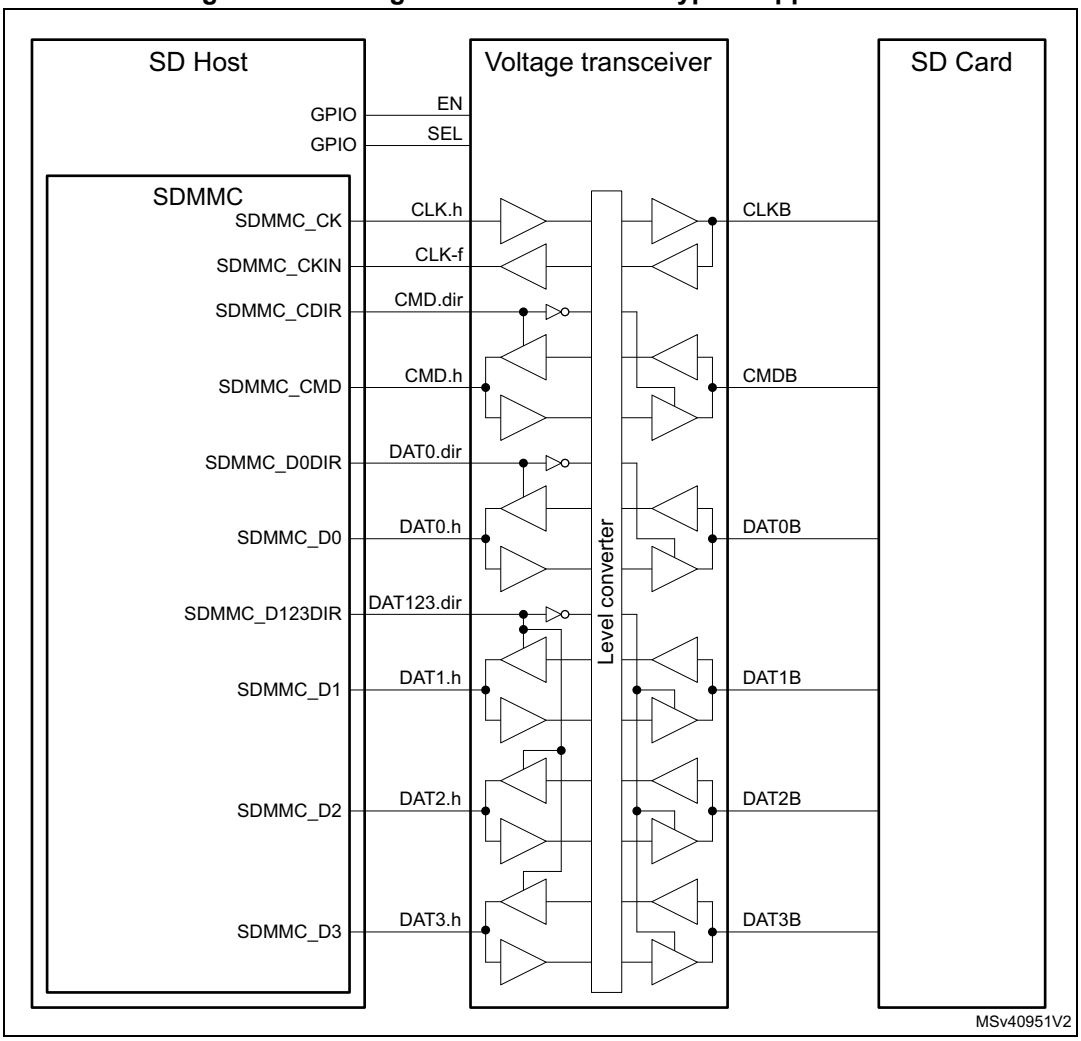
10. The host, on the VSWEND flag, checks SDMMC_D0 line using the BUSYD0 register bit, to confirm completion of voltage switch sequence:
    – When BUSYD0 is detected high, Voltage Switch has been completed successfully.
    – When BUSYD0 is detected low, Voltage Switch has failed, the host cycles the card power.

The minimum 5 ms time to stop the SDMMC_CK is derived from the internal un-gated SDMMC_CK clock, which has a maximum frequency of 25 MHz (SD mode), as set by the clock divider CLKDIV. The >5 ms time is counted by 2^{12} cycles (10.24 ms @ 400 kHz). If a lower SDMMC_CK frequency is selected by the clock divider CLKDIV the time for the SDMMC_CK clock to be stopped is longer.

The maximum 1 ms time for the card to drive the SDMMC_Dn and SDMMC_CMD lines high is derived from the internal ungated SDMMC_CK which has a maximum frequency of 25 MHz (SD mode), as set by the clock divider CLKDIV. The SDMMC checks the lines after >1 ms time which is counted by 2^9 cycles (1.28 ms @ 25 MHz). If a lower SDMMC_CK frequency is selected by the clock divider CLKDIV the time to check the lines is longer.

The signal voltage level is supported through an external voltage translation transceiver like STMicroelectronics ST6G3244ME.

Figure 259. Voltage switch transceiver typical application



To interface with an external driver (a voltage switch transceiver), next to the standard signals the SDMMC uses the following signals:

- **SDMMC_CKIN** feedback input clock
- **SDMMC_CDIR** I/O direction control for the CMD signal
- **SDMMC_D0DIR** I/O direction control for the SDMMC_D0 signal
- **SDMMC_D123DIR** I/O direction control for the SDMMC_D1, SDMMC_D2 and SDMMC_D3 signals

The voltage transceiver signals **EN** and **SEL** are to be handled through general-purpose I/O.

The polarity of the SDMMC_CDIR, SDMMC_D0DIR and SDMMC_D123DIR signals can be selected through SDMMC_POWER.DIRPOL control bit.

30.9 SDMMC interrupts

Table 231. SDMMC interrupts

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode
SDMMC	Command response CRC fail	CCRCFAIL	CCRCFAILIE	CCRCFAILC	Yes
SDMMC	Data block CRC fail	DCRCFAIL	DCRCFAILIE	DCRCFAILC	Yes
SDMMC	Command response timeout	CTIMEOUT	CTIMEOUTIE	CTIMEOUTC	Yes
SDMMC	Data timeout	DTIMEOUT	DTIMEOUTIE	DTIMEOUTC	Yes
SDMMC	Transmit FIFO underrun	TXUNDERR	TXUNDERRIE	TXUNDERRC	Yes
SDMMC	Receive FIFO overrun	RXOVERR	RXOVERRIE	RXOVERRC	Yes
SDMMC	Command reponse received	CMDREND	CMDRENDIE	CMDRENDC	Yes
SDMMC	Comand sent	CMDSENT	CMDSENTIE	CMDSENTC	Yes
SDMMC	Data transfer ended	DATAEND	DATAENDIE	DATAENDC	Yes
SDMMC	Data transfer hold	DHOLD	DHOLDIE	DHOLDC	Yes
SDMMC	Data block sent or received	DBCKEND	DBCKENDIE	DBCKENDC	Yes
SDMMC	Data transfer aborted	DABORT	DABORTIE	DABORTC	Yes
SDMMC	Transmit FIFO half empty	TXFIFOHE	TXFIFOHEIE	N/A	Yes
SDMMC	Receive FIFO half full	RXFIFOHF	RXFIFOHFIE	N/A	Yes
SDMMC	Transmit FIFO full	TXFIFOF	N/A	N/A	Yes
SDMMC	Receive FIFO full	RXFIFOF	RXFIFOFIE	N/A	Yes
SDMMC	Transmit FIFO empty	TXFIFOE	TXFIFOEIE	N/A	Yes
SDMMC	Receive FIFO empty	RXFIFOE	N/A	N/A	Yes
SDMMC	Command response end of busy	BUSYD0END	BUSYD0ENDIE	BUSYD0ENDC	Yes
SDMMC	SDIO interrupt	SDIOIT	SDIOITIE	SDIOITC	Yes
SDMMC	Boot acknowledgment fail	ACKFAIL	ACKFAILIE	ACKFAILC	Yes
SDMMC	Boot acknowledgment timeout	ACKTIMEOUT	ACKTIMEOUTIE	ACKTIMEOUTC	Yes
SDMMC	Voltage switch timing	VSWEND	VSWENDIE	VSWENDC	Yes
SDMMC	SDMM_CK stopped in voltage switch	CKSTOP	CKSTOPIE	CKSTOPC	Yes
SDMMC	IDMA transfer error	IDMATE	IDMATEIE	IDMATEC	Yes
SDMMC	IDMA buffer transfer complete	IDMABTC	IDMABTCIE	IDMABTCC	Yes

30.10 SDMMC registers

The device communicates to the system via 32-bit control registers accessible via AHB slave interface.

The peripheral registers have to be accessed by words (32-bit). Byte (8-bit) and half-word (16-bit) accesses trigger an AHB bus error.

30.10.1 SDMMC power control register (SDMMC_POWER)

Address offset: 0x000

Reset value: 0x0000 0000

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

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

Bit 4 **DIRPOL**: Data and command direction signals polarity selection

This bit can only be written when the SDMMC is in the power-off state (PWRCTRL = 00).

0: Voltage transceiver I/Os driven as output when direction signal is low.

1: Voltage transceiver I/Os driven as output when direction signal is high.

Bit 3 **VSWITCHEN**: Voltage switch procedure enable

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).

This bit is used to stop the SDMMC_CK after the voltage switch command response:

0: SDMMC_CK clock kept unchanged after successfully received command response.

1: SDMMC_CK clock stopped after successfully received command response.

Bit 2 **VSWITCH**: Voltage switch sequence start

This bit is used to start the timing critical section of the voltage switch sequence:

0: Voltage switch sequence not started and not active.

1: Voltage switch sequence started or active.

Bits 1:0 **PWRCTRL[1:0]**: SDMMC state control bits

These bits can only be written when the SDMMC is not in the power-on state (PWRCTRL ≠ 11).

These bits are used to define the functional state of the SDMMC signals:

00: After reset, Reset: the SDMMC is disabled and the clock to the Card is stopped, SDMMC_D[7:0], and SDMMC_CMD are HiZ and SDMMC_CK is driven low. When written 00, power-off: the SDMMC is disabled and the clock to the card is stopped, SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are driven high.

01: Reserved (When written 01, PWRCTRL value does not change)

10: Power-cycle, the SDMMC is disabled and the clock to the card is stopped, SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are driven low.

11: Power-on: the card is clocked, The first 74 SDMMC_CK cycles the SDMMC is still disabled. After the 74 cycles the SDMMC is enabled and the SDMMC_D[7:0], SDMMC_CMD and SDMMC_CK are controlled according the SDMMC operation.

Any further write is ignored, PWRCTRL value keeps 11.

30.10.2 SDMMC clock control register (SDMMC_CLKCR)

Address offset: 0x004

Reset value: 0x0000 0000

This register controls the SDMMC_CK output clock, the sdmmc_rx_ck receive clock, and the bus width.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELCLKRX[1:0]		BUS SPEED	DDR	HWFC_EN	NEG EDGE
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WID BUS[1:0]		Res.	PWR SAV	Res.	Res.	CLKDIV[9:0]									
rw	rw		rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

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

Bits 21:20 **SELCLKRX[1:0]**: Receive clock selection

These bits can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

00: sdmmc_io_in_ck selected as receive clock

01: SDMMC_CKIN feedback clock selected as receive clock

10: sdmmc_fb_ck tuned feedback clock selected as receive clock

11: Reserved (select sdmmc_io_in_ck)

Bit 19 **BUSSPEED**: Bus speed for selection of SDMMC operating modes

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

0: DS, HS, SDR12, SDR25, Legacy compatible, High speed SDR, High speed DDR bus speed mode selected

1: SDR50, DDR50, SDR104, HS200 bus speed mode selected

Bit 18 **DDR**: Data rate signaling selection

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

DDR rate must only be selected with 4-bit or 8-bit wide bus mode. (WIDBUS > 00). DDR = 1 has no effect when WIDBUS = 00 (1-bit wide bus).

DDR rate must only be selected with clock division > 1 (CLKDIV > 0).

0: SDR Single data rate signaling

1: DDR double data rate signaling

Bit 17 **HWFC_EN**: Hardware flow control enable

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

0: Hardware flow control is disabled

1: Hardware flow control is enabled

When Hardware flow control is enabled, the meaning of the TXFIFOE and RXFIFOF flags change, see SDMMC status register definition in *Section 30.10.11*.

Bit 16 **NEGEDGE**: SDMMC_CK dephasing selection bit for data and command

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0).

When clock division = 1 (CLKDIV = 0), this bit has no effect. Data and Command change on SDMMC_CK falling edge.

0: When clock division >1 (CLKDIV > 0) and DDR = 0:

- Command and data changed on the sdmmc_ker_ck falling edge succeeding the rising edge of SDMMC_CK.
- SDMMC_CK edge occurs on sdmmc_ker_ck rising edge.

When clock division >1 (CLKDIV > 0) and DDR = 1:

- Command changed on the sdmmc_ker_ck falling edge succeeding the rising edge of SDMMC_CK.
- Data changed on the sdmmc_ker_ck falling edge succeeding a SDMMC_CK edge.
- SDMMC_CK edge occurs on sdmmc_ker_ck rising edge.

1: When clock division >1 (CLKDIV > 0) and DDR = 0:

- Command and data changed on the same sdmmc_ker_ck rising edge generating the SDMMC_CK falling edge.

When clock division >1 (CLKDIV > 0) and DDR = 1:

- Command changed on the same sdmmc_ker_ck rising edge generating the SDMMC_CK falling edge.
- Data changed on the SDMMC_CK falling edge succeeding a SDMMC_CK edge.
- SDMMC_CK edge occurs on sdmmc_ker_ck rising edge.

Bits 15:14 **WIDBUS[1:0]**: Wide bus mode enable bit

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

00: Default 1-bit wide bus mode: SDMMC_D0 used (Does not support DDR)

01: 4-bit wide bus mode: SDMMC_D[3:0] used

10: 8-bit wide bus mode: SDMMC_D[7:0] used

Bit 13 Reserved, must be kept at reset value.

Bit 12 **PWRSAV**: Power saving configuration bit

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0)

For power saving, the SDMMC_CK clock output can be disabled when the bus is idle by setting PWRSAV:

0: SDMMC_CK clock is always enabled

1: SDMMC_CK is only enabled when the bus is active

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:0 **CLKDIV[9:0]**: Clock divide factor

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0).

This field defines the divide factor between the input clock (sdmmc_ker_ck) and the output clock (SDMMC_CK): SDMMC_CK frequency = sdmmc_ker_ck / [2 * CLKDIV].

0x000: SDMMC_CK frequency = sdmmc_ker_ck / 1 (Does not support DDR)

0x001: SDMMC_CK frequency = sdmmc_ker_ck / 2

0x002: SDMMC_CK frequency = sdmmc_ker_ck / 4

0x0XX: ..

0x080: SDMMC_CK frequency = sdmmc_ker_ck / 256

0xXXX: ..

0x3FF: SDMMC_CK frequency = sdmmc_ker_ck / 2046

Note: While the SD/SDIO card or e•MMC is in identification mode, the SDMMC_CK frequency must be less than 400 kHz.

The clock frequency can be changed to the maximum card bus frequency when relative card addresses are assigned to all cards.

At least seven sdmmc_hclk clock periods are needed between two write accesses to this register. SDMMC_CK can also be stopped during the Read Wait interval for SD I/O cards: in this case the SDMMC_CLKCR register does not control SDMMC_CK.

30.10.3 SDMMC argument register (SDMMC_ARGR)

Address offset: 0x008

Reset value: 0x0000 0000

This register contains a 32-bit command argument, which is sent to a card as part of a command message.

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

Bits 31:0 **CMDARG[31:0]**: Command argument

These bits can only be written by firmware when CPSM is disabled (CPSMEN = 0). Command argument sent to a card as part of a command message. If a command contains an argument, it must be loaded into this register before writing a command to the command register.

30.10.4 SDMMC command register (SDMMC_CMDR)

Address offset: 0x00C

Reset value: 0x0000 0000

This register contains the command index and command type bits. The command index is sent to a card as part of a command message. The command type bits control the command path state machine (CPSM).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMD SUS PEND
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BOOT EN	BOOT MODE	DT HOLD	CPSM EN	WAITP END	WAIT INT	WAITRESP[1:0]		CMD STOP	CMD TRANS	CMDINDEX[5:0]					
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **CMDSUSPEND**: The CPSM treats the command as a Suspend or Resume command and signals interrupt period start/end

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).
CMDSUSPEND = 1 and CMDTRANS = 0 Suspend command, start interrupt period when response bit BS = 0.
CMDSUSPEND = 1 and CMDTRANS = 1 Resume command with data, end interrupt period when response bit DF = 1.

Bit 15 **BOOTEN**: Enable boot mode procedure

0: Boot mode procedure disabled
1: Boot mode procedure enabled

Bit 14 **BOOTMODE**: Select the boot mode procedure to be used

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0)
0: Normal boot mode procedure selected
1: Alternative boot mode procedure selected

Bit 13 **DTHOLD**: Hold new data block transmission and reception in the DPSM

If this bit is set, the DPSM does not move from the Wait_S state to the Send state or from the Wait_R state to the Receive state.

Bit 12 **CPSMEN**: Command path state machine (CPSM) enable bit

This bit is written 1 by firmware, and cleared by hardware when the CPSM enters the Idle state.
If this bit is set, the CPSM is enabled.
When DTEN = 1, no command is transfered nor boot procedure is started. CPSMEN is cleared to 0.
During Read Wait with SDMMC_CK stopped no command is sent and CPSMEN is kept 0.

Bit 11 **WAITPEND**: CPSM waits for end of data transfer (CmdPend internal signal) from DPSM

This bit when set, the CPSM waits for the end of data transfer trigger before it starts sending a command.
WAITPEND is only taken into account when DTMODE = *e*•MMC stream data transfer, WIDBUS = 1-bit wide bus mode, DPSMACT = 1 and DTDIR = from host to card.

Bit 10 **WAITINT**: CPSM waits for interrupt request

If this bit is set, the CPSM disables command timeout and waits for an card interrupt request (Response).
If this bit is cleared in the CPSM Wait state, it causes the abort of the interrupt mode.

Bits 9:8 **WAITRESP[1:0]**: Wait for response bits

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).
They are used to configure whether the CPSM is to wait for a response, and if yes, which kind of response.
00: No response, expect CMDSENT flag
01: Short response, expect CMDREND or CCRCFAIL flag
10: Short response, expect CMDREND flag (No CRC)
11: Long response, expect CMDREND or CCRCFAIL flag

Bit 7 **CMDSTOP**: The CPSM treats the command as a Stop Transmission command and signals abort to the DPSM

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).

If this bit is set, the CPSM issues the abort signal to the DPSM when the command is sent.

Bit 6 **CMDTRANS**: The CPSM treats the command as a data transfer command, stops the interrupt period, and signals DataEnable to the DPSM

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).

If this bit is set, the CPSM issues an end of interrupt period and issues DataEnable signal to the DPSM when the command is sent.

Bits 5:0 **CMDINDEX[5:0]**: Command index

This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).

The command index is sent to the card as part of a command message.

Note: 1 At least seven sdmmc_hclk clock periods are needed between two write accesses to this register.

2 MultiMediaCard can send two kinds of response: short responses, 48 bits, or long responses,136 bits. SD card and SD I/O card can send only short responses, the argument can vary according to the type of response: the software distinguishes the type of response according to the send command.

30.10.5 SDMMC command response register (SDMMC_RESPCMDR)

Address offset: 0x010

Reset value: 0x0000 0000

This register contains the command index field of the last command response received. If the command response transmission does not contain the command index field (long or OCR response), the RESPCMD field is unknown, although it must contain 111111b (the value of the reserved field from the response).

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

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

Bits 5:0 **RESPCMD[5:0]**: Response command index

Read-only bitfield. Contains the command index of the last command response received.

30.10.6 SDMMC response x register (SDMMC_RESPxR)

Address offset: 0x010 + 0x004 * x, (x = 1 to 4)

Reset value: 0x0000 0000

These registers contain the status of a card, which is part of the received response.

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

Bits 31:0 **CARDSTATUS[31:0]**: Card status according table below
See *Table 232*.

The card status size is 32 or 128 bits, depending on the response type.

Table 232. Response type and SDMMC_RESPxR registers

Register(1)	Short response	Long response
SDMMC_RESP1R	Card status[31:0]	Card status [127:96]
SDMMC_RESP2R	all 0	Card status [95:64]
SDMMC_RESP3R	all 0	Card status [63:32]
SDMMC_RESP4R	all 0	Card status [31:0](2)

1. The most significant bit of the card status is received first.
2. The SDMMC_RESP4R register LSB is always 0.

30.10.7 SDMMC data timer register (SDMMC_DTIMER)

Address offset: 0x024

Reset value: 0x0000 0000

This register contains the data timeout period, in card bus clock periods.

A counter loads the value from this register, and starts decrementing when the data path state machine (DPSM) enters the Wait_R or Busy state. If the timer reaches 0 while the DPSM is in either of these states, the timeout status flag is set.

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

Bits 31:0 **DATATIME[31:0]**: Data and R1b busy timeout period

This bit can only be written when the CPSM and DPSM are not active (CPSMACT = 0 and DPSMACT = 0).

Data and R1b busy timeout period expressed in card bus clock periods.

Note: A data transfer must be written to the data timer register and the data length register before being written to the data control register.

30.10.8 SDMMC data length register (SDMMC_DLENR)

Address offset: 0x028

Reset value: 0x0000 0000

This register contains the number of data bytes to be transferred. The value is loaded into the data counter when data transfer starts.

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

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:0 **DATALENGTH[24:0]**: Data length value

This register can only be written by firmware when DPSM is inactive (DPSMACT = 0).

Number of data bytes to be transferred.

When DDR = 1 DATALENGTH is truncated to a multiple of 2. (The last odd byte is not transfered)

When DATALENGTH = 0 no data are transfered, when requested by a CPSMEN and CMDTRANS = 1 also no command is transfered. DTEN and CPSMEN are cleared to 0.

Note: For a block data transfer, the value in the data length register must be a multiple of the block size (see SDMMC_DCTRL). A data transfer must be written to the data timer register and the data length register before being written to the data control register.

For an SDMMC multibyte transfer the value in the data length register must be between 1 and 512.

30.10.9 SDMMC data control register (SDMMC_DCTRL)

Address offset: 0x02C

Reset value: 0x0000 0000

This register controls the data path state machine (DPSM).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	FIFO RST	BOOT ACK EN	SDIO EN	RW MOD	RW STOP	RW START	DBLOCKSIZE[3:0]				DTMODE[1:0]		DTDIR	DTEN
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **FIFORST**: FIFO reset, flushes any remaining data

This bit can only be written by firmware when IDMAEN= 0 and DPSM is active (DPSMACT = 1). This bit only takes effect when a transfer error or transfer hold occurs.

0: FIFO not affected.

1: Flush any remaining data and reset the FIFO pointers. This bit is automatically cleared to 0 by hardware when DPSM gets inactive (DPSMACT = 0).

Bit 12 **BOOTACKEN**: Enable the reception of the boot acknowledgment

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

0: Boot acknowledgment disabled, not expected to be received

1: Boot acknowledgment enabled, expected to be received

Bit 11 **SDIOEN**: SD I/O interrupt enable functions

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

If this bit is set, the DPSM enables the SD I/O card specific interrupt operation.

Bit 10 **RWMOD**: Read Wait mode

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).

0: Read Wait control using SDMMC_D2

1: Read Wait control stopping SDMMC_CK

Bit 9 **RWSTOP**: Read Wait stop

This bit is written by firmware and auto cleared by hardware when the DPSM moves from the R_W state to the Wait_R or Idle state.

0: No Read Wait stop

1: Enable for Read Wait stop when DPSM is in the R_W state.

Bit 8 **RWSTART**: Read Wait start

If this bit is set, Read Wait operation starts.

Bits 7:4 **DBLOCKSIZE[3:0]**: Data block size

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
Define the data block length when the block data transfer mode is selected:
0000: Block length = 2^0 = 1 byte
0001: Block length = 2^1 = 2 bytes
0010: Block length = 2^2 = 4 bytes
0011: Block length = 2^3 = 8 bytes
0100: Block length = 2^4 = 16 bytes
0101: Block length = 2^5 = 32 bytes
0110: Block length = 2^6 = 64 bytes
0111: Block length = 2^7 = 128 bytes
1000: Block length = 2^8 = 256 bytes
1001: Block length = 2^9 = 512 bytes
1010: Block length = 2^{10} = 1024 bytes
1011: Block length = 2^{11} = 2048 bytes
1100: Block length = 2^{12} = 4096 bytes
1101: Block length = 2^{13} = 8192 bytes
1110: Block length = 2^{14} = 16384 bytes
1111: Reserved
When DATALENGTH is not a multiple of DBLOCKSIZE, the transfered data is truncated at a multiple of DBLOCKSIZE. (None of the remaining data are transfered.)
When DDR = 1, DBLOCKSIZE = 0000 must not be used. (No data are transfered)

Bits 3:2 **DTMODE[1:0]**: Data transfer mode selection

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
00: Block data transfer ending on block count.
01: SDIO multibyte data transfer.
10: *e*•MMC Stream data transfer. (WIDBUS must select 1-bit wide bus mode)
11: Block data transfer ending with STOP_TRANSMISSION command (not to be used with DTEN initiated data transfers).

Bit 1 **DTDIR**: Data transfer direction selection

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
0: From host to card.
1: From card to host.

Bit 0 **DTEN**: Data transfer enable bit

This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0). This bit is cleared by hardware when data transfer completes.
This bit must only be used to transfer data when no associated data transfer command is used (must not be used with SD or *e*•MMC cards).
0: Do not start data transfer without CPSM data transfer command.
1: Start data transfer without CPSM data transfer command.

30.10.10 SDMMC data counter register (SDMMC_DCNTR)

Address offset: 0x030

Reset value: 0x0000 0000

This register loads the value from the data length register (see SDMMC_DLENR) when the DPSM moves from the Idle state to the Wait_R or Wait_S state. As data is transferred, the counter decrements the value until it reaches 0. The DPSM then moves to the Idle state and

when there has been no error, and no transmit data transfer hold, the data status end flag (DATAEND) is set.

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

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:0 **DATACOUNT[24:0]**: Data count value

When read, the number of remaining data bytes to be transferred is returned. Write has no effect.

Note: This register must be read only after the data transfer is complete, or hold. When reading after an error event the read data count value may be different from the real number of data bytes transfered.

30.10.11 SDMMC status register (SDMMC_STAR)

Address offset: 0x034

Reset value: 0x0000 0000

This register is a read-only register. It contains two types of flag:

- Static flags (bits [28, 21, 11:0]): these bits remain asserted until they are cleared by writing to the SDMMC interrupt Clear register (see SDMMC_ICR)
- Dynamic flags (bits [20:12]): these bits change state depending on the state of the underlying logic (for example, FIFO full and empty flags are asserted and deasserted as data while written to the FIFO)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	IDMA BTC	IDMA TE	CK STOP	VSW END	ACK TIME OUT	ACK FAIL	SDIOIT	BUSY D0END	BUSY D0	RX FIFOE	TX FIFOE	RX FIFOF	TX FIFOF
			r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RX FIFO HF	TX FIFO HE	CPSM ACT	DPSM ACT	DA BORT	DBCK END	DHOLD	DATA END	CMD SENT	CMDR END	RX OVERR	TX UNDER R	D TIME OUT	C TIME OUT	DCRC FAIL	CCRC FAIL
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **IDMABTC**: IDMA buffer transfer complete

The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 27 **IDMATE**: IDMA transfer error

The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 26 **CKSTOP**: SDMMC_CK stopped in Voltage switch procedure

The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 25 **VSWEND**: Voltage switch critical timing section completion
The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 24 **ACKTIMEOUT**: Boot acknowledgment timeout
The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 23 **ACKFAIL**: Boot acknowledgment received (boot acknowledgment check fail)
The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 22 **SDIOIT**: SDIO interrupt received
The interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 21 **BUSYD0END**: end of SDMMC_D0 Busy following a CMD response detected
This indicates only end of busy following a CMD response. This bit does not signal busy due to data transfer. Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.
0: card SDMMC_D0 signal does NOT signal change from busy to not busy.
1: card SDMMC_D0 signal changed from busy to NOT busy.

Bit 20 **BUSYD0**: Inverted value of SDMMC_D0 line (Busy), sampled at the end of a CMD response and a second time 2 SDMMC_CK cycles after the CMD response
This bit is reset to not busy when the SDMMCD0 line changes from busy to not busy. This bit does not signal busy due to data transfer. This is a hardware status flag only, it does not generate an interrupt.
0: card signals not busy on SDMMC_D0.
1: card signals busy on SDMMC_D0.

Bit 19 **RXFIFOE**: Receive FIFO empty
This is a hardware status flag only, does not generate an interrupt. This bit is cleared when one FIFO location becomes full.

Bit 18 **TXFIFOE**: Transmit FIFO empty
This bit is cleared when one FIFO location becomes full.

Bit 17 **RXFIFOF**: Receive FIFO full
This bit is cleared when one FIFO location becomes empty.

Bit 16 **TXFIFOF**: Transmit FIFO full
This is a hardware status flag only, does not generate an interrupt. This bit is cleared when one FIFO location becomes empty.

Bit 15 **RXFIFOHF**: Receive FIFO half full
There are at least half the number of words in the FIFO. This bit is cleared when the FIFO becomes half+1 empty.

Bit 14 **TXFIFOHE**: Transmit FIFO half empty
At least half the number of words can be written into the FIFO. This bit is cleared when the FIFO becomes half+1 full.

Bit 13 **CPSMACT**: Command path state machine active (not in Idle state)
This is a hardware status flag only, does not generate an interrupt.

Bit 12 **DPSMACT**: Data path state machine active (not in Idle state)
This is a hardware status flag only, does not generate an interrupt.

Bit 11 **DABORT**: Data transfer aborted by CMD12
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 10 **DBCKEND**: Data block sent/received
DBCKEND is set when:
- CRC check passed and DPSM moves to the R_W state
or
- IDMAEN = 0 and transmit data transfer hold and DATACOUNT >0 and DPSM moves to Wait_S.
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 9 **DHOLD**: Data transfer Hold
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 8 **DATAEND**: Data transfer ended correctly
DATAEND is set if data counter DATACOUNT is zero and no errors occur, and no transmit data transfer hold.
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 7 **CMDSENT**: Command sent (no response required)
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 6 **CMDREND**: Command response received (CRC check passed, or no CRC)
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 5 **RXOVERR**: Received FIFO overrun error (masked by hardware when IDMA is enabled)
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 4 **TXUNDERR**: Transmit FIFO underrun error (masked by hardware when IDMA is enabled)
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 3 **DTIMEOUT**: Data timeout
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 2 **CTIMEOUT**: Command response timeout
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.
The Command Timeout period has a fixed value of 64 SDMMC_CK clock periods.

Bit 1 **DCRCFAIL**: Data block sent/received (CRC check failed)
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Bit 0 **CCRCFAIL**: Command response received (CRC check failed)
Interrupt flag is cleared by writing corresponding interrupt clear bit in SDMMC_ICR.

Note: FIFO interrupt flags must be masked in SDMMC_MASKR when using IDMA mode.

30.10.12 SDMMC interrupt clear register (SDMMC_ICR)

Address offset: 0x038

Reset value: 0x0000 0000

This register is a write-only register. Writing a bit with 1 clears the corresponding bit in the SDMMC_STAR status register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	IDMA BTCC	IDMA TEC	CK STOPC	VSW ENDC	ACK TIME OUTC	ACK FAILC	SDIO ITC	BUSY D0 ENDC	Res.	Res.	Res.	Res.	Res.
			rw	rw	rw	rw	rw	rw	rw	rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	D ABORT C	DBCK ENDC	DHOLD C	DATA ENDC	CMD SENTC	CMDR ENDC	RX OVERR C	TX UNDER RC	D TIME OUTC	C TIME OUTC	DCRC FAILC	CCRC FAILC
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **IDMABTCC**: IDMA buffer transfer complete clear bit
Set by software to clear the IDMABTC flag.
0: IDMABTC not cleared
1: IDMABTC cleared

Bit 27 **IDMATEC**: IDMA transfer error clear bit
Set by software to clear the IDMATE flag.
0: IDMATE not cleared
1: IDMATE cleared

Bit 26 **CKSTOPC**: CKSTOP flag clear bit
Set by software to clear the CKSTOP flag.
0: CKSTOP not cleared
1: CKSTOP cleared

Bit 25 **VSWENDC**: VSWEND flag clear bit
Set by software to clear the VSWEND flag.
0: VSWEND not cleared
1: VSWEND cleared

Bit 24 **ACKTIMEOUTC**: ACKTIMEOUT flag clear bit
Set by software to clear the ACKTIMEOUT flag.
0: ACKTIMEOUT not cleared
1: ACKTIMEOUT cleared

Bit 23 **ACKFAILC**: ACKFAIL flag clear bit
Set by software to clear the ACKFAIL flag.
0: ACKFAIL not cleared
1: ACKFAIL cleared

Bit 22 **SDIOITC**: SDIOIT flag clear bit
Set by software to clear the SDIOIT flag.
0: SDIOIT not cleared
1: SDIOIT cleared

Bit 21 **BUSYD0ENDC**: BUSYD0END flag clear bit
Set by software to clear the BUSYD0END flag.
0: BUSYD0END not cleared
1: BUSYD0END cleared

Bits 20:12 Reserved, must be kept at reset value.

Bit 11 **DABORTC**: DABORT flag clear bit
Set by software to clear the DABORT flag.
0: DABORT not cleared
1: DABORT cleared

Bit 10 **DBCKENDC**: DBCKEND flag clear bit
Set by software to clear the DBCKEND flag.
0: DBCKEND not cleared
1: DBCKEND cleared

Bit 9 **DHOLDC**: DHOLD flag clear bit
Set by software to clear the DHOLD flag.
0: DHOLD not cleared
1: DHOLD cleared

Bit 8 **DATAENDC**: DATAEND flag clear bit
Set by software to clear the DATAEND flag.
0: DATAEND not cleared
1: DATAEND cleared

Bit 7 **CMDSENTC**: CMDSENT flag clear bit
Set by software to clear the CMDSENT flag.
0: CMDSENT not cleared
1: CMDSENT cleared

Bit 6 **CMDRENDC**: CMDREND flag clear bit
Set by software to clear the CMDREND flag.
0: CMDREND not cleared
1: CMDREND cleared

Bit 5 **RXOVERRC**: RXOVERR flag clear bit
Set by software to clear the RXOVERR flag.
0: RXOVERR not cleared
1: RXOVERR cleared

Bit 4 **TXUNDERRC**: TXUNDERR flag clear bit
Set by software to clear TXUNDERR flag.
0: TXUNDERR not cleared
1: TXUNDERR cleared

Bit 3 **DTIMEOUTC**: DTIMEOUT flag clear bit
Set by software to clear the DTIMEOUT flag.
0: DTIMEOUT not cleared
1: DTIMEOUT cleared

Bit 2 **CTIMEOUTC**: CTIMEOUT flag clear bit
Set by software to clear the CTIMEOUT flag.
0: CTIMEOUT not cleared
1: CTIMEOUT cleared

Bit 1 **DCRCFAILC**: DCRCFAIL flag clear bit
Set by software to clear the DCRCFAIL flag.
0: DCRCFAIL not cleared
1: DCRCFAIL cleared

Bit 0 **CCRCFAILC**: CCRCFAIL flag clear bit
Set by software to clear the CCRCFAIL flag.
0: CCRCFAIL not cleared
1: CCRCFAIL cleared

30.10.13 SDMMC mask register (SDMMC_MASKR)

Address offset: 0x03C

Reset value: 0x0000 0000

This register determines which status flags generate an interrupt request by setting the corresponding bit to 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	IDMA BTCIE	Res.	CK STOP IE	VSW ENDIE	ACK TIME OUTIE	ACK FAILIE	SDIO ITIE	BUSY D0 ENDIE	Res.	Res.	TX FIFO EIE	RX FIFO FIE	Res.
			rw		rw	rw	rw	rw	rw	rw			rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RX FIFO HFIE	TX FIFO HEIE	Res.	Res.	DA BORT IE	DBCK ENDIE	DHOLD IE	DATA ENDIE	CMD SENT IE	CMDR ENDIE	RX OVER RIE	TX UNDER RIE	D TIME OUTIE	C TIME OUTIE	DCRC FAILIE	CCRC FAILIE
rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **IDMABTCIE**: IDMA buffer transfer complete interrupt enable
Set and cleared by software to enable/disable the interrupt generated when the IDMA has transferred all data belonging to a memory buffer.
0: IDMA buffer transfer complete interrupt disabled
1: IDMA buffer transfer complete interrupt enabled

Bit 27 Reserved, must be kept at reset value.

Bit 26 **CKSTOPIE**: Voltage switch clock stopped interrupt enable
Set and cleared by software to enable/disable interrupt caused by voltage switch clock stopped.
0: Voltage switch clock stopped interrupt disabled
1: Voltage switch clock stopped interrupt enabled

Bit 25 **VSWENDIE**: Voltage switch critical timing section completion interrupt enable
Set and cleared by software to enable/disable the interrupt generated when voltage switch critical timing section completion.
0: Voltage switch critical timing section completion interrupt disabled
1: Voltage switch critical timing section completion interrupt enabled

Bit 24 **ACKTIMEOUTIE**: Acknowledgment timeout interrupt enable
Set and cleared by software to enable/disable interrupt caused by acknowledgment timeout.
0: Acknowledgment timeout interrupt disabled
1: Acknowledgment timeout interrupt enabled

Bit 23 **ACKFAILIE**: Acknowledgment fail interrupt enable
Set and cleared by software to enable/disable interrupt caused by acknowledgment fail.
0: Acknowledgment fail interrupt disabled
1: Acknowledgment fail interrupt enabled

Bit 22 **SDIOITIE**: SDIO mode interrupt received interrupt enable
Set and cleared by software to enable/disable the interrupt generated when receiving the SDIO mode interrupt.
0: SDIO mode interrupt received interrupt disabled
1: SDIO mode interrupt received interrupt enabled

Bit 21 **BUSYD0ENDIE**: BUSYD0END interrupt enable
Set and cleared by software to enable/disable the interrupt generated when SDMMC_D0 signal changes from busy to NOT busy following a CMD response.
0: BUSYD0END interrupt disabled
1: BUSYD0END interrupt enabled

Bits 20:19 Reserved, must be kept at reset value.

Bit 18 **TXFIFOEIE**: Tx FIFO empty interrupt enable
Set and cleared by software to enable/disable interrupt caused by Tx FIFO empty.
0: Tx FIFO empty interrupt disabled
1: Tx FIFO empty interrupt enabled

Bit 17 **RXFIFOFIE**: Rx FIFO full interrupt enable
Set and cleared by software to enable/disable interrupt caused by Rx FIFO full.
0: Rx FIFO full interrupt disabled
1: Rx FIFO full interrupt enabled

Bit 16 Reserved, must be kept at reset value.

Bit 15 **RXFIFOHFIE**: Rx FIFO half full interrupt enable
Set and cleared by software to enable/disable interrupt caused by Rx FIFO half full.
0: Rx FIFO half full interrupt disabled
1: Rx FIFO half full interrupt enabled

Bit 14 **TXFIFOHEIE**: Tx FIFO half empty interrupt enable
Set and cleared by software to enable/disable interrupt caused by Tx FIFO half empty.
0: Tx FIFO half empty interrupt disabled
1: Tx FIFO half empty interrupt enabled

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **DABORTIE**: Data transfer aborted interrupt enable
Set and cleared by software to enable/disable interrupt caused by a data transfer being aborted.
0: Data transfer abort interrupt disabled
1: Data transfer abort interrupt enabled

Bit 10 **DBCKENDIE**: Data block end interrupt enable
Set and cleared by software to enable/disable interrupt caused by data block end.
0: Data block end interrupt disabled
1: Data block end interrupt enabled

Bit 9 **DHOLDIE**: Data hold interrupt enable
Set and cleared by software to enable/disable the interrupt generated when sending new data is hold in the DPSM Wait_S state.
0: Data hold interrupt disabled
1: Data hold interrupt enabled

Bit 8 **DATAENDIE**: Data end interrupt enable
Set and cleared by software to enable/disable interrupt caused by data end.
0: Data end interrupt disabled
1: Data end interrupt enabled

Bit 7 **CMDSENTIE**: Command sent interrupt enable
Set and cleared by software to enable/disable interrupt caused by sending command.
0: Command sent interrupt disabled
1: Command sent interrupt enabled

Bit 6 **CMDRENDIE**: Command response received interrupt enable
Set and cleared by software to enable/disable interrupt caused by receiving command response.
0: Command response received interrupt disabled
1: command Response received interrupt enabled

Bit 5 **RXOVERRIE**: Rx FIFO overrun error interrupt enable
Set and cleared by software to enable/disable interrupt caused by Rx FIFO overrun error.
0: Rx FIFO overrun error interrupt disabled
1: Rx FIFO overrun error interrupt enabled

Bit 4 **TXUNDERRIE**: Tx FIFO underrun error interrupt enable
Set and cleared by software to enable/disable interrupt caused by Tx FIFO underrun error.
0: Tx FIFO underrun error interrupt disabled
1: Tx FIFO underrun error interrupt enabled

Bit 3 **DTIMEOUTIE**: Data timeout interrupt enable
Set and cleared by software to enable/disable interrupt caused by data timeout.
0: Data timeout interrupt disabled
1: Data timeout interrupt enabled

Bit 2 **CTIMEOUTIE**: Command timeout interrupt enable
Set and cleared by software to enable/disable interrupt caused by command timeout.
0: Command timeout interrupt disabled
1: Command timeout interrupt enabled

Bit 1 **DCRCFAILIE**: Data CRC fail interrupt enable
Set and cleared by software to enable/disable interrupt caused by data CRC failure.
0: Data CRC fail interrupt disabled
1: Data CRC fail interrupt enabled

Bit 0 **CCRCFAILIE**: Command CRC fail interrupt enable
Set and cleared by software to enable/disable interrupt caused by command CRC failure.
0: Command CRC fail interrupt disabled
1: Command CRC fail interrupt enabled

30.10.14 SDMMC acknowledgment timer register (SDMMC_ACKTIMER)

Address offset: 0x040

Reset value: 0x0000 0000

This register contains the acknowledgment timeout period, in SDMMC_CK bus clock periods.

A counter loads the value from this register, and starts decrementing when the data path state machine (DPSM) enters the Wait_Ack state. If the timer reaches 0 while the DPSM is in this states, the acknowledgment timeout status flag is set.

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

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:0 **ACKTIME[24:0]**: Boot acknowledgment timeout period
This bit can only be written by firmware when CPSM is disabled (CPSMEN = 0).
Boot acknowledgment timeout period expressed in card bus clock periods.

Note: The data transfer must be written to the acknowledgment timer register before being written to the data control register.

30.10.15 SDMMC data FIFO threshold register (SDMMC_FIFOTHRR)

Address offset: 0x044

Reset value: 0x0000 0000

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

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

Bits 3:0 **THR[3:0]**: FIFO threshold
This bit can only be written by software when DPSM is inactive (DPSMACT = 0).
Define the data FIFO threshold:
0b0000:threshold = 0 byte (no threshold)
Others: threshold = $2^{(N-1)}$ bytes, threshold must not be programmed bigger than the FIFO size, 256 words.

Note: When THR > DBLOCKSIZE, blocks may not be transferred.

30.10.16 SDMMC DMA control register (SDMMC_IDMACTRLR)

Address offset: 0x050

Reset value: 0x0000 0000

The receive and transmit FIFOs can be read or written as 32-bit wide registers. The FIFOs contain 32 entries on 32 sequential addresses. This enables the CPU to use its load and store multiple operands to read from/write to the FIFO.

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

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

Bit 1 **IDMABMODE**: Buffer mode selection
This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
0: Single buffer mode.
1: Linked list mode.

Bit 0 **IDMAEN**: IDMA enable
This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
0: IDMA disabled
1: IDMA enabled

30.10.17 SDMMC IDMA buffer size register (SDMMC_IDMABSIZER)

Address offset: 0x054

Reset value: 0x0000 0000

This register contains the buffer size when in linked list configuration.

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

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:6 **IDMABNDT[10:0]**: Number of bytes per buffer

This 11-bit value must be multiplied by 16 to get the size of the buffer in 32-bit words and by 64 to get the size of the buffer in bytes.
Example: IDMABNDT = 0x001: buffer size = 16 words = 64 bytes.
Example: IDMABNDT = 0x400: buffer size = 16384 words = 64 Kbytes.
These bits can only be written by firmware when DPSM is inactive (DPSMACT = 0).

Bits 5:0 Reserved, must be kept at reset value.

30.10.18 SDMMC IDMA buffer base address register (SDMMC_IDMABASER)

Address offset: 0x058

Reset value: 0x0000 0000

This register contains the memory buffer base address in single buffer configuration and linked list configuration.

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

Bits 31:0 **IDMABASE[31:0]**: Buffer memory base address bits [31:2], must be word aligned (bit [1:0] are always 0 and read only)

This register can be written by firmware when DPSM is inactive (DPSMACT = 0), and can dynamically be written by firmware when DPSM active (DPSMACT = 1).

30.10.19 SDMMC IDMA linked list address register (SDMMC_IDMALAR)

Address offset: 0x064

Reset value: 0x0000 0000

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

Bit 31 **ULA**: Update SDMMC_IDMALAR from linked list when in linked list mode (SDMMC_IDMACTRLR.IDMABMODE select linked list mode)
This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
0: SDMMC_IDMALAR is not to be updated, last linked list item.
1: SDMMC_IDMALAR is to be updated from linked list table.

Bit 30 **ULS**: Update SDMMC_IDMABSIZE from the next linked list when in linked list mode (SDMMC_IDMACTRLR.IDMABMODE select linked list mode and ULA = 1)
This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
0: SDMMC_IDMABSIZER is not to be updated from next linked list table.
1: SDMMC_IDMABSIZER is to be updated from next linked list table.

Bit 29 **ABR**: Acknowledge linked list buffer ready
This bit can only be written by firmware when DPSM is inactive (DPSMACT = 0).
This bit is not taken into account when starting the first linked list buffer from the software programmed register information. ABR is only taken into account on subsequent loaded linked list items.
0: Loaded linked list buffer is not ready (this causes a linked list IDMA transfer error to be generated).
1: Loaded linked list buffer ready acknowledge. Linked list buffer data are transfered by IDMA.

Bits 28:16 Reserved, must be kept at reset value.

Bits 15:2 **IDMALA[13:0]**: Word aligned linked list item address offset
Linked list item offset pointer to the base of the next linked list item structure.
Linked list item base address is IDMABA + IDMALA.
These bits can only be written by firmware when DPSM is inactive (DPSMACT = 0).

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

30.10.20 SDMMC IDMA linked list memory base register (SDMMC_IDMABAR)

Address offset: 0x068

Reset value: 0x0000 0000

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

Bits 31:2 **IDMABA[29:0]**: Word aligned Linked list memory base address
Linked list memory base pointer.
These bits can only be written by firmware when DPSM is inactive (DPSMACT = 0).

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

30.10.21 SDMMC data FIFO registers x (SDMMC_FIFORx)

Address offset: 0x080 + 0x004 * x, (x =0 to 15)

Reset value: 0x0000 0000

The receive and transmit FIFOs can be only read or written as word (32-bit) wide registers. The FIFOs contain 16 entries on sequential addresses. This enables the CPU to use its load and store multiple operands to read from/write to the FIFO. The FIFO register interface takes care of correct data alignment inside the FIFO, the FIFO register address used by the CPU does matter.

When accessing SDMMC_FIFOR with half word or byte access an AHB bus fault is generated.

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

Bits 31:0 **FIFODATA[31:0]**: Receive and transmit FIFO data

This register can only be read or written by firmware when the DPSM is active (DPSMACT = 1).

The FIFO data occupies 16 entries of 32-bit words.

30.10.22 SDMMC register map

Table 233. SDMMC register map

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	SDMMC_ POWER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIRPOL	VSWITCHEN	VSWITCH	PWRCTRL[1:0]	
	Reset value																												0	0	0	0	0
0x004	SDMMC_ CLKCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELCLKRX[1:0]		BUSSPEED	DDR	HWFC_EN	NEGEDGE	WIDBUS[1:0]		Res.	PWRSAV	Res.	Res.	CLKDIV[9:0]									
	Reset value											0	0	0	0	0	0	0	0		0			0	0	0	0	0	0	0	0	0	0
0x008	SDMMC_ ARGR	CMDARG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	SDMMC_ CMDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CMDSUSPEND	BOOTEN	BOOTMODE	DTHOLD	CPSMEN	WAITPEND	WAITINT	WAITRESP[1:0]		CMDSTOP	CMDTRANS	CMDINDEX[5:0]					
	Reset value																																

Table 233. SDMMC register map (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x010	SDMMC_ RESPCMDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RESPCMD[5:0]					
	Reset value																											0	0	0	0	0	0
0x014	SDMMC_ RESP1R	CARDSTATUS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	SDMMC_ RESP2R	CARDSTATUS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	SDMMC_ RESP3R	CARDSTATUS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	SDMMC_ RESP4R	CARDSTATUS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	SDMMC_ DTIMER	DATATIME[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	SDMMC_ DLENR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATALENGTH[24:0]																								
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	SDMMC_ DCTRLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FIFORST	BOOTACKEN	SDIOEN	RWMOD	RWSTOP	RWSTART	DBLOCK SIZE[3:0]				DTMODE[1:0]		DTDIR	DTEN
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	SDMMC_ DCNTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATACOUNT[24:0]																								
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	SDMMC_ STAR	Res.	Res.	Res.	IDMABTC	IDMATE	CKSTOP	VSWEND	ACKTIMEOUT	ACKFAIL	SDIOIT	BUSYD0END	BUSYD0	RXFIFOE	TXFIFOE	RXFIFOF	TXFIFOF	RXFIFOHF	TXFIFOHE	CPSMACT	DPSMACT	DABORT	DBCKEND	DHOLD	DATAEND	CMDSENT	CMDREND	RXOVERR	TXUNDERR	DTIMEOUT	CTIMEOUT	DCRCFAIL	CCRCFAIL
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	SDMMC_ ICR	Res.	Res.	Res.	IDMABTCC	IDMATEC	CKSTOPC	VSWENDC	ACKTIMEOUTC	ACKFAILC	SDIOITC	BUSYD0ENDC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DABORTC	DBCKENDC	DHOLDC	DATAENDC	CMDSENTC	CMDRENDC	RXOVERRC	TXUNDERRC	DTIMEOUTC	CTIMEOUTC	DCRCFAILC	CCRCFAILC
	Reset value				0	0	0	0	0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0
0x03C	SDMMC_ MASKR	Res.	Res.	Res.	IDMABTCIE	Res.	CKSTOPIE	VSWENDIE	ACKTIMEOUTIE	ACKFAILIE	SDIOITIE	BUSYD0ENDIE	Res.	Res.	TXFIFOEIE	RXFIFOFIE	Res.	RXFIFOHFIE	TXFIFOHEIE	Res.	Res.	DABORTIE	DBCKENDIE	DHOLDIE	DATAENDIE	CMDSENTIE	CMDRENDIE	RXOVERRIE	TXUNDERRIE	DTIMEOUTIE	CTIMEOUTIE	DCRCFAILIE	CCRCFAILIE
	Reset value				0		0	0	0	0	0	0			0	0		0	0			0	0	0	0	0	0	0	0	0	0	0	0

Table 233. SDMMC register map (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x040	SDMMC_ ACKTIMER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACKTIME[24:0]																								
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	SDMMC_ FIFOTHRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	THR[3:0]			
	Reset value																													0	0	0	0
0x048 - 0x04C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x050	SDMMC_ IDMACTRLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDMABMODE	IDMAEN
	Reset value																															0	0
0x054	SDMMC_ IDMABSIZER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDMABNDT[10:0]											Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																0	0	0	0	0	0	0	0	0	0	0						
0x058	SDMMC_ IDMABASER	IDMABASE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x05C - 0x060	Reserved	Res.																															
0x064	SDMMC_ IDMALAR	ULA	ULS	ABR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDMALA[13:0]														Res.	Res.
	Reset value	0	0	0														0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x068	SDMMC_ IDMABAR	IDMABA[29:0]																														Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x06C - 0x07C	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x080 + 0x04 * x, (x=0..15)	SDMMC_ FIFORx	FIFODATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

31 Delay block (DLYB)

31.1 DLYB introduction

The delay block (DLYB) is used to generate two output clocks that are dephased from 2 input clocks. The phase of each output clock must be programmed by the user application. The output clocks are then used to clock the data transmitted/received by another peripheral such as an SDMMC.

31.2 DLYB main features

The delay block has the following features:

- Frequency range for Lock mode is [50 MHz, 208 MHz]
- 32 phases (delay taps)
- Bypass mode for operation below 50 MHz

31.3 DLYB implementation

Table 234. STM32N6x5/x7xx features

DLYB associated peripheral	DLYBSD1	DLYBSD2
SDMMC1	X	-
SDMMC2	-	X

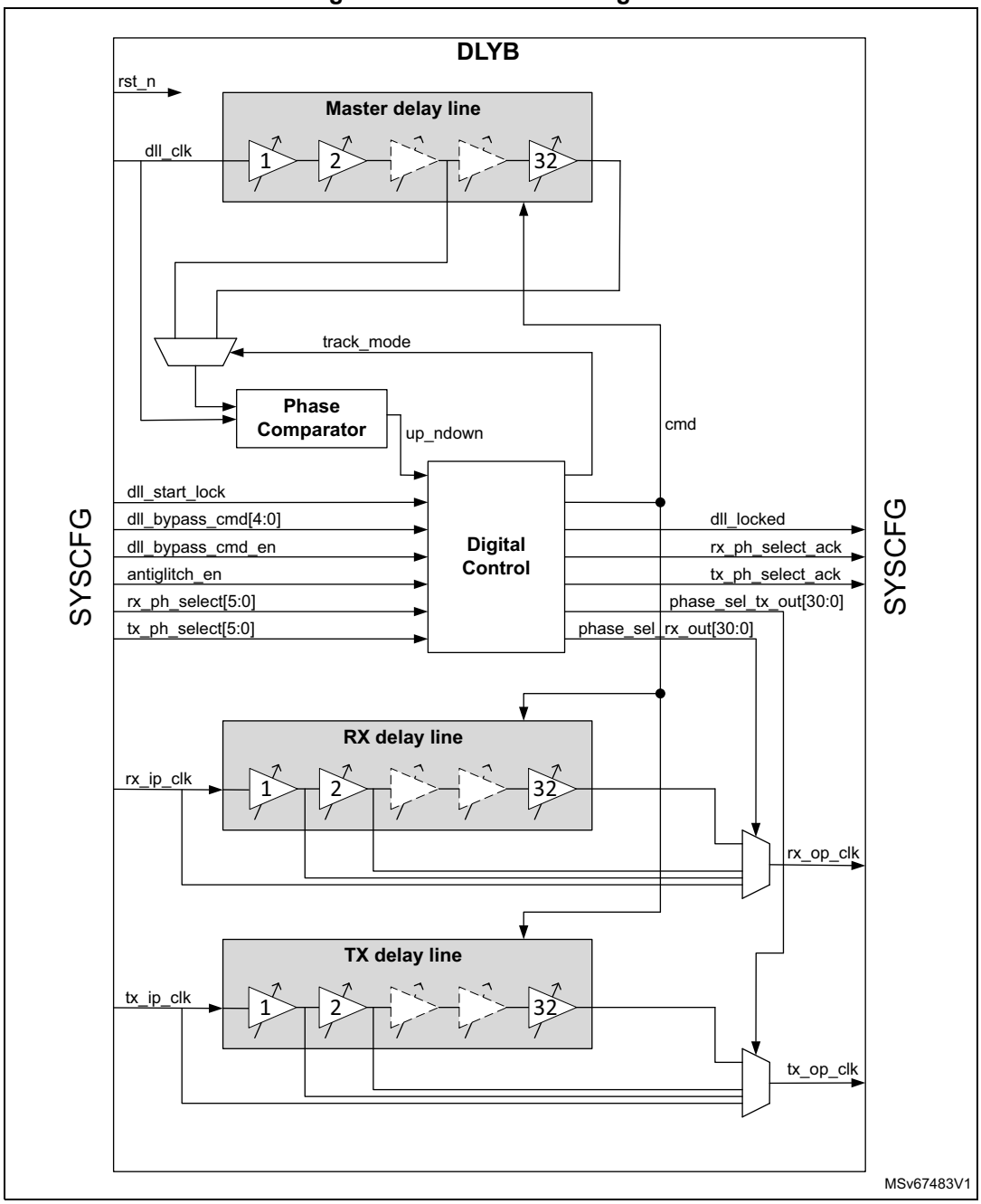
31.4 DLYB functional description

31.4.1 DLYB diagram

The delay block includes the following sub-blocks (shown in the figure below):

- Digital block to generate commands for delay lines and phase selection. This block embeds lock FSM.
- Delay line supporting the unit delays
- Output clock selection multiplexer

Figure 260. DLYB block diagram



31.4.2 DLYB internal signals

Table 235 lists the DLYB functional signals.

Table 235. DLYB functional signals

Signal name	Signal type	Description
dll_clk	Digital input	Free-running input clock used as reference clock for the Master delay line and FSM blocks
tx_ip_clk	Digital input	Input clock of the TX slave delay line
rx_ip_clk	Digital input	Input clock of the RX slave delay line
tx_op_clk	Digital output	Selected output clock of the TX slave delay line
rx_op_clk	Digital output	Selected output clock of the RX slave delay line
rst_n	Digital input	Asynchronous active low reset
dll_start_lock	Digital input	Enable of DLL when configured in Lock mode
dll_locked	Digital output	Lock indicator. When high, it indicates that the FSM has finished the lock sequence. The DLL can be used
tx_ph_select[5:0]	Digital input	Selection of the slave DLL phase shift over 33 for TX path 000000: 0 tap selection 000001: 1 tap selection 011111: 31 taps selection 1xxxxx: 32 taps selection
tx_ph_select_ack	Digital output	When high: phase selected by tx_ph_select is acknowledged When low: output clock tx_op_clk is tied low
rx_ph_select[5:0]	Digital input	Selection of the slave DLL phase shift over 33 for RX path 000000: 0 tap selection 000001: 1 tap selection 011111: 31 taps selection 1xxxxx: 32 taps selection
rx_ph_select_ack	Digital output	When high: phase selected by rx_ph_select is acknowledged When low: output clock rx_op_clk is tied low
antiglitch_en	Digital input	When high: output clocks (tx_op_clk or rx_op_clk) when minimum delay is selected (tx_ph_select = 0 or rx_ph_select=0) are gated to avoid any glitch when phase selection changes When low: output clocks (tx_op_clk or rx_op_clk) when minimum delay is selected (tx_ph_select = 0 or rx_ph_select=0) are not gated (possible glitches if input clocks (tx_ip_clk or rx_ip_clk) are enabled when phase selection changes)
dll_bypass_cmd_en	Digital input	Active high: use dll_bypass_cmd for command generation
dll_bypass_cmd[4:0]	Digital input	permits to force delays for all taps when dll_bypass_cmd_en high: 00000: not functional 00001: pass gate 0 is activated (min delay) 00010: pass gate 1 is activated 11000: pass gate 23 is activated 11001 to 11111: pass gate 23 is activated (max delay)

Table 236 lists the DLYB interconnections. The SDMMCx DLYB registers (DLYBSDx) are highlighted in light gray, RCC registers in dark gray and peripheral ports in blue.

Table 236. DLYB interconnection

Signal name	Source/destination	
	SDMMC1	SDMMC2
dll_clk	ClkDll (sdmmc1_ClkDll)	ClkDll (sdmmc2_ClkDll)
rx_ip_clk	CKio (sdmmc1_io_CKio)	CKio (sdmmc2_io_CKio)
tx_ip_clk	Not used	
rx_op_clk	FB_CLK (sdmmc1_fb_clk)	FB_CLK (sdmmc2_fb_clk)
tx_op_clk	Not used	
rst_n	RCC_MISCRSTR.SDMMC1DLLRST	RCC_MISCRSTR.SDMMC2DLLRST
dll_start_lock	DLYBSD_CFG(1) SDMMC_DLL_EN	DLYBSD_CFG(2) SDMMC_DLL_EN
dll_locked	DLYBSD_STATUS(1) SDMMC_DLL_LOCK	DLYBSD_STATUS(2) SDMMC_DLL_LOCK
tx_ph_select[5:0]	Not used	
tx_ph_select_ack	Not used	
rx_ph_select[5:0]	DLYBSD_CFG(1) SDMMC_RX_TAP_SEL[5:0]	DLYBSD_CFG(2) SDMMC_RX_TAP_SEL[5:0]
rx_ph_select_ack	DLYBSD_STATUS(1) SDMMC_RX_TAP_SEL_ACK	DLYBSD_STATUS(2) SDMMC_RX_TAP_SEL_ACK
antiglitch_en	DLYBSD_CFG(1) SDMMC_DLL_ANTIGLITCH_EN	DLYBSD_CFG(2) SDMMC_DLL_ANTIGLITCH_EN
dll_bypass_cmd_en	DLYBSD_CFG(1) SDMMC_DLL_BYP_EN	DLYBSD_CFG(2) SDMMC_DLL_BYP_EN
dll_bypass_cmd[4:0]	DLYBSD_CFG(1) SDMMC_DLL_BYP_CMD[4:0]	DLYBSD_CFG(2) SDMMC_DLL_BYP_CMD[4:0]

1. Register in DLYB1 instance.
2. Register in DLYB2 instance.

31.4.3 General description

The delay block is structured around one master delay line and two slave delay lines (RX and TX) and there are two operating modes:

- Lock mode (*dll_bypass_cmd_en* = 0)
  - This mode can only be used when the frequency of dll_clk is in the range [50 MHz, 208 MHz]
  - The delay block is enabled by setting *dll_start_lock* to '1'.
  - The delay block locks on the frequency of the *dll_clk* input clock and compensates for voltage and temperature variations. This process determines the command to be applied to each of the 32 delay units (a.k.a. tap) of the master delay line for the total master delay to represent one period of *dll_clk*. This command is copied to each delay unit of the RX and TX delay lines. The command varies with voltage and temperature for the DLL to remain locked. This guarantees that each delay unit of the RX and TX delay lines represents 1/32th of the input clock period.
  - Independent delay values can be selected for the RX and TX delay lines by controlling *rx_ph_select[5:0]* and *tx_ph_select[5:0]* parameters. The granularity of the delay is 1/32th of the input clock period.
- Bypass mode (*dll_bypass_cmd_en* = 1)
  - This mode is required to be able to insert a delay between the *rx_ip_clk* (resp. *tx_ip_clk*) input clock and the *rx_op_clk* (resp. *tx_op_clk*) output clock when the operating frequency is lower than the minimum frequency supported by the delay block in lock mode (frequency < 50 MHz).
  - In this mode, delays in the RX and TX delay lines are used as "pure" delays, i.e. without trying to lock on the frequency of the *dll_clk* input clock. There is therefore no voltage nor temperature compensation in this mode and the RX and TX delay values vary with voltage and temperature. The command applied to each delay unit of the RX and TX delay lines is controlled by *dll_bypass_cmd[4:0]*. Each delay unit includes 24 delay elements. The number of delay elements which are enabled in each delay unit depends on the value of *dll_bypass_cmd[4:0]*.
  - Independent delay values can be selected for the RX and TX delay lines by controlling *rx_ph_select[5:0]* and *tx_ph_select[5:0]* parameters. The total delay inserted by the RX (resp. TX) delay line depends on *rx_ph_select[5:0]* (resp. *tx_ph_select[5:0]*) and *dll_bypass_cmd[4:0]* values.

31.4.4 Lock mode procedure

For frequencies greater or equal to 50MHz, procedure is given below:

1. When *rst_n* is low, DLYB lock FSM is put in Idle mode. Note that *dll_start_lock* must be set to ‘0’ during hard reset.
2. Once the input signal *dll_clk* is stable (after a frequency switch for example) the lock sequence can be started by setting *dll_start_lock* to ‘1’ and maintain this value.
3. During lock sequence DLYB can not be used until *dll_locked* flag is set to ‘1’.
4. Once it is done, delay line for master and slaves are PVT compensated (tracking V and T evolution but not a frequency switch).
5. Then the phase can be programmed using *tx_ph_select[5:0]* and *rx_ph_select[5:0]* input signals. The selected delays are applied to TX output clock and RX output clock once *tx_ph_select_ack* and *rx_ph_select_ack* are set to ‘1’ respectively.
6. If another lock sequence is needed, for a frequency switch for example, *dll_start_lock* signal must go to ‘0’ and sequence can be restarted with step 2.

31.4.5 Bypass mode procedure

For frequencies lower the 50 MHz, procedure is given below:

1. When *rst_n* is low, DLYB lock FSM is put in Idle mode. Note that *dll_start_lock* must be set to ‘0’ during hard reset.
2. Delay block is enabled by setting *dll_bypass_cmd_en* to ‘1’. The command applied to every delay unit is defined by *dll_bypass_cmd[4:0]* input (the value 0 is reserved and must not be used).
3. The phase can be programmed using *tx_ph_select[5:0]* and *rx_ph_select[5:0]* input signals. The selected delays are applied to TX output clock and RX output clock once *tx_ph_select_ack* and *rx_ph_select_ack* are set to ‘1’ respectively. For minimum delay, *tx_ph_select[5:0]* (resp. *rx_ph_select[5:0]*) must be set to 0.

31.5 DLYB SDMMC registers description

There is one DLYB module associated to each SDMMC (see *Section 2.3: Memory organization*). It contains registers to configure the DLL for the RX path. The registers are described below.

31.5.1 Delay block SDMMC DLL configuration (DLYBSD_CFG)

Address offset: 0x0

Reset value: 0x0040 0000

SDMMC DLL configuration register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC_DLL_ANTIGLITCH_EN	SDMMC_DLL_BYP_CMD[4:0]					SDMMC_DLL_BYP_EN
									rw	rw	rw	rw	rw	rw	rw

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

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **SDMMC_DLL_ANTIGLITCH_EN**: Antiglitch logic enabled when 1

Bits 21:17 **SDMMC_DLL_BYP_CMD[4:0]**: Bypass command

It is applied on RX path and corresponds to the number of delay cells activated (when SDMMC_DLL_BYP_EN is high).

Bit 16 **SDMMC_DLL_BYP_EN**: DLL configuration

0: Lock mode

1: Pure delay

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:1 **SDMMC_RX_TAP_SEL[5:0]**: selection of RX delay

In 1/32th of clock period.

Bit 0 **SDMMC_DLL_EN**: DLL enable

Used to enable the DLL when configured in lock mode.

31.5.2 Delay block SDMMC DLL status (DLYBSD_STATUS)

Address offset: 0x4

Reset value: 0x0000 0002

SDMMC DLL status register.

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

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

Bit 1 **SDMMC_RX_TAP_SEL_ACK**: SDMMC RX delay selection acknowledge
This bit indicates that RX delay line has been configured with the delay selected by SDMMC_RX_TAP_SEL[5:0].

Bit 0 **SDMMC_DLL_LOCK**: SDMMC DLL lock
This bit indicates that DLL is locked to the input frequency.

31.5.3 DLYB register map

Table 237. DLYB register map and reset values

Offset	Register name Reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0	**DLYBSD_CFG**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC_DLL_ANTIGLITCH_EN	SDMMC_DLL_BYP_CMD					SDMMC_DLL_BYP_EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC_RX_TAP_SEL						SDMMC_DLL_EN
	Reset value										1	0	0	0	0	0	0										0	0	0	0	0	0	0

Table 237. DLYB register map and reset values (continued)

Offset	Register name Reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x4	**DLYBSD_ STATUS**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDMMC_RX_TAP_SEL_ACK	SDMMC_DLL_LOCK
	Reset value																															1	0

Refer to *Section 2.3 on page 163* for the register boundary addresses.

32 Analog-to-digital converters (ADC)

32.1 ADC introduction

This section describes the implementation of up to 2 ADCs:

- ADC1 and ADC2 are tightly coupled and can operate in dual mode (ADC1 is master)

Each ADC consists of one 12-bit successive approximation analog-to-digital converter.

Each ADC has up to 20 multiplexed channels. A/D conversion of the various channels can be performed in single, continuous, scan, or discontinuous mode. The result of the ADC is stored in a left-aligned or right-aligned (default configuration) 32-bit data register.

The ADCs are mapped on the AHB bus to allow fast data handling.

The analog watchdog features allow the application to detect if the input voltage goes outside the user-defined high or low thresholds.

A built-in hardware oversampler allows improving analog performances while off-loading the related computational burden from the CPU.

An efficient low-power mode is implemented to allow very low consumption at low frequency.

32.2 ADC main features

- High-performance features
  - Up to 2 ADCs which can operate in dual mode
    ADC1 is connected to 18 external channels + 2 internal channels
    ADC2 is connected to 17 external channels + 3 internal channels
  - 12, 10, 8 or 6-bit configurable resolution
  - ADC conversion time independent from the AHB bus clock frequency
  - Faster conversion time by lowering resolution
  - Management of single-ended and differential inputs (programmable per channels)
  - AHB slave bus interface to allow fast data handling
  - Offset calibration support
  - Channel-wise programmable sampling time
  - Flexible sampling time control
  - Fixed latency for a trigger to start of sampling
  - Up to 4 injected channels (analog inputs assignment to regular or injected channels is fully configurable)
  - Data alignment with in-built data coherency
  - Data can be managed by DMA for regular channel conversions
  - Data can be routed to MDF for post processing
  - Four dedicated data registers for the injected channels
- Low-power features
  - Speed adaptive low-power mode to reduce ADC consumption when operating at low frequency

  - Allows slow bus frequency application while keeping optimum ADC performance
  - Provides automatic control to avoid ADC overrun in low AHB bus clock frequency application (autodelayed mode)
- Oversampler
  - 32-bit data register
  - Oversampling ratio adjustable from 2 to 1024
  - Programmable data right and left shifts
- Data preconditioning
  - Gain compensation
  - Offset compensation
- Analog input channels
  - External analog inputs (per ADC): up to 17 GPIO pads
  - 1 channel for the internal reference voltage (V_{REFINT})
  - 1 channel for monitoring the external VBAT power supply pin
  - 1 channel connected to the analog positive reference voltage V_{REF+}
  - 1 channel for monitoring V_{DDCORE} internal voltage
- Start-of-conversion can be initiated:
  - By software for both regular and injected conversions
  - By hardware triggers with configurable polarity (internal timer events or GPIO input events) for both regular and injected conversions
- Conversion modes
  - Each ADC can convert a single channel or can scan a sequence of channels
  - Single mode converts selected inputs once per trigger
  - Continuous mode converts selected inputs continuously
  - Discontinuous mode
- Interrupt generation at ADC ready, the end of sampling, the end of conversion (regular or injected), end of sequence conversion (regular or injected), analog watchdog 1, 2 or 3 or overrun events
- 3 analog watchdogs per ADC
- ADC input range: $V_{SSA} \leq V_{IN} \leq V_{REF+}$

Figure 261 shows the block diagram of one ADC.

32.3 ADC implementation

Table 238. ADC features

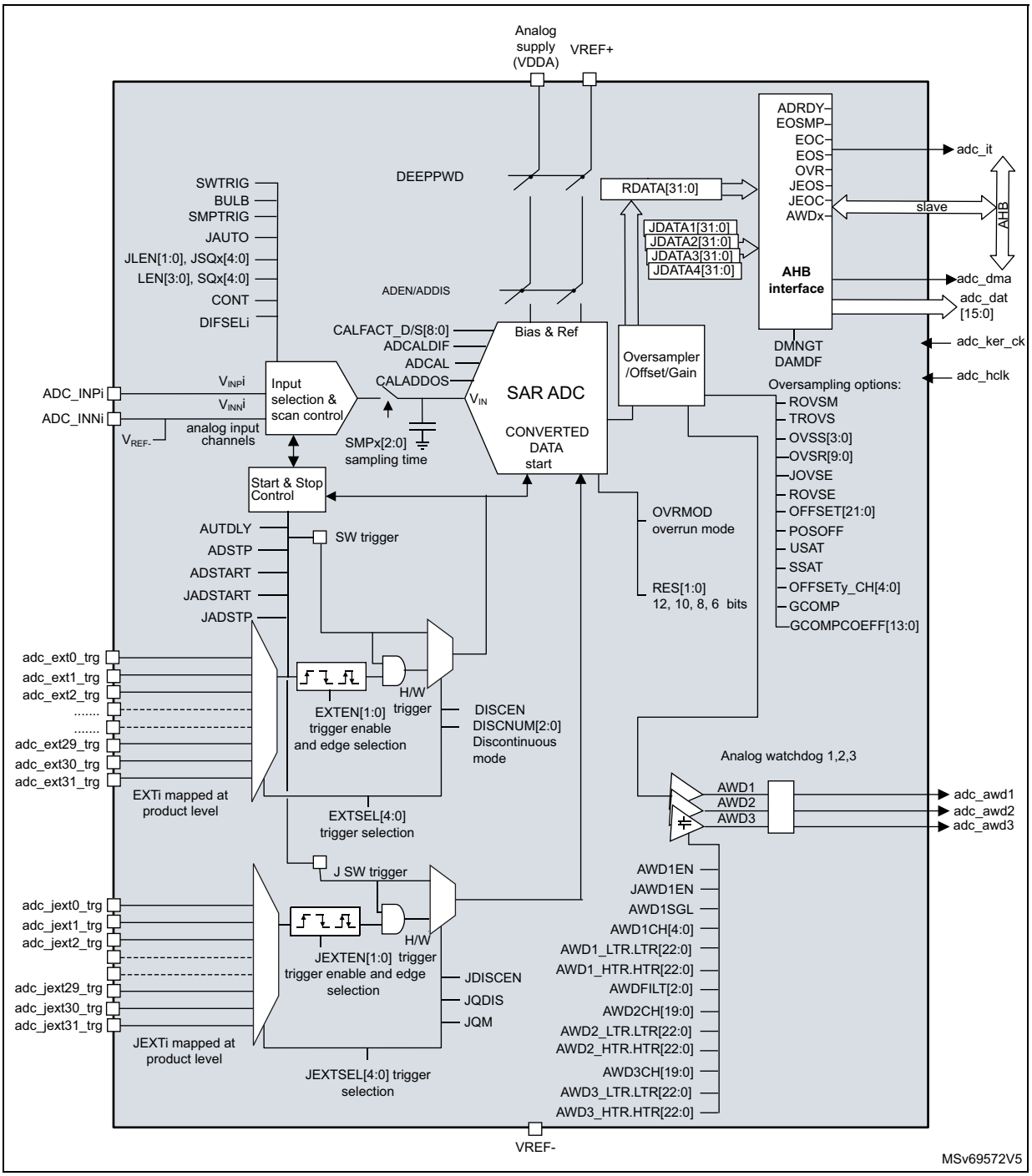
ADC modes/features	ADC1, ADC2
Resolution	12 bits
Maximum sampling speed	5 Msps (12-bit resolution)
Dual mode operation	X
Offset calibration	X
Single-end input	X
Differential input	X
Injected channel conversion	X
Oversampling	up to x1024
Data register	32 bits
DMA support	X
Parallel data output to MDF	X
Offset compensation	X
Gain compensation	X
Number of analog watchdog	3

32.4 ADC functional description

32.4.1 ADC block diagram

Figure 261 shows the ADC block diagram and *Table 239* gives the ADC pin description.

Figure 261. ADC block diagram



32.4.2 ADC pins and internal signals

Table 239. ADC input/output pins

Pin name	Signal type	Description
VDDA	Input, analog supply	Analog power supply
VSSA	Input, analog supply ground	Ground for analog power supply, equal to V_{SS}.
VREF+	Input, analog reference positive	The higher/positive reference voltage for the ADC.
VREF-	Input, analog reference negative	The lower/negative reference voltage for the ADC.
ADCx_INNi/INPi	Negative/positive external analog input signals	20 negative/positive external analog input channels (refer to *Section 32.4.4: ADC connectivity* for details)

Table 240. ADC internal input/output signals

Internal signal name	Signal type	Description
$V_{INP}i$	Positive analog input channels	Positive internal analog input channels connected either to ADCx_INP*i* external channels or to internal channels.
$V_{INN}i$	Negative analog input channels	Negative internal analog input channels connected either to ADCx_INN*i* external channels or to internal channels
adc_ext_trgi	Inputs	ADC external trigger inputs for regular conversions.These inputs are shared between the ADC master and the ADC slave.
adc_jext_trgi	Inputs	ADC external trigger inputs for the injected conversions. These inputs are shared between the ADC master and the ADC slave.
adc_awdy	Output	Internal analog watchdog output signal connected to on-chip timers. (y = Analog watchdog number 1,2,3)
adc_ker_ck	Input	ADC kernel clock
adc_hclk	Input	ADC peripheral clock
adc_it	Output	ADC interrupt
adc_dma	Output	ADC DMA request
adc_dat[15:0]	Output	ADC data outputs

Table 241. ADC1/2 interconnection

Signal name	Source/destination
ADC1 V_{INP}[17]	V_{REFINT} (output voltage from internal reference voltage)
ADC2 V_{INP}[16]	V_{BAT}/4 (VBAT pin input voltage divided by 4)
ADC2 V_{INP}[17]	V_{DDCORE} (internal logic core voltage)
ADC1 V_{INP}[19], ADC2 V_{INP}[19]	V_{REF+} (higher positive reference voltage)
adc_dat1[15:0]	mdf_adc1_dat[15:0]

Table 241. ADC1/2 interconnection

Signal name	Source/destination
adc_dat2[15:0]	mdf_adc2_dat[15:0]
adc_ext_trg0	exti11
adc_ext_trg1	tim1_oc1
adc_ext_trg2	tim1_oc2
adc_ext_trg3	tim1_oc3
adc_ext_trg4	tim1_trgo_cktim
adc_ext_trg5	tim1_trgo2_cktim
adc_ext_trg6	tim2_oc2
adc_ext_trg7	tim2_trgo_cktim
adc_ext_trg8	Reserved
adc_ext_trg9	Reserved
adc_ext_trg10	tim3_oc4
adc_ext_trg11	tim3_trgo_cktim
adc_ext_trg12	tim4_oc4
adc_ext_trg13	tim4_trgo_cktim
adc_ext_trg14	tim5_trgo_cktim
adc_ext_trg15	tim6_trgo_cktim
adc_ext_trg16	tim7_trgo_cktim
adc_ext_trg17	tim8_trgo_cktim
adc_ext_trg18	tim8_trgo2_cktim
adc_ext_trg19	tim9_oc1
adc_ext_trg20	tim9_trgo_cktim
adc_ext_trg21	tim12_trgo_cktim
adc_ext_trg22	tim15_trgo_cktim
adc_ext_trg23	tim18_trgo_cktim
adc_ext_trg24	lptim1_ch1
adc_ext_trg25	lptim2_ch1
adc_ext_trg26	lptim3_ch1
adc_ext_trg27	Reserved
adc_ext_trg28	Reserved
adc_ext_trg29	Reserved
adc_ext_trg30	Reserved
adc_ext_trg31	Reserved
adc_jext_trg0	exti15
adc_jext_trg1	tim1_oc4

Table 241. ADC1/2 interconnection

Signal name	Source/destination
adc_jext_trg2	Reserved
adc_jext_trg3	Reserved
adc_jext_trg4	tim1_trgo_cktim
adc_jext_trg5	tim1_trgo2_cktim
adc_jext_trg6	tim2_oc1
adc_jext_trg7	tim2_trgo_cktim
adc_jext_trg8	tim3_oc1
adc_jext_trg9	tim3_oc3
adc_jext_trg10	tim3_oc4
adc_jext_trg11	tim3_trgo_cktim
adc_jext_trg12	Reserved
adc_jext_trg13	tim4_trgo_cktim
adc_jext_trg14	tim5_trgo_cktim
adc_jext_trg15	tim6_trgo_cktim
adc_jext_trg16	tim7_trgo_cktim
adc_jext_trg17	tim8_trgo_cktim
adc_jext_trg18	tim8_trgo2_cktim
adc_jext_trg19	tim9_oc2
adc_jext_trg20	tim9_trgo_cktim
adc_jext_trg21	tim12_trgo_cktim
adc_jext_trg22	tim15_trgo_cktim
adc_jext_trg23	tim18_trgo_cktim
adc_jext_trg24	lptim1_ch2
adc_jext_trg25	lptim2_ch2
adc_jext_trg26	lptim3_ch2
adc_jext_trg27	Reserved
adc_jext_trg28	Reserved
adc_jext_trg29	Reserved
adc_jext_trg30	Reserved
adc_jext_trg31	Reserved

32.4.3 ADC clocks

Dual clock domain architecture

The dual clock-domain architecture means that the ADC kernel clock is independent from the AHB bus clock that is used to access ADC registers.

The adc_ker_ck input clock can be selected between different clock sources (see *Figure 262: ADC clock scheme*). This selection is done in the RCC (refer to section *Reset and clock control (RCC)* for more information):

1. The ADC clock can be provided by an internal or external clock source, which is independent and asynchronous from the AHB clock.
2. The ADC clock can be derived from AHB clock (selected in RCC).

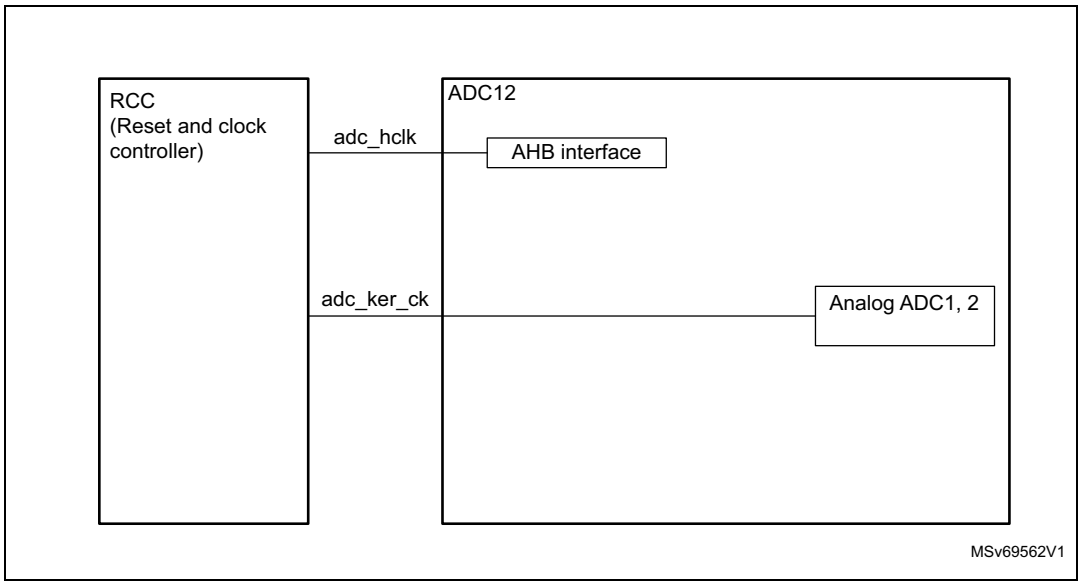
Option **1** has the advantage of achieving the maximum ADC clock frequency whatever the AHB clock scheme selected.

Option **2** corresponds to a pseudosynchronous clock. This can be useful when the ADC is triggered by a timer and the application requires that the ADC is accurately triggered without any uncertainty (otherwise, an uncertainty of the trigger instant time is added by the resynchronizations between the two clock domains). This accurate trigger is supported only by trgo or trgo2 timer triggers.

For further details on the synchronization between ADC and timers, refer to section *ADC synchronization* in each timer section, and to section *Peripheral clock distribution* in the RCC section.

The clock is configured through the RCC and must be compliant with the operating frequency specified in the device datasheet.

Figure 262. ADC clock scheme



Clock ratio constraint between ADC clock and AHB clock

There are no constraints to respect for the ratio between the ADC clock and the AHB clock. However, the ratio must be carefully chosen to avoid any overrun especially if the clock AHB is much slower than the ADC clock.

When adc_hclk operates at a higher frequency than adc_ker_ck, writing to the AHB register does not immediately update the ADC configuration due to clock domain crossing delays. To guarantee that the configuration is properly applied within the adc_ker_ck domain, it is necessary to wait for four adc_ker_ck clock cycles after the AHB register update.

32.4.4 ADC connectivity

ADC inputs are connected to external channels as well as internal sources as described below.

Figure 263. ADC1 connectivity

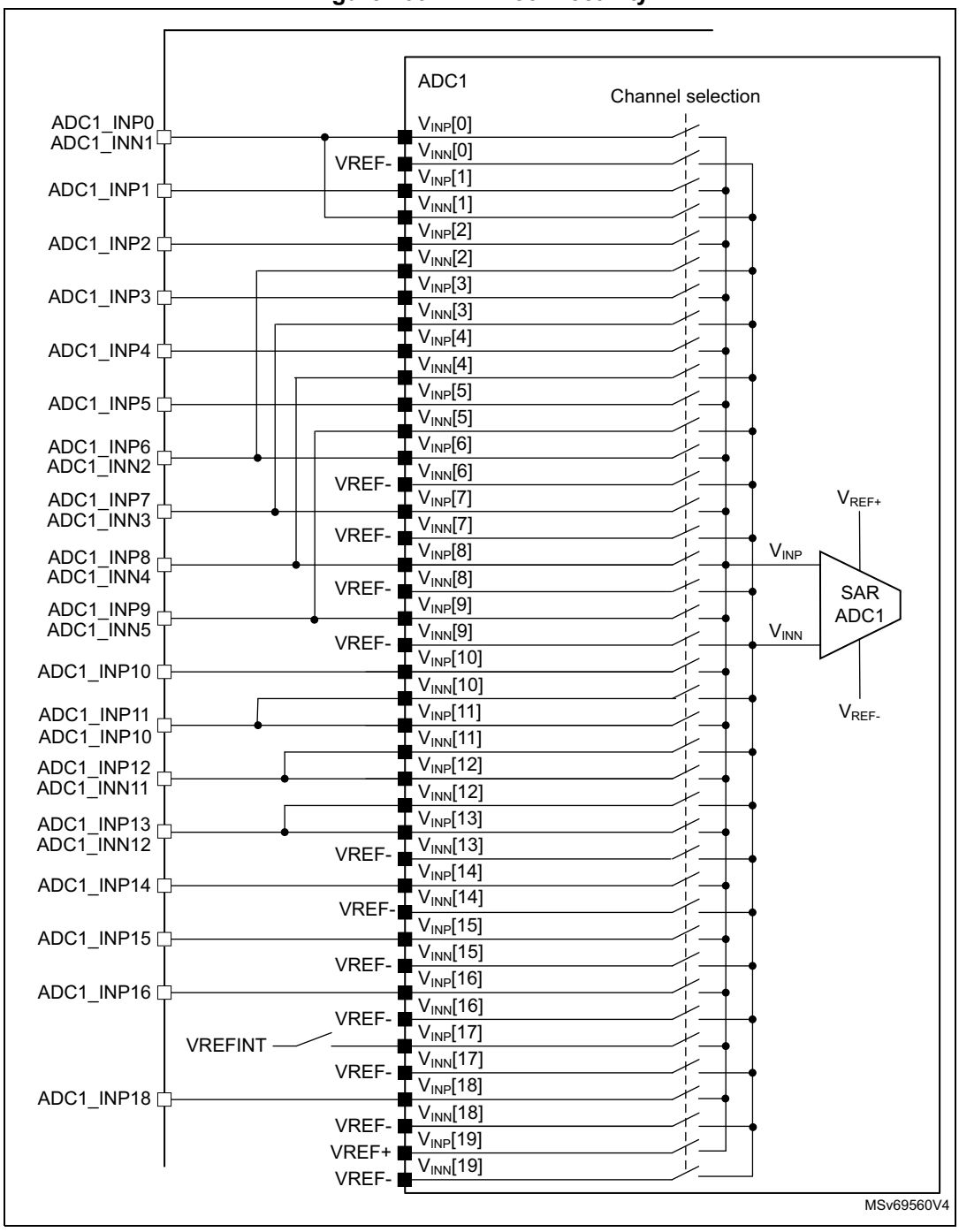
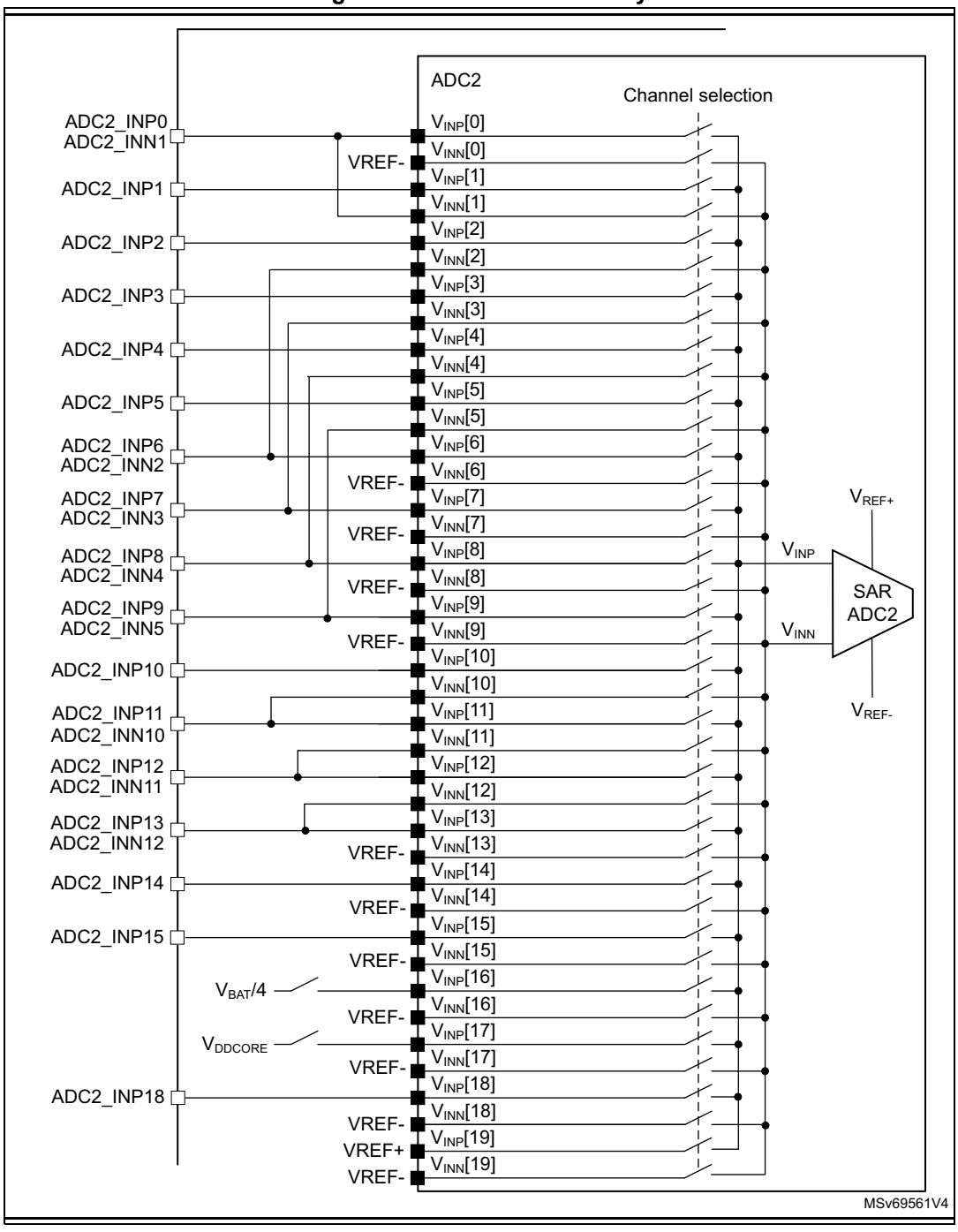


Figure 264. ADC2 connectivity



32.4.5 Slave AHB interface

The ADCs implement an AHB slave port for control/status register and data access. The features of the AHB interface are listed below:

- Word accesses
- Single cycle response
- Response to all read/write accesses to the registers with zero wait states.

The AHB slave interface does not support split/retry requests, and it can generate AHB error in case of wrong HSIZE.

32.4.6 ADC Deep-power-down mode (DEEPPWD)

By default, the ADC is in deep-power-down mode where its supply voltage is internally switched off to reduce the leakage currents (the reset state of DEEPPWD bit is 1 in the ADC_CR register).

To activate the ADC, first exit the Deep-power-down mode by clearing the DEEPPWD bit.

When ADC operations are complete, the ADC can be disabled by clearing the ADEN bit of the ADC_CR register.

Power can be saved by entering ADC Deep-power-down mode again (by setting DEEPPWD of ADC_CR register). This is particularly interesting before entering Stop mode.

32.4.7 Single-ended and differential input channels

ADC channels can be configured to be either single-ended inputs or differential inputs by programming DIFSEL[i] bits in the ADC_DIFSEL register. This configuration must be programmed when no ADC conversion is ongoing. Note that the DIFSEL[i] bits corresponding to single-ended channels must be programmed at 0.

In single-ended input mode, the analog voltage to be converted for channel "*i*" is the difference between the external voltage $V_{INP[i]}$ and V_{REF-}.

In differential input mode, the analog voltage to be converted for channel "*i*" is the difference between the external voltage $V_{INP[i]}$ (positive input) and $V_{INN[i]}$ (negative input).

In differential mode, the output data are unsigned data. When $V_{INP[i]}$ equals V_{REF-}, $V_{INN[i]}$ equals V_{REF+} and the output data is 0x000 (12-bit resolution mode). When $V_{INP[i]}$ equals V_{REF+}, $V_{INN[i]}$ equals V_{REF-} and the output data is 0xFFF.

$$\text{Converted value} = \frac{\text{ADC_Full_Scale}}{2} \times \left[1 + \frac{V_{INP} - V_{INN}}{V_{REF+}}\right]$$

When the ADC is configured in differential mode, the input signals are supposed to be differential (the common-mode voltage must be fixed at $V_{REF+}/2$ voltage).

Internal channels are used in single-ended mode only.

For a complete description of how the input channels are connected for each ADC, refer to *Section 32.4.4: ADC connectivity*.

Caution: When configuring channel "*i*" in differential input mode, its negative input voltage $V_{INN[i]}$ is connected to another channel. As a consequence, this channel is no longer usable in single-ended or differential mode, and must never be configured to be converted. Some channels

are shared between ADCs: this can make the channel on the other ADCs unusable. The only exception is interleaved mode for ADC master and the slave.

32.4.8 Calibration (ADCAL, CALADDOS, ADC_CALFACT)

A software calibration is required before launching the application. During the calibration sequence, the ADC calculates a 9-bit calibration factor, which is then internally applied to the ADC. Each ADC must follow the calibration procedure to calibrate the zero offset.

The calibration removes the offset error that may vary from chip to chip due to process or bandgap variation. It is a prerequisite to any ADC operation.

The calibration factor to be applied for single-ended input conversions is different from the factor to be applied for differential input conversions:

- Clear ADCALDIF before launching a calibration that is applied for single-ended input conversions.
- Set ADCALDIF before launching a calibration that is applied for differential input conversions.

The calibration needs to be run by combinations of hardware and software. When the calibration procedure is complete, the calibration factor must be written to the CALFACT_S[8:0] bits for the single-ended calibration factor and CALFACT_D[8:0] for the differential calibration factor.

The calibration factor can be written only if the ADC is enabled and no conversion is ongoing (ADEN = 1, ADSTART = 0 and JADSTART = 0). It is recommended to calibrate again the ADC if V_{REF+} voltage changes of more than 10% (see *Section 32.4.33: Monitoring the internal voltage reference*).

Software procedure to calibrate the ADC

1. Ensure DEEPPWD = 0, ADEN = 1 and wait until the ADRDY bit is set.
2. Set ADCAL and ensure CALADDOS = 0.
3. Select the calibration input mode by clearing ADCALDIF (single-ended input).
4. Set the ADSTART bit in the ADC_CR register.
5. Wait until the ADSTART bit is cleared or the EOC flag is set.
6. Read the ADC_DR register, then copy the converted data to the memory.
7. Repeat from step 4 several times (for example eight times).
8. Average the data stored in memory by dividing the accumulated data by the number of the conversions
9. If the averaged data is zero, set CALADDOS. Repeat all steps from step 4.
10. Store the averaged data to CALFACT_S[8:0].
11. Select the calibration input mode by setting ADCALDIF (differential input).
12. Keep the same CALADDOS setting as the one obtained during the single-end calibration.
13. Repeat steps 4 to 8.
14. Subtract 0x7FF from the averaged data. If the result is positive, store it in the CALFACT_D[8:0] bitfield. If it is negative, set CALADDOS, then repeat steps from 4 to 8.
15. Subtract again 0x7FF from the new averaged data. The resulting value is positive. Store it in CALFACT_D[8:0].
16. CALADDOS is now set, so clear ADCALDIF, and repeat steps 4 to 8.
17. Store the averaged data in CALFACT_S[8:0].
18. Clear ADCAL bit.

Software procedure to reinject a calibration factor into the ADC

1. Ensure ADSTART = 0 and JADSTART = 0 (no conversion ongoing).
2. Set ADCAL.
3. Program CALADDOS, CALFACT_S and CALFACT_D with the new calibration factors.
4. Clear ADCAL.

Note: CALFACT_S and CALFACT_D limit the maximum value of the ADC digital output. For example, in single-ended mode, if CALFACT_S = 100, the maximum ADC data output is 4095 - 100 = 3995. When the CALADDOS bit is set, an analog positive offset of about 256 is internally added to the ADC input to allow the calibration of negative offset devices. The resulting calibration factor is approximately 256. The maximum ADC output is around 4095 - 256 = 3839.

Converting single-ended and differential analog inputs with a single ADC

If the ADC has to convert both differential and single-ended inputs, two calibrations must be performed, one with ADCALDIF = 0 and one with ADCALDIF = 1.

32.4.9 ADC on-off control (ADEN, ADDIS, ADRDY)

First, follow the procedure described in *Section 32.4.6: ADC Deep-power-down mode (DEEPPWD)*).

Once DEEPPWD is cleared, the ADC can be enabled. It requires a t_{STAB} stabilization time before starting converting accurately (see *Figure 265*).

Two control bits enable or disable the ADC:

- Set ADEN to enable the ADC. The ADRDY flag is set once the ADC is ready for operation.
- Set ADDIS to disable the ADC. ADEN and ADDIS are automatically deasserted by hardware as soon as the analog ADC is effectively disabled.

Regular conversions can then start either by setting ADSTART (refer to *Section 32.4.20: Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN)*) or when an external trigger event occurs if regular triggers are enabled.

Injected conversions start by setting JADSTART or when an external injected trigger event occurs, if injected triggers are enabled.

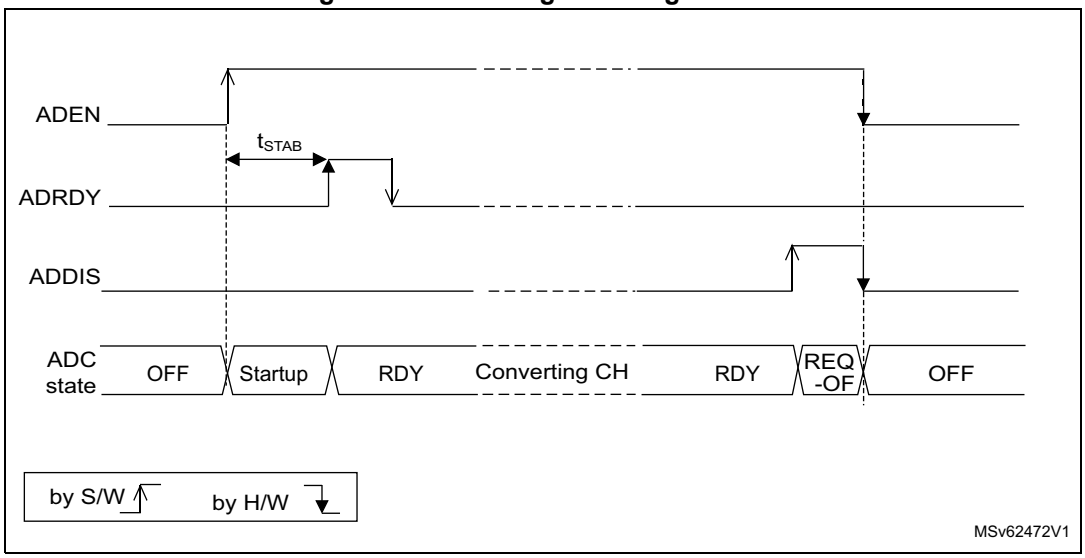
Software procedure to enable the ADC

1. Clear the ADRDY bit in the ADC_ISR register by programming it to 1.
2. Set ADEN.
3. Wait until ADRDY = 1 (ADRDY is set after the ADC startup time). This can be done by using the associated interrupt (ADRDYIE must be set).
4. Clear the ADRDY bit in the ADC_ISR register by programming it to 1 (optional).

Software procedure to disable the ADC

1. Check that both ADSTART = 0 and JADSTART = 0 to ensure that no conversion is ongoing. If required, stop all regular and injected ongoing conversions by setting ADSTP and JADSTP. Then wait until ADSTP = 0 and JADSTP = 0.
2. Set ADDIS.
3. If required by the application, wait for ADEN = 0, until the analog ADC is effectively disabled (ADDIS is automatically reset once ADEN = 0).

Figure 265. Enabling/disabling the ADC



32.4.10 Constraints when writing the ADC control bits

The software is allowed to write the RCC control bits to configure and enable the ADC clock (refer to RCC section) and configure the ADEN bit in the ADC_CR register, only if the ADC is disabled (ADEN must be cleared).

The software can write the ADSTART, JADSTART and ADDIS control bits in the ADC_CR register only if the ADC is enabled and there is no pending request to disable the ADC (ADEN must be equal to 1 and ADDIS to 0).

The following constraints apply to all the other control bits of the ADC_CFGRx, ADC_SMPRx, ADC_TRy, ADC_SQRy, ADC_JDRy, ADC_OFRy, ADC_OFCHRy and ADC_IER registers:

- For control bits related to configuration of regular conversions, the software is allowed to write them only if no regular conversions are ongoing (ADSTART must be cleared).
- For control bits related to configuration of injected conversions, the software is allowed to write them only if no injected conversions are ongoing (JADSTART must be cleared).
- ADC_TRy register thresholds can be modified when an analog-to-digital conversion is ongoing (refer to *Section 32.4.29: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWDCH of ADC_AWD2CR and ADC_AWD3CR, HTR, LTR, AWDFILT)* for details).
- To modify the all above control bits when the software trigger function is used, an ADSTP or JADSTP must be issued to make sure all activities are stopped, even if ADSTART and JADSTART are cleared.

The software is allowed to write the ADSTP or JADSTP control bits in the ADC_CR register only if the ADC is enabled and if there is no pending request to disable the ADC.

Note: Not all forbidden write accesses to ADC control bits are protected by hardware. In some cases, the ADC state may become unknown. To recover from this situation, the ADC must be disabled (ADEN = 0 as well as all the bits of the ADC_CR register).

32.4.11 Channel selection (ADC_SQRy, ADC_JSQR)

The ADC features up to 20 multiplexed channels per ADC, out of which:

- Up to 17 analog inputs coming from GPIO pads (ADCx_INN/INP[i]). Depending on the device, not all of them are available on GPIO pads.
- The ADC is connected to up to 4 internal analog inputs:
  - internal reference voltage (V_{REFINT})
  - V_{BAT} monitoring channel ($V_{BAT}/4$)
  - V_{REF+} positive reference voltage
  - V_{DDCORE} voltage

  To convert one of the internal analog channels, the corresponding analog sources must first be enabled by programming VREFEN, VBATEN in the ADCC_CCR registers, or by enabling the corresponding bit in the *ADC option register (ADC_OR)*.

Refer to *ADC interconnection* tables in *Section 32.4.2: ADC pins and internal signals* for the connection of the above internal analog inputs to ADC pins.

The conversions can be organized in two groups: regular and injected. A group consists of a sequence of conversions that can be done on any channel and in any order. For instance, it is possible to implement the conversion sequence in the following order: ADCx_INN/INP3, ADCx_INN/INP8, ADCx_INN/INP2, ADCx_INN/INP2, ADCx_INN/INP0, ADCx_INN/INP2, and ADCx_INN/INP2.

- A **regular group** is composed of up to 16 conversions. The regular channels and their order in the conversion sequence must be selected in the ADC_SQRy registers. The total number of conversions in the regular group must be written in the LEN[3:0] bits in the ADC_SQR1 register.
- An **injected group** is composed of up to four conversions. The injected channels and their order in the conversion sequence must be selected in the ADC_JSQR register. The total number of conversions in the injected group must be written in the JLEN[1:0] bits in the ADC_JSQR register.

ADC_SQRy registers must not be modified while a regular conversion is ongoing. ADC regular conversions must consequently be stopped by setting ADSTP (refer to *Section 32.4.19: Stopping an ongoing conversion (ADSTP, JADSTP)*).

32.4.12 Channel preselection register (ADC_PCSEL)

The PCSEL bit of the ADC_PCSEL register controls the analog switch integrated in the I/O.

For each channel selected through SQRx or JSQRx bits, the corresponding PCSEL bit must be configured in advance in the ADC_PCSEL register. The ADC input multiplexer selects the ADC input according to SQRx and JSQRx configuration with very high speed. The analog switch integrated in the I/O cannot react as fast as the ADC multiplexer. To avoid the delay due to the analog switch control on the I/O, it is necessary to preselect the input channels that are selected through the SQRx and JSQRx. The selection is based on the V_{INP} of each ADC input. For example, if the ADC converts ADCx_INN/INP1, PCSEL1 bit must also be set in ADC_PCSEL register. If ADCx_INN/INP1 is configured as a differential channel, both PCSEL1 and PCSEL0 must be set, which corresponds to ADC1_INN1 and ADC2_INN1.

Note: Configuring the PCSEL bit is not necessary for the internal channels (such as V_{REFINT}).

32.4.13 Channel-wise programmable sampling time (SMPR1, SMPR2)

Before starting a conversion, the ADC must establish a direct connection between the voltage source under measurement and the embedded sampling capacitor of the ADC. This sampling time must be sufficient for the input voltage source to charge the embedded capacitor to the input voltage level.

Each channel can be sampled with a different sampling time, which is programmable through the SMP[2:0] bits of the ADC_SMPR1 and ADC_SMPR2 registers. It is therefore possible to select among the following sampling time values:

- SMP = 000: 1.5 ADC clock cycles
- SMP = 001: 2.5 ADC clock cycles
- SMP = 010: 6.5 ADC clock cycles
- SMP = 011: 11.5 ADC clock cycles
- SMP = 100: 23.5 ADC clock cycles
- SMP = 101: 46.5 ADC clock cycles
- SMP = 110: 246.5 ADC clock cycles
- SMP = 111: 1499.5 ADC clock cycles

The total conversion time is calculated as follows (12-bit mode):

$$T_{CONV} = \text{Sampling time} + 13 \text{ ADC clock cycles}$$

Example

With $F_{adc_ker_ck}$ = 30 MHz and a sampling time of 1.5 ADC clock cycles:

$$T_{CONV} = (1.5 + 13.5) \times \text{ADC clock cycles} = 15 \text{ ADC clock cycles} = 500 \text{ ns}$$

The ADC notifies the end of the sampling phase by asserting the EOSMP flag (only for regular conversion).

Constraints on the sampling time

For each channel, SMP[2:0] bits must be programmed to respect a minimum and a maximum sampling times as specified in the ADC characteristics section of the datasheets.

Bulb sampling mode

The bulb sampling mode enables to obtain longer sampling time.

When the BULB bit is set in the ADC_CFGR2 register, the sampling period starts immediately after the last ADC conversion. A hardware or software trigger starts the conversion after the sampling time has been programmed in ADC_SMPRx registers. The very first ADC conversion after the ADC is enabled, is performed with the sampling time programmed in SMP bits. The bulb mode is effective starting from the second conversion.

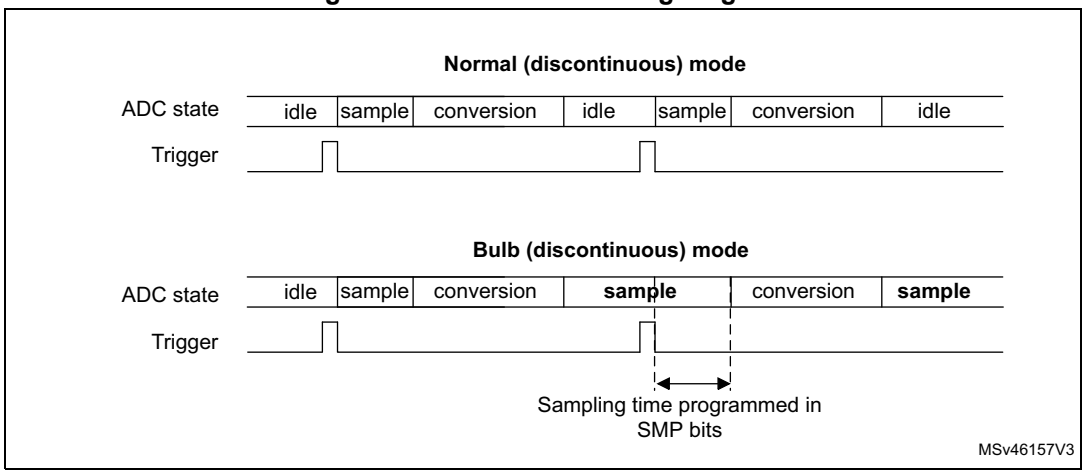
The maximum sampling time is limited (refer to the ADC characteristics section of the datasheet).

Bulb mode is supported exclusively in regular conversion; dual mode is not supported.

The BULB bit can be modified only when the ADEN bit of the ADC_CR register is cleared.

When the BULB bit is set, it is not allowed to set the SMPTRIG bit in ADC_CFGR2.

Figure 266. Bulb mode timing diagram



Sampling time control trigger mode

When the SMPTRIG bit is set, the sampling time programmed through SMPx bits is not applicable. The sampling time is controlled by the trigger signal edge.

When a hardware trigger is selected (EXTEN[1:0] = 01), each rising edge of the trigger signal starts the sampling period. A falling edge ends the sampling period and starts the conversion.

Due to the synchronization mechanism, the minimum allowed pulse width is seven adc_ker_ck periods.

When a software trigger is selected (EXTEN[1:0] = 00), the software trigger is not the ADSTART bit of ADC_CR but the SWTRIG bit. The SWTRIG bit has to be set to start the sampling period, and it has to be cleared to end the sampling period and start the conversion. In this mode, the minimum sampling time is limited to seven ADC clock cycles.

The maximum sampling time is limited (refer to the ADC characteristics section of the datasheet).

This mode is only compatible with the discontinuous mode. DISCNUM must be cleared for regular conversion mode. It is not supported for injected conversion mode and autoinjection mode.

When the SMPTRIG bit is set, setting the BULB bit is not allowed and only regular conversion is supported.

The sampling time control trigger mode is not compatible with the following dual modes: DUAL[4:0] = 0b00010, 0b00011, 0b00101, 0b00110, 0b00111, and 0b01001.

32.4.14 Single conversion mode

To enable the single conversion mode for regular channels, clear both CONT and DISCEN bits in ADC_CFGR1 register. To enable it on injected channel, clear JDISCEN bit.

In single conversion mode, the ADC performs once all the regular conversions of the programmed channels. This mode is started by one of the following events:

- ADSTART bit set in the ADC_CR register (for a regular channel)
- JADSTART bit set in the ADC_CR register (for an injected channel)
- An external hardware trigger event (for a regular or injected channel)

Inside the regular sequence, after each conversion is complete:

- The converted data are stored into the 32-bit 8-level FIFO accessible through the ADC_DR register,
- The EOC (end of regular conversion) flag is set, and
- An interrupt is generated if the EOCIE bit is set.

Inside the injected sequence, after each conversion is complete:

- The converted data are stored into one of the four 32-bit ADC_JDRy registers,
- The JEOC (end of injected conversion) flag is set, and
- An interrupt is generated if the JEOCIE bit is set.

After the regular sequence is complete:

- The EOS (end of regular sequence) flag is set, and
- An interrupt is generated if the EOSIE bit is set.

After the injected sequence is complete:

- The JEOS (end of injected sequence) flag is set, and
- An interrupt is generated if the JEOSIE bit is set.

The ADC then stops until a new external regular or injected trigger occurs or until the ADSTART or JADSTART bit is set again.

Note: To convert a single channel, program a sequence with a length of 1.

32.4.15 Continuous conversion mode (CONT = 1)

This mode applies to regular channels only.

In continuous conversion mode, when a software or hardware regular trigger event occurs, the ADC performs once all the regular conversions of the channels and then automatically restarts and continuously converts all the conversions of the sequence. This mode is started by setting the CONT bit of the ADC_CFGR1 register, either by an external trigger or by setting the ADSTART bit in the ADC_CR register.

Inside the regular sequence, after each conversion is complete:

- The converted data are stored into the eight 32-bit word FIFO accessible through the ADC_DR register.
- The EOC (end of conversion) flag is set, and
- An interrupt is generated if the EOCIE bit is set.

After the sequence of conversions is complete:

- The EOS (end of sequence) flag is set, and
- An interrupt is generated if the EOSIE bit is set.

Then, a new sequence restarts immediately and the ADC continuously repeats the conversion sequence.

Note: To convert a single channel, program a sequence with a length of 1.

It is not possible to have both discontinuous mode and continuous mode enabled, this means that it is forbidden to set both DISCEN and CONT bits.

Injected channels cannot be converted continuously. The only exception is when an injected channel is configured to be converted automatically after regular channels in continuous mode (using JAUTO bit) (refer to Section : Autoinjection mode).

32.4.16 Discontinuous mode (DISCEN, DISCNUM, JDISCEN)

Regular group mode

This mode is enabled by setting the DISCEN and CONT = 0 bit in the ADC_CFGR1 register.

It is used to convert a short sequence (subgroup) of n conversions (n ≤ 8) that is part of the sequence of conversions selected in the ADC_SQRy registers. The value of n is specified by writing to the DISCNUM[2:0] bits in the ADC_CFGR1 register.

When an external trigger occurs, it starts the next n conversions selected in the ADC_SQRy registers until all the conversions in the sequence are done. The total sequence length is defined by the LEN[3:0] bits in the ADC_SQR1 register.

Example:

- DISCEN = 1, n = 3, channels to be converted = 1, 2, 3, 6, 7, 8, 9, 10, 11
  - First trigger: channels converted are 1, 2, 3 (an EOC event is generated at the end of each conversion).
  - Second trigger: channels converted are 6, 7, 8 (an EOC event is generated at the end of each conversion).
  - Third trigger: channels converted are 9, 10, 11 (an EOC event is generated at the end of each conversion) and an EOS event is generated after the conversion of channel 11.
  - Fourth trigger: channels converted are 1, 2, 3 (an EOC event is generated at the end of each conversion).
  - ...
- DISCEN = 0, channels to be converted = 1, 2, 3, 6, 7, 8, 9, 10,11
  - First trigger: the complete sequence is converted: channel 1, then 2, 3, 6, 7, 8, 9, 10 and 11. Each conversion generates an EOC event and the last one also generates an EOS event.
  - All the next trigger events relaunch the complete sequence.

Note: The channel numbers referred to in the above example might not be available on all devices.

When a regular group is converted in discontinuous mode, no rollover occurs (the last subgroup of the sequence can have less than n conversions).

When all subgroups are converted, the next trigger starts the conversion of the first subgroup. In the example above, the fourth trigger reconverts the channels 1, 2 and 3 in the first subgroup.

It is not possible to have both discontinuous mode and continuous mode enabled. In this case (if DISCEN = 1, CONT = 1), the ADC behaves as if continuous mode was disabled.

Injected group mode

This mode is enabled by setting the JDISCEN bit in the ADC_CFGR1 register. It converts the sequence selected in the ADC_JSQR register, channel by channel, after an external injected trigger event. This is equivalent to discontinuous mode for regular channels with DISCNUM = 0.

When an external trigger occurs, it starts the next channel conversions selected in the ADC_JSQR registers until all the conversions in the sequence are done. The total sequence length is defined by the JLEN[1:0] bits in the ADC_JSQR register.

Example:

- JDISCEN = 1, channels to be converted = 1, 2, 3
  - First trigger: channel 1 converted (a JEOC event is generated)
  - Second trigger: channel 2 converted (a JEOC event is generated)
  - Third trigger: channel 3 converted and a JEOC event + a JEOS event are generated
  - ...

Note: The channel numbers referred to in the above example might not be available on all devices.

When all injected channels have been converted, the next trigger starts the conversion of the first injected channel. In the example above, the fourth trigger reconverts the first injected channel 1.

It is not possible to use both autoinjection mode and discontinuous mode simultaneously: the bits DISCEN and JDISCEN must be kept cleared by software when JAUTO is set.

32.4.17 Starting conversions (ADSTART, JADSTART)

ADC regular conversions can be started by setting ADSTART.

When ADSTART is set, the conversion starts:

- Immediately if EXTEN[1:0] = 00 (software trigger), or
- At the next active edge of the selected regular hardware trigger, if EXTEN[1:0] is not equal to 00.

The application software starts ADC injected conversions by setting JADSTART.

When JADSTART is set, the conversion starts:

- Immediately if JEXTEN[1:0] = 00 (software trigger), or
- At the next active edge of the selected injected hardware trigger, if JEXTEN[1:0] is not equal to 00.

Note: In autoinjection mode (JAUTO = 1), use the ADSTART bit to start regular conversions followed by autoinjected conversions (JADSTART must be kept cleared).

When the hardware trigger is enabled, ADSTART and JADSTART also indicate whether an ADC operation is ongoing. The ADC can be reconfigured while ADSTART and JADSTART are both cleared, indicating that the ADC is idle.

In the case of software triggering, ADSTP and/or JADSTP must be set before ADC reconfiguration.

ADSTART is deasserted by hardware:

- In single mode with software regular trigger (CONT = 0, EXTEN = 0x0): at any end of regular conversion sequence (EOS assertion)
- In discontinuous mode (CONT = 0, DISCEN = 1, DISCNUM= x): at any of subgroup sequence
- In all cases (CONT and EXTEN don't care), after execution of the ADSTP procedure asserted by the software

Note: In continuous mode (CONT = 1), ADSTART is not deasserted by hardware when EOS is asserted because the sequence is automatically relaunched.

When a hardware trigger is selected in single mode (CONT = 0 and EXTEN ≠ 0x00), ADSTART is not deasserted by hardware when EOS is asserted to help the software that does not need to reset ADSTART again for the next hardware trigger event. This ensures that no further hardware triggers are missed.

JADSTART is deasserted by hardware:

- In single mode with software injected trigger (JEXTEN = 0x0): at the end of each injected conversion sequence (JEOS assertion)
- In discontinuous mode (JDISCEN = 1): at the end of any conversion
- In all cases (JEXTEN=x), after execution of the JADSTP procedure asserted by the software

Note: When the software trigger is selected, the ADSTART bit must not be set if the EOC flag is still high.

In bulb mode (BULB = 1), the previous end-of-conversion automatically starts the sampling, then setting ADSTART starts the conversion including the programmed sampling time.

In sampling time control trigger mode (SMPTRIG = 1), SWTRIG bit must be set to start the conversion.

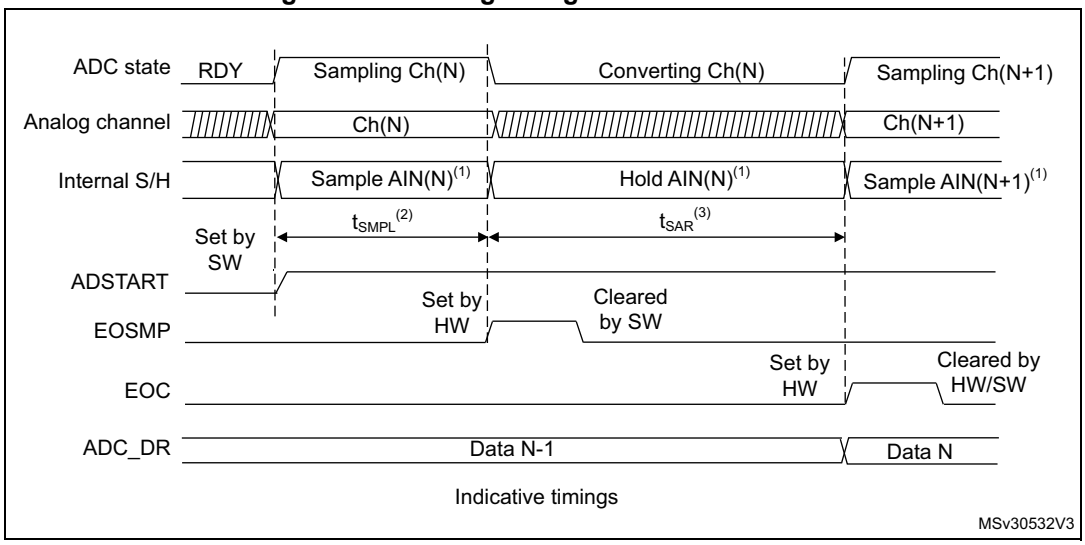
32.4.18 Timing

The elapsed time between the start of a conversion and the end of a conversion is the sum of the configured sampling time plus the successive approximation time depending on data resolution:

$T_{CONV} = T_{SMPL} + T_{SAR} = [1.5\ _{|min} + 13.5\ _{|12bit}] \times T_{ADC_CLK}$

$T_{CONV} = T_{SMPL} + T_{SAR} = 50\ ns\ _{|min} + 450\ ns\ _{|12bit} = 500\ ns$ (for $F_{adc_ker_ck} = 30$ MHz)

Figure 267. Analog-to-digital conversion time



1. AIN represents the input voltage captured by the internal Sample and Hold (S/H) capacitor.
2. t_{SMPL} depends on SMP[2:0].
3. t_{SAR} depends on RES[2:0].

Note: The bulb and the sampling time control trigger modes are not described in Figure 267: Analog-to-digital conversion time.

32.4.19 Stopping an ongoing conversion (ADSTP, JADSTP)

The software can decide to stop ongoing regular conversions by setting ADSTP and ongoing injected conversions by setting JADSTP.

Stopping conversions resets the ongoing ADC operation. The ADC can then be reconfigured (for example by changing the channel selection or the trigger). It is then ready for a new operation.

Injected conversions can be stopped while regular conversions are still ongoing and vice versa. This allows, for instance, the reconfiguration of the injected conversion sequence and triggers while regular conversions are still ongoing (and vice versa).

When the ADSTP bit is set by software, any ongoing regular conversion is aborted with the partial result discarded (the ADC_DR register is not updated with the current conversion).

When the JADSTP bit is set by software, any ongoing injected conversion is aborted with partial result discarded (the ADC_JDRy register is not updated with the current conversion). The scan sequence is also aborted and reset (meaning that relaunching the ADC restarts a new sequence).

Once this procedure is complete, ADSTP/ADSTART bits (for regular conversion), or JADSTP/JADSTART bits (for injected conversion) are deasserted by hardware and the software must poll ADSTART (or JADSTART) until the bit is reset before assuming the ADC is completely stopped.

Note: In autoinjection mode (JAUTO = 1), setting the ADSTP bit aborts both regular and injected conversions (JADSTP must not be used).

Figure 268. Stopping ongoing regular conversions

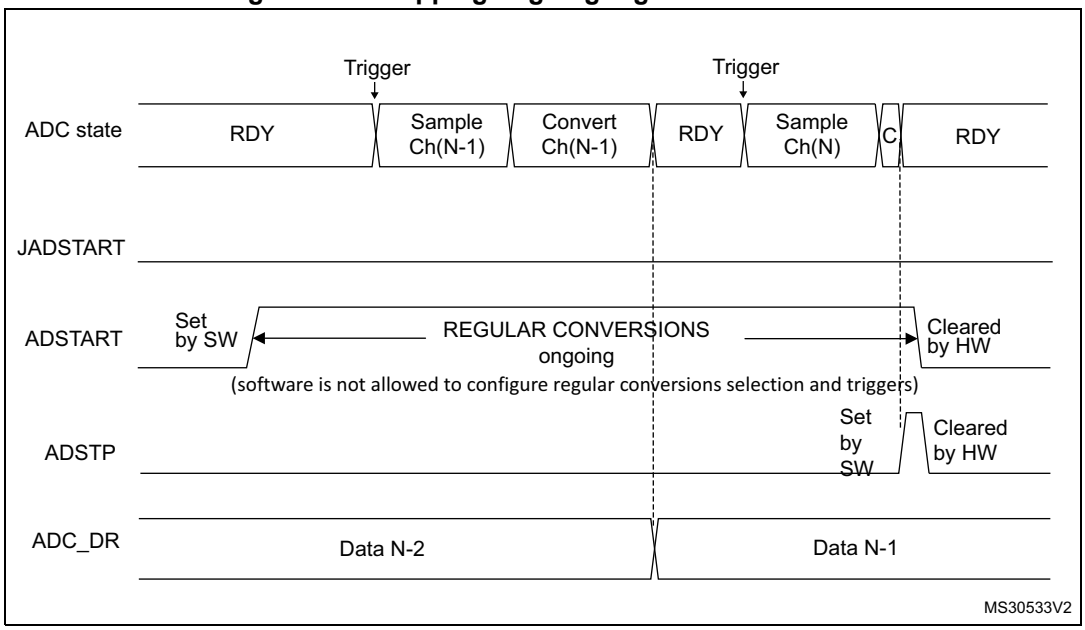
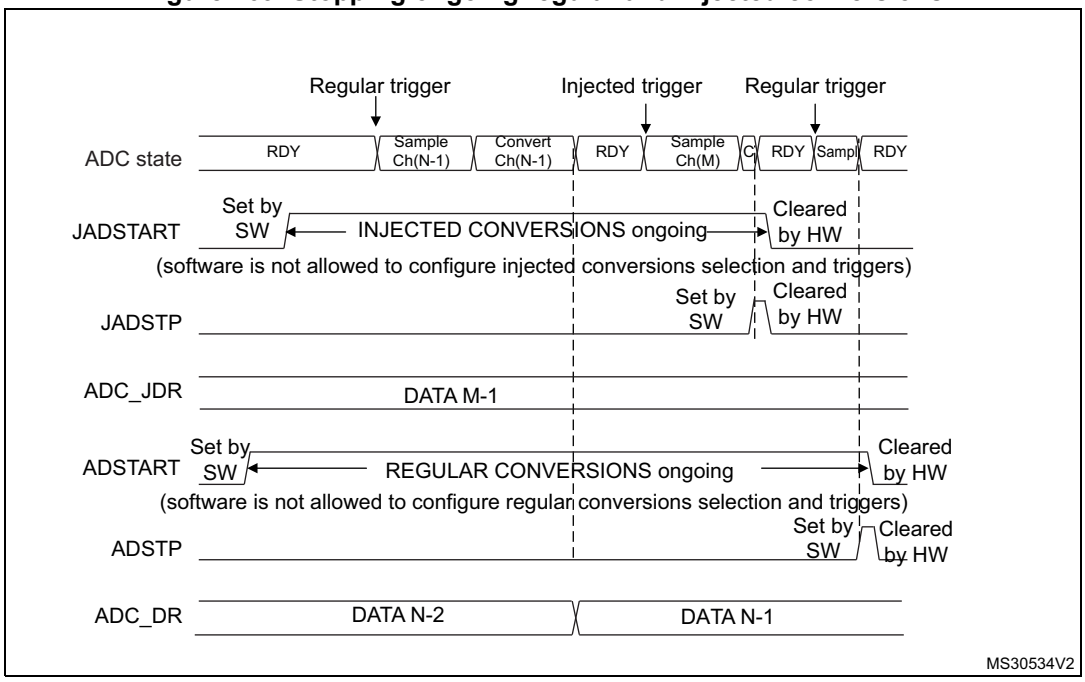


Figure 269. Stopping ongoing regular and injected conversions



32.4.20 Conversion on external trigger and trigger polarity (EXTSEL, EXTEN, JEXTSEL, JEXTEN)

A conversion or a sequence of conversions can be triggered either by software or by an external event (for example timer capture, input pins). If the EXTEN[1:0] control bits (for a regular conversion) or JEXTEN[1:0] bits (for an injected conversion) are different from 0b00, then external events are able to trigger a conversion with the selected polarity.

The regular trigger selection is effective once the software has set ADSTART while the injected trigger selection is effective once the software has set JADSTART bit.

All hardware triggers that occur while a conversion is ongoing are ignored:

- If ADSTART = 0, regular hardware triggers are ignored.
- If JADSTART = 0, injected hardware triggers are ignored.

Table 242 provides the correspondence between the EXTEN[1:0] and JEXTEN[1:0] values and the trigger polarity.

Table 242. Configuring the trigger polarity for regular external triggers

EXTEN[1:0]	Source
00	Hardware trigger detection disabled, software trigger detection enabled
01	Hardware trigger with detection on the rising edge (sampling time controlled in SMPTRIG mode)
10	Hardware trigger with detection on the falling edge
11	Hardware trigger with detection on both the rising and falling edges

Note: The polarity of the regular trigger cannot be changed on-the-fly.

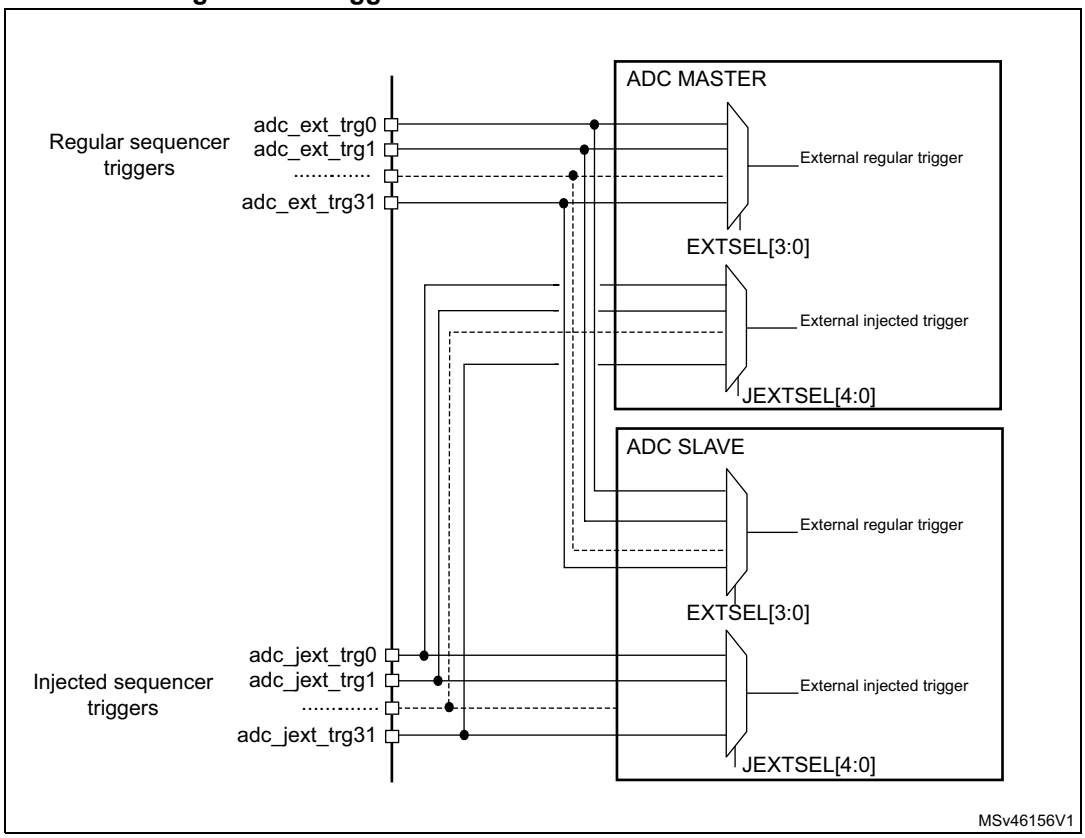
Table 243. Configuring the trigger polarity for injected external triggers

JEXTEN[1:0]	Source
00	Hardware Trigger detection disabled, software trigger detection enabled
01	Hardware Trigger with detection on the rising edge
10	Hardware Trigger with detection on the falling edge
11	Hardware Trigger with detection on both the rising and falling edges

The EXTSEL and JEXTSEL control bits select which, out of 32 possible events, can trigger the conversion of the regular and injected groups.

A regular group conversion can be interrupted by an injected trigger.

Figure 270. Triggers shared between ADC master and slave



Refer to *ADC interconnection* tables in *Section 32.4.2: ADC pins and internal signals* for the list of the external triggers.

32.4.21 Injected channel management

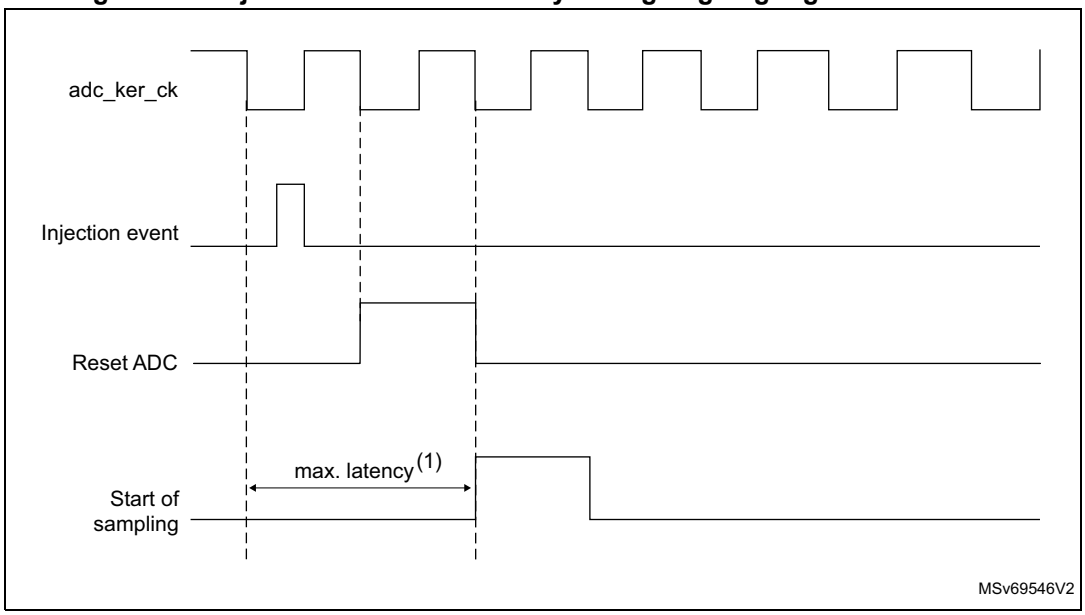
Triggered injection mode

To use triggered injection, the JAUTO bit in the ADC_CFGR1 register must be cleared.

1. Start the conversion of a group of regular channels either by an external trigger or by setting the ADSTART bit in the ADC_CR register.
2. If an external injected trigger occurs, or if the JADSTART bit in the ADC_CR register is set during the conversion of a regular group of channels, the current conversion is reset and the injected channel sequence switches are launched (all the injected channels are converted once if JDISCEN = 0).
3. Then, the regular conversion of the regular group of channels is resumed from the last interrupted regular conversion.
4. If a regular event occurs during an injected conversion, the injected conversion is not interrupted but the regular sequence is executed at the end of the injected sequence. *Figure 271* shows the corresponding timing diagram.

Note: When using triggered injection, one must ensure that the interval between trigger events is longer than the injection sequence.

Figure 271. Injected conversion latency during ongoing regular conversion



1. The maximum latency value can be found in the electrical characteristics of the device datasheet.

Autoinjection mode

If the JAUTO bit in the ADC_CFGR1 register is set, then the channels in the injected group are automatically converted after the regular group of channels. This can be used to convert a sequence of up to 20 conversions programmed in the ADC_SQRy and ADC_JSQR registers.

In this mode, the ADSTART bit in the ADC_CR register must be set to start regular conversions, followed by injected conversions (JADSTART must be kept cleared). Setting the ADSTP bit aborts both regular and injected conversions (JADSTP bit must not be used).

In this mode, external trigger on injected channels must be disabled.

Note: Autoinjection mode is compatible with discontinuous mode.

32.4.22 Programmable resolution (RES) - fast conversion mode

Faster conversions can be performed by reducing the ADC resolution.

The resolution can be configured to be either 12, 10, 8, or 6 bits by programming the RES[1:0] control bits. *Figure 276*, *Figure 277*, *Figure 278* and *Figure 279* show the conversion result format with respect to the resolution as well as to the data alignment.

Lower resolution allows faster conversion time for applications where high-data precision is not required. It reduces the conversion time spent by the successive approximation steps according to *Table 244*.

Table 244. T_{SAR} timings depending on resolution

RES (bits)	T_{SAR} (ADC clock cycles)	T_{SAR} (ns) at $F_{adc_ker_ck}$ = 30 MHz	T_{CONV} (ADC clock cycles) (with Sampling time= 1.5 ADC clock cycles)	T_{CONV} (ns) at $F_{adc_ker_ck}$ = 30 MHz
12	13.5 ADC clock cycles	450 ns	15 ADC clock cycles	500.0 ns
10	11.5 ADC clock cycles	383.33 ns	13 ADC clock cycles	433.33 ns
8	8.5 ADC clock cycles	203.33 ns	10 ADC clock cycles	333.33 ns
6	6.5 ADC clock cycles	216.67 ns	8 ADC clock cycles	266.66 ns

32.4.23 End of conversion and end of sampling phase (EOC, JEOC, EOSMP)

The ADC notifies the application at the end of each regular conversion (EOC) event and injected conversion (JEOC) event.

The ADC sets the EOC flag as soon as a new regular conversion data is available in the ADC_DR register. An interrupt can be generated if the EOCIE bit is set. The software can clear the EOC flag either by programming it to 1 or by reading ADC_DR. If ADC_DR FIFO is not emptied, the EOC flag is set again and an interrupt can be generated again.

The ADC sets the JEOC flag as soon as new injected conversion data is available in one of the ADC_JDRy registers. An interrupt can be generated if the JEOCIE bit is set. The software can clear the JEOC flag either by programming it to 1 or by reading the corresponding ADC_JDRy register.

The ADC also notifies the end of the sampling phase by setting the EOSMP flag (for regular conversions only). The EOSMP flag is cleared by software by programming it to 1. An interrupt can be generated if the EOSMPIE bit is set.

32.4.24 End of conversion sequence (EOS, JEOS)

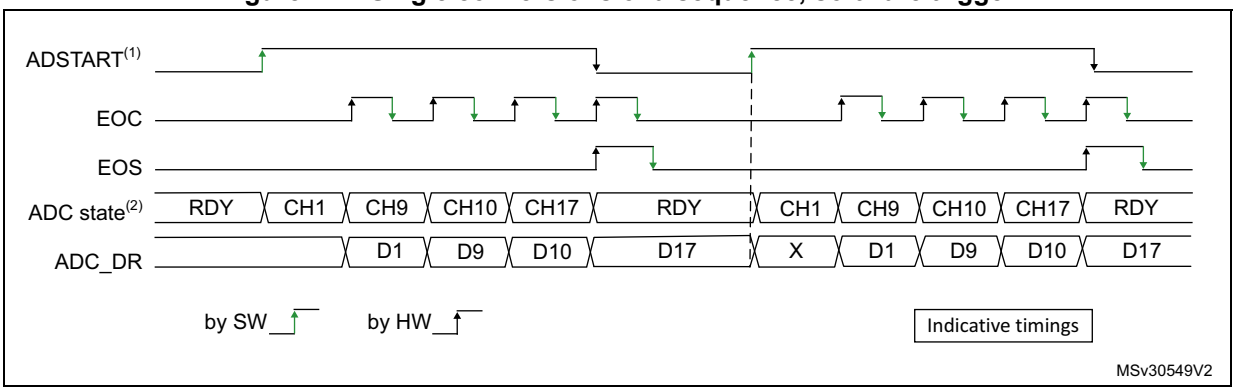
The ADC notifies the application at the end of each regular sequence (EOS) and injected sequence (JEOS) event.

The ADC sets the EOS flag as soon as the conversion of the last data of the regular sequence is complete. An interrupt can be generated if the EOSIE bit is set. The EOS flag is cleared by the software by programming it to 1.

The ADC sets the JEOS flag as soon as the conversion of the last data of the injected sequence is complete. An interrupt can be generated if the JEOSIE bit is set. The JEOS flag is cleared by the software by programming it to 1.

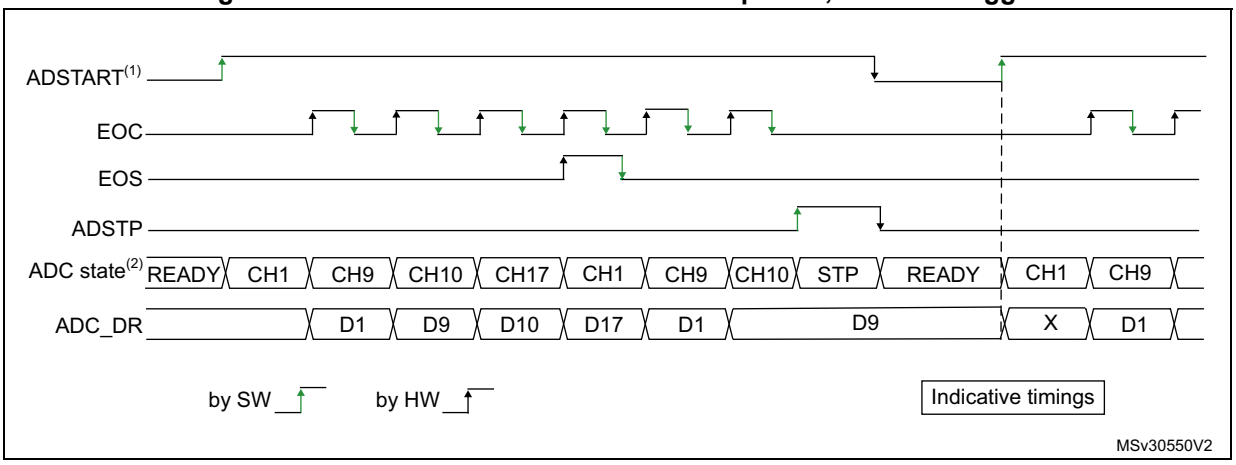
32.4.25 Timing diagram examples (single/continuous modes, hardware/software triggers)

Figure 272. Single conversions of a sequence, software trigger



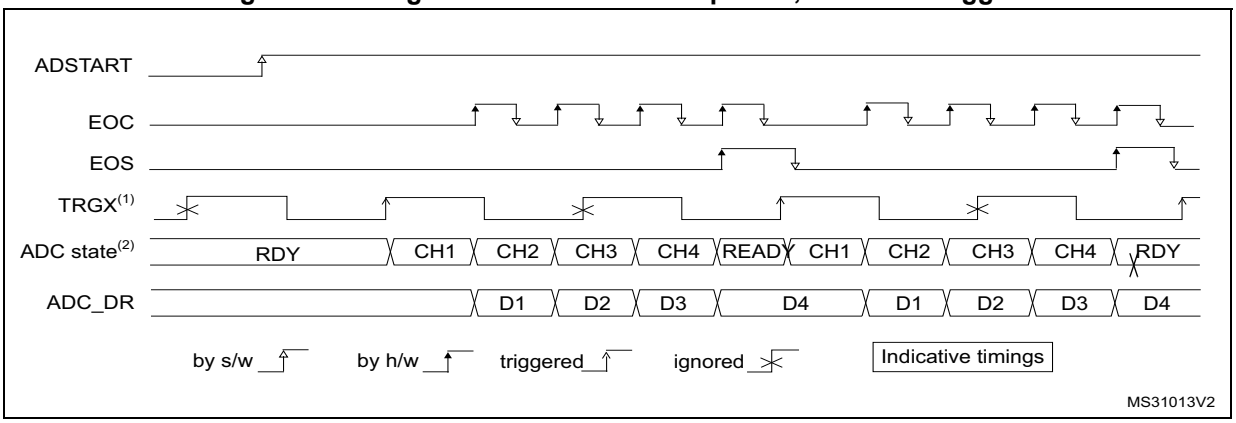
1. EXTEN[1:0] = 00, CONT = 0
2. Channels selected = 1, 9, 10, 17; AUTDLY = 0.

Figure 273. Continuous conversion of a sequence, software trigger



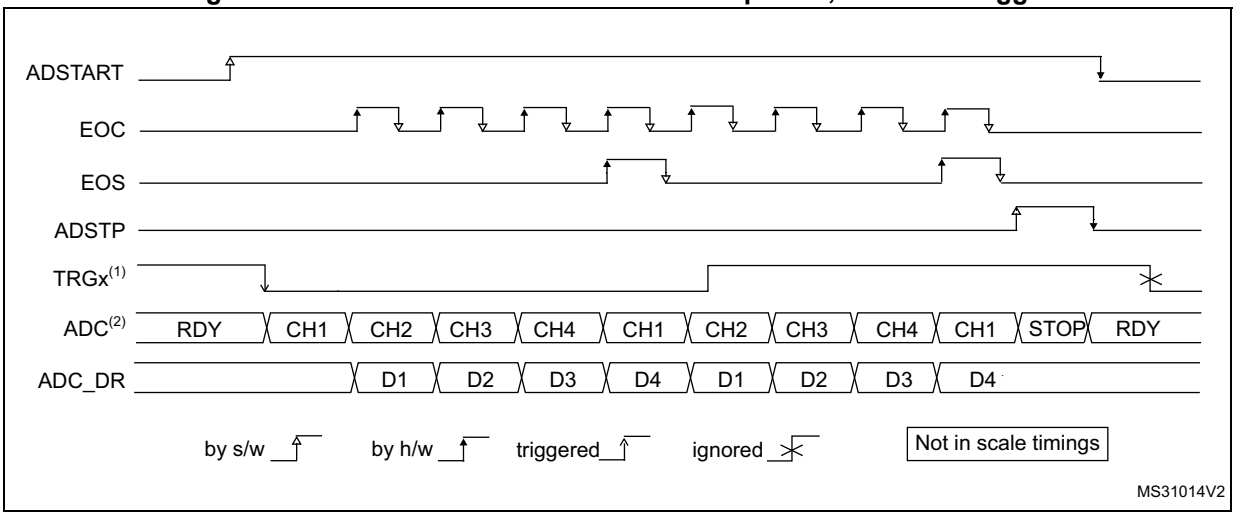
1. EXTEN[1:0] = 00, CONT = 1
2. Channels selected = 1, 9, 10, 17; AUTDLY = 0.

Figure 274. Single conversions of a sequence, hardware trigger



1. TRGx (overfrequency) is selected as trigger source, EXTEN[1:0] = 01, CONT = 0
2. Channels selected = 1, 2, 3, 4; AUTDLY = 0.

Figure 275. Continuous conversions of a sequence, hardware trigger



1. TRGx is selected as trigger source, EXTEN[1:0] = 10, CONT = 1.
2. Channels selected = 1, 2, 3, 4; AUTDLY = 0.

32.4.26 Data management

Data register, data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, JOVSS[a], JLSHIFT[a], USAT, SSAT, POSOFF)

Data and alignment

At the end of each regular channel conversion (when an EOC event occurs), the result of the converted data is stored in the 32-bit wide ADC_DR data register with FIFO (when OVRMOD = 0). Since the data register has eight levels of FIFO, the EOC flag is raised again as far as the FIFO is not emptied.

At the end of each injected channel conversion (when a JEOC event occurs), the result of the converted data is stored in the 32-bit wide corresponding ADC_JDRy data register.

The OVSS[3:0], LSHIFT[3:0], JOVSS[3:0], and JLSHIFT[3:0] bitfields in the ADC_CFGR2 register select the alignment of the data stored after the conversion. By default, data are right-aligned. Refer to *Figure 276*, *Figure 277*, *Figure 278*, and *Figure 279* for examples of data alignment.

Note: The data can be aligned in normal and oversampling mode. The OVSS[3:0] bitfield is only effective in oversampling mode.

JOVSS[3:0] and JLSHIFT[3:0] is reserved for injected conversions. Their usage is the same as OVSS[3:0] and LSHIFT[3:0], unless otherwise specifically described.

Offset

An unsigned offset value y (y = 1, 2, 3, 4) can be applied to a channel by programming a value different from 0 in the OFFSET[21:0] bitfield of the ADC_OFRy register. The channel to which the offset is applied is programmed to the OFFSETy_CH[4:0] bits of ADC_OFCFGRy register. The offset can be positive or negative depending on the value of POSOFF bit. When POSOFF is cleared, the converted value is subtracted by the user-defined offset written in OFFSET[21:0] bits. The result can be a negative value. The read data is consequently signed and the SEXT bit represents the extended sign value.

The offset value must be lower than the maximum conversion value (for example the maximum offset value is 0xFFF in 12-bit mode).

The offset can be used to convert unsigned data to signed data (for example the offset value is equal to 0x800 in 12-bit mode).

The offset correction is also supported in oversampling mode. In this mode, the offset is subtracted when all the oversampling operations are complete.

Note: If the OFFSET channel is selected but the offset value is 0x0000, this offset programming is ignored. If the same OFFSET channel is programmed in different registers multiple times, the first programmed register takes priority.

Table 250 describes how the computation is performed for all the possible ADC resolutions and offset settings.

a. Only for products that include this bit.

Table 245. Offset computation versus data resolution

Resolution (RES[1:0] bits)	Subtraction between raw converted data and offset		Result	Comments
	Raw converted Data, left aligned	Offset		
00: 12-bit	DATA[11:0]	OFFSET[11:0]	Signed or unsigned 24-bit data, right aligned to [11:0]	-
01: 10-bit	DATA[11:2],00	OFFSET[11:2]	Signed or unsigned 24-bit data, right aligned to [9:0]	The user must configure OFFSET[1:0] to 0b00.
10: 8-bit	DATA[11:4],0000	OFFSET[11:4]	Signed or unsigned 24 bit data right aligned to [7:0]	The user must configure OFFSET[3:0] to 0b0000.
11: 6-bit	DATA[11:6],000000	OFFSET[11:6]	Signed or unsigned 24 bit data right aligned to [5:0]	The user must configure OFFSET[5:0] to 0b000000.

Figure 276, *Figure 277*, *Figure 278* and *Figure 279* show alignments for signed and unsigned data. JOVSS and JLSHIFT are not described here, but they behave the same as OVSS and LSHIFT.

Figure 276. Right alignment (offset disabled, unsigned value)

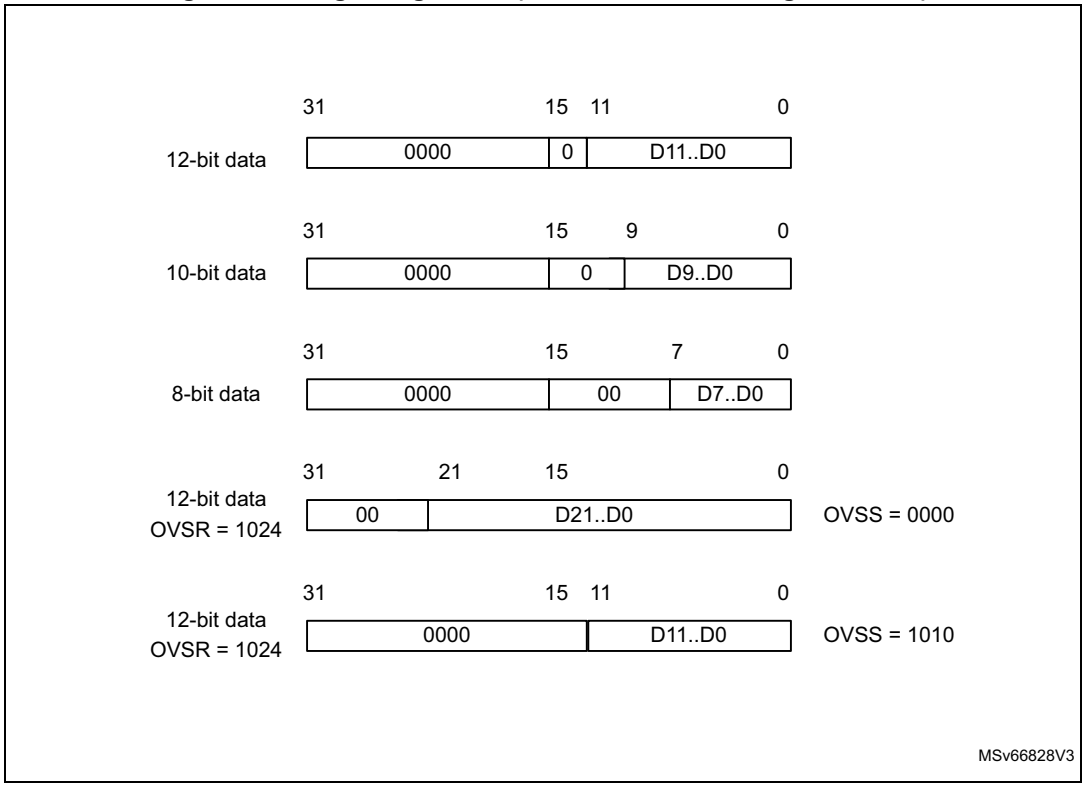


Figure 277. Right alignment (offset enabled, signed value)

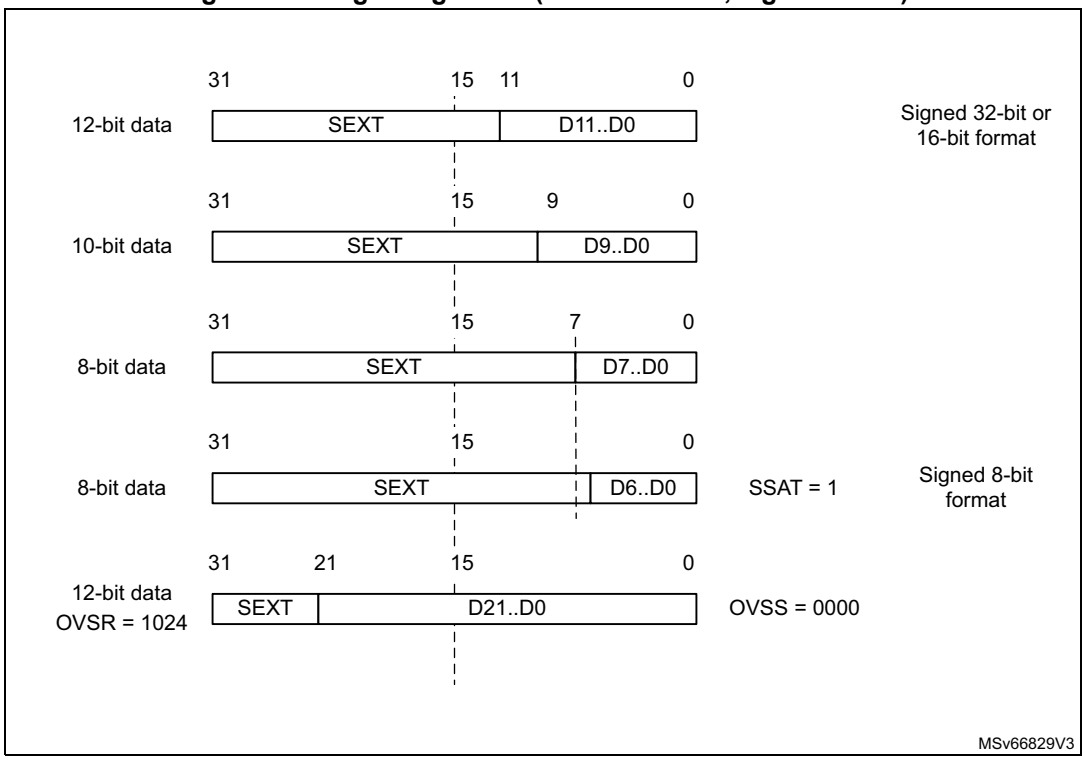


Figure 278. Left alignment (offset disabled, unsigned value)

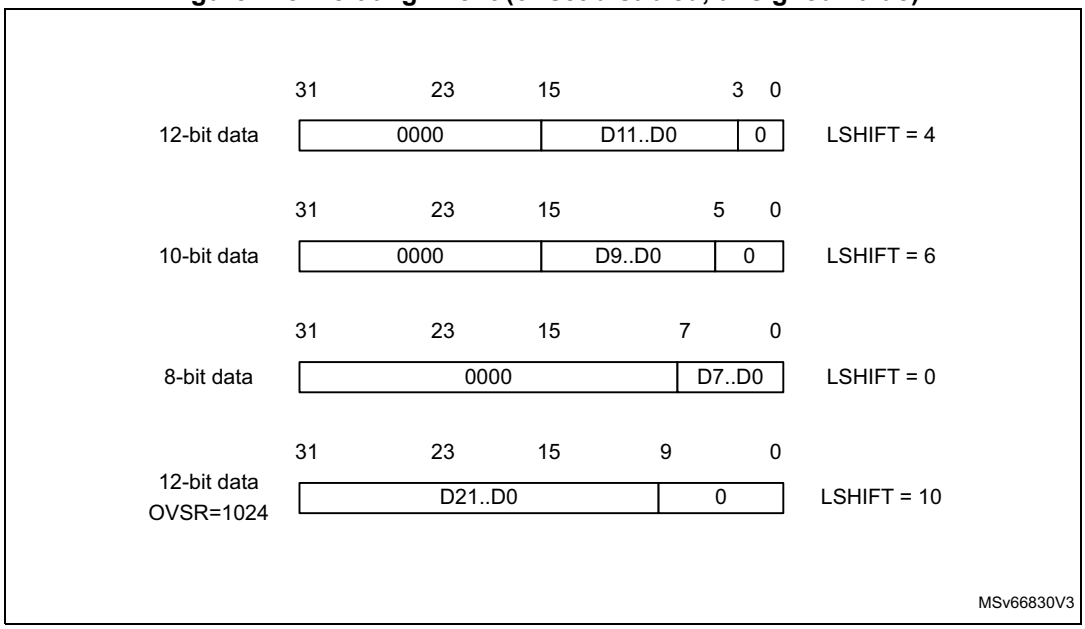
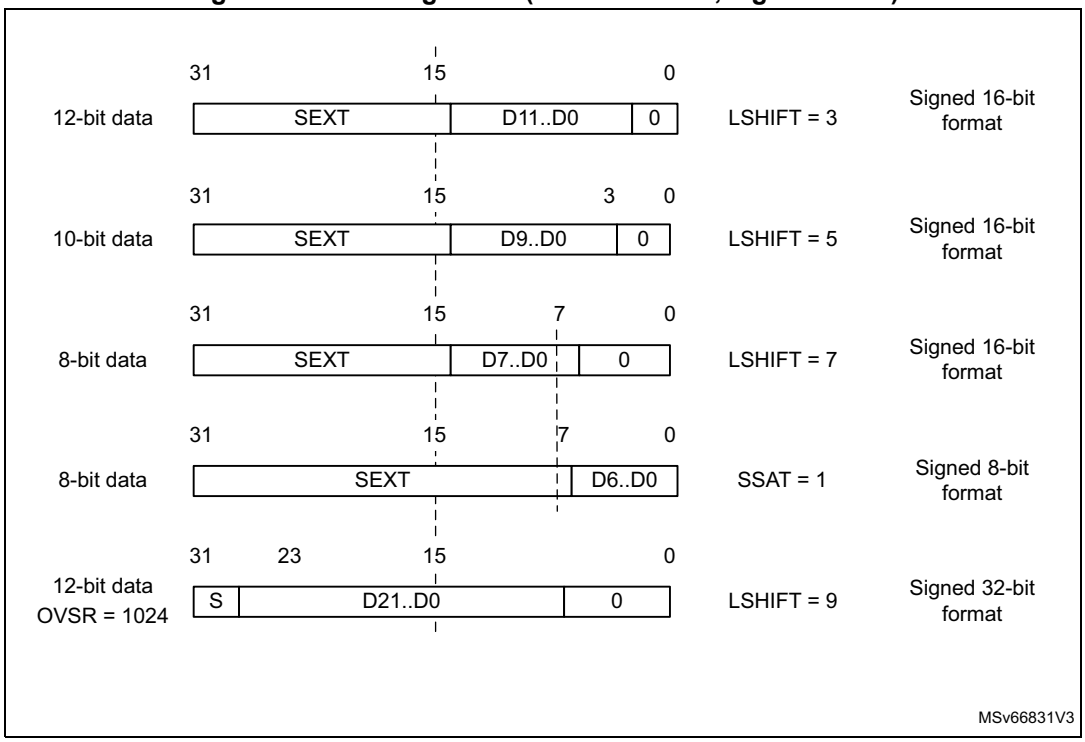


Figure 279. Left alignment (offset enabled, signed value)



Management of signed and unsigned saturation format (SSAT, USAT)

The offset correction might result in the data width to be wider than the original data.

To limit the original data width, the data saturation can be enabled through the SSAT and USAT bits of the ADC_OFCFGRy register.

Unsigned 12-bit data can be extended to 13-bit signed data by using an offset value different from 0x800.

The original data width can be preserved by setting SSAT bit to limit the data width to 12 bits.

Unsigned data can be saturated to the original data width by setting the USAT bit.

Table 246 shows the sign-extended data format corresponding to different resolutions.

Table 246.12-bit data formats

SSAT	USAT	Format	Data range (offset =0x800)
0	0	Sign-extended 13-bit significant data: SEXT[31:13] DATA[12:0]	0x0000 07FF - 0xFFFF F800
1	0	Sign-extended 12-bit significant data: SEXT[31:12] DATA[11:0]	0x7FF - 0x800

Table 246.12-bit data formats

SSAT	USAT	Format	Data range (offset =0x800)
0	1	Unsigned saturation 12-bit signification data: DATA[11:0]	0xFFF - 0x000
1	1	Reserved	-

Table 247 provides numerical examples for four different offset values.

Table 247. Numerical examples for 32-bit or 16-bit format (POSSPFF = 0)

Raw conversion result	Offset value	Result SSAT = 0 USAT = 0	Result SSAT = 0 USAT = 1	Result SSAT = 1 USAT = 0
0xFFF	0x800	0x0000 07FF	0x07FF	0x07FF
0x800		0x0000 0000	0x0000	0x0000
0x000		0xFFFF F800	0x0000	0xF800
0xFFF	0x820	0x0000 07DF	0x07DF	0x07DF
0x800		0xFFFF FFE0	0x0000	0xFFE0
0x000		0xFFFF F7E0	0x0000	0xF7E0
0xFFF	0x7E0	0x0000 081F	0x081F	0x07FF
0x800		0x0000 0020	0x0020	0x0020
0x000		0xFFFF F820	0x0000	0xF820
0xFFF	0x020	0x0000 0FDF	0x0FDF	0x0FDF
0x800		0x0000 07E0	0x07E0	0x07E0
0x000		0xFFFF FFE0	0x0000	0xFFE0

Caution: SSAT must not be used in conjunction with USAT. No hardware check is performed to ensure that this condition is respected.

Gain compensation

When the GCOMP bit is set in the ADC_GCOMP register, the gain compensation is activated on all the converted data. After each conversion, data is calculated using the following formula.

$$DATA = DATA(\text{adc result}) \times (GCOMPCOEFF) / 4096$$

As GCOMPCOEFF can be programmed from 0 to 16383, the actual gain compensation factor can range from 0 to 3.999756.

Before storing the resulting data in RDATAx or JDATAx bitfields, the LSB - 1 value is evaluated to round up the data and minimize the error.

The gain compensation is also effective for the oversampling. When the gain compensation is used in oversampling mode, the gain calculation is performed after the accumulation and

right-shift operations to minimize the power consumption (the gain calculation is done only once and not at each conversion).

The internal multiplier width is 32 bits and the input data width for the gain compensation must be less than 18 bits. When using oversampling with injected and regular conversion mode, the ROVSM bit of the ADC_CFGR2 register must be set to resume the pending conversion with the correct value.

When the gain compensation is activated, the ADC data latency is increased by one clock cycle. In continuous and scan conversion modes, the ADC conversion rate does not change.

ADC overrun (OVR, OVRMOD)

The overrun flag (OVR) issues a notification that the regular converted data has not been read (by the CPU or the DMA) before the ADC_DR FIFO (eight stages) is overflowed, or adc_hclk clock is too slow to manage the data.

The OVR flag is set when a new conversion completes while ADC_DR as single register or FIFO output is full. An interrupt is generated if the OVRIE bit is set.

When an overrun condition occurs, the ADC is still operating and can continue converting unless the software decides to stop and reset the sequence by setting ADSTP. The OVR flag is cleared by software by programming it to 1.

Data can be configured to be preserved or overwritten when an overrun event occurs by programming the control bit OVRMOD:

- OVRMOD = 0

  The overrun event preserves the data register from being overwritten: the old data are maintained up to ADC_DR FIFO depth (eight data) and the new conversion is discarded and lost. If OVR remains at 1, further conversions occur but the result data is also discarded. The FIFO can be emptied through ADC_DR read access by the CPU while the OVR flag is set.
- OVRMOD = 1

  The data register is overwritten with the last conversion result and the previous unread data is lost. In this mode, ADC_DR FIFO is disabled. If OVR remains at 1, any further conversion is performed normally and the ADC_DR register always contains the latest converted data. In this mode, DMA burst mode nor MDF output mode cannot be supported.

Figure 280. Example of overrun (OVRMOD = 0)

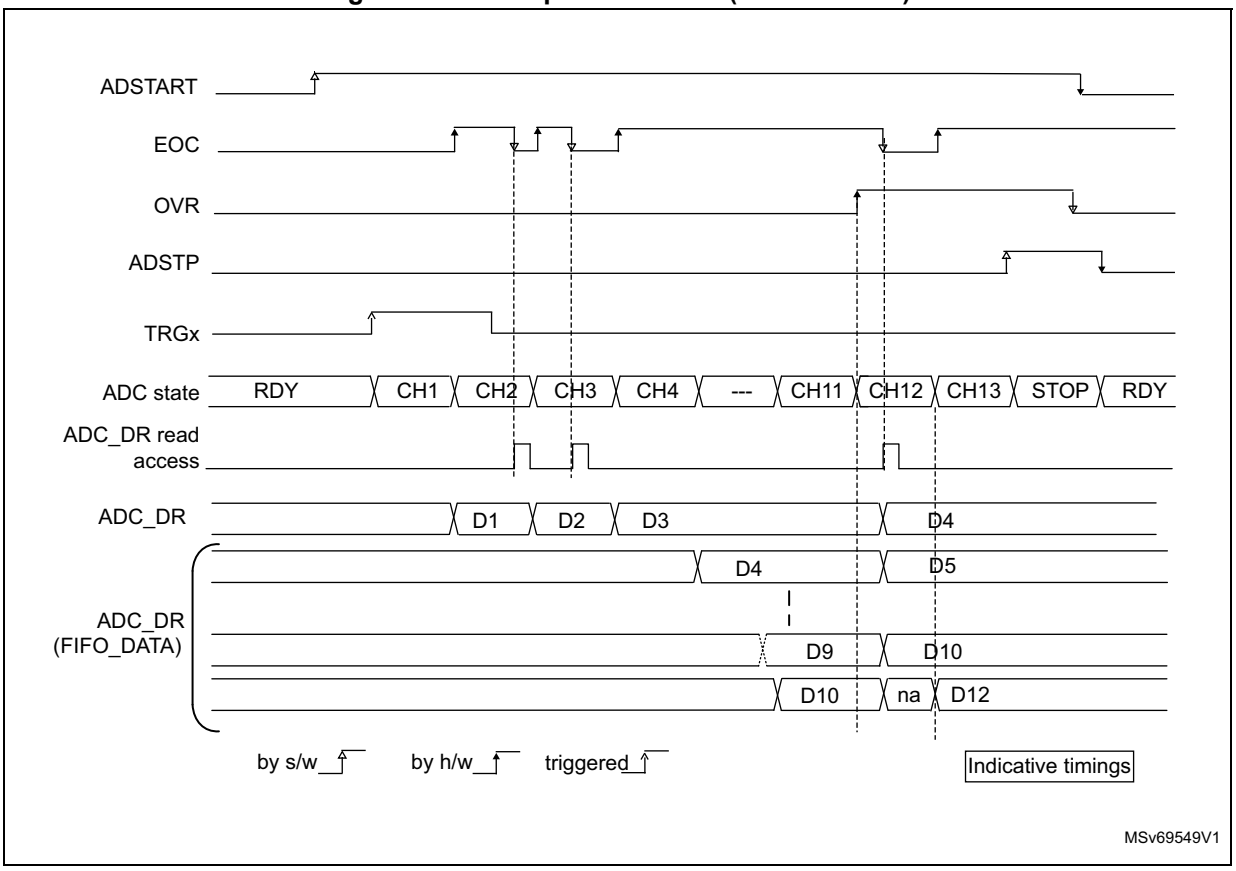
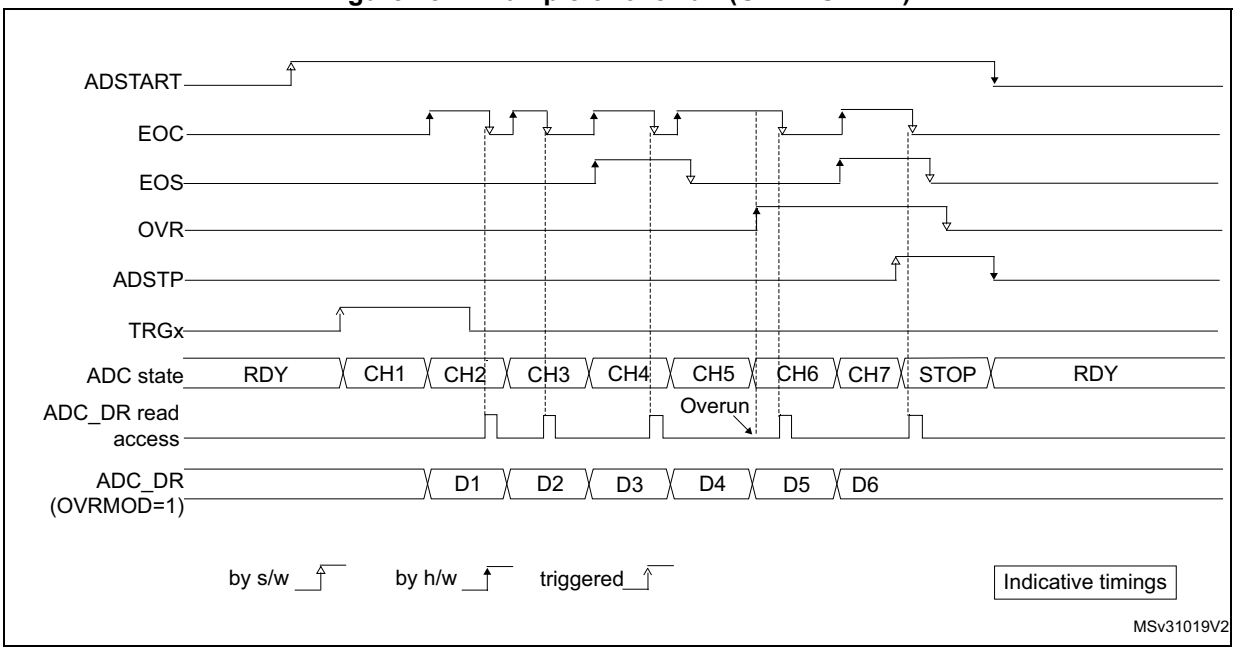


Figure 281. Example of overrun (OVRMOD = 1)



Note: Even if overrun may occur for injected channels, there is no overrun detection on these channels since there is a dedicated data register for each of the four injected channels.

Managing a sequence of conversions without using the DMA

If the conversions are slow enough, the conversion sequence can be handled by the software. In this case the software must use the EOC flag and its associated interrupt to handle each data. Each time a conversion is complete, EOC is set and the ADC_DR register can be read. OVRMOD must be configured to 0 to manage overrun events or FIFO overflow as errors.

Managing conversions without using the DMA and without overrun

It may be useful to let the ADC convert one or more channels without reading the data each time (if there is an analog watchdog for instance). In this case, the OVRMOD bit must be configured to 1 and the OVR flag must be ignored by the software. An overrun event does not prevent the ADC from continuing to convert and the ADC_DR register always contains the latest conversion.

Managing conversions using the DMA

When the DMA mode is enabled (DMNGT[1:0] = 01 or 11 in the ADC_CFGR1 register in single ADC mode), a DMA request is generated after each channel conversion. This allows the transfer of the converted data from the ADC_DR register to the destination location configured in the DMA.

Despite this, if an overrun occurs (OVR = 1) because the DMA cannot serve the DMA transfer request in time, the ADC stops generating DMA requests and the data corresponding to the new conversion is not transferred by the DMA, which means that all the data transferred to the RAM can be considered as valid. (OVRMOD = 0).

When OVRMOD = 1, even if DMA does not transfer the data immediately, new data overwrite ADC_DR, causing DMA to potentially skip data. As a result, some conversion results may be missed in the data transferred to RAM.

Depending on the configuration of the OVRMOD bit, the data are either preserved or overwritten (refer to *Section : ADC overrun (OVR, OVRMOD)*). If OVRMOD = 0 (data preserved), DMA transfer requests are blocked.

DMA can be configured through the DMNGT[1:0] bitfield of the ADC_CFGR1 register. Refer to *Section 32.4.31: Dual ADC modes* for information on DMA usage with dual ADC mode.

- DMA one-shot mode (DMNGT[1:0] = 01)

  This mode is suitable when the DMA is programmed to transfer a fixed number of data.
- DMA circular mode (DMNGT[1:0] = 11)

  This mode is suitable when programming the DMA when the number of transfers is unknown.

DMA one-shot mode (DMNGT[1:0] = 01)

In one-shot mode, the ADC generates a DMA transfer request each time a new conversion data is available and stops generating DMA requests once the DMA has reached the last DMA transfer (that is when DMA_LACK interrupt occurs, see *Direct-memory controller* section) even if a new conversion has been started.

When DMA transfers are complete (all the transfers configured in the DMA controller have been done):

- The content of the ADC data register is frozen.
- All ongoing conversions are aborted with the partial result discarded.
- The scan sequence is stopped and reset.
- The DMA and ADC are stopped.

DMA circular mode (DMNGT[1:0] = 11)

In circular mode, the ADC generates a DMA transfer request each time a new conversion data is available in the data register. This allows the configuration of the DMA in circular mode to handle a continuous analog input data stream.

DMA single-transfer or burst mode with FIFO (OVRMOD = 0)

The output data register features an eight-stage FIFO. Two different DMA requests are generated in parallel. When a data is available, an “SREQ single request” is generated. When four data are available, a “BREQ burst request” is generated. The DMA can be programmed either in single-transfer mode or in burst mode (four beats). The appropriate request line is selected by the DMA according to the mode selected. Refer to *Section Direct memory access controller* for further information.

Due to the existence of FIFO, DMA requests are generated until the FIFO is emptied. To stop DMA requests even when the FIFO is not emptied, set the ADSTP bit.

DMA for dual mode

Refer to *Section 32.4.31: Dual ADC modes*.

32.4.27 Managing conversions using the MDF

The ADC conversion results can be transferred directly to the MDF.

In this case, the DMNGT[1:0] bits must be set to 10.

The ADC transfers the 16 least significant bits of the regular data register to the MDF through adcx_dat[15:0] bus, which in turn resets the EOC flag once the transfer is effective.

The data format must be in 16-bit signed format:

ADC_DR[31:16] = Don’t care

ADC_DR[15] = Sign

ADC_DR[14:0] = Data

Any value above the 16-bit signed format is truncated.

To obtain a 16-bit signed format, the OFFSET function needs to be activated.

In this mode, OVRMOD = 1 is not supported.

32.4.28 Dynamic low-power features

Autodelayed conversion mode (AUTDLY)

The ADC implements an autodelayed conversion mode controlled by the AUTDLY configuration bit in ADC_CFGR1. Autodelayed conversions are useful to simplify the software as well as to optimize performance of an application clocked at low frequency where there must be risk of encountering an ADC overrun.

Regular conversion mode

When AUTDLY is set, a new conversion can start if the following conditions are met:

- The ADC_DR register is read or the EOC bit is cleared (see *Figure 282*).

This is a way to automatically adapt the speed of the ADC to the speed of the system that reads the data and prevent overrun errors.

The delay is inserted after each regular conversion.

Injected conversion mode

When AUTDLY is set, a new conversion can start only if the JEOS bit has been cleared (see *Figure 283*). Since there are four ADC_JDRy registers, the autodelay is performed at the end on the injected conversion sequence.

There is no delay inserted between each conversion of the injected sequence, except after the last one.

A hardware trigger event occurring during this delay is ignored.

No delay is inserted between regular and injected conversions. The autodelay for regular and injected modes is managed separately.

Caution: The behavior is slightly different in autoinjection mode (JAUTO = 1) where a new regular conversion can start only when the automatic delay of the previous injected sequence of conversion has ended (when JEOS has been cleared). This is to ensure that the software can read all the data of a given sequence before starting a new sequence (see *Figure 286*).

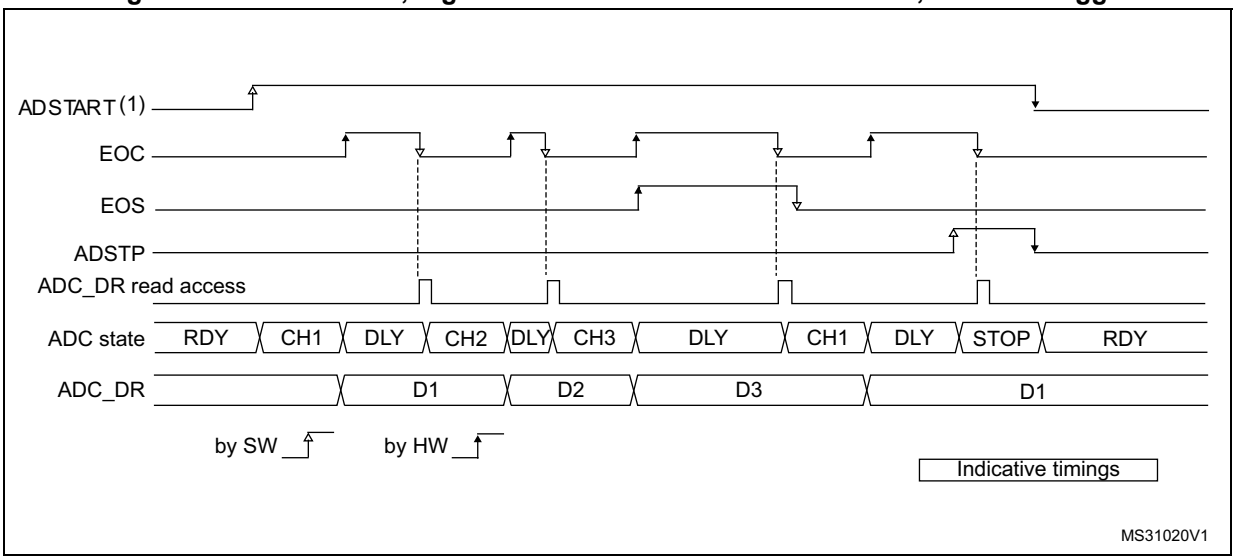
In autoinjection mode, there is no delay between the last regular conversion and the first injected conversion. The user must clear the last EOC flag before clearing the JEOS flag, otherwise the next sequence might start before EOC flag is cleared.

In autodelay mode, a hardware regular trigger event is ignored if it occurs during an already ongoing regular sequence or during the delay that follows the last regular conversion of the sequence. It is however considered pending if it occurs after this delay, even if it occurs during an injected sequence of the delay that follows it. The conversion then starts at the end of the delay of the injected sequence.

In autodelay mode with JAUTO = 0, a hardware injected trigger event is ignored if it occurs during an already ongoing injected sequence or during the delay that follows the last injected conversion of the sequence.

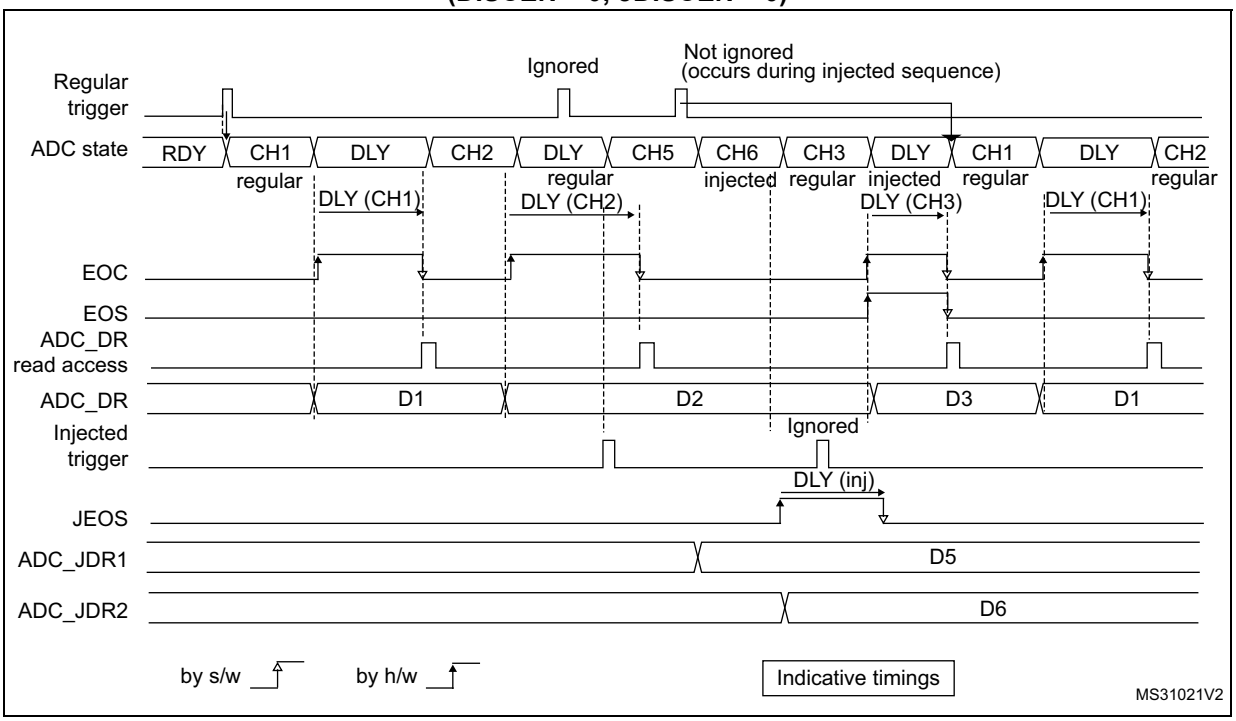
Note: AUTDLY mode is not supported with SMPTRIG or BULB mode.

Figure 282. AUTDLY = 1, regular conversion in continuous mode, software trigger



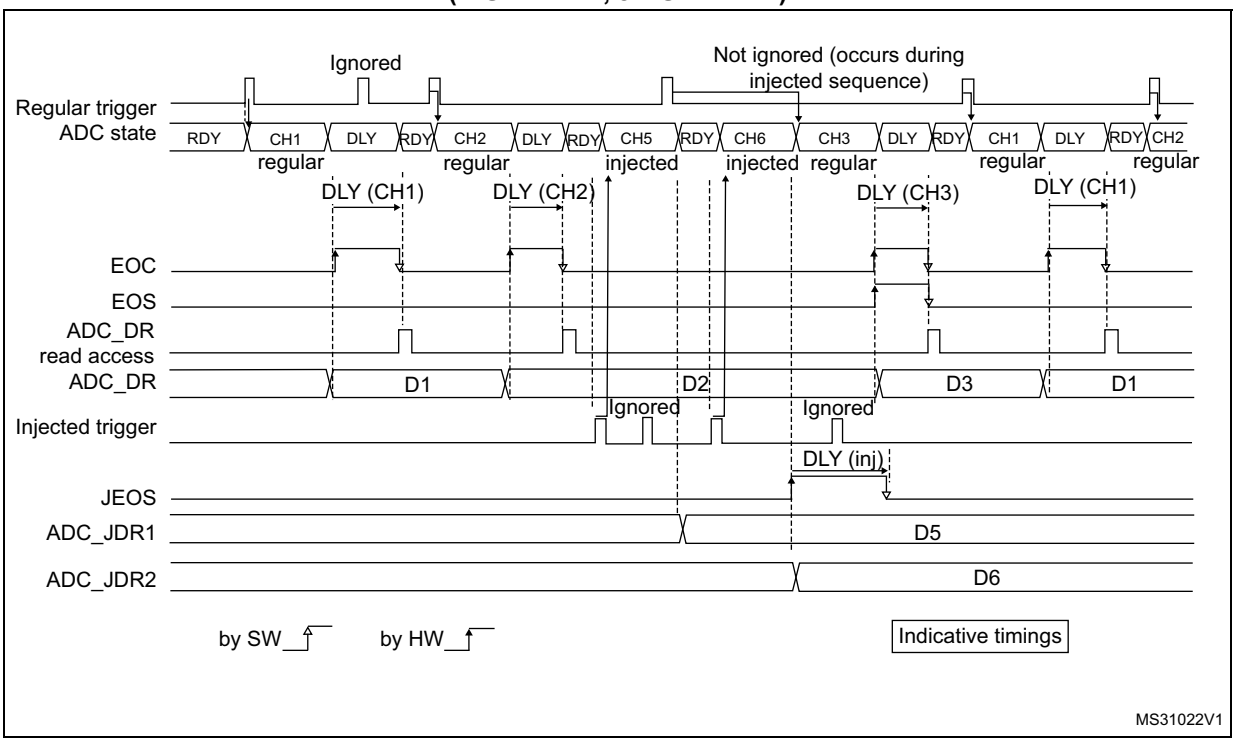
1. AUTDLY = 1, JAUTO = 0.
2. Regular configuration: EXTEN[1:0] = 00 (software trigger), CONT = 1, CHANNELS = 1, 2, 3.
3. Injected configuration DISABLED.

Figure 283. AUTDLY = 1, regular hardware conversions interrupted by injected conversions (DISCEN = 0; JDISCEN = 0)



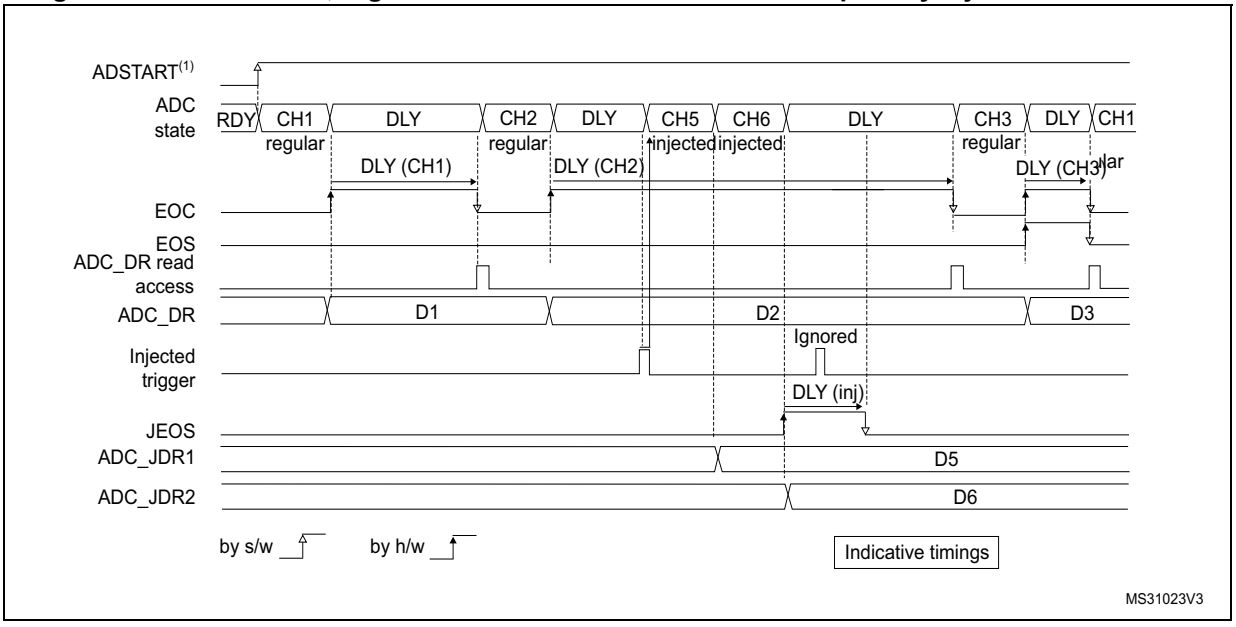
1. AUTDLY = 1, JAUTO = 0.
2. Regular configuration: EXTEN[1:0] = 01 (hardware trigger), CONT = 0, DISCEN = 0, CHANNELS = 1, 2, 3.
3. Injected configuration: JEXTEN[1:0] = 01 (hardware trigger), JDISCEN = 0, CHANNELS = 5, 6.

Figure 284. AUTDLY = 1, regular hardware conversions interrupted by injected conversions (DISCEN = 1, JDISCEN = 1)



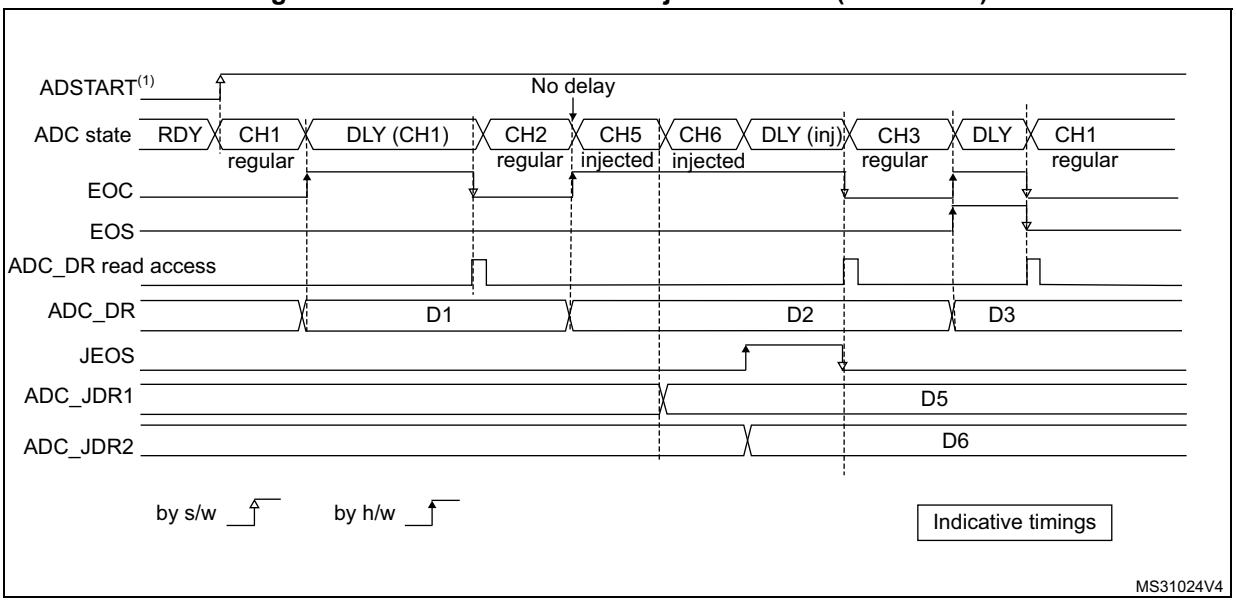
1. AUTDLY = 1, JAUTO = 0.
2. Regular configuration: EXTEN[1:0] = 01 (hardware trigger), CONT = 0, DISCEN = 1, DISCNUM = 1, CHANNELS = 1, 2, 3.
3. Injected configuration: JEXTEN[1:0] = 01 (hardware trigger), JDISCEN = 1, CHANNELS = 5, 6.

Figure 285. AUTDLY = 1, regular continuous conversions interrupted by injected conversions



1. AUTDLY = 1, JAUTO = 0.
2. Regular configuration: EXTEN[1:0] = 00 (software trigger), CONT = 1, DISCEN = 0, CHANNELS = 1, 2, 3.
3. Injected configuration: JEXTEN[1:0] = 01 (hardware trigger), JDISCEN = 0, CHANNELS = 5, 6.

Figure 286. AUTDLY = 1 in autoinjection mode (JAUTO = 1)



1. AUTDLY = 1.
2. Regular configuration: EXTEN[1:0] = 00 (software trigger), CONT = 1, DISCEN = 0, CHANNELS = 1, 2.
3. Injected configuration: JAUTO = 1, CHANNELS = 5, 6.

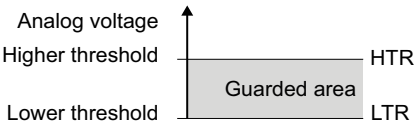
32.4.29 Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWDCH of ADC_AWD2CR and ADC_AWD3CR, HTR, LTR, AWDFILT)

Table 248. Analog window watchdog features

Feature	AWD1	AWD2	AWD3
Enable control	AWD1EN or JAWD1EN	ADC_AWD2CR	ADC_AWD3CR
Channel selection	AWD1CH[4:0]/AWD1SGL	ADC_AWD2CR	ADC_AWD3CR
Channel selection limitation	One or all channels	Any number of channels	
	Regular or injected	-	-
Threshold control	ADC_AWD1LTR / ADC_AWD1HTR	ADC_AWD2LTR / ADC_AWD2HTR	ADC_AWD3LTR / ADC_AWD3HTR
Filter configuration	AWDFILT[2:0] in ADC_AWDyHTR		
Interrupt	AWDy in ADC_ISR		

The three AWD analog watchdogs monitor whether some channels remain within a configured voltage range (window).

Figure 287. Analog watchdog guarded area



AWDy flag and interrupt

An interrupt can be enabled for each of the three analog watchdogs by setting AWDyIE in the ADC_IER register (y = 1, 2, 3).

AWDy (y = 1, 2, 3) flag is cleared by software by writing 1 to it, or by setting ADDIS.

The ADC conversion result is compared to the lower and higher thresholds before alignment.

Description of analog watchdog 1

The AWD analog watchdog 1 is enabled by setting the AWD1EN or JAWD1EN bit in the ADC_CFGR1 register. This watchdog monitors whether either one selected channel or all enabled channels remain within a configured voltage range (window).

Table 249 shows how the ADC_CFGR1 registers must be configured to enable the analog watchdog on one or more channels.

Table 249. Analog watchdog 1 channel selection

Channels guarded by the analog watchdog	AWD1SGL bit	AWD1EN bit	JAWD1EN bit
None	x	0	0
All injected channels	0	0	1
All regular channels	0	1	0
All regular and injected channels	0	1	1
Single(1) injected channel	1	0	1
Single(1) regular channel	1	1	0
Single(1) regular or injected channel	1	1	1

1. Selected by the AWD1CH[4:0] bits. The channels must also be programmed to be converted in the appropriate regular or injected sequence.

The AWD1 analog watchdog flag is set if the analog voltage converted by the ADC is below a lower threshold or above a higher threshold.

These thresholds are programmed through HTR[22:0] bits of the ADC_AWD1HTR register and LTR[22:0] bits of the ADC_AWD1LTR register for the analog watchdog 1.

The threshold can be up to 23 bits (12-bit resolution with oversampling, OVSR = 1024, offset and gain compensation)

When converting data with a resolution of less than 12 bits (defined by RES[1:0] bits), the LSB of the programmed thresholds must be kept cleared because the internal comparison is always performed on the full 12-bit raw converted data.

Table 250 describes how the comparison is performed for all the possible resolutions and for analog watchdog 1.

Table 250. Analog watchdog 1, 2, 3 comparison

Resolution (RES[1:0] bits)	Analog watchdog comparison between:		Comments
	Raw converted data, left aligned(1)	Thresholds	
00: 12-bit	DATA[11:0]	LTR[22:0] and HTR[22:0]	-
01: 10-bit	DATA[11:2],00	LTR[22:0] and HTR[22:0]	The user must configure LTR[1:0] to 0b00 and HTR[1:0] to 0b00.
10: 8-bit	DATA[11:4],0000	LTR[22:0] and HTR[22:0]	The user must configure LTR[3:0] to 0b0000 and HTR[3:0] to 0b0000.
11: 6-bit	DATA[11:6],000000	LTR[22:0] and HTR[22:0]	The user must configure LTR[5:0] to 0b000000 and HTR[5:0] to 0b000000.

1. Refer to *Section : Gain compensation* for additional details on analog watchdog comparison.

Description of analog watchdog 2 and 3

The second and third analog watchdogs are more flexible and can guard several selected channels by programming the corresponding bits in AWDCH[19:0] (of ADC_AWD2CR and ADC_AWD3CR).

The corresponding watchdog is enabled when any bit of AWDCH[19:0] (of ADC_AWD2CR and ADC_AWD3CR) is set.

The threshold can be up to 23 bits (12-bit resolution with oversampling, OVSR = 1024 and offset compensation in signed format with gain compensation) and is programmed through the ADC_AWD2HTR, ADC_AWD2LTR, ADC_AWD3HTR and ADC_AWD3LTR registers.

When converting data with a resolution of less than 12 bits (defined by RES[1:0]) bits), the LSBs of the programmed threshold must be kept cleared, the internal comparison being performed on the full 12-bits converted data (left aligned).

ADCx_AWDy_OUT signal output generation

Each analog watchdog is associated with an internal hardware signal ADC_AWDy_OUT (*y* being the watchdog number), which is directly connected to the ETR input (external trigger) of some of the on-chip timers. Refer to the on-chip timers sections to understand how to select the ADC_AWDy_OUT signal as ETR.

ADC_AWDy_OUT is activated when the associated analog watchdog is enabled:

- ADC_AWDy_OUT is set when a guarded conversion is outside the programmed thresholds.
- ADC_AWDy_OUT is reset after the end of the next guarded conversion that is inside the programmed thresholds (It remains at 1 if the next guarded conversions are still outside the programmed thresholds).
- ADC_AWDy_OUT is also reset when disabling the ADC (when setting ADDIS). Note that stopping regular or injected conversions (by setting ADSTP or JADSTP) has no influence on the generation of ADCx_AWDy_OUT.

Note: The AWDy flag is set by hardware and reset by software: the AWDy flag has no influence on the generation of ADC_AWDy_OUT (ex: ADC_AWDy_OUT can toggle while the AWDy flag remains at 1 if the software did not clear the flag).

The AWDy flag is cleared by programming it to 1 or by set ADDIS bit. ADSTP and JADSTP bits do not reset the AWDy flag.

Figure 288. ADC_AWDy_OUT signal generation (on all regular channels)

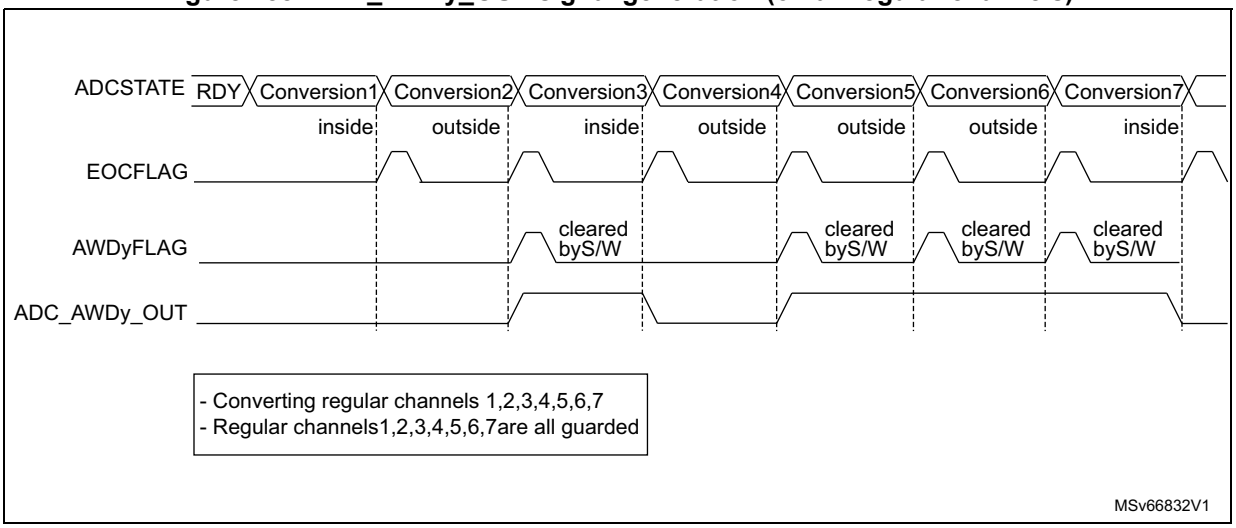


Figure 289. ADC_AWDy_OUT signal generation (AWDy flag not cleared by software)

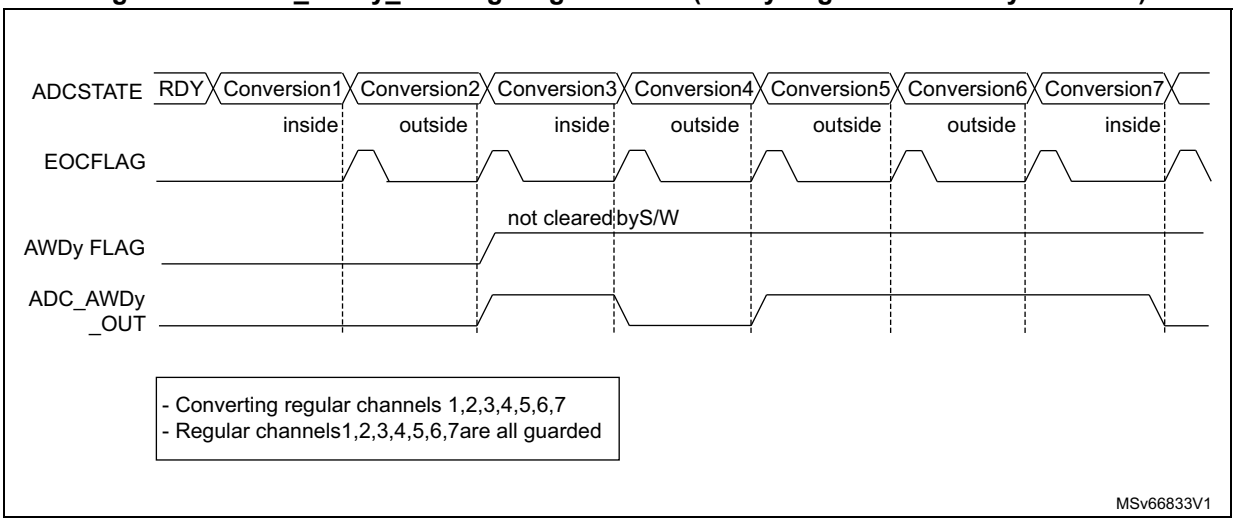


Figure 290. ADC_AWDy_OUT signal generation (on a single regular channel)

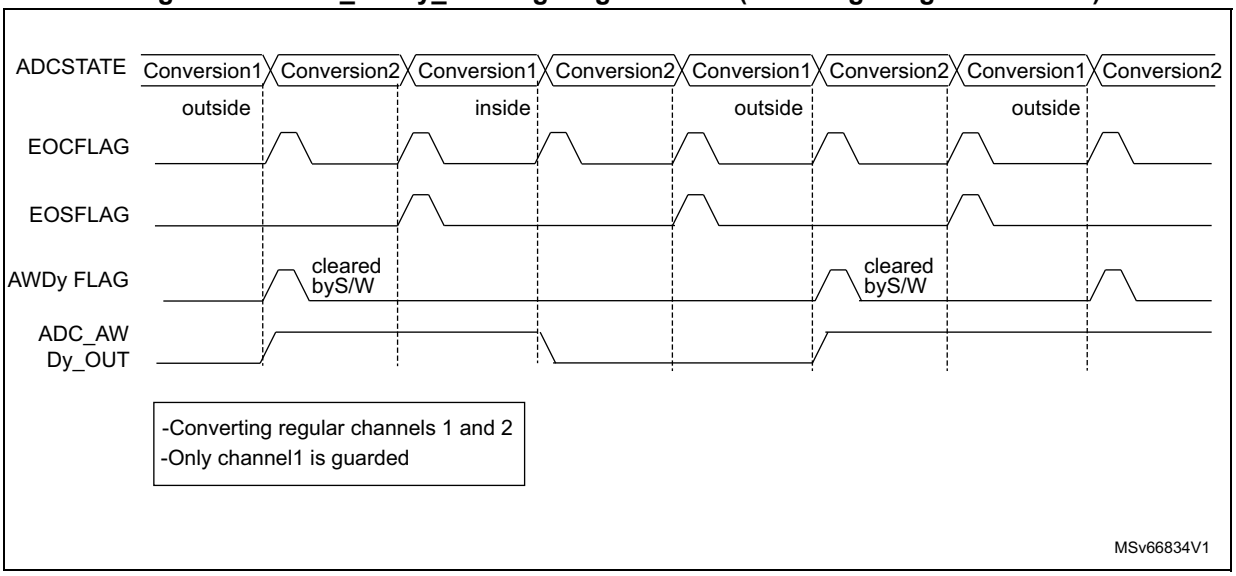
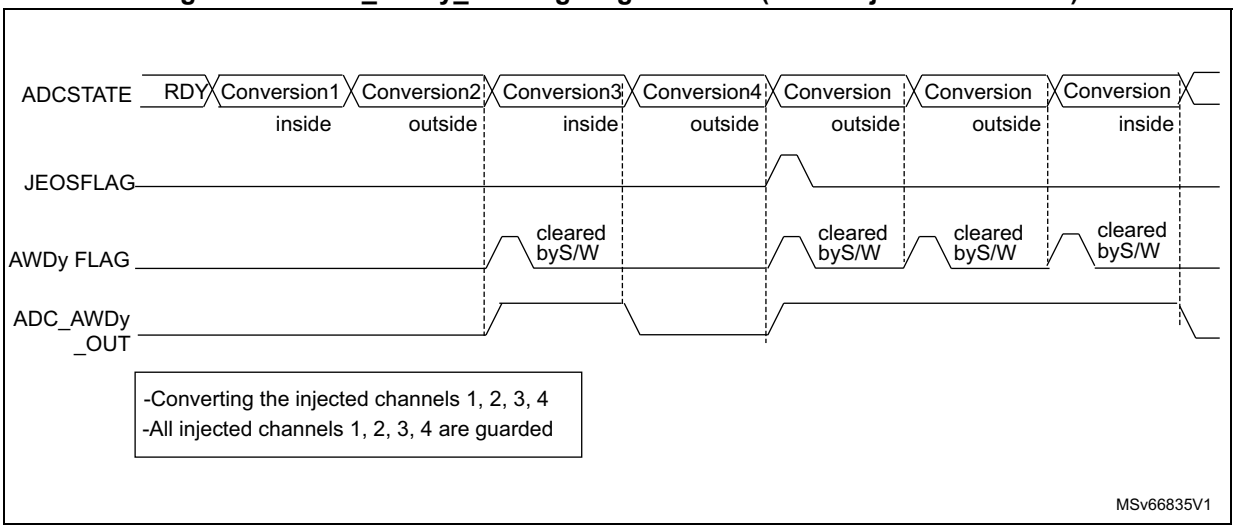


Figure 291. ADC_AWDy_OUT signal generation (on all injected channels)



Analog watchdog threshold control on the fly

LTR[22:0] and HTR[22:0] can be changed when an analog-to-digital conversion is ongoing (that is between the start of conversion and the end of conversion of the ADC internal state). If LTR[22:0] and HTR[22:0] are updated during the ADC conversion of the ADC guarded channel, resulting in analog watchdog thresholds to be applied from the next ADC conversion. The analog watchdog comparison is performed at each end of conversion. If the current ADC data is out of the new modified threshold, no interrupt and AWDy_OUT signal are issued. The Interrupt and the AWD generation only happen at the end of the conversion that started after the threshold update. If AWDy_OUT is already asserted, programming the new thresholds does not deassert the AWDy_OUT signal. However, the watchdog threshold must not be changed twice during the four adc_ker_ck periods.

Analog watchdog filter for watchdog 1

With analog watchdog, a valid ADC conversion data range can be configured through the ADC_AWD1LTR and ADC_AWD1HTR registers. When the conversion results have consecutively passed the threshold for more than the watchdog filter value (AWDFILT[2:0] + 1), AWD1_OUT is generated as well as the AWD1 flag. Before this value is reached, AWD1 output is not activated.

Analog watchdog with gain and offset compensation

When the gain and offset compensation are enabled, the analog watchdog compares the data after the compensation.

32.4.30 Oversampler

The oversampling unit performs data preprocessing to offload the device. It is able to handle multiple conversions and average them into a single data with increased data width, up to 22 bits.

It provides a result with the following form, where N and M can be adjusted:

$$\text{Result} = \frac{1}{M} \times \sum_{n=0}^{n=N-1} \text{Conversion}(t_n)$$

It enables the following functions to be performed by hardware: averaging, data rate reduction, SNR improvement and basic filtering.

The oversampling ratio N is defined by the OVSR[9:0] bits in the ADC_CFGR2 register, or by the JOVSS[3:0] bits in the ADC_CFGR3 register for parallel injected oversampling mode. They can range from 2x to 1024x. The division coefficient *M* consists of a right bit shift up to 10 bits. It is defined through the OVSS[3:0] bits of the ADC_CFGR2 register.

The summation unit can yield a result up to 22 bits, which can be left or right shifted at the end. When the right shift is selected, it is rounded to the nearest value using the least significant bits left apart by the shifting, before being finally transferred to the ADC_DR data register.

Figure 292 gives a numerical example of the processing, from a raw 22-bit accumulated data to the final 12-bit result.

Figure 292. 12-bit result oversampling with 10-bit right shift and rounding

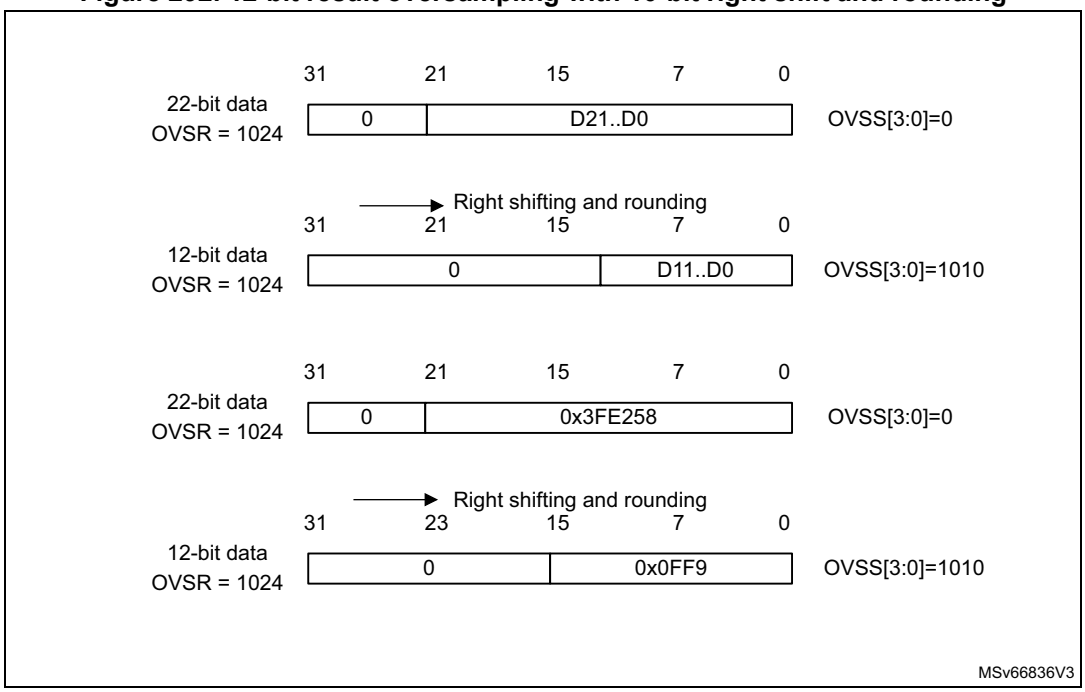


Table 280 gives data format for the various N and M combinations, for a raw conversion data equal to 0xFFF.

Table 251. Maximum output results versus N and M

Over sampling ratio	Max Raw data	No-shift OVSS = 0000	1-bit shift OVSS = 0001	2-bit shift OVSS = 0010	3-bit shift OVSS = 0011	4-bit shift OVSS = 0100	5-bit shift OVSS = 0101	6-bit shift OVSS = 0110	7-bit shift OVSS =0111	8-bit shift OVSS = 1000
x2	0x1FFE	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080	0x0040	0x0020
x4	0x3FFC	0x3FFC	x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100	0x0080	0x0040
x16	0xFFF0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400	0x0200	0x0100
x64	0x3FFC0	0x3FFC0	0x1FFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF	0x0800	0x0400
x256	0xFFF00	0xFFF00	0x7FF80	0x3FFC0	0x1FFE0	0xFFF0	0x7FF8	0x3FFC	0x1FFE	0x0FFF
x1024	0x3FFC00	0x3FFC00	0x1FFE00	0xFFF00	0x7FF80	0x3FFC0	0x1FFE0	0xFFF0	0x7FF8	0x3FFC

The conversion timing rate does not change in oversampling mode: the sample time is maintained equal during the whole oversampling sequence. A new data is provided every N conversion, with an equivalent delay equal to N x T_{CONV} = N x (t_{SMPL} + t_{SAR}). The flags are set as follows:

- The end of the sampling phase (EOSMP) is set after each sampling phase for regular conversion.
- The end of conversion (EOC) occurs once every N conversion, when the oversampled result is available.
- The end of sequence (EOS) occurs once the sequence of oversampled data is completed (that is after a total of N x sequence length conversions).

Unlike the conversion rate, the data latency changes in oversampling mode: each single conversion introduces an additional accumulation. As a result, an additional clock cycle is required when all the conversions are complete, after the accumulation and shift.
The EOC flag is set two additional clock cycles after the conversion is complete.

ADC operating modes supported in oversampling mode (single ADC mode)

In oversampling mode, most of the ADC operating modes are maintained:

- Single, discontinuous, or continuous conversion modes
- ADC conversions started either by software or triggers
- ADC stop during a conversion (abort)
- Data read via CPU or DMA with overrun detection
- Low-power modes (AUTDLY)
- Programmable resolution: in this case, the reduced conversion values (as per RES[1:0] bits of the ADC_CFGR1 register) are accumulated, truncated, rounded, and shifted in the same way as 12-bit conversions are.
- Autoinjected mode (JAUTO) if JOVSPAR = 0 and oversampling ratio is the same for both regular and injected channels.

Analog watchdog

The analog watchdog functionality is maintained, with the following differences:

- The RES[1:0] bits are ignored and the comparison is always done using the full 23-bit values HTR[22:0] and LTR[22:0].
- The comparison is performed on the oversampled accumulated value before shifting.

Triggered mode

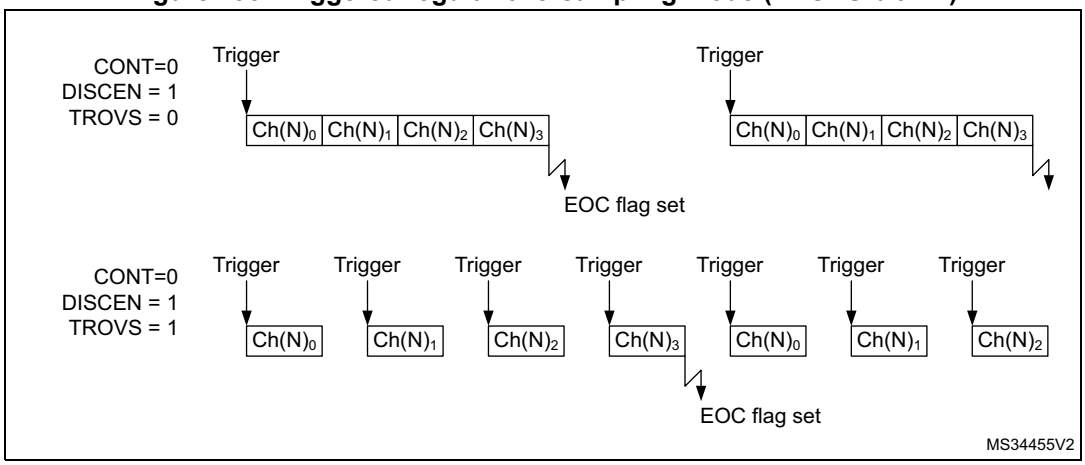
The oversampling can also be used for basic filtering purpose. Although it not a very powerful filter (slow roll-off and limited stop band attenuation), it can act as a notch filter to reject constant parasitic frequencies (typically coming from the mains or from a switched-mode power supply). For this purpose, a specific discontinuous mode can be enabled through the TROVS bit in ADC_CFGR2, to be able to achieve an oversampling frequency defined by the user and independent from the conversion time itself. The TROVS bit is supported with hardware trigger mode.

Figure 293 shows how conversions are started in response to triggers during discontinuous mode.

If the TROVS bit is set and OVSR ≠ 0, the content of the DISCEN bit is ignored and considered as 1.

If the TROVS bit is set, neither software trigger nor JAUTO is supported.

Figure 293. Triggered regular oversampling mode (TROVS bit = 1)



Injected and regular sequencer management in oversampling mode

In oversampling mode, injected and regular sequencers can have different behaviors. The oversampling can be enabled for both sequencers with some limitations if they have to be used simultaneously (this is related to a unique accumulation unit).

Oversampling regular channels only

The regular oversampling mode ROVSM bit defines how the regular oversampling sequence is resumed if it is interrupted by an injected conversion:

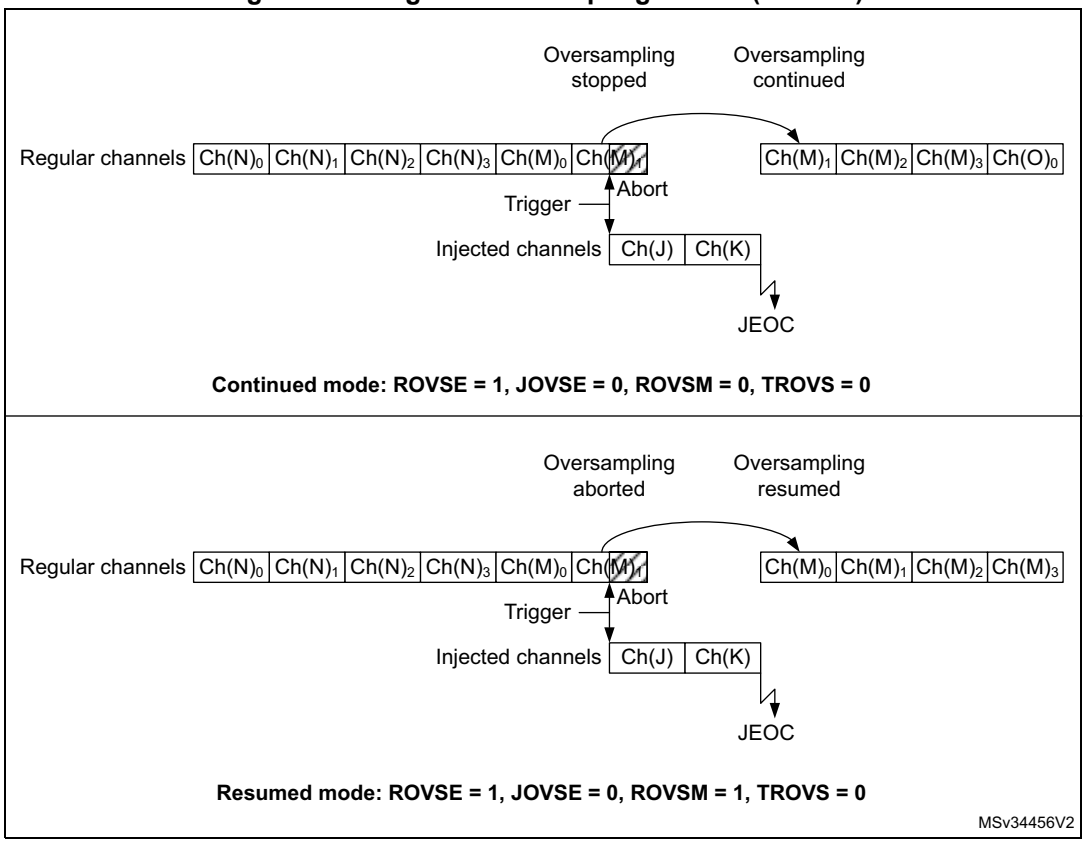
- In continuous mode (ROVSM = 0), the accumulation restarts from the last valid data (prior to the conversion abort request due to the injected trigger). This ensures that

oversampling is complete whatever the injection frequency (providing at least one regular conversion can be complete between triggers);

- In resumed mode (ROVSM = 1), the accumulation restarts from 0 (previous conversion results are ignored). This mode guarantees that all the data used for oversampling were converted back-to-back within a single time slot. Care must be taken to have a injection trigger period above the oversampling period length. If this condition is not respected, the oversampling cannot be complete and the regular sequence is blocked.

Figure 294 gives examples for a 4x oversampling ratio.

Figure 294. Regular oversampling modes (4x ratio)



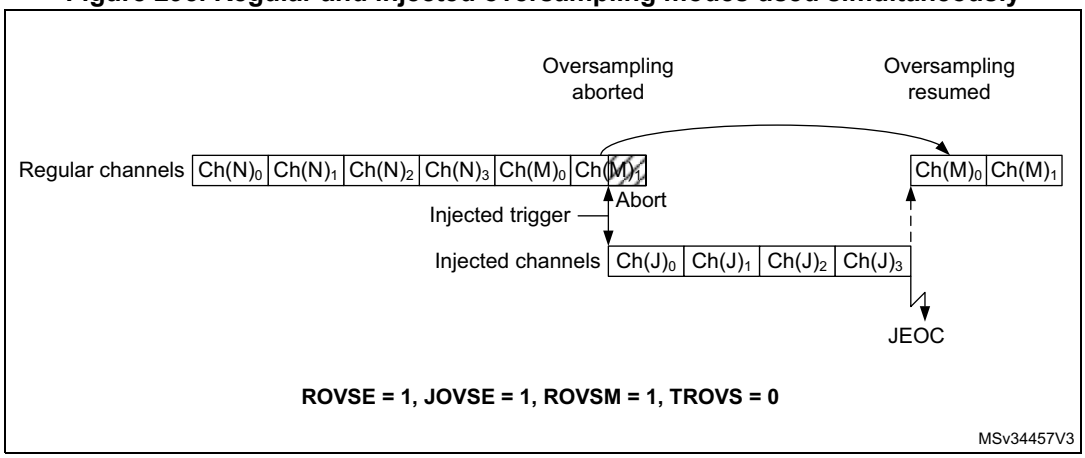
Oversampling injected channels only

The injected oversampling mode bit JOVSE enables oversampling solely for conversions in the injected sequencer.

Oversampling regular and injected channels

It is possible to have both ROVSE and JOVSE bits set. In this case, the regular oversampling must be resumed (ROVSM bit = 1), as shown in *Figure 295*. This limitation no longer applies in parallel injected oversampling mode.

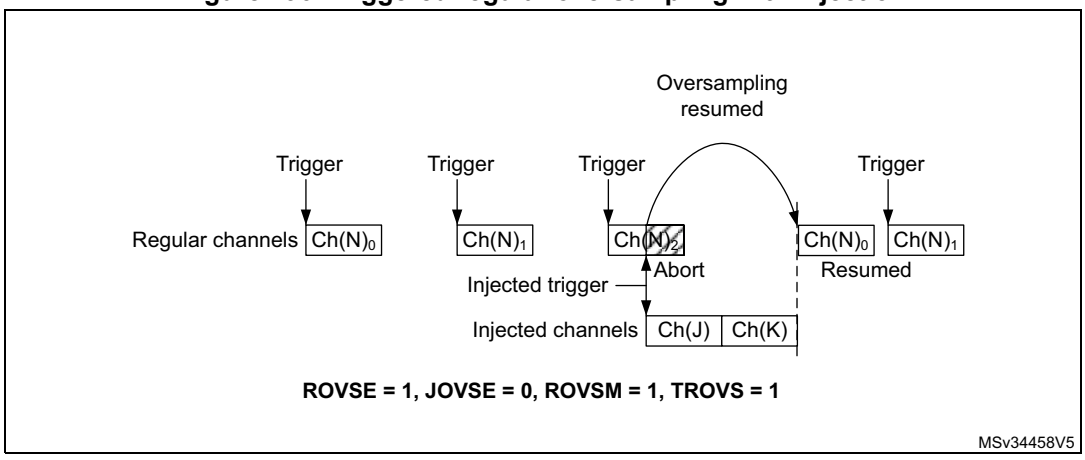
Figure 295. Regular and injected oversampling modes used simultaneously



Triggered regular oversampling with injected conversions

It is possible to have triggered regular mode with injected conversions. In this case, the injected oversampling mode must be disabled and the ROVSM bit set (the resumed mode is forced). The JOVSE bit must be reset. The behavior is shown in *Figure 296*.

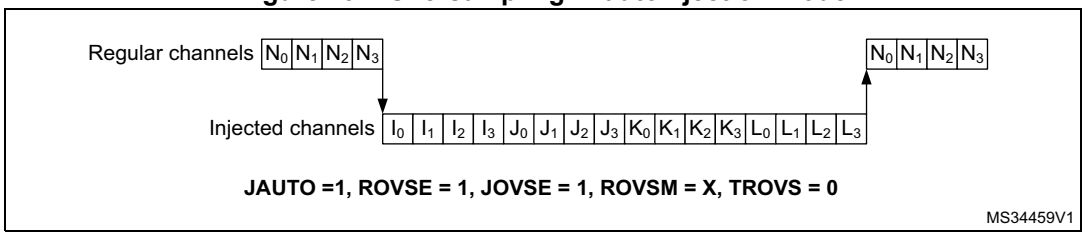
Figure 296. Triggered regular oversampling with injection



Autoinjection mode

It is possible to oversample autoinjected sequences and store all conversion results in registers. This mode is available only with both regular and injected oversampling active: JAUTO = 1, ROVSE = 1 and JOVSE = 1. Other combinations are not supported. The ROVSM bit is ignored in autoinjection mode. *Figure 297* shows how the conversions are sequenced.

Figure 297. Oversampling in autoinjection mode



Combined modes summary

Table 252 summarizes all combinations, including the modes that are not supported.

Table 252. Oversampler operating mode summary

Regular oversampling ROVSE	Injected oversampling JOVSE	Oversampler mode ROVSM 0 = continued 1 = resumed	Triggered regular mode TROVS	Comment
1	0	0	0	Regular continued mode
1	0	0	1	Not supported
1	0	1	0	Regular resumed mode
1	0	1	1	Triggered regular resumed mode
1	1	0	X	Not supported
1	1	1	0	Injected and regular resumed mode
1	1	1	1	Not supported
0	1	X	X	Injected oversampling

32.4.31 Dual ADC modes

Dual ADC modes can be used in devices with two ADCs or more (see *Figure 298*).

In dual ADC mode the start of conversion is triggered alternately or simultaneously by the ADC master to the ADC slave, depending on the mode selected by the bits DUAL[4:0] in the ADCC_CCR register.

Four possible modes are implemented:

- Injected simultaneous mode
- Regular simultaneous mode
- Interleaved mode
- Alternate trigger mode

It is also possible to use these modes combined in the following ways:

- Injected simultaneous mode + regular simultaneous mode
- Regular simultaneous mode + alternate trigger mode
- Injected simultaneous mode + interleaved mode

In dual ADC mode (when the DUAL[4:0] bits in the ADCC_CCR register are not equal to zero), the following bits are shared between the master and slave ADCs, with the slave ADC bits being ignored and always applying to the corresponding bits of the master ADC:

- ADC_CFGR1 register: DMNGT[2:0], RES[1:0], EXTEN[1:0], OVRMOD, AUTDLY, DISCEN, DISCNUM[2:0], JDISCEN, JAUTO, JDISCEN.
- ADC_CFGR2 register: ROVSE, JOVSE, TROVS, ROVSM, BULB, SWTRIG, SMPTRIG, OVSR[9:0].
- ADC_GCOMP register: GCOMP.
- ADC_CR register: ADSTART, ADSTP, JADSTART, JADSTP.
- ADC_CFGR1 register: CONT, DISCEN, DISCNUM[2:0].
- ADC_SQR1 register: LEN (for the regular conversion).
- ADC_JSQR register: JLEN[1:0], JEXTSEL[4:0], JEXTEN[1:0] (for the injected conversion).
- ADC_CFGR3 register: JOVSPAR, JOVSE, JOVSIVE, TJOVS, JOVSR[9:0] (for the injected conversion).

For dual mode usage, the sampling time of the master and slave channels must be equal.

To start a conversion in dual mode, the user must program the EXTEN[1:0], EXTSEL, JEXTEN[1:0] and JEXTSEL bits of the master ADC only to configure a software or hardware trigger and a regular or injected trigger. The EXTEN[1:0], JEXTEN[1:0], and JEXTSEL bits of the slave ADC are don’t care).

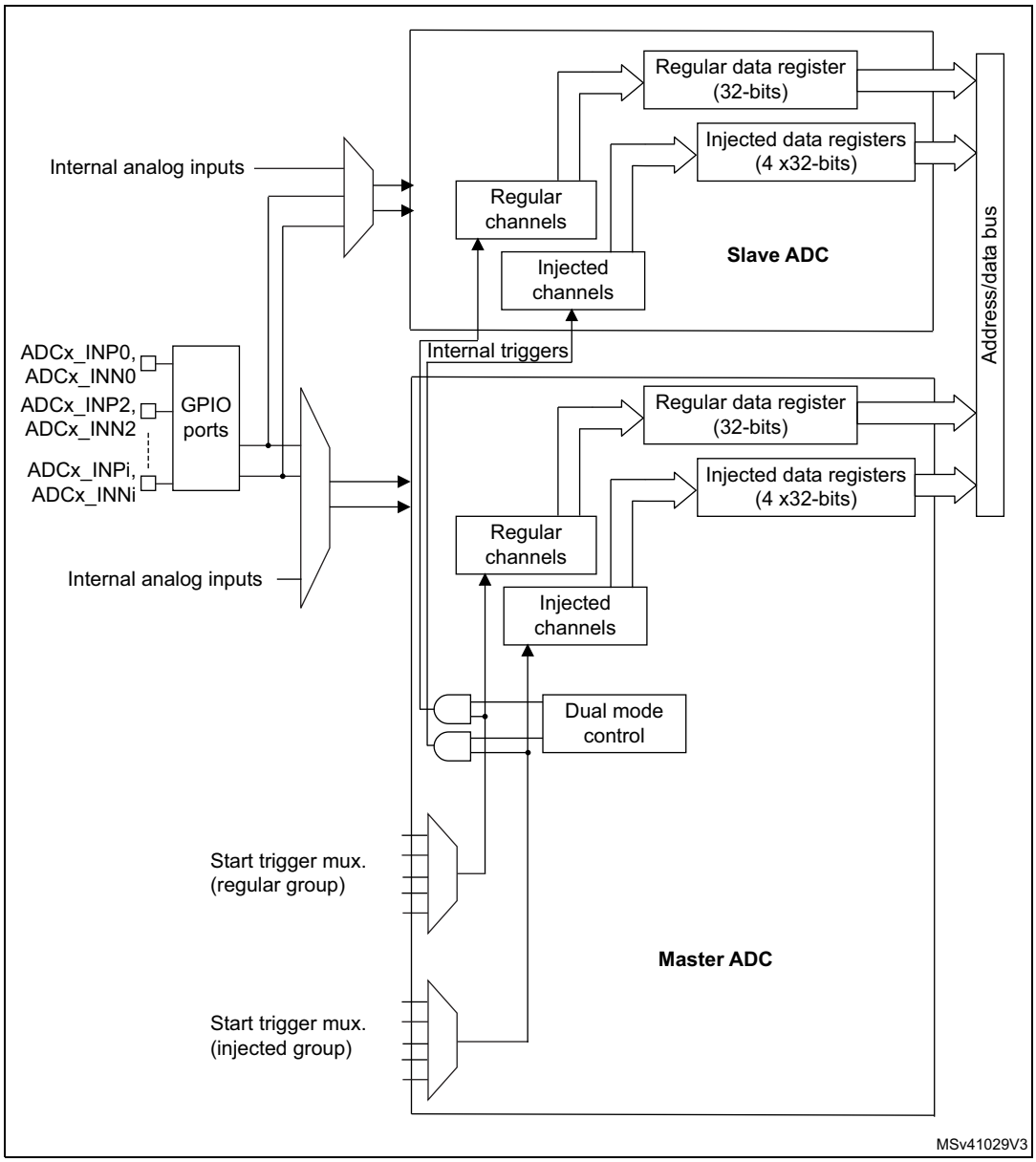
In regular simultaneous or interleaved modes, once the user sets the ADSTART or ADSTP bit of the master ADC, the corresponding bit of the slave ADC is automatically set. However, ADSTART or ADSTP bit of the slave ADC is not necessarily cleared at the same time as the master ADC bit.

In injected simultaneous or alternate trigger modes, once the user sets the JADSTART or JADSTP bit of the master ADC, the corresponding bit of the slave ADC is automatically set. However, JADSTART or JADSTP bit of the slave ADC is not necessarily cleared at the same time as the master ADC bit.

In dual ADC mode, the converted data of the master and slave ADC can be read in parallel, by reading the ADC common data register (ADCC_CDR). The flags can also be read in parallel by reading the dual-mode status register (ADCC_CSR).

Note: In dual mode, BULB, JAUTO, SMPTRIG are not supported.

Figure 298. Dual ADC block diagram



1. External triggers also exist on slave ADC but are not shown for the purposes of this diagram.
2. The ADC common data register (ADCC_CDR) contains both the master and slave ADC regular converted data.

Injected simultaneous mode with independent regular conversion

The injected simultaneous mode is selected by programming DUAL[4:0] bits to 0b00101.

This mode converts an injected group of channels. The external trigger source comes from the injected group multiplexer of the master ADC (selected by the JEXTSEL bits in the ADC_JSQR register).

Note: Do not convert the same channel on the two ADCs (no overlapping sampling times for the two ADCs when converting the same channel).

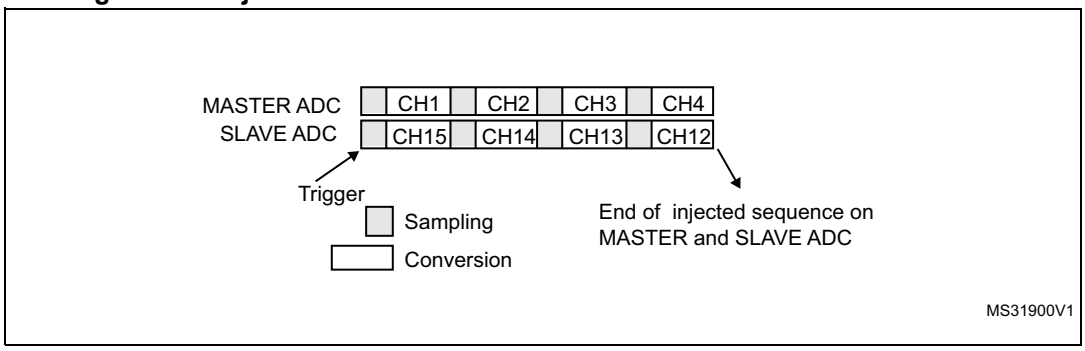
In simultaneous mode, the user software must convert sequences with the same length. Sampling times for channels with the same sequence number must be equal.

Regular conversions can be performed on one or all ADCs. In that case, they are independent of each other and are interrupted when an injected event occurs. They are resumed at the end of the injected conversion group.

The software is notified by interrupts when it can read the data:

- At the end of injected sequence of conversion event (JEOS) on the master ADC, the converted data is stored into the master ADC_JDRy registers and a JEOS interrupt is generated (if enabled)
- At the end of injected sequence of conversion event (JEOS) on the slave ADC, the converted data is stored into the slave ADC_JDRy registers and a JEOS interrupt is generated (if enabled)
- If the duration of the master-injected sequence is equal to the duration of the slave injected one (as shown in *Figure 299*), the software can enable only one of the two JEOS interrupts (for example the master JEOS) and read both converted data (from master ADC_JDRy and slave ADC_JDRy registers).

Figure 299. Injected simultaneous mode on four channels: Dual ADC mode



If JDISCEN is set, each simultaneous conversion of the injected sequence requires an injected trigger event to occur.

This mode can be combined with the AUTDLY mode:

- Once a simultaneous injected sequence of conversions has ended, a new injected trigger event is accepted only if both JEOS bits of the master and the slave ADC have been cleared (delay phase). Any new injected trigger events occurring during the ongoing injected sequence and the associated delay phase are ignored.
- Once a regular sequence of conversions of the master ADC has ended, a new regular trigger event of the master ADC is accepted only if the master data register (ADC_DR) has been read. Any new regular trigger events occurring for the master ADC during the ongoing regular sequence and the associated delay phases are ignored.
  There is the same behavior for regular sequences occurring on the slave ADC.

In this mode, the following register bits are shared between the master and slave ADCs:

JADSTART, JADSTP, JDISCEN, JOVSE, JLEN, JEXTEN, JEXTSEL, GCOMPEN, JOVSECT, JOVSISE, JOVSE, and TJOVS.

Regular simultaneous mode with independent injected conversions

The regular simultaneous mode is selected by programming DUAL[4:0] bits to 0b00110.

This mode applies to a regular group of channels. The external trigger source comes from the regular group multiplexer of the master ADC (selected by the EXTSEL bits in the ADC_CFGR1 register). A simultaneous trigger is provided to the slave ADC.

In this mode, independent injected conversions are supported. An injection request (either on the master or on the slave) aborts the current simultaneous conversions both for master and slave. They are restarted once the injected conversion is completed.

Note: Do not convert the same channel on the two ADCs (no overlapping sampling times for the two ADCs when converting the same channel).

In regular simultaneous mode, the user software must convert sequences with the same length. Sampling times for channels with the same sequence number must be equal.

The software is notified by interrupts when it can read the data:

- As the duration of the master regular sequence is equal to the duration of the slave one (as shown in *Figure 300*), the software can enable only one of the two EOC interrupts (for example the master EOC) and read both converted data from the common data register (ADCC_CDR).
  The user can read data from either ADCC_CDR or ADCC_CDR2. Once software selects one of these registers, it must continue using the same register until the ADSTP procedure is executed.

The regular data can also be read using the DMA. Several methods are possible:

- Use two DMA channels (one for the master and one for the slave). In this case DAMDF[1:0] bits must be kept cleared:
  a) Configure the DMA master ADC channel to read ADC_DR from the master. DMA requests are generated at each EOC event of the master ADC.
  b) Configure the DMA slave ADC channel to read ADC_DR from the slave. DMA requests are generated at each EOC event of the slave ADC.
- Configure the dual ADC mode data format DAMDF[1:0] bits, which leaves one DMA channel free for other uses:
  a) Set DAMDF[1:0] = 0b10 or 0b11 (depending on resolution).
  b) A single DMA channel is used (the one corresponding to the master). Configure the DMA master ADC channel to read the common ADC data register (ADCC_CDR or ADCC_CDR2).
  c) A single DMA request is generated each time both the master and slave EOC events occur. At that moment:
  – The slave ADC converted data is available in the upper half-word of the 32-bit ADCC_CDR register
  – The master ADC converted data is available in the lower half-word of the ADCC_CDR register.

Alternatively, data can be read from ADCC_CDR2:

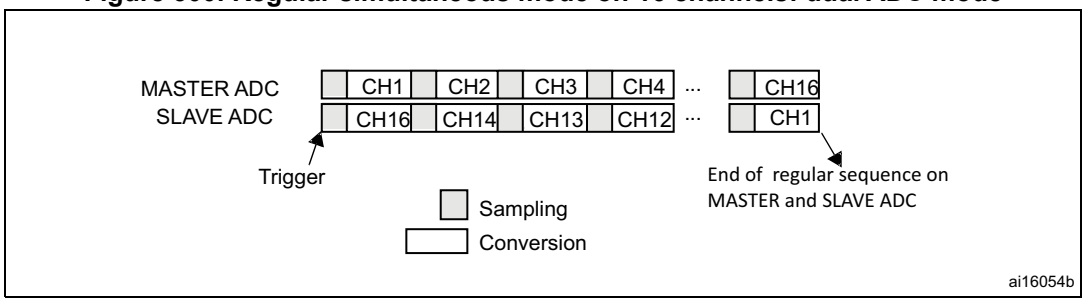
– The first read returns the master ADC data.

– The second read returns the slave ADC data.
After these reads, a DMA request is generated. The DMA transfers the slave data from the same data register and handles two transfers for dual mode operation.

d) Both EOC flags are cleared when the DMA reads the ADCC_CDR register.

Note: When DAMDF[1:0] = 0b10 or 0b11, the user must program the same number of conversions in the master and in the slave sequence. Otherwise, the remaining conversions do not generate a DMA request.

Figure 300. Regular simultaneous mode on 16 channels: dual ADC mode



If the DISCEN bit is set, each "*n*" simultaneous conversion of the regular sequence require a regular trigger event to occur ("n" is defined by DISCNUM).

This mode can be combined with the AUTDLY mode:

- Once a simultaneous conversion of the sequence has completed, the next conversion in the sequence starts only after the EOC flag of both the master and the slave are cleared.
- Once a simultaneous regular sequence of conversions has completed, a new regular trigger event is accepted only after the EOC flag of both the master and the slave are cleared. Any new regular trigger events occurring during the ongoing regular sequence and the associated delay phases are ignored.

The DMA can be used to handle data in regular simultaneous mode combined with AUTDLY mode, assuming that multi-DMA mode is used: bits DAMDF must be set to 0b10 or 0b11.

Interleaved mode with independent injected conversions

This mode is selected by programming DUAL[4:0] bits to 0b00111.

It can be started only on a regular group (usually one channel). The external trigger source comes from the regular channel multiplexer of the master ADC.

After an external trigger occurs:

- The master ADC starts immediately,
- The slave ADC starts after a delay of several ADC clock cycles after the sampling phase of the master ADC has completed (during the master ADC conversion period).

The minimum delay between two conversions in interleaved mode is configured in the DELAY bits in the ADCC_CCR register (see *Table DELAY bits versus ADC resolution*). This delay starts counting one half cycle after the end of the sampling phase of the master

conversion. This way, an ADC cannot start a conversion if the complementary ADC is still sampling its input (only one ADC can sample the input signal at a given time).

- The minimum possible DELAY is 1 to ensure that there is at least one cycle time between the opening of the analog switch of the master ADC sampling phase and the closing of the analog switch of the slave ADC sampling phase.
- The maximum DELAY is equal to the number of cycles corresponding to the selected resolution. However, the user must properly calculate this delay to ensure that an ADC does not start a conversion while the other ADC is still sampling its input.

As the CONT bit is set on both the master and slave ADCs, the selected regular channels of both ADCs are continuously converted.

The software is notified by interrupts when it can read the data at the end of each conversion event (EOC) on the master or slave ADC. A slave and master EOC interrupts are generated (if EOCIE is enabled) and the software can read the ADC_DR of the slave/master ADC.

Table 253. DELAY bits versus ADC resolution(1)

DELAY bits	12-bit resolution	10-bit resolution	8-bit resolution	6-bit resolution
0000	1 * $T_{adc_ker_ck}$	1 * $T_{adc_ker_ck}$	1 * $T_{adc_ker_ck}$	1 * $T_{adc_ker_ck}$
0001	2 * $T_{adc_ker_ck}$	2 * $T_{adc_ker_ck}$	2 * $T_{adc_ker_ck}$	2 * $T_{adc_ker_ck}$
0010	3 * $T_{adc_ker_ck}$	3 * $T_{adc_ker_ck}$	3 * $T_{adc_ker_ck}$	3 * $T_{adc_ker_ck}$
0011	4 * $T_{adc_ker_ck}$	4 * $T_{adc_ker_ck}$	4 * $T_{adc_ker_ck}$	4 * $T_{adc_ker_ck}$
0100	5 * $T_{adc_ker_ck}$	5 * $T_{adc_ker_ck}$	5 * $T_{adc_ker_ck}$	5 * $T_{adc_ker_ck}$
0101	6 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$
0110	7 * $T_{adc_ker_ck}$	7 * $T_{adc_ker_ck}$	7 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$
0111	8 * $T_{adc_ker_ck}$	8 * $T_{adc_ker_ck}$	8 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$
1000	9 * $T_{adc_ker_ck}$	9 * $T_{adc_ker_ck}$	8 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$
1001	10 * $T_{adc_ker_ck}$	10 * $T_{adc_ker_ck}$	8 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$
1010	11 * $T_{adc_ker_ck}$	11 * $T_{adc_ker_ck}$	8 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$
1011	12 * $T_{adc_ker_ck}$	11 * $T_{adc_ker_ck}$	8 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$
1100	13 * $T_{adc_ker_ck}$	11 * $T_{adc_ker_ck}$	8 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$
others	13 * $T_{adc_ker_ck}$	11 * $T_{adc_ker_ck}$	8 * $T_{adc_ker_ck}$	6 * $T_{adc_ker_ck}$

1. Preliminary information, subject to change.

Figure 301. Interleaved mode on 1 channel in continuous conversion mode: Dual ADC mode

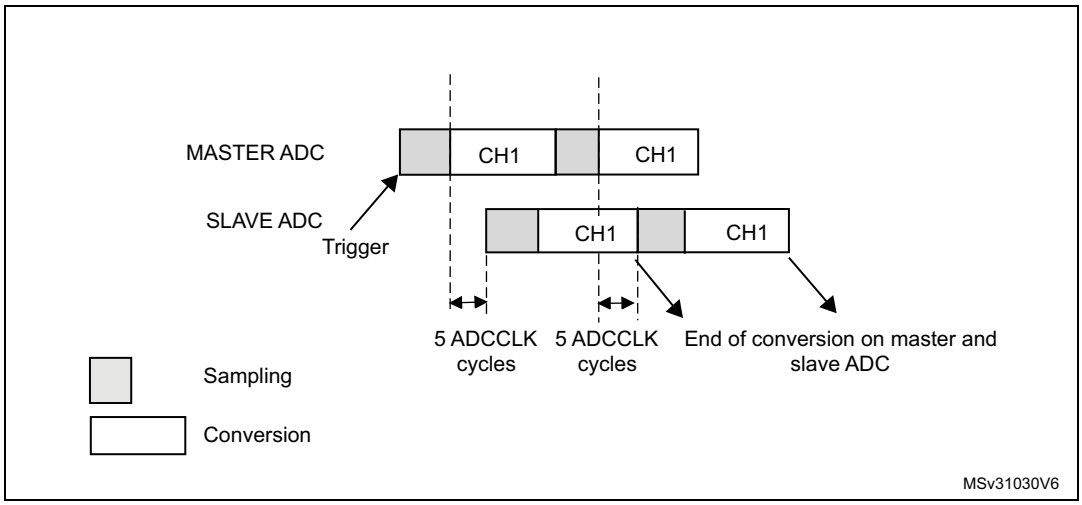
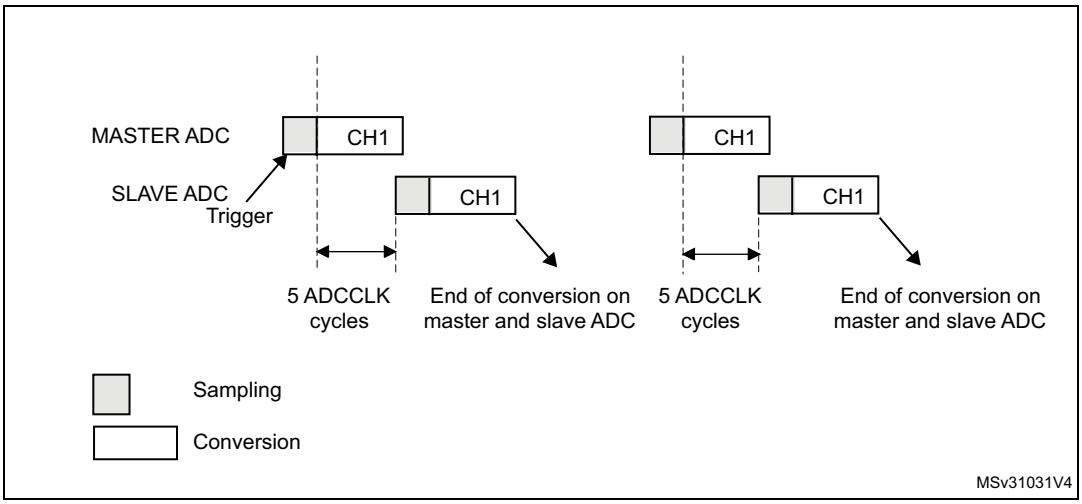


Figure 302. Interleaved mode on 1 channel in single conversion mode: Dual ADC mode



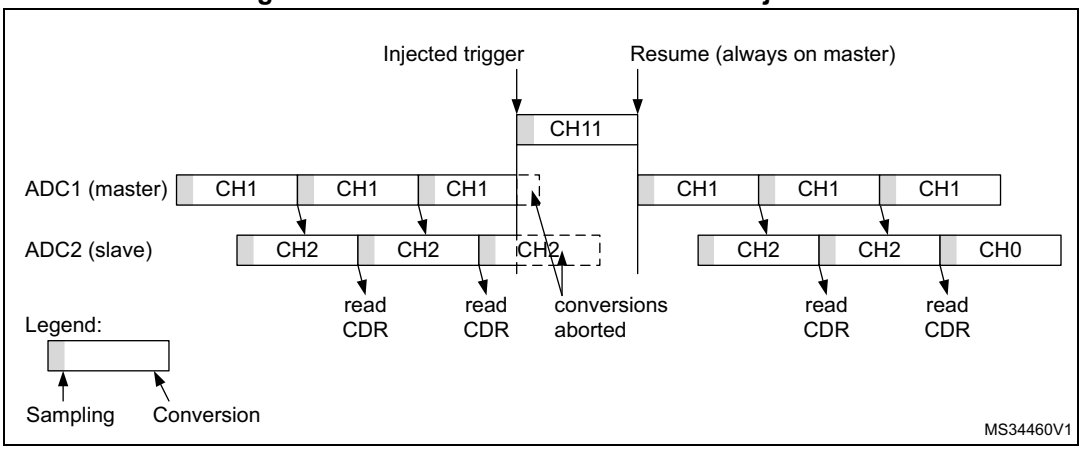
If the DISCEN bit is set, each "*n*" simultaneous conversion ("*n*" is defined by DISCNUM) of the regular sequence require a regular trigger event to occur.

In interleaved mode, injected conversions are supported. When injection is done (either on master or slave), both master and the slave regular conversions are aborted and the sequence is restarted from the master (see *Figure 303*).

Note: In interleaved mode, the ADCx_AWDy_OUT signal for the master ADC is synchronized with the EOC of the master ADC, but the interrupt flag is synchronized with the EOC/JEOC flag of the slave ADC.

In addition, the ROVSM = 0 setting is not supported in oversampling mode.

Figure 303. Interleaved conversion with injection



Alternate trigger mode with independent regular conversions

The alternate trigger mode is selected by programming DUAL[4:0] bits to 0b01001.

This mode can only be started on an injected group. The source of the external trigger comes from the multiplexer injected group of the master ADC.

This mode is only possible when selecting hardware triggers: JEXTEN[1:0] must not be 0x0.

Injected discontinuous mode disabled (JDISCEN = 0)

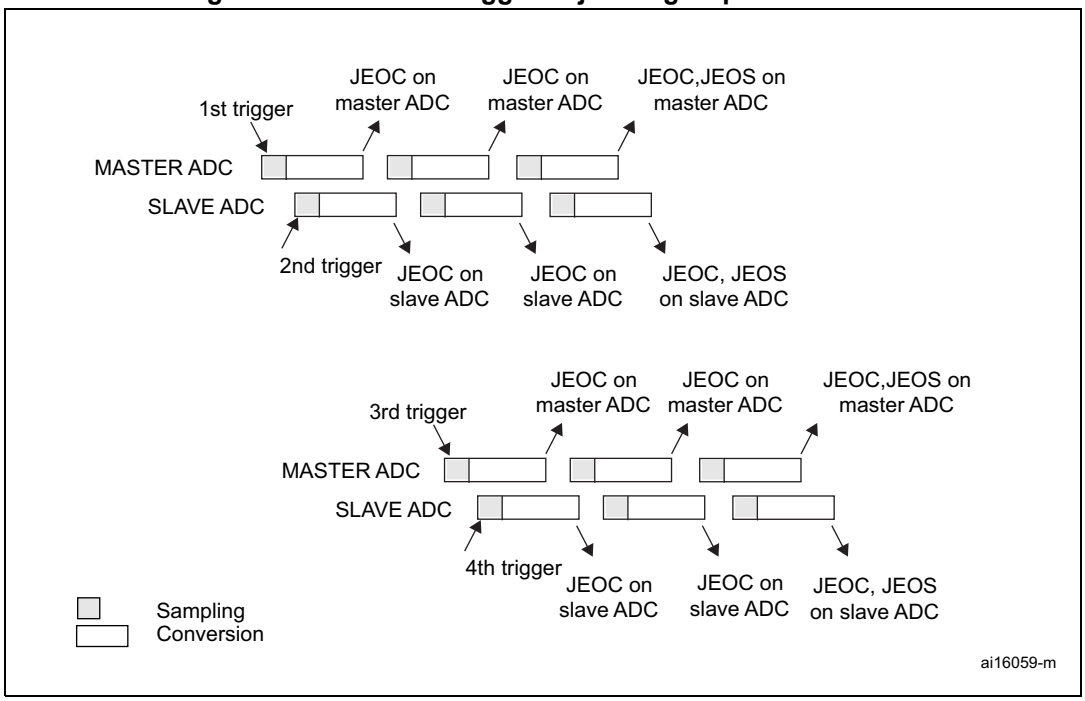
1. When the first trigger occurs, all injected master ADC channels in the group are converted.
2. When the second trigger occurs, all injected slave ADC channels in the group are converted.
3. And so on...

A JEOS interrupt, if enabled, is generated after all injected channels of the master ADC in the group have been converted.

A JEOS interrupt, if enabled, is generated after all injected channels of the slave ADC in the group have been converted.

JEOC interrupts, if enabled, can also be generated after each injected conversion.

Figure 304. Alternate trigger: injected group of each ADC



Note: Regular conversions can be enabled on one or all ADCs. In this case, the regular conversions are independent of each other. A regular conversion is interrupted when the ADC has to perform an injected conversion. It is resumed when the injected conversion is finished.

The time interval between two trigger events must be greater than or equal to one ADC clock period. The minimum time interval between two trigger events that start conversions on the same ADC is the same as in the single ADC mode.

Injected discontinuous mode enabled (JDISCEN = 1)

If the injected discontinuous mode is enabled for both master and slave ADCs:

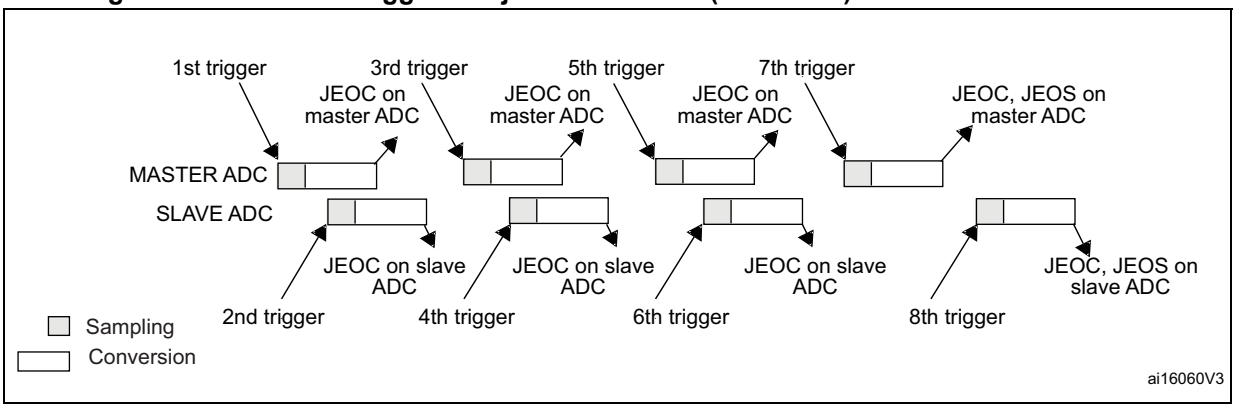
1. When the first trigger occurs, the first injected channel of the master ADC is converted.
2. When the second trigger occurs, the first injected channel of the slave ADC is converted.
3. And so on...

A JEOS interrupt, if enabled, is generated after all injected channels of the master ADC in the group have been converted.

A JEOS interrupt, if enabled, is generated after all injected channels of the slave ADC in the group have been converted.

JEOC interrupts, if enabled, can also be generated after each injected conversion.

Figure 305. Alternate trigger: 4 injected channels (each ADC) in discontinuous mode



Combined regular/injected simultaneous mode

This mode is selected by programming DUAL[4:0] bits to 0b00001.

The simultaneous conversion of a regular group can be interrupted to start the simultaneous conversion of an injected group.

Note: In combined regular/injected simultaneous mode, the user software must convert sequences with the same length.

Combined regular simultaneous and alternate trigger mode

This mode is selected by programming DUAL[4:0] bits to 0b00010.

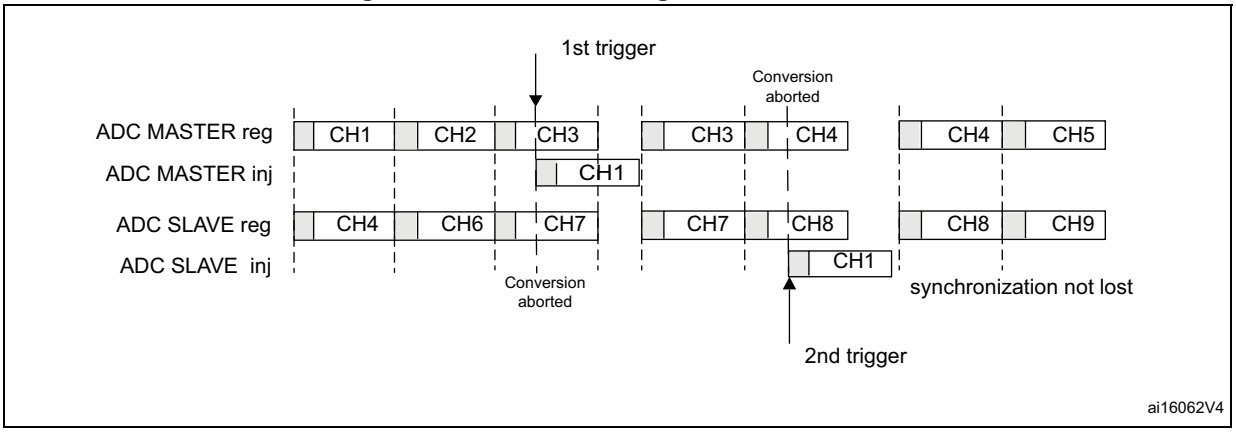
The simultaneous conversion of a regular group can be interrupted to start the alternate trigger conversion of an injected group. *Figure 306* shows the behavior of an alternate trigger interrupting a simultaneous regular conversion.

The injected alternate conversion immediately starts after the injected event. If a regular conversion is already running, in order to ensure synchronization after the injected conversion, the regular conversion of all (master/slave) ADCs is stopped and resumed synchronously at the end of the injected conversion.

Note: In combined regular simultaneous + alternate trigger mode, the user software must convert sequences with the same length.

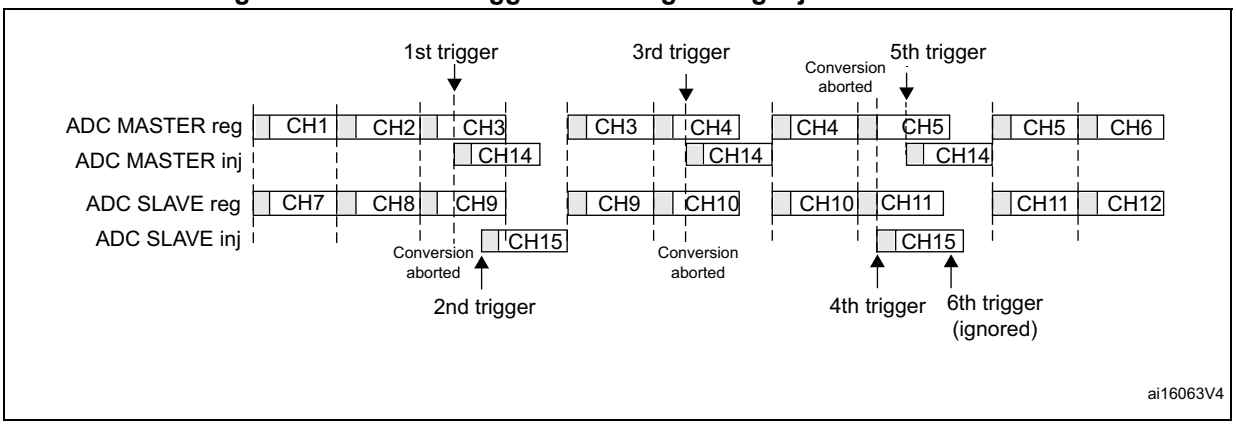
The software trigger is not supported for injected conversions.

Figure 306. Alternate + regular simultaneous



If a trigger occurs during an injected conversion that has interrupted a regular conversion, the alternate trigger is served. *Figure 307* shows the behavior in this case (note that the sixth trigger is ignored because the associated alternate conversion is not complete).

Figure 307. Case of trigger occurring during injected conversion



Combined injected simultaneous and interleaved mode

This mode is selected by programming DUAL[4:0] bits to 0b00011.

An interleaved conversion can be interrupted with a simultaneous injected event.

In this case, the interleaved conversion is immediately interrupted and the simultaneous injected conversion starts. At the end of the injected sequence, the interleaved conversion is resumed. When the interleaved regular conversion resumes, the first regular conversion which is performed is always the master's one. *Figure 308*, *Figure 309*, and *Figure 310* show examples of this behavior.

Note: In this mode, the ROVSM = 0 setting is not supported in oversampling mode.

Figure 308. Interleaved single channel CH0 with injected sequence CH11, CH12

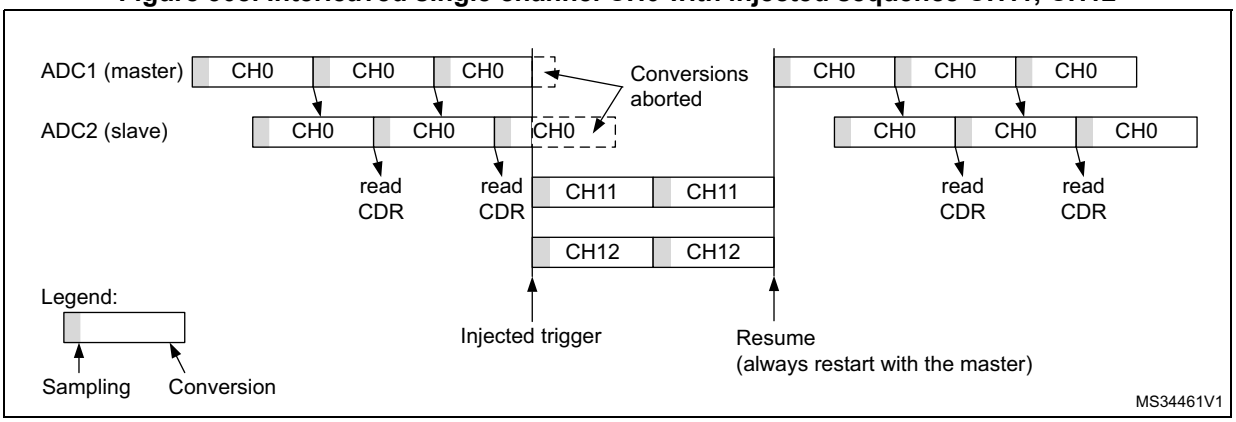


Figure 309. Two interleaved channels (CH1, CH2) with injected sequence CH11, CH12 - case 1: Master interrupted first

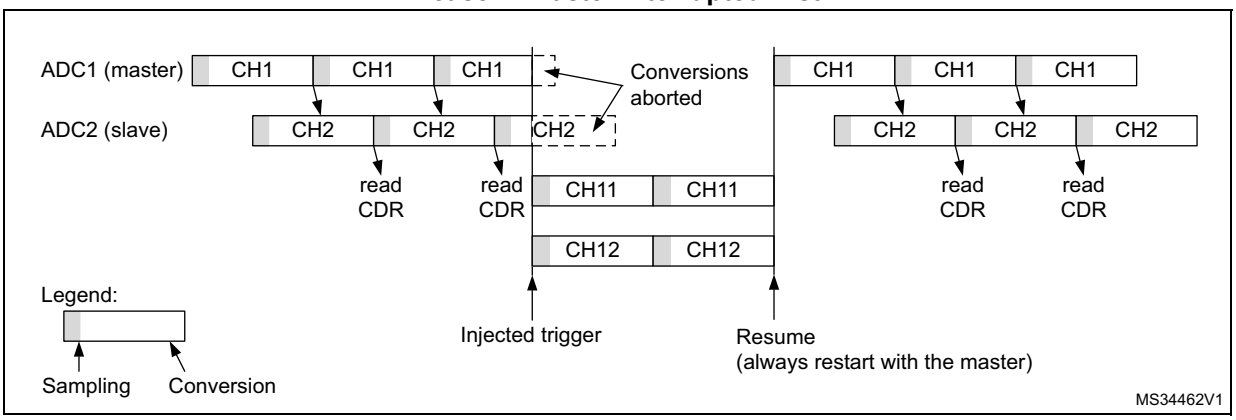
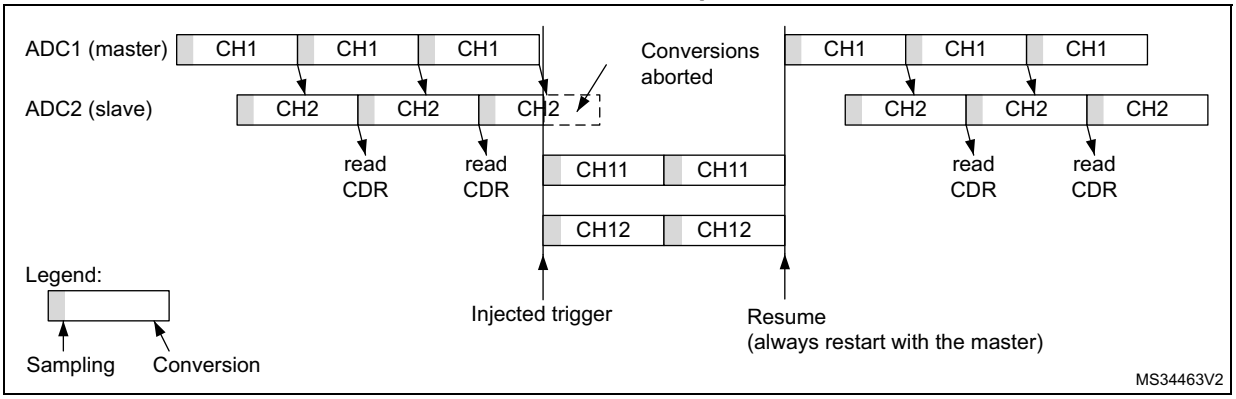


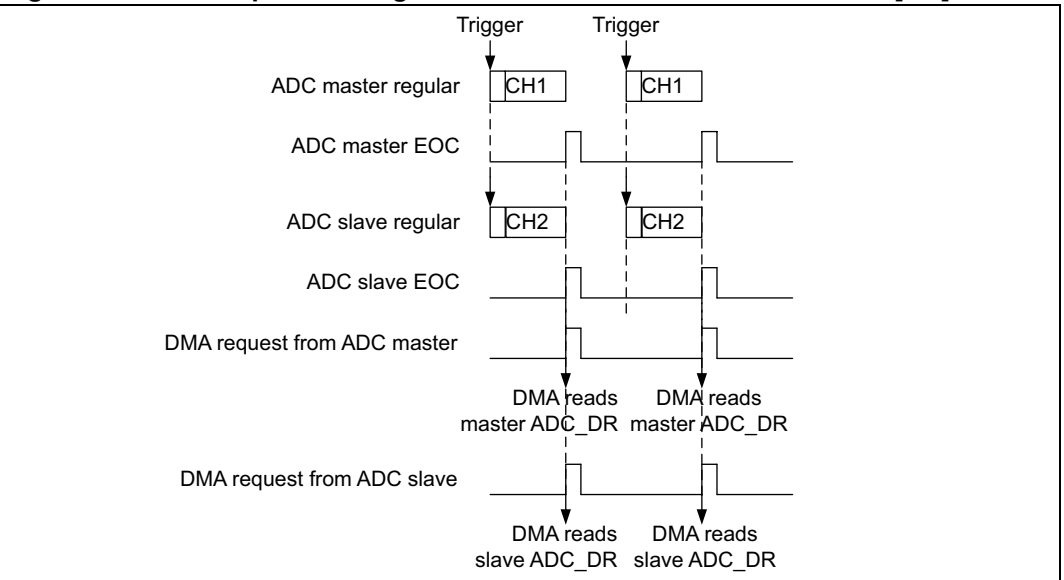
Figure 310. Two interleaved channels (CH1, CH2) with injected sequence CH11, CH12 - case 2: Slave interrupted first



DMA requests in dual ADC mode

In all dual ADC modes, it is possible to use two DMA channels (one for the master, one for the slave) to transfer the data, like in single mode (refer to *Figure 311: DMA requests in regular simultaneous mode when DAMDF[1:0] = 0b00*).

Figure 311. DMA requests in regular simultaneous mode when DAMDF[1:0] = 0b00



Configuration where each sequence contains only one conversion. Master and slave timings are equal.

MSv31032V3

In interleaved mode or in simultaneous regular mode, it is also possible to save one DMA channel and transfer both data using a single DMA channel. To do this, the DAMDF[1:0] bits must be configured in the ADCC_CCR register:

- **DAMDF[1:0] = 0b10, 32-bit format:**

  A DMA request is generated alternatively after the master and slave EOC events have occurred. At that time, the data are alternatively available in the 32-bit register ADCC_CDR2 32-bit register.

  This mode is used when the data width is above 16 bits (conversion data width can exceed 16 bits when functions such as oversampling, compensation and left-bit shift, are used).

  Behavior: A DMA request is generated each time a new 32-bit data is available:

  – First DMA request: ADCC_CDR2[31:0] = MST_ADC_DR[31:0]

  – Second DMA request: ADCC_CDR2[31:0] = SLV_ADC_DR[31:0]

- **DAMDF[1:0] = 0b10, 16-bit format**:

  A DMA request is generated alternatively after the master and slave EOC events have occurred. At that time, two data items are available and the 32-bit register ADCC_CDR contains the two half-words representing two ADC-converted data. The slave ADC data takes the upper half-word and the master ADC data takes the lower half-word. DMA transfers two data by one request.

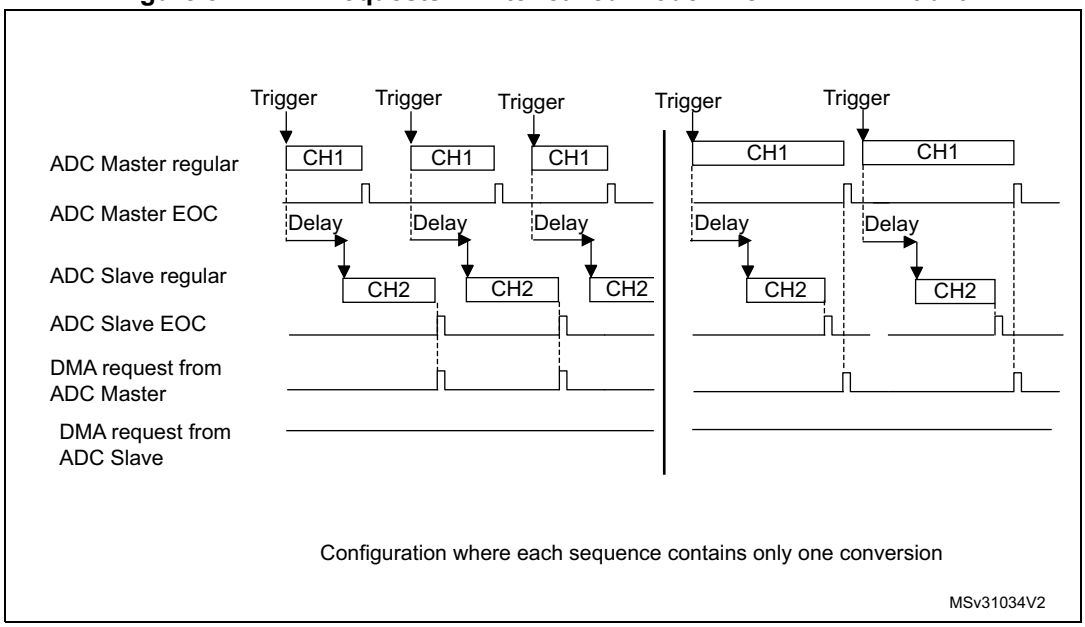
Any value above 16-bit in the master or the slave-converted data is truncated to the least 16 significant bits.

Behavior: A DMA request is generated each time a new 32-bit data is available:

– First DMA request:
ADCC_CDR[31:0] = (SLV_ADC_DR[15:0] << 16) | MST_ADC_DR[15:0]

– Second DMA request:
ADCC_CDR[31:0] = SLV_ADC_DR[15:0] | MST_ADC_DR[15:0]

Figure 312. DMA requests in interleaved mode when DAMDF = 0b10



Note: When using dual ADC mode, the user must take care to configure properly the duration of the master and slave conversions so that a DMA request is generated and served for reading both data (master + slave) before a new conversion data is available.

- **DAMDF[1:0] = 0b11**:

  This mode is similar to the DAMDF[1:0] = 0b10 configuration. The only difference is that on each DMA request (two data are available), two bytes representing two ADC converted data are transferred as a half-word.

  This mode is used when the resolution is 6 bits or 8 bits and data are not signed.

  Behavior: A DMA request is generated each time two data items are available:

  – First DMA request:
  ADCC_CDR[15:0] = (SLV_ADC_DR[7:0] << 8)| MST_ADC_DR[7:0]

  – Second DMA request:
  ADCC_CDR[15:0] = SLV_ADC_DR[7:0] | MST_ADC_DR[7:0]

Note: DMA burst mode is not supported in ADC dual mode.

Overrun detection when OVRMOD = 0

In dual ADC mode (when DUAL[4:0] is not equal to b00000), if an overrun is detected on one of the ADCs, DMA requests are no longer issued to ensure that all the data transferred to the RAM are valid (this behavior occurs whatever the DAMDF configuration). The EOC bit

corresponding to one ADC might remains set because the data register of this ADC contains valid data.

Overrun detection when OVRMOD = 1

In dual ADC mode, if an overrun occurs on either ADC, the data register is overwritten, causing continuous DMA requests. This results in loss of data coherency.

DMA one-shot mode/ DMA circular mode when multi-ADC mode is selected

When DAMDF mode is selected (0b10 or 0b11), DMNGT[1:0] bits of the ADCC_CCR register must also be configured to select between DMA one-shot mode and circular mode, as explained in section *Section : Managing conversions using the DMA*

Stopping the conversions in dual ADC modes

In dual ADC mode, the user software must set the ADSTP/JADSTP control bits in the master ADC to stop the conversions of both ADC. The other ADSTP control bit on the slave ADC has no effect in dual ADC mode.

Once both ADCs are effectively stopped, the ADSTART/JADSTART bits on the master and slave ADCs are both deasserted by hardware.

Analog watchdog in dual mode

The analog watchdog is supported in all dual modes.

Disabling the ADC in dual mode

During a dual-mode operation, the master ADC shares the ADC clock with the slave ADC. To disable the ADC correctly through the ADDIS bit of ADC_CR, set the ADSTP (or JADSP) bit on the master ADC. Once ADSTP (or JADSTP) becomes 0 both for the master and slave ADCs, set the ADDIS bit on the master ADC, then set the ADDIS bit on the slave ADC.

Autoinjection mode in dual mode

Autoinjection mode is not supported when DUAL[4:0] equals 0b00010, 0b00011, 0b00101, 0b00110, 0b00111, or 0b01001.

Additionally, parallel injected oversampling mode is not supported in autoinjection mode.

MDF mode in dual ADC interleaved mode

In dual ADC interleaved modes (DUAL[4:0] = 0b00011 or DUAL[4:0] = 0b00111), the ADC conversion results can be transferred directly to the MDF.

This mode is enabled by setting DMNGT[1:0] bits to 0b10 in the master ADC ADC_CFGR1 register.

The ADC transfers alternatively the 16 least significant bits of the regular data register from the master and the slave converter to a single channel of the MDF. Each transfer resets the EOC flag of each channel once the transfer is complete.

The data format must be 16-bit signed:

ADC_DR[15:12] = Sign extended

ADC_DR[11] = Sign

ADC_DR[10:0] = Data

To obtain 16-bit signed format, the software must configure OFFSET[11:0] bits to 0x800 with OFFSETy_CH[4:0] selecting the target channel set.

Only right aligned data can be set as MDF input format (see *Figure 277: Right alignment (offset enabled, signed value)*).

MDF mode in dual ADC simultaneous mode

It is not mandatory to use the MDF in dual ADC simultaneous mode since conversion data are handled by each individual channel. Single mode with same trigger source results in simultaneous conversion with MDF interface.

Dual ADC modes supported in oversampling mode

It is possible to have oversampling enabled when working in dual ADC configuration. In this case, the two ADCs must be programmed with the very same settings (including oversampling).

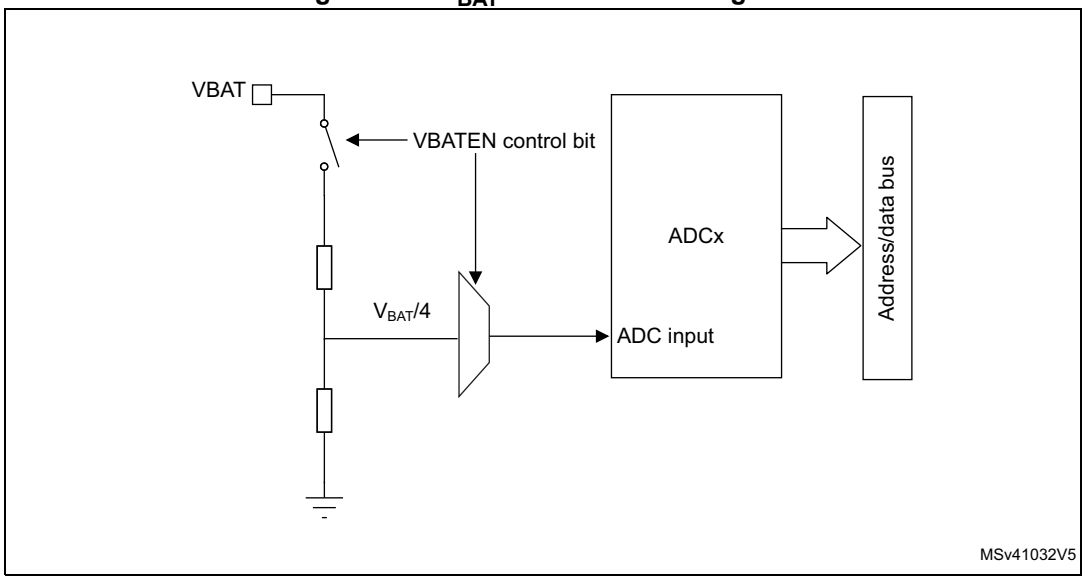
32.4.32 V_{BAT} supply monitoring

The VBATEN bit in the ADCC_CCR register is used to switch to the battery voltage. As the V_{BAT} voltage can be higher than V_{DDA}, to ensure the correct operation of the ADC, the V_{BAT} pin is internally connected to a bridge divider by 4. This bridge is automatically enabled when VBATEN is set, to connect $V_{BAT}/4$ to the ADC input channels (see *Table: ADC interconnection* in *Section 32.4.2: ADC pins and internal signals* for more details). As a consequence, the converted digital value is one fourth of the V_{BAT} voltage. To prevent any unwanted consumption on the battery, it is recommended to enable the bridge divider only when needed, for ADC conversion.

Refer to the electrical characteristics of the device datasheet for the sampling time value to be applied when converting the $V_{BAT}/4$ voltage.

Figure 313 shows the block diagram of the V_{BAT} sensing feature.

Figure 313. V_{BAT} channel block diagram



1. The VBATEN bit must be set to enable the conversion of internal channel for $V_{BAT}/4$.

32.4.33 Monitoring the internal voltage reference

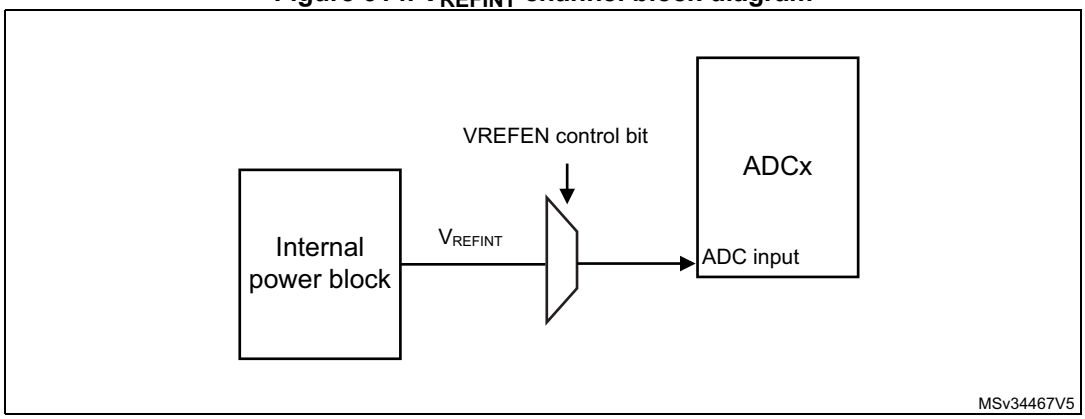
It is possible to monitor the internal voltage reference (V_{REFINT}) to have a reference point for evaluating the ADC V_{REF+} voltage level.

Refer to the *ADC interconnection* table in *Section 32.4.2: ADC pins and internal signals* for details on the ADC input channels to which the internal voltage reference is internally connected.

Refer to the electrical characteristics section of the device datasheet for the sampling time value to be applied when converting the internal voltage reference voltage.

Figure 314 shows the block diagram of the V_{REFINT} sensing feature.

Figure 314. V_{REFINT} channel block diagram



1. The VREFEN bit of the ADCC_CCR register must be set to enable the conversion of internal channels (V_{REFINT}).

Calculating the actual V_{REF+} voltage using the internal reference voltage

The power supply voltage applied to the device may be subject to variations or not precisely known. When V_{DDA} is connected to V_{REF+}, it is possible to compute the actual V_{DDA} voltage using the embedded internal reference voltage (V_{REFINT}). V_{REFINT} and its calibration data acquired by the ADC during the manufacturing process at V_{REF+} can be used to evaluate the actual V_{DDA} voltage level.

The following formula gives the actual V_{REF+} voltage supplying the device:

$$V_{REF+} = V_{REF+_Charac} \times VREFINT_CAL / VREFINT_DATA$$

Where:

- V_{REF+_Charac} is the value of V_{REF+} voltage characterized at V_{REFINT} during the manufacturing process. It is specified in the device datasheet.
- VREFINT_CAL is the VREFINT calibration value.
- VREFINT_DATA is the actual VREFINT output value converted by ADC.

Converting a supply-relative ADC measurement to an absolute voltage value

The ADC is designed to deliver a digital value corresponding to the ratio between V_{REF+} and the voltage applied on the converted channel.

For some applications, the V_{REF+} value is unknown and ADC converted values are right-aligned. In this case, it is necessary to convert this ratio into a voltage independent from V_{REF+}:

$$V_{CHANNELx} = \frac{V_{REF+}}{FULL_SCALE} \times ADC_DATA$$

By replacing V_{REF+} by the formula provided above, the absolute voltage value is given by the following formula

$$V_{CHANNELx} = \frac{V_{REF+_Charac} \times VREFINT_CAL \times ADC_DATA}{VREFINT_DATA \times FULL_SCALE}$$

For applications where V_{REF+} is known and ADC converted values are right-aligned, the absolute voltage value can be obtained by using the following formula:

$$V_{CHANNELx} = \frac{V_{REF+}}{FULL_SCALE} \times ADC_DATA$$

Where:

– VREFINT_CAL is the VREFINT calibration value.

– ADC_DATA is the value measured by the ADC on channel x (right-aligned).

– VREFINT_DATA is the actual VREFINT output value converted by the ADC.

– FULL_SCALE is the maximum digital value of the ADC output. For example with 12-bit resolution, we obtain $2^{12} - 1 = 4095$, or $2^8 - 1 = 255$ with 8-bit resolution.

Note: If ADC measurements are done using an output format other than 12-bit right-aligned, all the parameters must first be converted to a compatible format before the calculation is done.

32.4.34 Monitoring the supply voltage

The ADCs are connected to the internal supply voltage. To use the ADC to measure this voltage, enable the connection through ADC option register.

32.5 ADC interrupts

For each ADC, an interrupt can be generated:

- After ADC power-up, when the ADC is ready (ADRDY flag)
- On the end of any conversion for regular groups (EOC flag)
- On the end of a sequence of conversion for regular groups (EOS flag)
- On the end of any conversion for injected groups (JEOC flag)
- On the end of a sequence of conversion for injected groups (JEOS flag)
- When an analog watchdog detection occurs (AWD1, AWD2 and AWD3 flags)
- When the end of the sampling phase occurs (EOSMP flag)
- When the data overrun occurs (OVR flag)

Separate interrupt enable bits are available for flexibility.

Note: Interrupts are generated by the AHB clock domain. Since the ADC clock is adc_ker_ck, interrupt flags are delayed due to the synchronization between adc_ker_ck and AHB clock domains.

Table 254. ADC interrupts

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop, Standby mode
ADC	ADC ready	ADRDY	ADRDYIE	Set by hardware and cleared by software	Yes	No
	End of conversion of a regular group	EOC	EOCIE			
	End of conversion sequence of a regular group	EOS	EOSIE			
	End of conversion of an injected group	JEOC	JEOCIE			
	End of conversion sequence of an injected group	JEOS	JEOSIE			
	Analog watchdog 1 flag is set	AWD1	AWD1IE			
	Analog watchdog 2 flag is set	AWD2	AWD2IE			
	Analog watchdog 3 flag is set	AWD3	AWD3IE			
	End of sampling phase	EOSMP	EOSMPIE			
	Overrun	OVR	OVRIE			

32.6 ADC registers (for each ADC)

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

32.6.1 ADC interrupt and status register (ADC_ISR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	AWD3	AWD2	AWD1	JEOS	JEOC	OVR	EOS	EOC	EOSMP	ADRDY
						rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **AWD3**: Analog watchdog 3 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LTR[22:0] and HTR[22:0] of ADC_AWD3LTR and ADC_AWD3HTR registers. It is cleared by software writing 1 to it or when ADEN = 0.

0: No analog watchdog 3 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 3 event occurred

Bit 8 **AWD2**: Analog watchdog 2 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LTR[22:0] and HTR[22:0] of ADC_AWD2LTR and ADC_AWD2HTR registers. It is cleared by software writing 1 to it or when ADEN = 0.

0: No analog watchdog 2 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 2 event occurred

Bit 7 **AWD1**: Analog watchdog 1 flag

This bit is set by hardware when the converted voltage crosses the values programmed in the fields LTR[22:0] and HTR[22:0] of ADC_AWD1LTR and ADC_AWD1HTR registers. It is cleared by software writing 1 to it or when ADEN = 0.

0: No analog watchdog 1 event occurred (or the flag event was already acknowledged and cleared by software)

1: Analog watchdog 1 event occurred

Bit 6 **JEOS:** Injected channel end of sequence flag

This bit is set by hardware at the end of the conversions of all injected channels in the group. It is cleared by software writing 1 to it or when ADEN = 0.

0: Injected conversion sequence not complete (or the flag event was already acknowledged and cleared by software)

1: Injected conversions complete

Bit 5 **JEOC:** Injected channel end of conversion flag

This bit is set by hardware at the end of each injected conversion of a channel when a new data is available in the corresponding ADC_JDRy register. It is cleared by software writing 1 to it, when ADEN = 0, or by reading the corresponding ADC_JDRy register.

0: Injected channel conversion not complete (or the flag event was already acknowledged and cleared by software)

1: Injected channel conversion complete

Bit 4 **OVR**: ADC overrun

This bit is set by hardware when an overrun occurs on a regular channel (a new conversion has completed while the EOC flag was already set) or when adc_hclk clock is too slow to manage the data.

It is cleared by software writing 1 to it or when ADEN = 0.

0: No overrun occurred (or the flag event was already acknowledged and cleared by software)

1: Overrun has occurred

Bit 3 **EOS**: End of regular sequence flag

This bit is set by hardware at the end of the conversions of a regular sequence of channels. It is cleared by software writing 1 to it or when ADEN = 0.

0: Regular conversions sequence not complete (or the flag event was already acknowledged and cleared by software)

1: Regular conversions sequence complete

Bit 2 **EOC**: End of conversion flag

This bit is set by hardware at the end of each regular conversion of a channel when a new data is available in the ADC_DR register. It is cleared by software writing 1 to it, when ADEN = 0, or by reading the ADC_DR register.

0: Regular channel conversion not complete (or the flag event was already acknowledged and cleared by software)

1: Regular channel conversion complete

Bit 1 **EOSMP**: End of sampling flag

This bit is set by hardware during the conversion of any channel (only for regular channels), at the end of the sampling phase. It is cleared by software writing 1 or when ADEN = 0.

0: not at the end of the sampling phase (or the flag event was already acknowledged and cleared by software)

1: End of sampling phase reached

Bit 0 **ADRDY**: ADC ready

This bit is set by hardware after the ADC has been enabled (bit ADEN = 1) and when the ADC reaches a state where it is ready to accept conversion requests.

It is cleared by software writing 1 to it or when ADEN = 0.

0: ADC not yet ready to start conversion (or the flag event was already acknowledged and cleared by software)

1: ADC is ready to start conversion

32.6.2 ADC interrupt enable register (ADC_IER)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	AWD3IE	AWD2IE	AWD1IE	JEOSIE	JEOCIE	OVRIE	EOSIE	EOCIE	EOSMPIE	ADRDYIE
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **AWD3IE**: Analog watchdog 3 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog 2 interrupt.

0: Analog watchdog 3 interrupt disabled

1: Analog watchdog 3 interrupt enabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 8 **AWD2IE**: Analog watchdog 2 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog 2 interrupt.

0: Analog watchdog 2 interrupt disabled

1: Analog watchdog 2 interrupt enabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 7 **AWD1IE**: Analog watchdog 1 interrupt enable

This bit is set and cleared by software to enable/disable the analog watchdog 1 interrupt.

0: Analog watchdog 1 interrupt disabled

1: Analog watchdog 1 interrupt enabled

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 6 **JEOSIE**: End of injected sequence of conversions interrupt enable

This bit is set and cleared by software to enable/disable the end of injected sequence of conversions interrupt.

0: JEOS interrupt disabled

1: JEOS interrupt enabled. An interrupt is generated when the JEOS bit is set.

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 5 **JEOCIE**: End of injected conversion interrupt enable

This bit is set and cleared by software to enable/disable the end of an injected conversion interrupt.

0: JEOC interrupt disabled.

1: JEOC interrupt enabled. An interrupt is generated when the JEOC bit is set.

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 4 **OVRIE**: Overrun interrupt enable

This bit is set and cleared by software to enable/disable the Overrun interrupt of a regular conversion.

0: Overrun interrupt disabled

1: Overrun interrupt enabled. An interrupt is generated when the OVR bit is set.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 3 **EOSIE**: End of regular sequence of conversions interrupt enable

This bit is set and cleared by software to enable/disable the end of regular sequence of conversions interrupt.

0: EOS interrupt disabled

1: EOS interrupt enabled. An interrupt is generated when the EOS bit is set.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 2 **EOCIE**: End of regular conversion interrupt enable

This bit is set and cleared by software to enable/disable the end of a regular conversion interrupt.

0: EOC interrupt disabled.

1: EOC interrupt enabled. An interrupt is generated when the EOC bit is set.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 1 **EOSMPIE**: End of sampling flag interrupt enable for regular conversions

This bit is set and cleared by software to enable/disable the end of the sampling phase interrupt for regular conversions.

0: EOSMP interrupt disabled.

1: EOSMP interrupt enabled. An interrupt is generated when the EOSMP bit is set.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 0 **ADRDYIE**: ADC ready interrupt enable

This bit is set and cleared by software to enable/disable the ADC Ready interrupt.

0: ADRDY interrupt disabled

1: ADRDY interrupt enabled. An interrupt is generated when the ADRDY bit is set.

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

32.6.3 ADC control register (ADC_CR)

Address offset: 0x008

Reset value: 0x2000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADCAL	ADCALDIF	DEEPPWD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rs	rw	rw													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JADSTP	ADSTP	JADSTART	ADSTART	ADDIS	ADEN
										rs	rs	rs	rs	rs	rs

Bit 31 **ADCAL**: ADC calibration

This bit is set by software to enter ADC calibration mode.

0: Normal mode

1: Calibration mode

Note: The software is allowed to update the calibration factor by writing ADC_CALFACT only when ADEN = 1, ADCAL = 1, ADSTART = 0 and JADSTART = 0 (ADC enabled, Calibration mode enabled and no conversion ongoing).

Bit 30 **ADCALDIF**: Differential mode for calibration

This bit is set and cleared by software to configure the single-ended or differential input mode for the calibration.

0: Single-ended input calibration mode when ADCAL = 1

1: Differential input calibration mode when ADCAL = 1

Note: The software is allowed to write this bit only when the ADC is disabled and is not calibrating (ADCAL = 0, JADSTART = 0, JADSTP = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Bit 29 **DEEPPWD**: Deep-power-down enable

This bit is set and cleared by software to put the ADC in Deep-power-down mode.

0: ADC not in Deep-power down

1: ADC in Deep-power-down (default reset state)

Note: The software is allowed to write this bit only when the ADC is disabled (ADCAL = 0 and ADEN =0).

Bits 28:6 Reserved, must be kept at reset value.

Bit 5 **JADSTP**: ADC stop of injected conversion command

This bit is set by software to stop and discard an ongoing injected conversion (JADSTP command).

It is cleared by hardware when the conversion is effectively discarded and the ADC injected sequence and triggers can be reconfigured. The ADC is then ready to accept a new start of injected conversions (JADSTART command).

When a software trigger is used, JADSTART bit is cleared by hardware, but JADSTP bit must be programmed to reconfigure the ADC.

0: No ADC stop injected conversion command ongoing

1: Write 1 to stop injected conversions ongoing. Read 1 means that an ADSTP command is in progress.

Note: In autoinjection mode (JAUTO = 1), setting ADSTP bit aborts both regular and injected conversions (do not use JADSTP).

Bit 4 **ADSTP**: ADC stop of regular conversion command

This bit is set by software to stop and discard an ongoing regular conversion (ADSTP command).

It is cleared by hardware when the conversion is effectively discarded. and the ADC regular sequence and triggers can be re-configured. The ADC is then ready to accept a new start of regular conversions (ADSTART command).

When a software trigger is used, ADSTART bit is cleared by hardware. However it is necessary to write ADSTP bit to reconfigure the ADC.

0: No ADC stop regular conversion command ongoing

1: Write 1 to stop ongoing regular conversions. Read 1 means that an ADSTP command is in progress.

Note: In autoinjection mode (JAUTO = 1), setting ADSTP bit aborts both regular and injected conversions (do not use ADSTP).

Note: In dual ADC regular simultaneous and interleaved modes, the ADSTP bit of the master ADC must be used to stop regular conversions on both ADCs. The other ADSTP bit is inactive.

Bit 3 **JADSTART**: ADC start of injected conversion

This bit is set by software to start ADC conversion of injected channels. Depending on the configuration bits JEXTEN[1:0], a conversion starts immediately (software trigger configuration) or once an injected hardware trigger event occurs (hardware trigger configuration).

- In single conversion mode when software trigger is selected (JEXTSEL = 0x0 and JDISEN = 1), this bit is cleared by hardware immediately after injected conversion starts.
- In all cases, after the execution of the JADSTP command, this bit and JADSTP are cleared by hardware simultaneously.

0: No ADC injected conversion is ongoing.

1: Write 1 to start injected conversions. Read 1 means that the ADC is operating and eventually converting an injected channel.

Note: The software is allowed to set JADSTART only when ADEN = 1, ADDIS = 0 and JAUTO = 0 (ADC is enabled and there is no pending request to disable the ADC).

Note: In autoinjection mode (JAUTO = 1), regular and autoinjected conversions are started by setting ADSTART bit (JADSTART must be kept cleared).

Bit 2 **ADSTART**: ADC start of regular conversion

This bit is set by software to start ADC conversion of regular channels. Depending on the configuration bits EXTEN[1:0], a conversion starts immediately (software trigger configuration) or once a regular hardware trigger event occurs (hardware trigger configuration).

- In single conversion mode when software trigger is selected (EXTSEL = 0x0 and DISEN = 1), this bit is cleared by hardware immediately after regular conversion starts.
- In all cases, after the execution of the ADSTP command, this bit and ADSTP are cleared by hardware simultaneously.

0: No ADC regular conversion is ongoing.

1: Write 1 to start regular conversions. Read 1 means that the ADC is operating and eventually converting a regular channel.

Note: The software is allowed to set ADSTART only when ADEN = 1 and ADDIS = 0 (ADC is enabled and there is no pending request to disable the ADC).

Note: In autoinjection mode (JAUTO = 1), regular and autoinjected conversions are started by setting ADSTART bit (JADSTART must be kept cleared)

Bit 1 **ADDIS**: ADC disable command

This bit is set by software to disable the ADC (ADDIS command) and place it into power-down state (OFF state).

It is cleared by hardware once the ADC is effectively disabled (ADEN is also cleared by hardware at this time).

0: no ADDIS command ongoing

1: Write 1 to disable the ADC. Read 1 means that an ADDIS command is in progress.

Note: The software is allowed to set ADDIS only when ADEN = 1, ADDIS = 0 and both ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 0 **ADEN**: ADC enable control

This bit is set by software to enable the ADC. The ADC is effectively ready to operate once the ADRDY flag has been set.

It is cleared by hardware when the ADC is disabled, after the execution of the ADDIS command.

0: ADC is disabled (OFF state)

1: Write 1 to enable the ADC.

Note: The software is allowed to set ADEN only when all the bits of ADC_CR registers are cleared (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

32.6.4 ADC configuration register (ADC_CFGR1)

Address offset: 0x00C

Reset value: 0x8000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	AWD1CH[4:0]					JAUTO	JAWD1EN	AWD1EN	AWD1SGL	Res.	JDISCEN	DISCNUM[2:0]			DISCEN
	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	AUTDLY	CONT	OVRMOD	EXTEN[1:0]		EXTSEL[4:0]					Res.	RES[1:0]		DMNGT[1:0]	
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:26 **AWD1CH[4:0]**: Analog watchdog 1 channel selection

These bits are set and cleared by software. They select the input channel to be guarded by the analog watchdog.

00000: ADC analog input channel 0 monitored by AWD1

00001: ADC analog input channel 1 monitored by AWD1

.....

10011: ADC analog input channel 19 monitored by AWD1

Others: reserved, must not be used

Note: Some channels are not connected physically. Keep the corresponding AWD1CH[4:0] setting to the reset value.

Note: The channel selected by AWD1CH must be also selected into the SQRi or JSQRi registers. The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 25 **JAUTO:** Automatic injected group conversion

This bit is set and cleared by software to enable/disable automatic injected group conversion after regular group conversion.

0: Automatic injected group conversion disabled

1: Automatic injected group conversion enabled

Note: The software is allowed to write this bit only when ADEN = 0.

Note: When dual mode is enabled (DUAL bits in ADCC_CCR register are not equal to zero), the JAUTO bit of the slave ADC is not valid and controlled by the JAUTO bit of the master ADC.

Bit 24 **JAWD1EN**: Analog watchdog 1 enable on injected channels

This bit is set and cleared by software

0: Analog watchdog 1 disabled on injected channels

1: Analog watchdog 1 enabled on injected channels

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 23 **AWD1EN**: Analog watchdog 1 enable on regular channels

This bit is set and cleared by software

0: Analog watchdog 1 disabled on regular channels

1: Analog watchdog 1 enabled on regular channels

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 22 **AWD1SGL**: Enable the watchdog 1 on a single channel or on all channels

This bit is set and cleared by software to enable the analog watchdog on the channel identified by the AWD1CH[4:0] bits or on all the channels

0: Analog watchdog 1 enabled on all channels

1: Analog watchdog 1 enabled on a single channel

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 21 Reserved, must be kept at reset value.

Bit 20 **JDISCEN:** Discontinuous mode on injected channels

This bit is set and cleared by software to enable/disable discontinuous mode on the injected channels of a group.

0: Discontinuous mode on injected channels disabled

1: Discontinuous mode on injected channels enabled

Note: The software is allowed to write this bit only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Note: It is not possible to use autoinjection mode and discontinuous mode simultaneously: the DISCEN and JDISCEN bits must be kept cleared by software when JAUTO is set.

Note: When dual mode is enabled (DUAL bits of ADCC_CCR register are not equal to zero), the JDISCEN bit of the slave ADC is not valid and controlled by the JDISCEN bit of the master ADC.

Bits 19:17 **DISCNUM[2:0]:** Discontinuous mode channel count

These bits are written by software to define the number of regular channels to be converted in discontinuous mode, after receiving an external trigger.

000: 1 channel

001: 2 channels

...

111: 8 channels

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: When dual mode is enabled (DUAL bits in ADCC_CCR register are not equal to zero), the DISCNUM[2:0] bits of the slave ADC are not valid and controlled by the DISCNUM[2:0] bits of the master ADC.

Bit 16 **DISCEN**: Discontinuous mode for regular channels

This bit is set and cleared by software to enable/disable discontinuous mode for regular channels.

0: Discontinuous mode for regular channels disabled

1: Discontinuous mode for regular channels enabled

Note: It is not possible to have both discontinuous mode and continuous mode enabled: DISCEN and CONT cannot both be set (DISCEN takes priority).

Note: It is not possible to use autoinjection mode and discontinuous mode simultaneously: the DISCEN and JDISCEN bits must be kept cleared by software when JAUTO is set.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: When regular dual mode is enabled (DUAL bits in ADCC_CCR register are not equal to zero), the DISCEN bit of the slave ADC is not valid and controlled by the DISCEN bit of the master ADC.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **AUTDLY**: Delayed conversion mode

This bit is set and cleared by software to enable/disable the autodelayed conversion mode.

0: Autodelayed conversion mode off

1: Autodelayed conversion mode on

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Note: When dual mode is enabled (DUAL bits in ADCC_CCR register are not equal to zero), the AUTDLY bit of the slave ADC is not valid and controlled by the AUTDLY bit of the master ADC.

Bit 13 **CONT**: Single / continuous conversion mode for regular conversions

This bit is set and cleared by software. If it is set, regular conversion takes place continuously until it is cleared.

0: Single conversion mode

1: Continuous conversion mode

Note: It is not possible to have both discontinuous mode and continuous mode enabled: DISCEN and CONT cannot both be set (DISCEN takes priority).

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: When regular dual mode is enabled (DUAL bits in ADCC_CCR register are not equal to zero), the CONT bit of the slave ADC is not valid and controlled by the CONT bit of the master ADC.

Bit 12 **OVRMOD**: Overrun mode

This bit is set and cleared by software and configure the way data overrun is managed.

0: ADC_DR register is preserved with the old data when an overrun is detected.

1: ADC_DR register is overwritten with the last conversion result when an overrun is detected.

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bits 11:10 **EXTEN[1:0]**: External trigger enable and polarity selection for regular channels

These bits are set and cleared by software to select the external trigger polarity and enable the trigger of a regular group.

00: Hardware trigger detection disabled (conversions can be launched by software)

01: Hardware trigger detection on the rising edge

10: Hardware trigger detection on the falling edge

11: Hardware trigger detection on both the rising and falling edges

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bits 9:5 **EXTSEL[4:0]**: External trigger selection for regular group

These bits select the external event used to trigger the start of conversion of a regular group:

00000: adc_ext_trg0

00001: adc_ext_trg1

00010: adc_ext_trg2

00011: adc_ext_trg3

00100: adc_ext_trg4

00101: adc_ext_trg5

00110: adc_ext_trg6

00111: adc_ext_trg7

...

11111: adc_ext_trg31

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 4 Reserved, must be kept at reset value.

Bits 3:2 **RES[1:0]**: Data resolution

These bits are written by software to select the resolution of the conversion.

00: 12-bit

01: 10-bit

10: 8-bit

11: 6-bit

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 1:0 **DMNGT[1:0]**: Data management configuration

These bits are set and cleared by software to select how the ADC interface output data are managed.

00: Regular conversion data stored in DR only

01: DMA one-shot mode selected

10: MDF mode selected

11: DMA circular mode selected

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no conversion is ongoing).

32.6.5 ADC configuration register 2 (ADC_CFGR2)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LSHIFT[3:0]				Res.	Res.	OVSR[9:0]									
rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMPTRIG	SWTRIG	BULB	Res.	Res.	ROVSM	TROVS	OVSS[3:0]				Res.	Res.	Res.	Res.	ROVSE
rw	rw	rw			rw	rw	rw	rw	rw	rw					rw

Bits 31:28 **LSHIFT[3:0]**: Left shift factor

This bitfield is set and cleared by software to define the left shifting applied to the final result with or without oversampling.

0000: No left shift
0001: 1-bit left shift
0010: 2-bit left shift
0011: 3-bit left shift
0100: 4-bit left shift
0101: 5-bit left shift
0110: 6-bit left shift
0111: 7-bit left shift
1000: 8-bit left shift
1001: 9-bit left shift
1010: 10-bit left shift
1011: 11-bit left shift
1100: 12-bit left shift
1101: 13-bit left shift
1110: 14-bit left shift
1111: 15-bit left shift

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:16 **OVSR[9:0]**: Oversampling ratio

This bitfield is set and cleared by software to define the oversampling ratio.

0: 1x (no oversampling)
1: 2x
2: 3x
...
1023: 1024x

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bit 15 **SMPTRIG**: Sampling time control trigger mode

This bit is set and cleared by software to enable the sampling time control trigger mode.

0: Sampling time control trigger mode disabled
1: Sampling time control trigger mode enabled

If EXTEN[1:0] bits are set to 01, the sampling time starts on the trigger rising edge, and the conversion starts on the trigger falling edge.

The SMPTRIG bit must not be set when the BULB bit is set.

When EXTEN[1:0] bits are set to 00, set SWTRIG to start the sampling.

Note: The software is allowed to write this bit only when ADEN = 0. When the discontinuous mode is used, only DISCNUM[2:0] = 000 configuration is compatible with sampling time control trigger mode*.*

Bit 14 **SWTRIG**: Software trigger bit for sampling time control trigger mode

This bit is set and cleared by software to trigger the conversion in sampling time control trigger mode.

0: Software trigger starts the conversion for sampling time control trigger mode
1: Software trigger starts the sampling for *sampling time control trigger mode*

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing), SMPTRIG = 1, and EXTEN[0:1] = 00.

Bit 13 **BULB**: Bulb sampling mode

This bit is set and cleared by software to enable the bulb sampling mode.

0: Bulb sampling mode disabled

1: Bulb sampling mode enabled. The sampling period starts just after the previous end of conversion.

BULB bit must not be set when the SWTRIG bit is set.

The very first ADC conversion is performed with the sampling time specified in SMPx bits.

Note: The software is allowed to write this bit only when ADEN = 0. When the discontinuous mode is used, only DISCNUM[2:0] = 000 configuration is compatible with bulb mode.

Bits 12:11 Reserved, must be kept at reset value.

Bit 10 **ROVSM**: Regular oversampling mode

This bit is set and cleared by software to select the regular oversampling mode.

0: Continued mode: When injected conversions are triggered, the oversampling is temporary stopped and continued after the injection sequence (oversampling buffer is maintained during injected sequence)

1: Resumed mode: When injected conversions are triggered, the current oversampling is aborted and resumed from start after the injection sequence (oversampling buffer is zeroed by injected sequence start)

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing). It is recommended to clear both JOVSE and GCOMP when ROVSM = 0.

Bit 9 **TROVS**: Triggered regular oversampling

This bit is set and cleared by software to enable triggered oversampling

0: All oversampled conversions for a channel are done consecutively following a trigger

1: Each oversampled conversion for a channel needs a new trigger

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 8:5 **OVSS[3:0]**: Oversampling shift

This bitfield is set and cleared by software to define the right shifting applied to the raw oversampling result.

0000: No shift
0001: 1-bit shift
0010: 2-bit shift
0011: 3-bit shift
0100: 4-bit shift
0101: 5-bit shift
0110: 6-bit shift
0111: 7-bit shift
1000: 8-bit shift
1001: 9-bit shift
1010: 10-bit shift
Other: reserved, must not be used

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing).

Bits 4:1 Reserved, must be kept at reset value.

Bit 0 **ROVSE**: Regular oversampling enable

This bit is set and cleared by software to enable regular oversampling.

0: Regular oversampling disabled
1: Regular oversampling enabled

Note: The software is allowed to write this bit only when ADSTART = 0 (which ensures that no conversion is ongoing)

32.6.6 ADC sample time register 1 (ADC_SMPR1)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	SMP9[2:0]			SMP8[2:0]			SMP7[2:0]			SMP6[2:0]			SMP5[2:1]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP5[0]	SMP4[2:0]			SMP3[2:0]			SMP2[2:0]			SMP1[2:0]			SMP0[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:0 **SMPx[2:0]:** Channel x sampling time selection (x = 9 to 0)

These bits are written by software to select the sampling time individually for each channel. During sample cycles, the channel selection bits must remain unchanged.

000: 1.5 ADC clock cycles
001: 2.5 ADC clock cycles
010: 6.5 ADC clock cycles
011: 11.5 ADC clock cycles
100: 23.5 ADC clock cycles
101: 46.5 ADC clock cycles
110: 246.5 ADC clock cycles
111: 1499.5 ADC clock cycles

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Note: Some channels are not connected physically. Keep the corresponding SMPx[2:0] setting to the reset value.

32.6.7 ADC sample time register 2 (ADC_SMPR2)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	SMP19[2:0]			SMP18[2:0]			SMP17[2:0]			SMP16[2:0]			SMP15[2:1]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMP15[0]	SMP14[2:0]			SMP13[2:0]			SMP12[2:0]			SMP11[2:0]			SMP10[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:0 **SMPx[2:0]:** Channel x sampling time selection (x = 19 to 10)

These bits are written by software to select the sampling time individually for each channel. During sampling cycles, the channel selection bits must remain unchanged.

000: 1.5 ADC clock cycles
001: 2.5 ADC clock cycles
010: 6.5 ADC clock cycles
011: 11.5 ADC clock cycles
100: 23.5 ADC clock cycles
101: 46.5 ADC clock cycles
110: 246.5 ADC clock cycles
111: 1499.5 ADC clock cycles

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Note: Some channels are not connected physically. Keep the corresponding SMPx[2:0] setting to the reset value.

32.6.8 ADC channel preselection register (ADC_PCSEL)

Address offset: 0x01C

Reset value: 0x0000 0000

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **PCSEL[19:0]**: Channel i ($V_{INP[i]}$) preselection

These bits are written by software to preselect the input channel I/O to be converted.

0: Input channel i ($V_{INP[i]}$) is not preselected for conversion, the result of the ADC conversion for this channel is wrong

1: Input channel i ($V_{INP[i]}$) is preselected for conversion

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Configuring the PCSEL bit is not necessary for the internal channels (such as VREFINT).

32.6.9 ADC regular sequence register 1 (ADC_SQR1)

Address offset: 0x030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SQ4[4:0]					Res.	SQ3[4:0]					Res.	SQ2[4]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ2[3:0]				Res.	SQ1[4:0]					Res.	Res.	LEN[3:0]			
rw	rw	rw	rw		rw	rw	rw	rw	rw			rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ4[4:0]:** 4th conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 4th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ3[4:0]:** 3rd conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 3rd in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ2[4:0]:** 2nd conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 2nd in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ1[4:0]:** 1st conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 1st in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bits 5:4 Reserved, must be kept at reset value.

Bits 3:0 **LEN[3:0]:** Regular channel sequence length

These bits are written by software to define the total number of conversions in the regular channel conversion sequence.
0000: 1 conversion
0001: 2 conversions
...
1111: 16 conversions

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

32.6.10 ADC regular sequence register 2 (ADC_SQR2)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SQ9[4:0]					Res.	SQ8[4:0]					Res.	SQ7[4]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ7[3:0]				Res.	SQ6[4:0]					Res.	SQ5[4:0]				
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ9[4:0]:** 9th conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 9th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ8[4:0]:** 8th conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 8th in the regular conversion sequence
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ7[4:0]:** 7th conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 7th in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ6[4:0]:** 6th conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 6th in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ5[4:0]:** 5th conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 5th in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

32.6.11 ADC regular sequence register 3 (ADC_SQR3)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SQ14[4:0]					Res.	SQ13[4:0]					Res.	SQ12[4]
			rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SQ12[3:0]				Res.	SQ11[4:0]					Res.	SQ10[4:0]				
rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **SQ14[4:0]:** 14th conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 14th in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 23 Reserved, must be kept at reset value.

Bits 22:18 **SQ13[4:0]:** 13th conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 13th in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 17 Reserved, must be kept at reset value.

Bits 16:12 **SQ12[4:0]:** 12th conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 12th in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 11 Reserved, must be kept at reset value.

Bits 10:6 **SQ11[4:0]:** 11th conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 11th in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ10[4:0]:** 10th conversion in regular sequence
These bits are written by software with the channel number (0 to 19) assigned as the 10th in the regular conversion sequence.
Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

32.6.12 ADC regular sequence register 4 (ADC_SQR4)

Address offset: 0x03C

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	SQ16[4:0]					Res.	SQ15[4:0]				
					rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:6 **SQ16[4:0]:** 16th conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 16th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **SQ15[4:0]:** 15th conversion in regular sequence

These bits are written by software with the channel number (0 to 19) assigned as the 15th in the regular conversion sequence.

Note: The software is allowed to write these bits only when ADSTART = 0 (which ensures that no regular conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

32.6.13 ADC regular data register (ADC_DR)

Address offset: 0x040

Reset value: 0x0000 0000

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

Bits 31:0 **RDATA[31:0]**: Regular data converted

These bits are read-only. They contain the conversion result from the last converted regular channel. The data are left- or right-aligned as described in *Section 32.4.26: Data management*.

32.6.14 ADC injected sequence register (ADC_JSQR)

Address offset: 0x04C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
JSQ4[4:0]					Res.	JSQ3[4:0]					Res.	JSQ2[4:1]			
rw	rw	rw	rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
JSQ2[0]	Res.	JSQ1[4:0]					JEXTEN[1:0]		JEXTSEL[4:0]					JLEN[1:0]	
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 **JSQ4[4:0]:** 4th conversion in the injected sequence

These bits are written by software with the channel number (0 to 19) assigned as the 4th in the injected conversion sequence.

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 26 Reserved, must be kept at reset value.

Bits 25:21 **JSQ3[4:0]:** 3rd conversion in the injected sequence

These bits are written by software with the channel number (0 to 19) assigned as the 3rd in the injected conversion sequence.

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 20 Reserved, must be kept at reset value.

Bits 19:15 **JSQ2[4:0]:** 2nd conversion in the injected sequence

These bits are written by software with the channel number (0 to 19) assigned as the 2nd in the injected conversion sequence.

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bit 14 Reserved, must be kept at reset value.

Bits 13:9 **JSQ1[4:0]:** 1st conversion in the injected sequence

These bits are written by software with the channel number (0 to 19) assigned as the 1st in the injected conversion sequence.

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bits 8:7 **JEXTEN[1:0]**: External trigger enable and polarity selection for injected channels

These bits are set and cleared by software to select the external trigger polarity and enable the trigger of an injected group.

00: Hardware trigger detection disabled (conversions can be launched by software)

01: Hardware trigger detection on the rising edge

10: Hardware trigger detection on the falling edge

11: Hardware trigger detection on both the rising and falling edges

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bits 6:2 **JEXTSEL[4:0]**: External trigger selection for injected group

These bits select the external event used to trigger the start of conversion of an injected group:

00000: adc_jext_trg0

00001: adc_jext_trg1

00010: adc_jext_trg2

00011: adc_jext_trg3

00100: adc_jext_trg4

00101: adc_jext_trg5

00110: adc_jext_trg6

00111: adc_jext_trg7

...

11111: adc_jext_trg31

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Bits 1:0 **JLEN[1:0]:** Injected channel sequence length

These bits are written by software to define the total number of conversions in the injected channel conversion sequence.

00: 1 conversion

01: 2 conversions

10: 3 conversions

11: 4 conversions

Note: The software is allowed to write these bits only when JADSTART = 0 (which ensures that no injected conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for conversion.

32.6.15 ADC offset y configuration register (ADC_OFCFGRy)

Address offset: 0x050 + 0x004 * (y -1), (y = 1 to 4)

Reset value: 0x0000 0000

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

Bits 31:27 **OFFSET_CH[4:0]:** Channel selection for the data offset y

These bits are written by software to define the channel to which the offset programmed in bits OFFSET[21:0] applies.

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for the data offset y.

Bit 26 **SSAT**: Signed saturation enable

This bit is set and cleared by software to enable the signed saturation feature.(see *Section : Data register, data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, JOVSS, JLSHIFT[a], USAT, SSAT, POSOFF)*).

0: Offset is subtracted maintaining the data integrity and extending converted data size (13-bit signed format)

1: Offset is subtracted and result is saturated to maintain the converted data size

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 25 **USAT**: Unsigned saturation enable

This bit is set and cleared by software to enable the unsigned saturation feature.

0: Offset is subtracted maintaining the data integrity

1: Offset is subtracted and result is saturated to maintain the converted data size

Note: The software is allowed to write these bits only when ADSTART = and JADSTART = 0 (which ensures that no conversion is ongoing).

Bit 24 **POSOFF**: Positive offset enable

This bit is set and cleared by software to enable the positive offset

0: Negative offset

1: Positive offset

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

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

32.6.16 ADC offset y register (ADC_OFRy)

Address offset: 0x060 + 0x004 * (y -1), (y= 1 to 4)

Reset value: 0x0000 0000

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

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

Bits 21:0 **OFFSET[21:0]:** Data offset y for the channel programmed in OFFSETy_CH[4:0] bits

These bits are written by software to define the offset y to be subtracted from the raw converted data when converting a channel (can be regular or injected). The channel to which applies the data offset y must be programmed in the OFFSETy_CH[4:0] bits. The conversion result can be read from in the ADC_DR (regular conversion) or from in the ADC_JDRyi registers (injected conversion).

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

32.6.17 ADC gain compensation register (ADC_GCOMP)

Address offset: 0x070

Reset value: 0x0000 1000

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

Bit 31 **GCOMP:** Gain compensation mode

This bit is set and cleared by software to enable the Gain compensation mode.

0: Regular ADC operation mode

1: Gain compensation enabled and applied on all channels.

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensure that no conversion is ongoing)

Bits 30:14 Reserved, must be kept at reset value.

Bits 13:0 **GCOMPCOEFF[13:0]:** Gain compensation coefficient

These bits are set and cleared by software to program the gain compensation coefficient.

00 1000 0000 0000: gain factor of 0.5

...

01 0000 0000 0000: gain factor of 1

10 0000 0000 0000: gain factor of 2

11 0000 0000 0000: gain factor of 3

...

The coefficient is divided by 4096 to get the gain factor ranging from 0 to 3.9999756

Note: This gain compensation is only applied when GCOMP bit is set.

Note: The software is allowed to write this bit only when ADSTART = 0 and JADSTART = 0 (which ensure that no conversion is ongoing).

32.6.18 ADC injected channel y data register (ADC_JDRy)

Address offset: 0x080 + 0x004 * (y - 1), (y = 1 to 4)

Reset value: 0x0000 0000

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

Bits 31:0 **JDATA[31:0]:** Injected data

These bits are read-only. They contain the conversion result from injected channel y. The data are left- or right-aligned as described in *Section 32.4.26: Data management*.

32.6.19 ADC analog watchdog 2 configuration register (ADC_AWD2CR)

Address offset: 0x0A0

Reset value: 0x0000 0000

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **AWDCH[19:0]**: Analog watchdog 2 channel selection

These bits are set and cleared by software. They enable and select the input channels to be guarded by the analog watchdog 2.

AWDCH[i] = 0: ADC analog input channel i is not monitored by AWD2

AWDCH[i] = 1: ADC analog input channel i is monitored by AWD2

When AWDCH[19:0] = 000..0, the analog Watchdog 2 is disabled

Note: The channels selected by AWDCH must be also selected in the SQi or JSQi bits.

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for the analog watchdog.

32.6.20 ADC analog watchdog 3 configuration register (ADC_AWD3CR)

Address offset: 0x0A4

Reset value: 0x0000 0000

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **AWDCH[19:0]**: Analog watchdog 3 channel selection

These bits are set and cleared by software. They enable and select the input channels to be guarded by the analog watchdog 3.

AWDCH[i] = 0: ADC analog input channel i is not monitored by AWD3

AWDCH[i] = 1: ADC analog input channel i is monitored by AWD3

When AWDCH[19:0] = 000..0, the analog Watchdog 3 is disabled.

Note: The channels selected by AWDCH must be also selected in the SQi or JSQi bits.

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Note: Some channels are not connected physically and must not be selected for the analog watchdog.

32.6.21 ADC analog watchdog 1 lower threshold register (ADC_AWD1LTR)

Address offset: 0x0A8

Reset value: 0x0000 0000

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

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:0 **LTR[22:0]**: Analog watchdog 1 lower threshold

These bits are set and cleared by software to define the lower threshold for analog watchdog 1.

Refer to *Section 32.4.29: Analog window watchdog (AWD1EN, JAWD1EN, AWD1SGL, AWD1CH, AWDCH of ADC_AWD2CR and ADC_AWD3CR, HTR, LTR, AWDFILT)*.

32.6.22 ADC analog watchdog 1 higher threshold register (ADC_AWD1HTR)

Address offset: 0x0AC

Reset value: 0x003F FFFF

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

Bits 31:29 **AWDFILT[2:0]**: Analog watchdog filtering parameter

These bits are set and cleared by software.

000: No filtering, one detection generates an AWD1 flag or an interrupt

001: two consecutive detections generate an AWD1 flag or an interrupt

...

111: Eight consecutive detections generate an AWD1 flag or an interrupt

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

Bits 28:23 Reserved, must be kept at reset value.

Bits 22:0 **HTR[22:0]**: Analog watchdog 1 higher threshold

These bits are set and cleared by software to define the higher threshold for analog watchdog 1. Refer to *Analog windows watchdog* section.

32.6.23 ADC analog watchdog 2 lower threshold register (ADC_AWD2LTR)

Address offset: 0x0B0

Reset value: 0x0000 0000

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

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:0 **LTR[22:0]**: Analog watchdog 2 lower threshold

These bits are set and cleared by software to define the lower threshold for analog watchdog 2. Refer to *Analog windows watchdog* section.

32.6.24 ADC analog watchdog 2 higher threshold register (ADC_AWD2HTR)

Address offset: 0x0B4

Reset value: 0x003F FFFF

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

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:0 **HTR[22:0]**: Analog watchdog 2 higher threshold

These bits are set and cleared by software to define the higher threshold for analog watchdog 2. Refer to *Analog windows watchdog* section.

32.6.25 ADC analog watchdog 3 lower threshold register (ADC_AWD3LTR)

Address offset: 0x0B8

Reset value: 0x0000 0000

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

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:0 **LTR[22:0]**: Analog watchdog 3 lower threshold

These bits are set and cleared by software to define the lower threshold for analog watchdog 3. Refer to *Analog windows watchdog* section.

32.6.26 ADC analog watchdog 3 higher threshold register (ADC_AWD3HTR)

Address offset: 0x0BC

Reset value: 0x003F FFFF

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

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:0 **HTR[22:0]**: Analog watchdog 3 higher threshold

These bits are set and cleared by software to define the higher threshold for analog watchdog 3. Refer to *Analog windows watchdog* section.

32.6.27 ADC differential mode selection register (ADC_DIFSEL)

Address offset: 0x0C0

Reset value: 0x0000 0000

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **DIFSEL[19:0]:** Differential mode for channels 19 to 0.

These bits are set and cleared by software. They allow to select if a channel is configured as single-ended or differential or pseudodifferential mode.

DIFSEL[i] = 0: ADC analog input channel is configured in single-ended mode

DIFSEL[i] = 1: ADC analog input channel i is configured in differential or pseudodifferential mode

Note: The DIFSEL bits corresponding to channels that are either connected to a single-ended I/O port or to an internal channel must be kept their reset value (single-ended input mode).

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (which ensures that no conversion is ongoing).

32.6.28 ADC calibration factors (ADC_CALFACT)

Address offset: 0x0C4

Reset value: 0x0000 0000

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

Bit 31 **CALADDOS**: Calibration additional offset

This bit is set by software to add a positive offset at the input stage of ADC for calibration purpose.

0: Calibration additional positive offset disabled

1: Calibration additional positive offset enabled

Note: The software is allowed to write these bits only when ADSTART = 0, and JADSTART = 0 (ADC is enabled, calibration mode is selected, and no conversion is ongoing).

Bits 30:25 Reserved, must be kept at reset value.

Bits 24:16 **CALFACT_D[8:0]**: Calibration factors in differential mode

These bits are written by hardware or by software.

Once a differential input calibration is complete, they are updated by hardware with the calibration factors.

The software can write these bits with a new calibration factor. If the new calibration factor is different from the current one stored in the analog ADC, it is applied once a new differential conversion is launched.

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (ADC is enabled, no calibration is ongoing and no conversion is ongoing).

Bits 15:9 Reserved, must be kept at reset value.

Bits 8:0 **CALFACT_S[8:0]**: Calibration factors In single-ended mode

These bits are written by hardware or by software.

Once a single-ended input calibration is complete, they are updated by hardware with the calibration factors.

The software can write these bits with a new calibration factor. If the new calibration factor is different from the current one stored in the analog ADC, it is applied once a new single-ended conversion is launched.

Note: The software is allowed to write these bits only when ADSTART = 0 and JADSTART = 0 (ADC is enabled, no calibration is ongoing and no conversion is ongoing).

This bit can only be cleared by a power-on reset.

32.6.29 ADC option register (ADC_OR)

Address offset: 0x0D0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OP2	OP1	OP0
													rw	rw	rw

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

Bit 2 **OP2:** Option bit 2

This bit is set and cleared by software to enable/disable the V_{DDCORE} channel.

0: V_{DDCORE} channel disabled

1: V_{DDCORE} channel enabled

Note: This bit is only available in ADC2 and ADC3.

Bit 1 **OP1:** Option bit 1

This bit is set and cleared by software to choose the ADC bandgap source

0: ADC internal bandgap disabled

1: ADC internal bandgap enabled

Note: The software is allowed to write this bit only when ADC is disabled.

Bit 0 **OP0:** Option bit 0

This bit is set and cleared by software to enable the internal reference buffer.

0: ADC internal reference voltage buffer disabled

1: ADC internal reference voltage buffer enabled

Note: The software is allowed to write this bit only when ADC is disabled.

32.7 ADC common registers

These registers define the control and status registers common to master and slave ADCs.

32.7.1 ADC common status register (ADCC_CSR)

Address offset: 0x000

Reset value: 0x0000 0000

This register provides an image of the flags of the different ADCs. Nevertheless it is read-only and does not allow to clear the different flags. Instead each flag must be cleared by writing 0 to it in the corresponding ADC_ISR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	AWD3_SLV	AWD2_SLV	AWD1_SLV	JEOS_SLV	JEOC_SLV	OVR_SLV	EOS_SLV	EOC_SLV	EOSMP_SLV	ADRDY_SLV
						r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	AWD3_MST	AWD2_MST	AWD1_MST	JEOS_MST	JEOC_MST	OVR_MST	EOS_MST	EOC_MST	EOSMP_MST	ADRDY_MST
						r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:26 Reserved, must be kept at reset value.

Bit 25 **AWD3_SLV:** Analog watchdog 3 flag of the slave ADC
This bit is a copy of the AWD3 bit in the corresponding ADC_ISR register.

Bit 24 **AWD2_SLV:** Analog watchdog 2 flag of the slave ADC
This bit is a copy of the AWD2 bit in the corresponding ADC_ISR register.

Bit 23 **AWD1_SLV:** Analog watchdog 1 flag of the slave ADC
This bit is a copy of the AWD1 bit in the corresponding ADC_ISR register.

Bit 22 **JEOS_SLV:** End of injected sequence flag of the slave ADC
This bit is a copy of the JEOS bit in the corresponding ADC_ISR register.

Bit 21 **JEOC_SLV:** End of injected conversion flag of the slave ADC
This bit is a copy of the JEOC bit in the corresponding ADC_ISR register.

Bit 20 **OVR_SLV:** Overrun flag of the slave ADC
This bit is a copy of the OVR bit in the corresponding ADC_ISR register.

Bit 19 **EOS_SLV:** End of regular sequence flag of the slave ADC. This bit is a copy of the EOS bit in the corresponding ADC_ISR register.

Bit 18 **EOC_SLV:** End of regular conversion of the slave ADC
This bit is a copy of the EOC bit in the corresponding ADC_ISR register.

Bit 17 **EOSMP_SLV:** End of sampling phase flag of the slave ADC
This bit is a copy of the EOSMP2 bit in the corresponding ADC_ISR register.

Bit 16 **ADRDY_SLV:** Slave ADC ready
This bit is a copy of the ADRDY bit in the corresponding ADC_ISR register.

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

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **AWD3_MST:** Analog watchdog 3 flag of the master ADC
This bit is a copy of the AWD3 bit in the corresponding ADC_ISR register.

Bit 8 **AWD2_MST:** Analog watchdog 2 flag of the master ADC
This bit is a copy of the AWD2 bit in the corresponding ADC_ISR register.

Bit 7 **AWD1_MST:** Analog watchdog 1 flag of the master ADC
This bit is a copy of the AWD1 bit in the corresponding ADC_ISR register.

Bit 6 **JEOS_MST:** End of injected sequence flag of the master ADC
This bit is a copy of the JEOS bit in the corresponding ADC_ISR register.

Bit 5 **JEOC_MST:** End of injected conversion flag of the master ADC
This bit is a copy of the JEOC bit in the corresponding ADC_ISR register.

Bit 4 **OVR_MST:** Overrun flag of the master ADC
This bit is a copy of the OVR bit in the corresponding ADC_ISR register.

Bit 3 **EOS_MST:** End of regular sequence flag of the master ADC
This bit is a copy of the EOS bit in the corresponding ADC_ISR register.

Bit 2 **EOC_MST:** End of regular conversion of the master ADC
This bit is a copy of the EOC bit in the corresponding ADC_ISR register.

Bit 1 **EOSMP_MST:** End of Sampling phase flag of the master ADC
This bit is a copy of the EOSMP bit in the corresponding ADC_ISR register.

Bit 0 **ADRDY_MST:** Master ADC ready
This bit is a copy of the ADRDY bit in the corresponding ADC_ISR register.

32.7.2 ADC common control register (ADCC_CCR)

Address offset: 0x008

Reset value: 0x0000 0000

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

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

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **VBATEN**: VBAT enable
This bit is set and cleared by software to enable/disable the V_{BAT} channel.
0: V_{BAT} channel disabled
1: V_{BAT} channel enabled
Note: V_{BAT} is not available on all ADC instances. Refer to Section 32.4.4: ADC connectivity for details.

Bit 23 Reserved, must be kept at reset value.

Bit 22 **VREFEN**: V_{REFINT} enable
This bit is set and cleared by software to enable/disable the V_{REFINT} channel.
0: V_{REFINT} channel disabled
1: V_{REFINT} channel enabled
Note: V_{REFINT} is not available on all ADC instances. Refer to Section 32.4.4: ADC connectivity for details.
Note: The software is allowed to write this bit only when ADEN = 0.

Bits 21:16 Reserved, must be kept at reset value.

Bits 15:14 **DAMDF[1:0]:** Dual ADC mode data format
This bitfield are set and cleared by software. It specifies the data format in the common data register ADCC_CDR and ADCC_CDR2.
00: Dual ADC mode without data packing (ADCC_CDR and ADCC_CDR2 registers not used).
01: Reserved
10: Data formatting mode for any data width (ADCC_CDR data register is used when the data width is less than 16 bits, otherwise ADCC_CDR2 register is used)
11: Data formatting mode for data width lower that 8 bits (ADCC_CDR data register is used)
Note: The software is allowed to write these bits only when ADEN = 0 (ADC is disabled).

Bit 13 Reserved, must be kept at reset value.

Bit 12 Reserved, must be kept at reset value.

Bits 11:8 **DELAY[3:0]:** Delay between two sampling phases
These bits are set and cleared by software. These bits are used in dual interleaved modes. Refer to *Table 253* for the value of ADC resolution versus DELAY bits values.
Note: The software is allowed to write these bits only when the ADCs are disabled (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DUAL[4:0]:** Dual ADC mode selection
These bits are written by software to select the operating mode.
All the ADCs independent:
00000: Independent mode
The following settings apply to dual mode, master and slave ADCs working together
00001: Combined regular simultaneous + Injected simultaneous mode
00010: Combined regular simultaneous + Alternate trigger mode
00011: Combined interleaved mode + Injected simultaneous mode
00101: Injected simultaneous mode only
00110: Regular simultaneous mode only
00111: Interleaved mode only
01001: Alternate trigger mode only
Others: Reserved
All other combinations are reserved and must not be programmed
Note: The software is allowed to write these bits only when the ADCs are disabled (ADCAL = 0, JADSTART = 0, ADSTART = 0, ADSTP = 0, ADDIS = 0 and ADEN = 0).

32.7.3 ADC common regular data register for dual mode (ADCC_CDR)

Address offset: 0x00C

Reset value: 0x0000 0000

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

Bits 31:16 **RDATA_SLV[15:0]:** Regular data of the slave ADC

In dual mode, these bits contain the regular data of the slave ADC. Refer to *Section 32.4.31: Dual ADC modes*.

The data alignment is applied as described in *Data register, data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, JOVSS, JLSHIFT[a], USAT, SSAT, POSOFF)*)

Bits 15:0 **RDATA_MST[15:0]**: Regular data of the master ADC.

In dual mode, these bits contain the regular data of the master ADC. Refer to *Section 32.4.31: Dual ADC modes*.

The data alignment is applied as described in *Data register, data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, JOVSS, JLSHIFT[a], USAT, SSAT, POSOFF)*)

In MDMA=0b11 mode, bits 15:8 contains SLV_ADC_DR[7:0], bits 7:0 contains MST_ADC_DR[7:0].

32.7.4 ADC common regular data register for dual mode (ADCC_CDR2)

Address offset: 0x010

Reset value: 0x0000 0000

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

Bits 31:0 **RDATA_ALT[31:0]**: Regular data of the master/slave alternated ADCs.

In dual mode, these bits contain the regular 32-bit data of the master and the slave ADC. Refer to *Section 32.4.31: Dual ADC modes*.

The data alignment is applied as described in *Data register, data alignment and offset (ADC_DR, ADC_JDRy, OFFSETy, OFFSETy_CH, OVSS, LSHIFT, JOVSS, JLSHIFT[a], USAT, SSAT, POSOFF)*)

32.8 ADC register map

ADC1 and ADC2 are master and slave ADC, respectively.

Table 255. ADC register map and reset values for each ADC

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	ADC_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWD3	AWD2	AWD1	JEOS	JEOC	OVR	EOS	EOC	EOSMP	ADRDY
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x004	ADC_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWD3IE	AWD2IE	AWD1IE	JEOSIE	JEOCIE	OVRIE	EOSIE	EOCIE	EOSMPIE	ADRDYIE
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x08	ADC_CR	ADCAL	ADCALDIF	DEEPPWD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JADSTP	ADSTP	JADSTART	ADSTART	ADDIS	ADEN
	Reset value	0	0	1																								0	0	0	0	0	0
0x00C	ADC_CFGR1	Res.	AWD1CH[4:0]					JAUTO	JAWD1EN	AWD1EN	AWD1SGL	Res.	JDISCEN	DISCNUM[2:0]			DISCEN	Res.	AUTDLY	CONT	OVRMOD	EXTEN[1:0]		EXTSEL[4:0]					Res.	RES[1:0]		DMNGT[1:0]	
	Reset value	0x8000 0000																															
0x010	ADC_CFGR2	LSHIFT[3:0]				Res.	Res.	OVSR[9:0]										SMPTRIG	SWTRIG	BULB	Res.	Res.	ROMVSM	TROVS	OVSS[3:0]				Res.	Res.	Res.	Res.	ROVSE
	Reset value	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0					0
0x014	ADC_SMPR1	Res.	Res.	SMP9[2:0]			SMP8[2:0]			SMP7[2:0]			SMP6[2:0]			SMP5[2:0]			SMP4[2:0]			SMP3[2:0]			SMP2[2:0]			SMP1[2:0]			SMP0[2:0]	
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	ADC_SMPR2	Res.	Res.	SMP19[2:0]			SMP18[2:0]			SMP17[2:0]			SMP16[2:0]			SMP15[2:0]			SMP14[2:0]			SMP13[2:0]			SMP12[2:0]			SMP11[2:0]			SMP10[2:0]	
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	ADC_PCSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCSEL[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020-0x02C	Reserved																																
0x030	ADC_SQR1	Res.	Res.	Res.	SQ4[4:0]					Res.	SQ3[4:0]					Res.	SQ2[4:0]					Res.	SQ1[4:0]					Res.	Res.	LEN[3:0]			
	Reset value				0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0			0	0	0	0
0x034	ADC_SQR2	Res.	Res.	Res.	SQ9[4:0]					Res.	SQ8[4:0]					Res.	SQ7[4:0]					Res.	SQ6[4:0]					Res.	Res.	SQ5[4:0]			
	Reset value				0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0			0	0	0	0
0x038	ADC_SQR3	Res.	Res.	Res.	SQ14[4:0]					Res.	SQ13[4:0]					Res.	SQ12[4:0]					Res.	SQ11[4:0]					Res.	Res.	SQ10[4:0]			
	Reset value				0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0			0	0	0	0
0x03C	ADC_SQR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SQ16[4:0]					Res.	SQ15[4:0]			
	Reset value																							0	0	0	0	0		0	0	0	0
0x040	ADC_DR	RDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 255. ADC register map and reset values for each ADC (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x044-0x048	Reserved																																
0x04C	ADC_JSQR	JSQ4[4:0]					Res.	JSQ3[4:0]					Res.	JSQ2[4:0]					Res.	JSQ1[4:0]					JEXTEN[1:0]		JEXTSEL[4:0]					JLEN[1:0]	
	Reset value	0	0	0	0	0		0	0	0	0	0		0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	ADC_OFCFGR1	OFFSET1_CH[4:0]					SSAT	USAT	POSOF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0																								
0x054	ADC_OFCFGR2	OFFSET2_CH[4:0]					SSAT	USAT	POSOF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0																								
0x058	ADC_OFCFGR3	OFFSET3_CH[4:0]					SSAT	USAT	POSOF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0																								
0x05C	ADC_OFCFGR4	OFFSET4_CH[4:0]					SSAT	USAT	POSOF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0																								
0x060	ADC_OFR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[21:0]																					
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x064	ADC_OFR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[21:0]																					
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x068	ADC_OFR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[21:0]																					
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x06C	ADC_OFR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[21:0]																					
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x070	ADC_GCOMP	GCOMP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GCOMPCOEFF[13:0]													
	Reset value	0																		0	1	0	0	0	0	0	0	0	0	0	0	0	0
0x074-0x07C	Reserved																																
0x080	ADC_JDR1	JDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084	ADC_JDR2	JDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 255. ADC register map and reset values for each ADC (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x088	ADC_JDR3	JDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08C	ADC_JDR4	JDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x090-0x09C	Reserved																																
0x0A0	ADC_AWD2CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDCH[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A4	ADC_AWD3CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AWDCH[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A8	ADC_AWD1LTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTR[22:0]																						
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0AC	ADC_AWD1HTR	AWDFILT [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	HTR[22:0]																						
	Reset value	0	0	0							0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0B0	ADC_AWD2LTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTR[22:0]																						
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B4	ADC_AWD2HTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HTR[22:0]																						
	Reset value										0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0B8	ADC_AWD3LTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LTR[22:0]																						
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0BC	ADC_AWD3HTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HTR[22:0]																						
	Reset value										0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0C0	ADC_DIFSEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIFSEL[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C4	ADC_CALFACT	CALADDOS	Res.	Res.	Res.	Res.	Res.	Res.	CALFACT_D[8:0]									Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALFACT_S[8:0]								
	Reset value	0							0	0	0	0	0	0	0	0	0								0	0	0	0	0	0	0	0	0
0x0C8-0x0CC	Reserved																																
0x0D0	ADC_OR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OP2	OP1	OP0
	Reset value																														0	0	0
0x0D4-0x0FC	Reserved																																

Table 256. ADC register map and reset values (master and slave ADC common registers)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	ADCC_CSR	Res.	Res.	Res.	Res.	Res.	Res.	AWD3_SLV	AWD2_SLV	AWD1_SLV	JEOS_SLV	JEOC_SLV	OVR_SLV	EOS_SLV	EOC_SLV	EOSMP_SLV	ADRDY_SLV	Res.	Res.	Res.	Res.	Res.	Res.	AWD3_MST	AWD2_MST	AWD1_MST	JEOS_MST	JEOC_MST	OVR_MST	EOS_MST	EOC_MST	EOSMP_MST	ADRDY_MST
		slave ADC																						master ADC									
	Reset value							0	0	0	0	0	0	0	0	0	0							0	0	0	0	0	0	0	0	0	0
0x004	Reserved	Res.																															
0x008	ADCC_CCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBATEN	Res.	VREFEN	Res.				Res.	Res.	DAMDF[1:0]		Res.	Res.	DELAY[3:0]				Res.	Res.	Res.	DUAL[4:0]				
	Reset value								0		0							0	0			0	0	0	0				0	0	0	0	0
0x00C	ADCC_CDR	RDATA_SLV[15:0]																RDATA_MST[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	ADCC_CDR2	RDATA_ALT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	Reserved																																

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

33 Digital temperature sensor (DTS)

33.1 Introduction

The DTS is a high-precision low-power junction temperature sensor (TS). It is composed of a configurable controller plus two embedded temperature sensors, TS0 and TS1. The controller features a generic interface that enables the DTS to be accessed in read and write modes, through the APB bus.

33.2 DTS main features

The temperature sensor main features are the following:

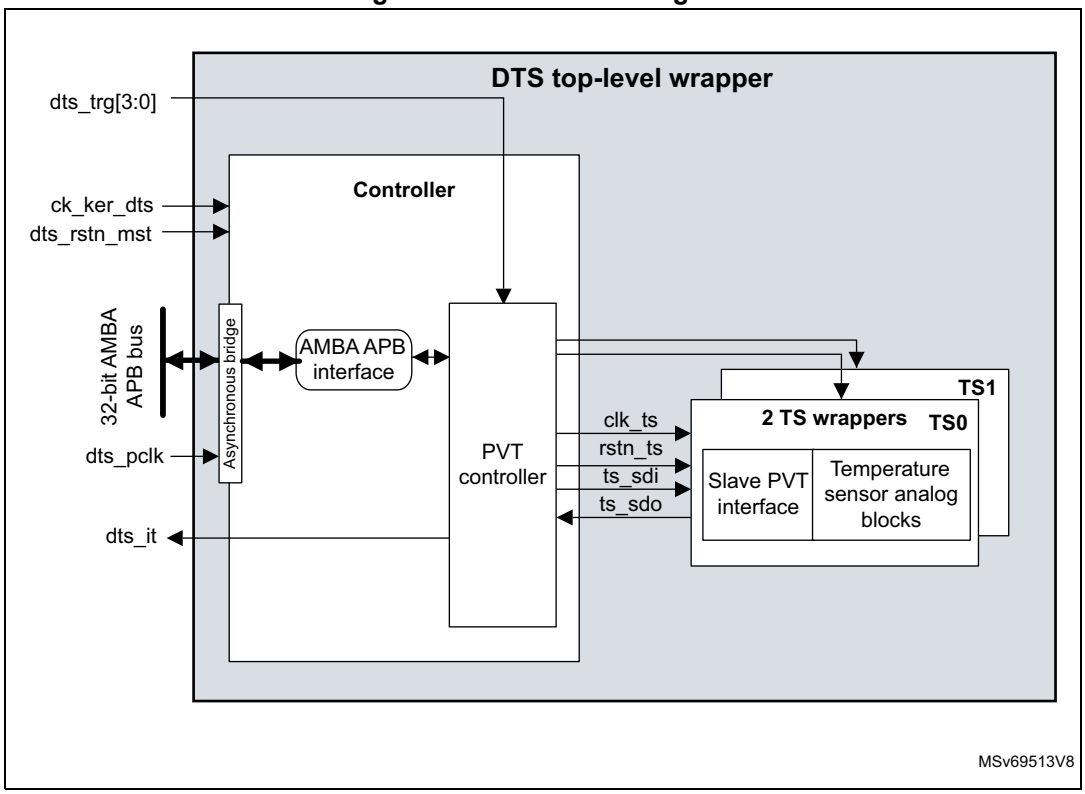
- For each sensor:
  - Two programmable (rise or fall) hardware alarms incorporating hysteresis
  - Status registers recording the minimum and maximum data values received
- A power-up timer with IRQ to support manual operation

33.3 DTS functional description

33.3.1 DTS block diagram

Figure 315 shows the DTS block diagram.

Figure 315. DTS block diagram



The DTS incorporates an APB interface configured to control two temperature sensors, TS0 and TS1, through a process, voltage and temperature (PVT) controller.

33.3.2 DTS pins and internal signals

Table 257. DTS internal input/output signals

Internal signal name	Signal type	Description
dts_pclk	Input	APB bus clock
ck_ker_dts	Input	DTS main clock
dts_rstn_mst	Input	Master reset (asynchronous, active low)
AMBA APB bus	Input/output	APB Interface
dts_it	Output	IRQ interrupt from the DTS peripheral
dts_trg[3:0]	Inputs	Trigger interface
Temperature sensor (TS) interface		
clk_ts	Input	Temperature sensor SDA clock

Table 257. DTS internal input/output signals (continued)

Internal signal name	Signal type	Description
rstn_ts	Input	Temperature sensor SDA reset (asynchronous, active low)
ts_sdi	Input	Temperature sensor SDA slave data Input
ts_sdo	Output	Temperature sensor SDA slave data output

Table 258. DTS interconnection

Internal signal name	Source/Destination
dts_trg[0]	lptim4_out
dts_trg[1]	lptim2_ch1
dts_trg[2]	lptim3_ch1
dts_trg[3]	exti13

33.3.3 DTS reset and clocks

DTS clocks

The DTS is clocked by ck_ker_dts.

The PVT controller incorporates a clock synthesizer in the TS block. The synthesizers generate the clock and timing strobes for the slave serial data adapters (SDA) and the clock for the embedded TS blocks (clk_ts).

The synthesizer CLK_SYNTH_HI[7:0]/CLK_SYNTH_LO[7:0] counters allow the integer division of the ck_ker_dts clock to generate the TS clock (clk_ts) at frequency ratios from 1/2 to 1/512 (see *DTS TSC clock synthesizer register (DTS_TSCCLKSYNTHR)*). Separate counters are used to generate timing control strobes that are aligned with the clk_ts clock. These timing strobes are used by the master serial adapters to control the sample point of the serial data inputs and the hold time of the serial data outputs.

The system clock ck_ker_dts frequency is determined by the RCC. CLK_SYNTH_HI[7:0] and CLK_SYNTH_LO[7:0] must be programmed so that the ratio CLK_SYNTH_HI[7:0]/CLK_SYNTH_LO[7:0] enables to obtain a clk_ts serial clock that falls within the TS operating frequency range (from 4 to 8 MHz).

The TS frequency is defined by:

$$\text{clk_ts frequency} = 1 / ((\text{CLK_SYNTH_HI[7:0]} + 1) + (\text{CLK_SYNTH_HI[7:0]} + 1)) \times \text{ck_ker_dts frequency}$$

For example:

For ck_ker_dts = 64.0 MHz and a TS operating frequency ranging between 4 and 8 MHz, setting the ck_ker_dts to clk_ts divider factor to 8:1 (CLK_SYNTH_HI[7:0] and CLK_SYNTH_LO[7:0] both programmed to 0x03), returns an TS operating frequency of 64.0 MHz / 8 = 8 MHz, which falls within the acceptable operating frequency range of 4 to 8 MHz. This enables to obtain an overall divider factor of 8 and a ck_ker_dts serial clock duty cycle of 50/50.

For ck_ker_dts = 8 MHz, the configuration must be:

CLK_SYNTH_LO[7:0] = 0x0

CLK_SYNTH_HI[7:0] = 0x0

CLK_SYNTH_HOLD[3:0]= 0x1

clk_ts = 4 MHz

clk_ts duty cycle = 50 %

The CLK_SYNTH_HOLD[3:0] value must be programmed to accommodate the worst-case serial data path propagation delays without exceeding the synthesized clock period, this means that:

$$\text{CLK_SYNTH_HOLD[3:0]} \leq \text{CLK_SYNTH_HI[7:0]} + \text{CLK_SYNTH_LO[7:0]} + 1$$

DTS reset

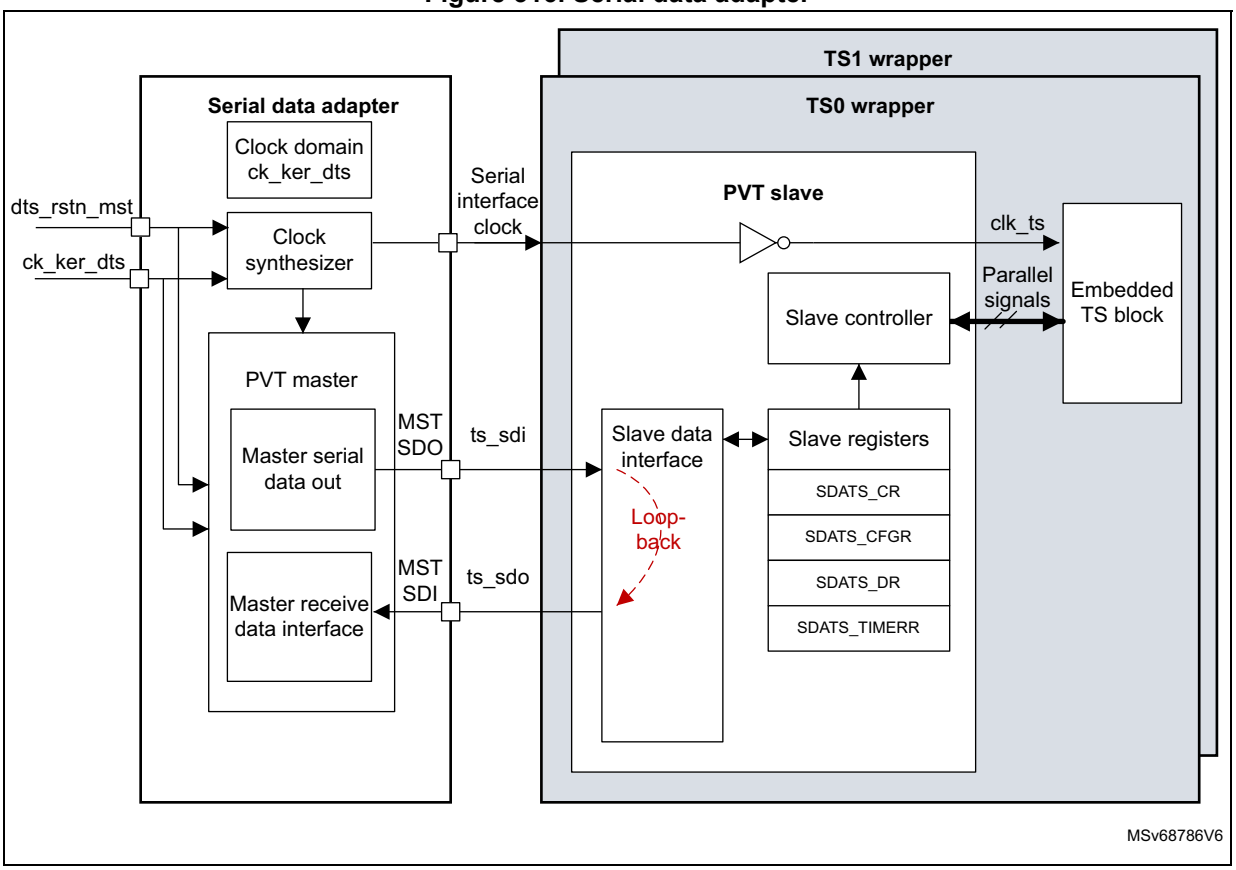
The DTS is reset using the asynchronous active-low dts_rstn_mst input.

When the master dts_rstn_mst is asserted, the entire DTS is reset, all APB internal registers and the PVT controller. It also powers down and resets the embedded TS blocks.

When de-asserted, the APB interface responds to requests after a three clock-cycle delay.

33.3.4 DTS serial data adapter (SDA)

Figure 316. Serial data adapter



The serial data adapter (SDA) allows the TS blocks to be configured and controlled via serial data interfaces (SDIF). The purpose of the SDA is to minimize the number of signals that need to be routed from the PVT controller to each TS block.

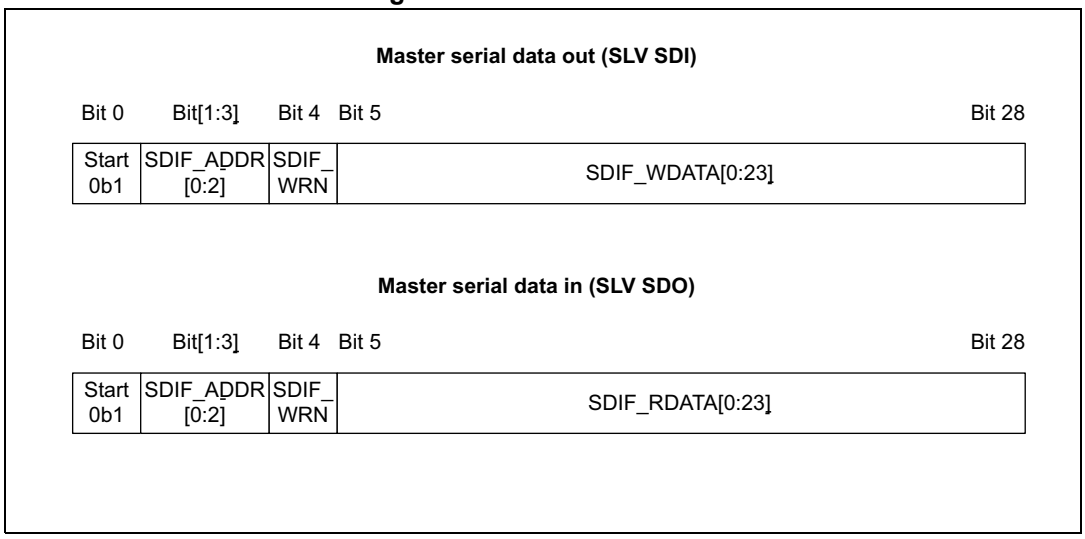
A master usually transmits data to all its active PVT slave devices. However, it is possible to inhibit the programming of one or more slaves to allow different configuration options to be applied. To achieve power saving, it is also possible to disable completely a TS block by disabling its clock and forcing reset.

The serial data interface also provides loop-back functionality (shown in red in *Figure 316*) to support system integration testing without having to power up the embedded TS block.

SDA data format

The serial data streams consist of an active-high start bit, 3-bit register address (SDIF_ADDR[2:0]), register write/not read enable (SDIF_WRN), and 24-bit data field (SDIF_WDATA[23:0]). The input and output serial data stream formats are shown in *Figure 317*. It must be noted that the serial data streams are little endian, meaning that the individual data fields are shifted LSB first.

Figure 317. SDA data format



SDA read

When reading data from the SDA slave registers (SDIF_WRN = 0), the SLV SDI bit stream is shifted to SLV SDO (delayed by one clk_ts cycle) and the register read data value is output on SDIF_RDATA[23:0]. The read data value is then made available in the PVT controller DTS_TSnSDIFRDATAR register (where *n* corresponds to the TS instance number).

SDA write

When writing data to the SDA slave registers (SDIF_WRN = 1), the SLV SDI bit stream is automatically shifted to SLV SDO and SDIF_RDATA[23:0] = SDIF_WDATA[23:0]. The write data loop-back value is then made available in the corresponding PVT controller DTS_TSnSDIFRDATAR register.

A special loop-back mode is available by writing to the (read-only) SDATS_DR register when SDIF_WRN = 1 and SDIF_ADDR[2:0] = 0x3. In this mode, the loop-back data value is also written to the PVT controller TS data registers (DTS_TSnSDIFRDATAR). When writing loop-back data to the TS data registers, the SDA TS_DATA[23:0] field **must** be set to 0x0.

The DTS_TSnSDIFRDATAR loop-back functionality can be used to verify the integrity of the serial data transfer between the PVT controller and remote PVT slave devices.

The special DTS_TSnSDIFRDATAR loop-back functionality is provided to test the PVT controller operation without having to power up the embedded TS block, for example when a fault or alarm occurs.

The loop-back functionality is disabled whenever the PVT slave is configured to automatic data recovery when the TS_AUTO bit of PVT SDATS_CR register is asserted.

SDA automatic data recovery

The automatic data recovery is enabled through TS_AUTO bit of SDATS_CR register.

In automatic data recovery mode, the SDIF_LOCK flag is set in the DTS_TSCSDIF_SR register, the SDA interface becomes write-only and all SDA read requests are ignored. In this mode, the SLV automatically streams TS_DATA[23:0] to the PVT controller each time new data samples become available, while setting SDIF_ADDR[2:0] to 0x3 and SDIF_WRN to 1. The automatically recovered data are made available in the PVT controller DTS_TSnSDIFRDATAR and DTS_TSnSDIFRDATAR registers.

SDA SDIF programming

The SDA slave is accessed via the TS SDIF control register DTS_TSCSDIF_CFGR, by setting the SDIF_WDATA[23:0], the SDIF_ADDR[2:0] and the SDIF_WRN, and asserting the SDIF_PROG bit. This SDA read or write request is normally set up using a single PVT register write access.

Once a programming request has been issued, the SDIF_BUSY flag in the DTS_TSCSDIF_SR register is set, and any additional read or write programming requests is ignored until the current access completes.

SDA controller operation

The SDA slave controller features two operating modes: manual (default) and automatic. In both modes, the embedded TS block is configured through the SDATS_CFGR register.

In manual operating mode, when the TS_AUTO bit of SDATS_CR register is de-asserted, the embedded TS block is controlled directly through the SDATS_CR register fields TS_PD, TS_RSTN, TS_RUN_ONCE, TS_RUN_CONT and TS_CLOAD. In manual mode, the user is responsible for ensuring that the PD, RSTN, CLOAD and RUN signals are applied in the correct order and at appropriate times. Sample data from the TS block are captured in the SDA slave SDATS_DR register at the end of the conversion. The user must either poll the SDATS_DR register or leave sufficient time for the conversion to complete before reading the SDATS_DR register in order to recover the sample data from the slave SDA.

In automatic mode, when the TS_AUTO bit is asserted in SDATS_CR register, the embedded TS block is controlled by the slave SDA. When either TS_RUN_ONCE or TS_RUN_CONT bit of the TS_SDATS_CR register is asserted, the slave SDA sequences the PD, RSTN, CLOAD, and RUN signals that drive the TS block to perform one or more conversions. In this mode sample data from the TS block are captured in the SDA slave SDATS_DR register at the end of the conversion before being automatically streamed to the PVT controller.

In trigger mode, when a trigger is asserted at DTS top level, this value is compared to the value programmed in bits 18 to 16 (TRIG[2:0]) of the SDATS_CFGR register. If it is equal, the TS_RUN_ONCE bit is automatically asserted. Sample data from each TS block are captured in the SDA slave SDATS_DR register at the end of the conversion.

SDA slave registers

The serial data adapter registers are accessible through the SDIF interface of the PVT controller.

Table 259. SDA slave registers

SDIF_ADDR[2:0]	Reset value	Register	Description
0x0	0x000001	SDATS_CR	TS control register (see *Table 260*)
0x1	0x000000	SDATS_CFGR	TS configuration register (see *Table 261*)
0x3	0x000000	SDATS_DR	TS data / SDA loop-back register (see *Table 262*)
0x4	-	Reserved	-
0x5	0x000000	SDATS_TIMERR	TS power-up Timer register (see *Table 260*)

Table 260. SDA TS control register (SDATS_CR)

Bitfield range	Bitfield description	Access type
Bits 23:10	Reserved, must be kept at reset value.	-
Bit 9	**TS_STOP:** TS force stop This self-clearing bit immediately returns the slave TS controller to the default TS control state and terminates any serial data activity. This is not a clean shut-down and it may invalidate any data operations currently in progress. 0: TS not stopped 1: TS stopped	rs
Bit 8	**TS_AUTO:** TS automatic mode This bit is used in conjunction with TS_RUN_CONT and TS_RUN_ONCE to automatically power up the embedded TS and run conversions. Asserting TS_RUN_CONT triggers continuous conversions, while asserting TS_RUN_ONCE triggers a single conversion. If TS_AUTO is de-asserted while conversions are running, the controller waits for any active conversions to complete and then returns to the default TS control state. 0: Automatic mode disabled 1: Automatic mode enabled	rw
Bits 7:5	Reserved, must be kept at reset value.	-
Bit 4	**TS_CLOAD:** TS configuration load This self-clearing bit provides the embedded TS cload input in the default TS control state (TS automatic mode disabled). 0: TS configuration not loaded 1: TS configuration loaded	rs
Bit 3	**TS_RUN_CONT:** TS run continuous The TS_RUN_CONT control bit is ORed with TS_RUN_ONCE in order to provide the embedded TS run input in the default TS control state (TS automatic mode disabled). This bit is active high.	rw

Table 260. SDA TS control register (SDATS_CR) (continued)

Bitfield range	Bitfield description	Access type
Bit 2	**TS_RUN_ONCE:** TS run once Provided TS_RUN_CONT is not asserted, this self-clearing bit asserts the embedded TS run input for one clock cycle when it is in the default TS control state (TS automatic mode disabled). This bit is active on the rising edge.	rs
Bit 1	**TS_RSTN:** TS reset This bit provides the embedded TS rstn input in the default TS control state (TS automatic mode must be disabled). 0: TS reset 1: TS not reset	rw
Bit 0	**TS_PD:** TS power-down This bit provides the embedded TS pd input when in the default TS control state (TS automatic mode disabled). 0: No TS power-down 1: TS power-down	rw

Table 261. SDA TS configuration register (SDATS_CFGR)

Bitfield range	Bitfield description	Access type
Bits 23:20	Reserved, must be kept at reset value.	-
Bit 19	**CLKDIS:** TS clock gating disable bit 0: Clock enabled 1: Clock disabled	rw
Bits 18:16	**TRIG[2:0]:** Trigger selection bits 000: no trigger source selected 001: dts_trg[0] selected 010: dts_trg[1] selected 011: dts_trg[2] selected 100: dts_trg[3] selected Others: Reserved, must not be used Refer to *Section 33.3.7: Temperature sensor configuration* for details.	rw
Bits 15:8	**TS1_CFG[7:0]:** TS1 configuration TS1_CFG[7:5]: TS1 resolution 000:12 bits 001:10 bits 010: 8 bits Other: reserved, must not be used TS1_CFGR[4]: TS1 output mode 0: parallel interface 1: serial interface (default) TS1_CFGR[3:0]: TS1 communication enable bit 0000: no communication 0001: communication enabled. Others: Reserved, must not be used.	rw

Table 261. SDA TS configuration register (SDATS_CFGR)

Bitfield range	Bitfield description	Access type
Bits 7:0	**TS0_CFG[7:0]:** TS0 configuration TS0_CFG[7:5]: TS0 resolution 000:12 bits 001:10 bits 010: 8 bits Other: reserved, must not be used TS0_CFGR[4]: TS0 output mode 0: parallel interface 1: serial interface (default) TS0_CFGR[3:0]: TS0 communication enable bit 0000: no communication 0001: communication enabled. Others: Reserved, must not be used.	rw

Table 262. SDA TS data register (SDATS_DR)

Bitfield range	Bitfield description	Access type
Bits 23:19	Reserved, must be kept at reset value.	-
Bit 18	**TS_DONE:** TS sample done flag This bit provides a way to identify when a new data sample has been recovered while manually polling the SDATS_DR register. In manually polling, the TS_DONE flag is cleared when the SDATS_DR register is read. The sample done flag is asserted when new sample data becomes available. It is automatically cleared in TS automatic mode as the SDATS_DR value is transmitted to the PVT controller. 0: No new data sample 1: New data sample	r
Bit 17	**TS_FAULT:** TS sample fault When asserted, this bit indicates that the embedded TS has signaled a fault. The TS_FAULT flag is updated when new data are received. 0: No TS sample fault 1: TS sample fault	r
Bit 16	**TS_TYPE:** TS sample type The TS_TYPE flag is updated when new data is received. 0: Indicates valid data 1 Indicates either analog access, signature select or fault debug mode	r
Bits 15:0	**TS_DATA[15:0]:** TS sample data	r

Table 263. SDA TS timer register (SDATS_TIMERR)

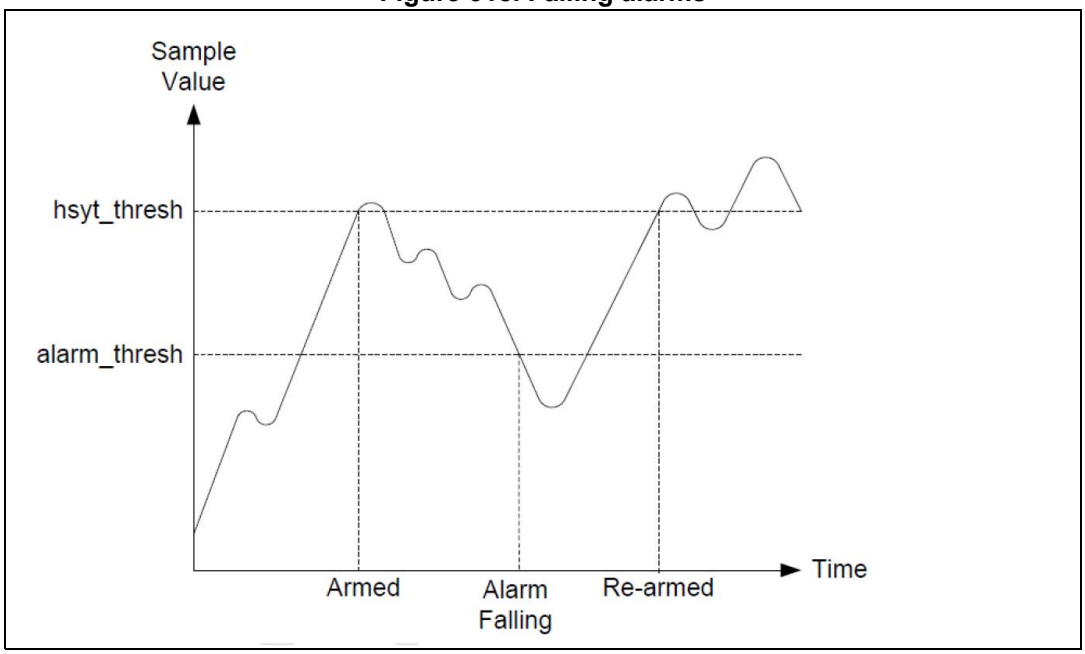
Bitfield range	Bitfield description	Access type
Bits 23:16	Reserved, must be kept at reset value.	-
Bits 15:0	**TS_PU_DLY[15:0]:** TS power-up delay The value is expressed in units of clk_ts cycles. Typical power-up delays are 256 cycles.	rw

33.3.5 Alarms

The DTS provides two alarms, alarm A and alarm B, for each TS block. The alarms feature programmable hysteresis and can be configured to detect either rising or falling sample values. They are disabled by setting the hysteresis threshold to a value that is equal to the alarm threshold (default condition). Once enabled, the alarms generate an interrupt if the value of the recovered data sample equals or exceeds the rising or falling alarm threshold. Once an alarm has been triggered, the hysteresis threshold provides a recovery window during which the alarm cannot trigger again.

Falling alarms

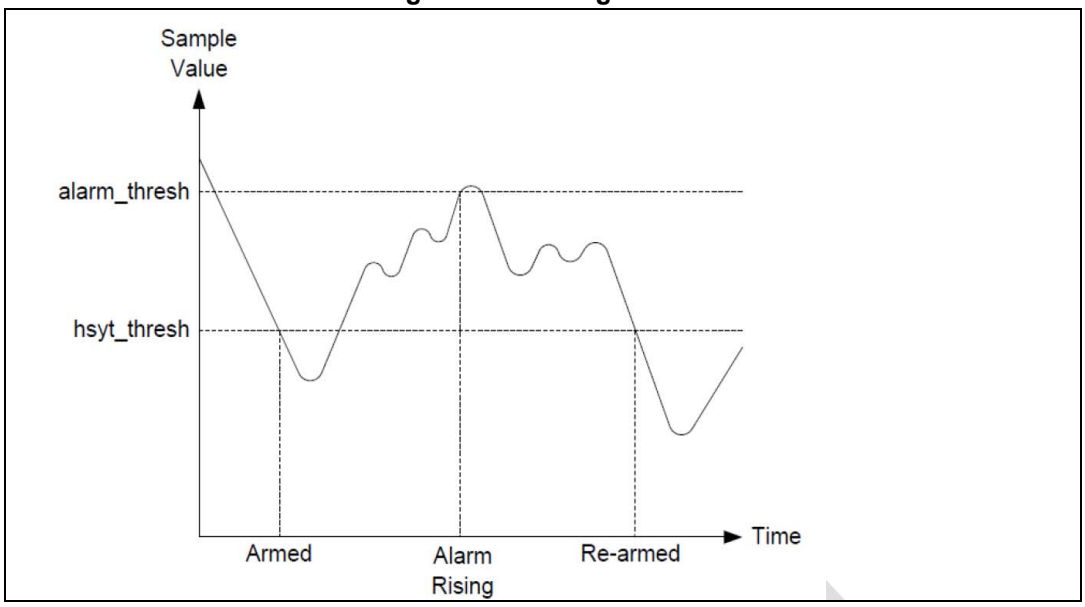
Figure 318. Falling alarms



Setting the hysteresis threshold greater than the alarm threshold creates a falling alarm. When configured as a falling alarm, the alarm is armed when the recovered sample value is greater than or equal to the hysteresis threshold. Once armed, a falling alarm triggers when the recovered sample value is less than or equal to the alarm threshold. An alarm is automatically disarmed once triggered and does not rearm until the recovered sample value is again greater than or equal to the hysteresis threshold. The alarm is also disarmed if an invalid sample value is received, that is if unexpected fault or type flags are detected.

Rising alarms

Figure 319. Rising alarms



Setting the hysteresis threshold less than the alarm threshold creates a rising alarm. When configured as a rising alarm, the alarm is armed when:

1. The associated clock synthesizer is first enabled (initial conversion).
2. The recovered sample value is less than or equal to the hysteresis threshold.
3. An invalid sample value is received, that is when an unexpected fault or type flags are detected.

Once armed, a rising alarm triggers when the recovered sample value is greater than or equal to the alarm threshold. The alarm is automatically disarmed once triggered and does not rearm until the recovered sample value is again less than or equal to the hysteresis threshold.

33.3.6 Main APB programming routines

This section gives several examples of routines to configure the PVT controller, program the SDA registers, and perform single or continuous temperature conversions.

Configure the PVT controller

1. STEP 1: Configure the PVT controller Interrupts:
   a) Write DTS_PVT_IER register:

      The timer is only intended to be used when using the TS block in bypass mode, so this interrupt must remain disabled .

   b) Write DTS_TSn_IER register:

      In the unlikely event of the TS block signaling a fault, the recovered data values are considered invalid and any active alarm is disarmed until the fault is removed. The fault IRQ is generated the first time a TS that an TS block signals a fault. If the fault is not cleared by the user, then no further interrupts are raised. It is recommended to enable always this interrupt.

2. STEP 2: Configure the PVT controller alarms:
   a) Write DTS_TSnALARMA_CFGR register:

      Alarms are enabled when the ALARM_THRESH is not equal to the HYST_THRESH.

      When enabling alarms, make sure that the corresponding IRQ is also enabled.
   b) Write DTS_TSnALARMB_CFGR register:

      Alarms are enabled when the ALARM_THRESH is not equal to the HYST_THRESH.

      When enabling alarms, make sure the corresponding IRQ is also enabled.
3. STEP 3: PVT sample control:
   a) Write DTS_TSCSMPL_CR register:

      When SMPL_DISCARD is set, TS data samples are temporarily discarded, but the sample counter continues to increment.

      When SMPL_CTR_HOLD is asserted, the sample counter is prevented from rolling over when the maximum sample count is reached.

      When SMPL_CTR_DISABLE is asserted, the sample counter is disabled.
4. STEP 4: PVT sample control:
   a) Write DTS_TSnHILORESETR for each TS instance.

Configure the SDA registers

1. STEP 1: Program the SDA registers:
   a) Read DTS_TSCSDIF_SR register:

      The SDIF_LOCK flag must be low, indicating that SDIF automatic mode has not been enabled.

      Before proceeding further, make sure that the serial data interface is inactive by checking that the SDIF_BUSY flag is low.
   b) Write DTS_TSCSDIF_CR register to fetch the remote slave SDA register value:

      SDIF_PROG = 0x1

      SDIF_WRN = 0x1

      SDIF_ADDR[2:0]: this bitfield selects the desired SDA register.

      SDIF_WDATA[23:0]: this bitfield provides the SDA write data value.

      Program the SDA register bits in the following order:

      SDATS_TIMERR register

      SDATS_CFGR register

      SDATS_CR register (see notes below).

Note: Setting the TS_AUTO bit and either TS_RUN_ONCE or TS_RUN_CONT bit in the SDATS_CR register initiates a conversion and sets the SDIF_LOCK flag. Once the SDIF_LOCK flag is set, the SDA registers become write-only.

The SDIF_WDATA[23:0] value is also reflected in the DTS_TSnSDIFRDATAR *registers.*

2. STEP 2: Read the SDA register values:
   a) Read DTS_TSCSDIF_SR register:
      The SDIF_LOCK flag must be low, indicating SDIF automatic mode has not been enabled.
      Before proceeding further, make sure the serial data interface is inactive by checking that the SDIF_BUSY flag is low.
   b) Write DTS_TSCSDIF_CR register to fetch the remote slave SDA register value:
      SDIF_PROG = 0x1
      SDIF_WRN = 0x0
      SDIF_ADDR[2:0]: this bitfield selects the desired SDA register.
      SDIF_WDATA[23:0]: this bitfield is not used and can be set to any value.

Note: Once the SDIF_LOCK flag is set, the SDA registers become write-only and any read requests is ignored.

The read data value (SDIF_RDATA[23:0]) is also reflected in the DTS_TSnSDIFRDATAR *register.*

   c) Read the SDA register value from the DTS_TSnSDIFRDATAR register*s*.

Run once

1. STEP 1:
   a) Read DTS_TSCSDIF_SR register:
      The SDIF_LOCK flag is low indicating that the SDIF automatic sample mode is not enabled.
      Verify that the serial data interface is not in use by checking that the SDIF_BUSY flag is low.
   b) Write DTS_TSCSDIF_CR register:
      SDIF_PROG = 0x1
      SDIF_WRN = 0x1
      SDIF_ADDR[2:0] = 0x0 to select the SDATS_CR register
      SDIF_WDATA[23:0] = 0x000104 to assert TS_AUTO and the self-clearing TS_RUN_ONCE bits
2. STEP 2: Wait for a sample done IRQ
   a) Read the DTS_TSnSDIFDONER register:
      Verify that one or more SDATS_DR registers have been updated by checking that the SDIF_SMPL_DONE flags are set to 0x1.
   b) Read the DTS_TSCSMPLCNTR register:
      The sample counter value contained in DTS_TSCSMPLCNTR increments whenever TS data (TS_DATA[23:0]) are updated.
   c) Read the DTS_TSnSDIFRDATAR register:
      Read the TS sample data, type and fault values.
      If the fault bit is set, the sample data value is unreliable, and must be discarded.
      The SDIF_SMPL_DONE flag in the DTS_TSnSDIFDONER register is automatically cleared when the SDATS_DR register is read.

Continuous run

1. STEP 1
   a) Read DTS_TSCSDIF_SR register:
      The SDIF_LOCK flag is low to indicate that the SDIF automatic sample mode is not enabled.
      Verify that the serial data interface is not in use by checking that the SDIF_BUSY flag is low.
   b) Write DTS_TSCSDIF_CR register:
      SDIF_PROG = 0x1
      SDIF_WRN = 0x1
      SDIF_ADDR[2:0] = 0x0
      SDIF_WDATA[23:0] = 0x000108 (value to assert TS_AUTO and the run continuous TS_RUN_CONT bits)
2. STEP 2: Wait for a sample done IRQ
   a) Read the DTS_TSnSDIFDONER register:
      Verify that the SDATS_DR registers have been updated by checking that the sample done SDIF_SMPL_DONE flags are set to 0x1.
   b) Read DTS_TSCSMPLCNTR register:
      The sample counter increments whenever the DTS_TSnSDIFRDATAR value is updated.
   c) Read DTS_TSnSDIFRDATAR register:
      Read the TS sample data, type and fault values.
      If the fault bit is set, the sample data value is unreliable, and must be discarded.
      The SDIF_SMPL_DONE flag in the DTS_TSnSDIFDONER register is automatically cleared when the SDATS_DR register is read.

33.3.7 Temperature sensor configuration

The temperature sensor can be configured through the internal register *SDA TS configuration register (SDATS_CFGR)*.

Selecting the conversion mode

The sensor operates only in a single mode to read the temperature value. This mode is selected by setting SDATS_CFGR[3:0] bits to 0b0001.

The temperature value is obtained using the formula provided in *Calculating the temperature*. It does not require any calibration.

Configuring the output options

The conversion time can be configures through SDATS_CFGR[7:5] bits. It can be shortened to reduce the counter output value with the drawback of degrading the temperature measurement accuracy (see *Table 264*).

The output data are available in the TS_DATA[15:0] bitfield of the SDATS_DR register.

Data can be output in parallel or serial mode depending on the setting of SDATS_CFGR[4] bit:

- 0: parallel output

  All the data bits are sent simultaneously through TS_DATA[11:0].
- 1: serial output (default mode)

  The data are sent serially on bit 11 of the TS_DATA bitfield.

  Serial data are always 12-bit wide whatever the output options. Using a lower resolution (such as 10 bits) results in similar output values with trailing zeroes.

Table 264. Output resolution configuration

Resolution	SDATS_CFGR[7:5] bits	Accuracy degradation	Comments
12 bits	0b000	-	Temperature value available on TS_DATA[11:0]
10 bits	0b001	±0.4 °C	Temperature value available on TS_DATA[11:2]
8 bits	0b010	±1.5 °C	Temperature value available on TS_DATA[11:4]
Others not allowed	0b011	-	Not allowed
	0b1XX	-	Not allowed

Calculating the temperature

The digital output is converted to an uncalibrated temperature value by using the following formula:

$$\text{Temp (°C)} = G + H \times \text{Eqbs} + J \times F_{clk_ts}$$

Where:

Temp = Temperature in degrees Celsius

Eqbs = Nbs/Cal5 – 0.5, Nbs being the digital output, TS_DATA[11:0], latched after conversion

F_{clk_ts} = Temperature sensor SDA clock frequency (in MHz)

For G, H, J and Cal5 values, please refer to device datasheet"

Configuring the triggers

Triggers can be configured through bits 18 to 16 (TRIG[2:0]) of the SDATS_CFGR register:

TRIG[2:0] = 0b001: dts_trg[0] is selected.

TRIG[2:0] = 0b010: dts_trg[1] is selected.

TRIG[2:0] = 0b011: dts_trg[2] is selected.

TRIG[2:0] = 0b100: dts_trg[3] is selected.

Disabling the clock

The TS clock can be disabled by setting the CLKDIS of the SDATS_CFGR, when the TS is not used and it does not operate in automatic mode.

33.4 DTS interrupts

The DTS provides a single active high (level) hardware interrupt output from the controller. This is the logical OR of all the embedded TS interrupt sources and the common delay timer.

The enabling, masking, and status of the interrupts are readily available through the IRQ control registers. The interrupts from each TS instance are coalesced into a single interrupt, the status of which can be observed through the DTS_TSn_ISR register. The index number of a set bit indicates the source TS number with an interrupt condition.

Each TS block can generate an interrupt based on the following sources:

- The irq_smpl indicates that sample data have been updated for the corresponding TS instance.
- The irq_alarma indicates that the recovered sample value has exceeded the programmed alarm A threshold for the corresponding TS instance.
- The irq_alarmb indicates that the recovered sample value has exceeded the programmed alarm B threshold for the corresponding TS instance.
- The irq_fault indicates that the corresponding TS instance has signaled a fault.

When an interrupt event occurs, the rising edge is detected and the corresponding flag is set in the interrupt source register, DTS_TSn_ISR. The IRQ status bits are individually gated (logical AND) with their corresponding interrupt enabled in the DTS_TSn_IER register to generate interrupt request bits. These IRQ bits are then ORed together and registered by ck_ker_dts to generate the interrupt outputs for IRQ control registers.

Once set the interrupt source bits are sticky, they must be cleared by writing a 1 to the corresponding bit of the DTS_TSn_ICR register. This also clears the associated IRQ if it has been enabled.

Interrupt events are triggered when the following flags are asserted or the force bits in the DTS_TSnIRQTESTR register are set. The forcing allows interrupts to be manually generated for software test purposes.

- irq_smpl: asserted whenever a new TS data sample is recovered.
- irq_alarma: asserted whenever the alarm A thresholds have been reached.
- irq_alarmb: asserted whenever the alarm B thresholds have been reached.
- irq_fault: asserted whenever the TS block asserts its fault output.

Generally, the embedded temperature sensors run continuously relying on their alarms to signal out-of-bounds conditions. The user would usually disable the sample done flags for these TS types unless they wish to log every data sample.

Table 265. DTS interrupts

Interrupt event	Event flag	Enable control bit	Interrupt clear method
irq_smpl	IRQ_STATUS_DONE	IRQ_EN_DONE	IRQ_CLEAR_DONE
irq_alarma	IRQ_STATUS_ALARMA	IRQ_EN_ALARMA	IRQ_CLEAR_ALARMA

Table 265. DTS interrupts

Interrupt event	Event flag	Enable control bit	Interrupt clear method
irq_alarmb	IRQ_STATUS_ALARMB	IRQ_EN_ALARMB	IRQ_CLEAR_ALARMB
irq_fault	IRQ_STATUS_FAULT	IRQ_EN_FAULT	IRQ_CLEAR_FAULT

33.5 DTS registers

The APB interface supports 32-bit word aligned addresses (byte access is not supported).

The memory map contains a mixture of read-only, read-write, and write-only register fields. The write-only bits are self-clearing bits that are generally used to create strobes. Reserved register fields and any undefined register addresses are read-only and return zero when read.

Table 266 shows the DTS memory map overview.

Table 266. DTS register memory map

Address boundary	Register bank	Register block size	Address locations	Description
0x0000 - 0x003F	PVT common registers	0x40	16	PVT common registers
0x0040 - 0x007F	PVT interrupt registers	0x40	16	PVT interrupt registers
0x0080 - 0x00BF	TS common registers	0x40	16	TS common registers
0x00C0 - 0x00FF	TS 0 individual registers	0x40	16	TS0 dedicated registers (such as IRQ, alarms and samples)
0x0100 - 0x013F	TS 1 individual registers	0x40	16	TS1 dedicated registers (such as IRQ, alarms and samples)

33.5.1 DTS PVT common registers

The PVT common registers are used to configure the timer and read back the lock status.

DTS PVT register lock register (DTS_PVTREG_LOCKR)

The DTS can be locked by software using the DTS_PVTREG_LOCKR register. Any write to the register locks all DTS registers except for the PVT common registers.

Address offset: 0x010

Reset value: 0x0000 0000

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

Bits 31:0 **LOCK[31:0]:** PVT software lock register
Any write to this register locks all DTS registers except for the PVT common registers. Once locked, the DTS registers are read-only. The software lock can be removed by writing 0x1ACC E551.

DTS PVT lock status register (DTS_PVTLOCK_SR)

Address offset: 0x014

Reset value: 0x0000 0000

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

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

Bit 1 **HW_LOCK_STATUS:** Hardware lock input status
This bit indicates the status of the hardware lock input to the PVT controller.

Bit 0 **SW_LOCK_STATUS:** Software lock input status
This bit indicates the current state of the lock register.

DTS PVT timer control register (DTS_PVTTMR_CR)

Address offset: 0x020

Reset value: 0x0000 0000

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

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **TMR_RUN:** Timer count enable bit
This bit enables the timer count function.

Bits 15:0 **TMR_DELAY[15:0]:** Timer delay
This bitfield can be used to configure the timer delay.

DTS PVT timer status register (DTS_PVTTMR_SR)

Address offset: 0x024

Reset value: 0x0000 0000

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

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

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

Bit 1 **TMR_DONE:** Counter delay expiration flag
This bit is set when the counter delay has expired.

Bit 0 **TMR_BUSY:** Counter busy flag
This bit is set while the counter is active.

33.5.2 PVT IRQ registers

The PVT IRQ registers are present in all build configurations of the PVT controller.

DTS PVT IRQ enable register (DTS_PVT_IER)

Address offset: 0x040

Reset value: 0x0000 0000

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

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

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

Bit 1 **TS_IRQ_ENABLE:** TS IRQ source enable bit
Set this bit to enable the TS IRQ to the processor.

Bit 0 **TMR_IRQ_ENABLE:** Timer IRQ source enable bit
Set this bit to enable the timer IRQ to the processor.

DTS PVT IRQ timer mask register (DTS_PVTIRQTRMASKR)

Address offset: 0x050

Reset value: 0x0000 0000

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

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

Bit 0 **TMR_IRQ_MASK:** Timer IRQ source mask bit
Set this bit to mask the timer IRQ to the processor.

DTS PVT IRQ TS mask register (DTS_TS_MR)

Address offset: 0x054

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_IRQ_MASK	TS0_IRQ_MASK
														rw	rw

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

Bit 1 **TS1_IRQ_MASK:** TS1 IRQ source mask bit
Set this bit to mask TS1 IRQ to the processor.

Bit 0 **TS0_IRQ_MASK:** TS0 IRQ source mask bit
Set this bit to mask TS0 IRQ to the processor.

DTS PVT IRQ timer status register (DTS_PVTTR_SR)

Address offset: 0x060

Reset value: 0x0000 0000

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

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

Bit 0 **TMR_IRQ_STATUS:** Timer IRQ status bit after masking
This bit reflects the timer IRQ status after the mask has been applied.

DTS PVT IRQ TS status register (DTS_TS_ISR)

Address offset: 0x064

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_IRQ_STATUS	TS0_IRQ_STATUS
														r	r

Bits 2 Reserved, must be kept at reset value.

Bit 1 **TS1_IRQ_STATUS:** TS1 IRQ status bit after masking
This bit reflects the timer IRQ status after the mask has been applied.
Note: The bit is set when the corresponding TS block has issued an IRQ interrupt to the processor.

Bit 0 **TS0_IRQ_STATUS:** TS0 IRQ status bit after masking
This bit reflects the timer IRQ status after the mask has been applied.
Note: The bit is set when the corresponding TS block has issued an IRQ interrupt to the processor.

DTS PVT IRQ timer raw status register (DTS_PVTTMRRAW_ISR)

Address offset: 0x070

Reset value: 0x0000 0000

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

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

Bit 0 **TMR_IRQ_RAW_STATUS:** TMR IRQ status bit before masking
This bit reflects the timer IRQ status before the mask is applied.

DTS PVT IRQ TS raw status register (DTS_TSRAW_ISR)

Address offset: 0x074

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_IRQ_RAW_STATUS	TS0_IRQ_RAW_STATUS
														r	r

Bits 2 Reserved, must be kept at reset value.

Bit 1 **TS1_IRQ_RAW_STATUS:** TS1 IRQ status bit before masking
This bit reflects the timer IRQ status before the mask is applied.

Bit 0 **TS0_IRQ_RAW_STATUS:** TS0 IRQ status bit before masking
This bit reflects the timer IRQ status before the mask is applied.

33.5.3 TS common registers

DTS TSC clock synthesizer register (DTS_TSCCLKSYNTHR)

Address offset: 0x080

Reset value: 0x0001 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLK_SYTH_EN	Res.	Res.	Res.	Res.	CLK_SYNTH_HOLD[3:0]			
							rw					rw	rw	rw	rw

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

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **CLK_SYTH_EN:** Synthesized clk_ts enable bit
0: Disabled
1: Enabled

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **CLK_SYNTH_HOLD[3:0]:** SDA master-to-SDA slave output hold delay/SDA slave-to-SDA master input setup delay
The delay is counted from the rising edge of clk_ts. It is expressed in units of ck_ker_dts cycles. The default value is 0b0001.

Bits 15:8 **CLK_SYNTH_HI[7:0]:** Synthesized clk_ts high period
This is a zero-Indexed value expressed in units of ck_ker_dts cycles.

Bits 7:0 **CLK_SYNTH_LO[7:0]:** Synthesized clk_ts low period
This is a zero-Indexed value expressed in units of ck_ker_dts cycles.

DTS TSC SDIF interface disable register (DTS_TSCSDIFDISABLER)

Address offset: 0x084

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_SDIF_DISABLE	TS0_SDIF_DISABLE
														rw	rw

Bits 2 Reserved, must be kept at reset value.

Bit 1 **TS1_SDIF_DISABLE:** TS1 serial data interface (SDIF) disable bit

When this bit is asserted, it completely disables TS1 instance by forcing TS1 clock and reset low.

0: Enabled

1: Disabled

Bit 0 **TS0_SDIF_DISABLE:** TS0 serial data interface (SDIF) disable bit

When this bit is asserted, it completely disables the selected TS0 instance by forcing TS0 clock and reset low.

0: Enabled

1: Disabled

Note: Any disable action must be applied before SDIF programming begins.

DTS TSC SDIF status register (DTS_TSCSDIF_SR)

Address offset: 0x088

Reset value: 0x0000 0000

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

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

Bit 1 **SDIF_LOCK:** SDIF locked flag

This bit is set when the slave automatic mode is requested, to indicate that the serial data interface (SDIF) is now a write-only interface (see *Section : SDA controller operation*).

In slave automatic mode, the data register (SDATS_DR) is automatically updated within the PVT controller whenever new TS data become available

0: Unlocked

1: Locked

Bit 0 **SDIF_BUSY:** SDIF busy flag

When asserted, this bit indicates either that the serial data interface (SDIF) is busy and any SDIF programming requests are ignored, or the clock synthesizer/disable SDIF state is being updated.

0: SDIF not busy/no clock synthesizer/disable SDIF state update ongoing

1: SDIF busy/clock synthesizer/disable SDIF state update ongoing

DTS TSC SDIF register (DTS_TSCSDIF_CR)

Address offset: 0x08C

Reset value: 0x0000 0000

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

Bit 31 **SDIF_PROG:** Serial interface program request

This self-clearing bit is ignored if the SDIF_BUSY or the SDIF_LOCK flag is set, and the request is a read request (SDIF_WRN = 0).

0: No SDIF program request

1: SDIF program requested

Bits 30:28 Reserved, must be kept at reset value.

Bit 27 **SDIF_WRN:** Serial interface write/no read control bit

0: SDIF read

1: SDIF write

Bits 26:24 **SDIF_ADDR[2:0]:** Serial interface register address

This bitfield contains the address of the target SDIF register, SDIF_ADDR[2:0].

Bits 23:0 **SDIF_WDATA[23:0]:** Serial interface write data

This bitfield contains the write data SDIF_DATA[23:0].

DTS TSC SDIF halt register (DTS_TSCSDIFHALTR)

Address offset: 0x090

Reset value: 0x0000 0000

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

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

Bit 0 **SDIF_STOP:** Serial data interface (SDIF) stop
This self-clearing bit immediately stops all SDIF programming activity in the unlikely event of it becoming locked. It must not be required during normal operation.
0: Not stopped
1: Stopped

DTS TSC SDIF control register (DTS_TSCSDIF_CFGR)

Address offset: 0x094

Reset value: 0x0000 0000

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

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

Bits 1:0 **SDIF_INHIBIT[1:0]:** Serial data interface (SDIF) programming inhibit
When asserted, this bit inhibits the serial programming of the selected TS instance(s). The inhibition must be applied before SDIF programming begins.
00: No Inhibition
01: TS0 inhibited/TS1 serial programming activated (SPA)
10: TS1 inhibited/TS0 serial programming activated (SPA)
11: TS1 and TS0 inhibited

DTS TSC sample control register (DTS_TSCSMPL_CR)

Address offset: 0x0A0

Reset value: 0x0000 0000

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

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

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

Bit 2 **SMPL_DISCARD:** Sample discard bit

This feature can be used to temporarily discard received data samples. When this bit is set, data recovery is disabled but the external TS may continue running.

0: Data samples not discarded

1: Data samples discarded

Bit 1 **SMPL_CTR_HOLD:** Sample counter hold bit

When this bit is asserted, the sample counter is prevented from rolling over when the maximum sample count is reached.

0: Counter not on hold

1: Counter on hold

Bit 0 **SMPL_CTR_DISABLE:** Sample counter disable bit

0: Sample counter enabled

1: Sample counter disabled

DTS TSC sample clear register (DTS_TSCSDIFSMPLCLRR)

Address offset: 0x0A4

Reset value: 0x0000 0000

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

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

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

Bit 0 **SMPL_CNTER_CLEAR:** Sample counter clear bit

The sample counter value is set to zero when this self-clearing bit is programmed to 1.

0: Counter value not cleared

1: Counter value cleared

DTS TSC sample count register (DTS_TSCSMPLCNTR)

Address offset: 0x0A8

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SMPL_COUNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **SMPL_COUNT[15:0]:** Sample counter

The 16-bit counter increments each time a new data sample is available. The counter is cleared by writing 1 to DTS_TSCSDIFSMPLCLRR. The sample counter overflow operation is determined by SMPL_CTR_HOLD.

33.5.4 TS individual registers

DTS TSn IRQ enable register (DTS_TSn_IER)

Address offset: 0x0C0 + 0x40 * n, (n = 0 to 1)

Reset value: 0x0000 0000

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

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

Bit 4 **IRQ_EN_ALARMB:** Alarm B IRQ enable bit

Bit 3 **IRQ_EN_ALARMA:** Alarm A IRQ enable bit

Bit 2 Reserved, must be kept at reset value.

Bit 1 **IRQ_EN_DONE:** Sample done IRQ enable bit

Bit 0 **IRQ_EN_FAULT:** Fault IRQ enable bit

DTS TSn IRQ status register (DTS_TSn_ISR)

Address offset: 0x0C4 + 0x40 * n, (n = 0 to 1)

Reset value: 0x0000 0000

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

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

Bit 4 **IRQ_STATUS_ALARMB:** Alarm B IRQ status bit

Bit 3 **IRQ_STATUS_ALARMA:** Alarm A IRQ status bit

Bit 2 Reserved, must be kept at reset value.

Bit 1 **IRQ_STATUS_DONE:** Sample done IRQ status bit

Bit 0 **IRQ_STATUS_FAULT:** Fault IRQ status bit

DTS TSn IRQ clear register (DTS_TSn_ICR)

Address offset: 0x0C8 + 0x40 * n, (n = 0 to 1)

Reset value: 0x0000 0000

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

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

Bit 4 **IRQ_CLEAR_ALARMB:** Alarm B IRQ clear bit

Bit 3 **IRQ_CLEAR_ALARMA:** Alarm A IRQ clear bit

Bit 2 Reserved, must be kept at reset value.

Bit 1 **IRQ_CLEAR_DONE:** Sample done IRQ clear bit

Bit 0 **IRQ_CLEAR_FAULT:** Fault IRQ clear bit

DTS TSn IRQ test register (DTS_TSnIRQTESTR)

Address offset: 0x0CC + 0x40 * n, (n = 0 to 1)

Reset value: 0x0000 0000

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

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

Bit 4 **IRQ_TEST_ALARMB:** Alarm B IRQ test bit

Bit 3 **IRQ_TEST_ALARMA:** Alarm A IRQ test bit

Bit 2 Reserved, must be kept at reset value.

Bit 1 **IRQ_TEST_DONE:** Sample done IRQ test bit

Bit 0 **IRQ_TEST_FAULT:** Fault IRQ test bit

DTS TSn SDIF RDATA register (DTS_TSnSDIFRDATAR)

Address offset: 0x0D0 + 0x40 * n, (n = 0 to 1)

Reset value: 0x0000 0000

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

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:0 **SDIF_RDATA[23:0]:** SDIF read data

DTS TSn SDIF done register (DTS_TSnSDIFDONER)

Address offset: 0x0D4 + 0x40 * n, (n = 0 to 1)

Reset value: 0x0000 0000

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

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

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

Bit 0 **SDIF_SMPL_DONE:** Sample done flag

The sample done flag is asserted when a new data sample is available. The sample done flag is automatically cleared when the associated data register is read.

0: No new data sample available

1: New data sample available

DTS TSn SDIF data register (DTS_TSnSDIFDATAR)

Address offset: 0x0D8 + 0x40 * n, (n = 0 to 1)

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SAMPLE_DATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

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

Bit 17 **SAMPLE_FAULT:** Sample fault

This bit is set to indicate that the TS block has signaled a fault. The sample data must be discarded.

Bit 16 **SAMPLE_TYPE:** TS sample type

0: Indicates valid data

1: Indicates either analog access, signature select or fault debug mode

The TS_TYPE flag is updated when new data are received.

Bits 15:0 **SAMPLE_DATA[15:0]:** Sample data.

Reading this register clears the state of the SDIF_SMPL_DONE flag in DTS_TSnSDIFDONER register.

DTS TSn alarm A configuration register (DTS_TSnALARMA_CFGR)

Address offset: 0x0E0 + 0x40 * n, (n = 0 to 1)

Reset value: 0x0000 0000

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

Bits 31:16 **ALARMA_THRESH[15:0]:** Alarm A threshold

Bits 15:0 **HYSTA_THRESH[15:0]:** Alarm A hysteresis threshold

DTS TSn alarm B configuration register (DTS_TSnALARMB_CFGR)

Address offset: 0x0E4 + 0x40 * n, (n = 0 to 1)

Reset value: 0x0000 0000

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

Bits 31:16 **ALARMB_THRESH[15:0]:** Alarm B threshold

Bits 15:0 **HYSTB_THRESH[15:0]:** Alarm B hysteresis threshold

DTS TSn high/low sample register (DTS_TSnHLSAMPLER)

Address offset: 0x0E8 + 0x40 * n, (n = 0 to 1)

Reset value: 0x0000 FFFF

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

Bits 31:16 **SMPL_HI[15:0]:** Highest valid data sample value received

Bits 15:0 **SMPL_LO[15:0]:** Lowest valid data sample value received

DTS TSn high/low reset register (DTS_TSnHILORESETR)

Address offset: 0x0EC + 0x40 * n, (n = 0 to 1)

Reset value: 0x0000 0000

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

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

Bit 1 **SMPL_HI_CLR:** Sample high clear 0

All the bits of the stored sample high value are initialized to 0 when written.

Bit 0 **SMPL_LO_SET:** Sample Low Set

All the bits of the stored sample low value are initialized to 1 when written.

33.5.5 DTS register map

Table 267. PVT common register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000-0x00C	Reserved	Reserved																															
0x010	DTS_PVTREG_LOCKR	LOCK[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	DTS_PVTLOCK_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HW_LOCK_STATUS	SW_LOCK_STATUS
	Reset value																															0	0
0x018 - 0x01C	Reserved	Reserved																															
0x020	DTS_PVTTMR_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TMR_RUN	TMR_DELAY[15:0]															
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	DTS_PVTTMR_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TMR_DONE	TMR_BUSY
	Reset value																															0	0
0x030 - 0x03C	Reserved	Reserved																															

Table 268. PVT IRQ register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x040	DTS_PVT_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS_IRQ_ENABLE	TMR_IRQ_ENABLE
	Reset value																															0	0
0x044 - 0x04C	Reserved	Reserved																															
0x050	DTS_PVTIRQTRMASKR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TMR_IRQ_MASK
	Reset value																																0
0x054	DTS_TS_MR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_IRQ_MASK	TS0_IRQ_MASK
	Reset value																															0	0

Table 268. PVT IRQ register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x058 - 0x05C	Reserved	Reserved																															
0x060	DTS_PVTTR_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TMR_IRQ_STATUS
	Reset value																																0
0x064	DTS_TS_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_IRQ_STATUS	TS0_IRQ_STATUS
	Reset value																															0	0
0x068 - 0x06C	Reserved	Reserved																															
0x070	DTS_PVTTMR RAW_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TMR_IRQ_RAW_STATUS
	Reset value																																0
0x074	DTS_TSRAW_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_IRQ_RAW_STATUS	TS0_IRQ_RAW_STATUS
	Reset value																															0	0
0x078 - 0x07C	Reserved	Reserved																															

Table 269. DTS TSC clock synthesizer register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x080	DTS_TSCCLK SYNTHR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLK_SYTH_EN	Res.	Res.	Res.	Res.	CLK_SYNTH _HOLD[3:0]				CLK_SYNTH_HI[7:0]								CLK_SYNTH_LO[7:0]							
	Reset value								0					0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 269. DTS TSC clock synthesizer register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x084	DTS_TSCSDIF DISABLER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS1_SDIF_DISABLE	TS0_SDIF_DISABLE
	Reset value																															0	0
0x088	DTS_TSCSDIF_ SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDIF_LOCK	SDIF_BUSY
	Reset value																															0	0
0x08C	DTS_TSCSDIF_ CR	SDIF_PROG	Res.	Res.	Res.	SDIF_WRN	SDIF_ADDR [2:0]			SDIF_WDATA[23:0]																							
	Reset value	0				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x090	DTS_TSCSDIF HALTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDIF_STOP
	Reset value																																0
0x094	DTS_TSCSDIF_ CFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDIF_INHIBIT [1:0]	
	Reset value																															0	0
0x098 - 0x09C	Reserved	Reserved																															
0x0A0	DTS_TSCSMPL _CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMPL_DISCARD	SMPL_CTR_HOLD	SMPL_CTR_DISABLE
	Reset value																														0	0	0
0x0A4	DTS_TSCSDIF SMPLCLRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMPL_CNTER_CLEAR
	Reset value																																0
0x0A8	DTS_TSCSMPL CNTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMPL_COUNT[15:0]																	
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B0 - 0x0BF	Reserved	Reserved																															

Table 270. DTS TS individual register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0C0 + 0x40 * n, (n = 0 to 1)	DTS_TSn_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IRQ_EN_ALARMB	IRQ_EN_ALARMA	Res.	IRQ_EN_DONE	IRQ_EN_FAULT
	Reset value																												0	0		0	0
0x0C4 + 0x40 * n, (n = 0 to 1)	DTS_TSn_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IRQ_STATUS_ALARMB	IRQ_STATUS_ALARMA	Res.	IRQ_STATUS_DONE	IRQ_STATUS_FAULT
	Reset value																												0	0		0	0
0x0C8 + 0x40 * n, (n = 0 to 1)	DTS_TSn_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IRQ_CLEAR_ALARMB	IRQ_CLEAR_ALARMA	Res.	IRQ_CLEAR_DONE	IRQ_CLEAR_FAULT
	Reset value																												0	0		0	0
0x0CC + 0x40 * n, (n = 0 to 1)	DTS_TSnIRQ TESTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IRQ_TEST_ALARMB	IRQ_TEST_ALARMA	Res.	IRQ_TEST_DONE	IRQ_TEST_FAULT
	Reset value																												0	0		0	0
0x0D0 + 0x40 * n, (n = 0 to 1)	DTS_TSnSDIF RDATAR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDIF_RDATA[23:0]																							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0D4 + 0x40 * n, (n = 0 to 1)	DTS_TSnSDIF DONER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDIF_SMPL_DONE
	Reset value																																0
0x0D8 + 0x40 * n, (n = 0 to 1)	DTS_TSnSDIF DATAR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SAMPLE_FAULT	SAMPLE_TYPE	SAMPLE_DATA[15:0]															
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0DC - 0x0E8	Reserved	Reserved																															
0x0E0 + 0x40 * n, (n = 0 to 1)	DTS_TSn ALARMA_ CFGR	HYSTA_THRESH[15:0]																ALARMA_THRESH[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 270. DTS TS individual register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0E4 + 0x40 * n, (n = 0 to 1)	DTS_TSn ALARMB_ CFGR	HYSTB_THRESH[15:0]																ALARMB_THRESH[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0E8 + 0x40 * n, (n = 0 to 1)	DTS_TSnHL SAMPLER	SMPL_HI[15:0]																SMPL_LO[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0EC + 0x40 * n, (n = 0 to 1)	DTS_TSnHILO RESETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMPL_HI_CLR	SMPL_LO_CLR
	Reset value																															0	0

Refer to *Section 2.3* for the register boundary addresses.

34 Voltage reference buffer (VREFBUF)

34.1 VREFBUF introduction

The devices embed a voltage reference buffer which can be used as voltage reference for the on-chip ADCs, and also as voltage reference for external components through the VREF+ pin.

34.2 VREFBUF implementation

The table below describes the VREFBUF voltages typical values:

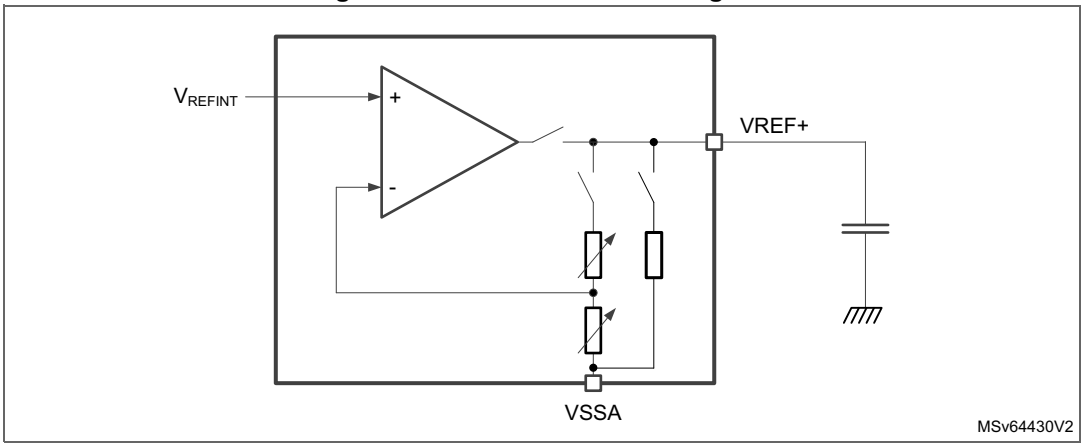
Table 271. VREFBUF typical values

Symbol	Value
VREFBUF0	1.21 V
VREFBUF1	1.5 V

Note: Refer to the product datasheet for more details.

34.3 VREFBUF functional description

Figure 320. VREFBUF block diagram



The internal voltage reference buffer is an operational amplifier, with programmable gain. The amplifier input is connected to the internal voltage reference V_{REFINT}. The VREFBUF supports two voltages[a], which are configured with VRS bits in the VREFBUF_CSR register:

- VRS = 000: VREFBUF0 voltage selected
- VRS = 001: VREFBUF1 voltage selected

The internal voltage reference can be configured in four different modes depending on ENVR and HIZ bits configuration. These modes are provided in the table below:

Table 272. VREF buffer modes

ENVR	HIZ	VREF buffer configuration
0	0	VREFBUF buffer off mode: – V_{REF+} pin pulled-down to V_{SSA}
0	1	External voltage reference mode (default value): – VREFBUF buffer off – V_{REF+} pin input mode
1	0	Internal voltage reference mode: – VREFBUF buffer on – V_{REF+} pin connected to VREFBUF buffer output
1	1	Hold mode: – VREF is enable without output buffer, VREF+ pin voltage is hold with the external capacitor – VRR detection disabled and VRR bit keeps last state

After enabling the VREFBUF by setting ENVR bit and clearing HIZ bit in the VREFBUF_CSR register, the user must wait until VRR bit is set, meaning that the voltage reference output has reached its expected value.

34.4 VREFBUF trimming

The VREFBUF output voltage is factory-calibrated by ST. At reset, and each time the VRS setting is changed, the calibration data is automatically loaded to the TRIM register.

Optionally user can trim the output voltage by changing the TRIM register bits directly. In this case, the VRS setting has no more effect on the TRIM register until the device is reset.

a. The minimum V_{DDA} voltage depends on VRS setting, refer to the product datasheet.

34.5 VREFBUF power sequence

While VREFBUF is activated, V_{DDA} supply changing from 1.8 V to 0 V can cause large current flowing from the V_{REF+} output capacitor to the V_{DDA} power domain through on-chip parasitic resistance. To avoid this current, the following procedure is required:

- While V_{DDA} is available, configure VREFBUF in buffer off mode.
- Wait for V_{REF+} to discharge (through internal pull-down) to less than 100 mV (typically 1 ms, depending on the capacitor value).
- Switch off V_{DDA} supply to 0 V.

34.6 VREFBUF registers

34.6.1 VREFBUF control and status register (VREFBUF_CSR)

Address offset: 0x00

Reset value: 0x0000 0002

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

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 **VRS[2:0]**: Voltage reference scale

These bits select the value generated by the voltage reference buffer.

VRS = 000: VREFBUF0 voltage selected.

VRS = 001: VREFBUF1 voltage selected.

Others: Reserved

Note: Refer to the product datasheet for each VREFBUFx voltage setting value.
The software can program this bitfield only when the VREFBUF is disabled (ENVR=0).

Bit 3 **VRR:** Voltage reference buffer ready

0: the voltage reference buffer output is not ready.

1: the voltage reference buffer output reached the requested level.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **HIZ:** High impedance mode

This bit controls the analog switch to connect or not the V_{REF+} pin.

0: V_{REF+} pin is internally connected to the voltage reference buffer output.

1: V_{REF+} pin is high impedance.

Refer to *Table 272: VREF buffer modes* for the mode descriptions depending on ENVR bit configuration.

Bit 0 **ENVR:** Voltage reference buffer mode enable

This bit is used to enable the voltage reference buffer mode.

0: Internal voltage reference mode disable (external voltage reference mode).

1: Internal voltage reference mode (reference buffer enable or hold mode) enable.

34.6.2 VREFBUF calibration control register (VREFBUF_CCR)

Address offset: 0x04

Reset value: 0x0000 00XX

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

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

Bits 5:0 **TRIM[5:0]**: Trimming code

The TRIM code is a 6-bit unsigned data (minimum 000000, maximum 111111) that is set and updated according the mechanism described below.

Reset:

TRIM[5:0] is automatically initialized with the VRS = 0 trimming value stored in the OTP memory during the production test.

VRS change:

TRIM[5:0] is automatically initialized with the trimming value (corresponding to VRS setting) stored in the OTP memory during the production test.

Write in TRIM[5:0]:

User can modify the TRIM[5:0] with an arbitrary value. This is permanently disabling the control of the trimming value with VRS (until the device is reset).

Note: If the user application performs the trimming, the trimming code must start from 000000 to 111111 in ascending order.

34.6.3 VREFBUF register map

The following table gives the VREFBUF register map and the reset values.

Table 273. VREFBUF register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	VREFBUF_CSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VRS[2:0]			VRR	Res.	HIZ	ENVR
	Reset value																										0	0	0	0		1	0
0x04	VREFBUF_CCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIM[5:0]					
	Reset value																											x	x	x	x	x	x

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

35 Multi-function digital filter (MDF)

35.1 MDF introduction

The multi-function digital filter (MDF) is a high-performance module dedicated to the connection of external sigma-delta (ΣΔ) modulators. It is mainly targeted for the following applications:

- audio capture signals
- motor control
- metering

The MDF features 6 digital serial interfaces (SITFx) and digital filters (DFLTx) with flexible digital processing options to offer up to 24-bit final resolution.

The DFLTx of the MDF also include the filters of the audio digital filter (ADF).

The MDF can receive, via its serial interfaces, streams coming from various digital sensors.

The MDF supports the following standards allowing the connection of various ΣΔ modulator sensors:

- SPI interface
- Manchester coded 1-wire interface
- PDM interface

A flexible bitstream matrix (BSMX) allows the connection of any incoming bitstream to any filter.

The MDF converts an input data stream into clean decimated digital data words. This conversion is done thanks to low-pass digital filters and decimation blocks. In addition it is possible to insert a high-pass filter or a DC offset correction block.

The conversion speed and resolution are adjustable according to configurable parameters for digital processing: filter type, filter order, decimation ratio, integrator length. The maximum output data resolution is up to 24 bits. There are two conversion modes: single conversion and continuous modes. The data can be automatically stored in a system RAM buffer through DMA, thus reducing the software overhead.

A flexible trigger interface can be used to control the conversion start. This timing control can trigger simultaneous conversions or insert a programmable delay between conversions.

The MDF features an out-off limit detectors (OLD) function. There is one OLD for each digital filter chain. Independent programmable thresholds are available for each OLD, making it very suitable for over-current detection.

A short circuit detector (SCD) is also available for every selected bitstream. The SCD is able to detect a short-circuit condition with a very short latency. Independent programmable thresholds are offered in order to define the short circuit condition.

The digital processing is performed using only the kernel clock. The MDF requests the bus interface clock (AHB clock) only when data must be transfered or when a specific event requests the attention of the system processor.

35.2 MDF main features

- AHB interface
- 6 serial digital inputs:
  - configurable SPI interface to connect various digital sensors
  - configurable Manchester coded interface support
  - compatible with PDM interface to support digital microphones
- 2 common clocks input/output for ΣΔ modulators
- Flexible matrix (BSMX) for connection between filters and digital inputs
- 2 inputs to connect the internal ADCs
- 6 flexible digital filter paths, including:
  - A configurable CIC filter:
    - Can be split into 2 CIC filters: high-resolution filter and out-off limit detector
    - Can be configured in $Sinc^4$ filter
    - Can be configured in $Sinc^5$ filter
    - Adjustable decimation ratio
  - A reshape filter to improve the out-off band rejection and in-band ripple
  - A high-pass filter to cancel the DC offset
  - An offset error cancellation
  - Gain control
  - Saturation blocks
  - An out-off limit detector
- Short-circuit detector
- Clock absence detector
- 16- or 24-bit signed output data resolution
- Continuous or single conversion
- Possibility to delay independently each bitstream
- Various trigger possibilities
- Break generation on out-of limit or short-circuit detector events
- Autonomous functionality in Stop mode(s)
- DMA can be used to read the conversion data
- Interrupts services

35.3 MDF implementation

The devices embed one MDF instance and one ADF instance, both being digital filters with common features.

Table 274. MDF features (1)

Modes or features	MDF1
Number of filters (DFLT) and serial interfaces (SITF)	6
MDF_CKI[x] connected to pads	X

Table 274. MDF features (1) (continued)

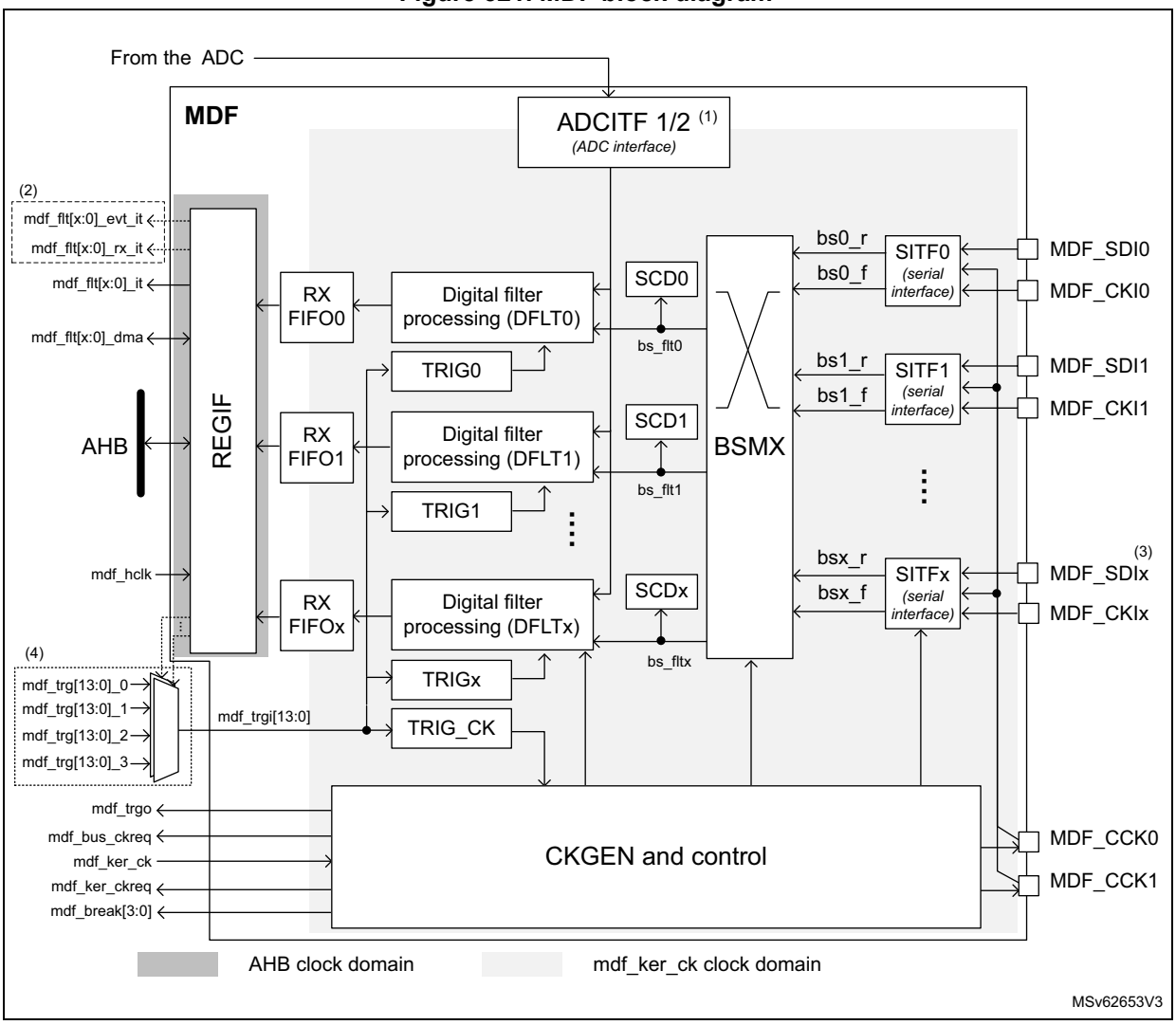
Modes or features	MDF1
Sound Activity Detection (SAD)	-
RXFIFO depth (number of 24-bit words)	4
Number of trigger inputs	14
ADC connected to ADCITF1	ADC1
ADC connected to ADCITF2	ADC2
Motor dedicated features (SCD, OLD, OEC, INT, snapshot, break)	X
Main path with CIC4, CIC5	X
Main path with CIC1,2, 3 or FastSinc	X
RSFLT, HPF, SAT, SCALE, DLY, Discard functions	X
Global Interrupts MDF_FLT[0:0] (mdf_it[0:0])	X
Receive interrupts MDF_RX[0:0] (mdf_rx_it[0:0])	-
Event interrupts MDF_EVT[0:0] (mdf_evt_it[0:0])	-
Autonomous in Stop mode	X(2)
Only DMA of filter 0 connected	-
All DMA channels connected	X

1. 'X' = supported, '-' = not supported.
2. Only in Stop SVOS high mode.

35.4 MDF functional description

35.4.1 MDF block diagram

Figure 321. MDF block diagram



1. Refer to *Section 35.3: MDF implementation* to check if the ADCITF is available, and which ADCs are connected.
2. Not always implemented, refer to the vector table of the product for details.
3. Refer to *Section 35.3: MDF implementation* to check the number of filter instances available.
4. The number of trigger inputs depends on the product. Refer to *Section 35.3: MDF implementation* for details.

35.4.2 MDF pins and internal signals

Table 275. MDF external pins

Name	Signal type	Comment
MDF_CKIy (y = 0 to 5)(1)	Input	Dedicated clock signals from external sensors
MDF_SDIy (y = 0 to 5)(1)	Input	Data signal from external sensors
MDF_CCKy (y = 0, 1)	Input/output	Clock outputs for external sensor, or common clock input from external sensors

1. The number of inputs available depends on the number of filters. Refer to *Section 35.3* for details.

Table 276. MDF internal signals

Name	Signal type	Comment
mdf_trgi[13:0]	Input	Trigger inputs in order to control the acquisition (see *Table 277: MDF trigger connections* for details)
mdf_trgo	Output	Trigger output for synchronizing several MDFs
mdf_break[3:0]	Output	Break signals event generation from over-current detector or short-circuit detector (see the next table for details)
mdf_flt[5:0]_dma(1)	Input/output	DMA request/acknowledge signals for each filter processing chain
mdf_flt[5:0]_it(1)	Output	Global interrupt signals, for each MDF filter
mdf_flt[5:0]_rx_it(1)(2)	Output	Receive interrupt signals, for each MDF filter.
mdf_flt[5:0]_evt_it(1)(2)	Output	Event interrupt signals, for each MDF filter.
mdf_bus_ckreq	Output	Bus interface clock request output
mdf_ker_ckreq	Output	Kernel clock request output
mdf_ker_ck	Input	Kernel clock input
mdf_hclk	Input	AHB bus interface clock input
mdf_adcitf1_dat[15:0]	Input	ADCITF1 data input
mdf_adcitf2_dat[15:0]	Input	ADCITF2 data input

1. The number of signals available depends on the number of filters. Refer to *Section 35.3* for details.
2. Not always connected. See the interrupt vector table for details.

The table below shows the way the trigger inputs of the MDF are connected.

Table 277. MDF trigger connections

Trigger name	Trigger source
mdf_trgi0	EXTI15
mdf_trgi1	EXTI11
mdf_trgi2	tim1_trgo
mdf_trgi3	tim1_trgo2
mdf_trgi4	tim3_trgo

Table 277. MDF trigger connections (continued)

Trigger name	Trigger source
mdf_trgi5	tim4_trgo
mdf_trgi6	tim6_trgo
mdf_trgi7	tim7_trgo
mdf_trgi8	tim8_trgo
mdf_trgi9	tim8_trgo2
mdf_trgi10	tim15_trgo
mdf_trgi11	lptim1_ch1
mdf_trgi12	adf_sad_det signal (from ADF)
mdf_trgi13	adf_trgo signal (from ADF)
mdf_trgo	-

The table below shows the way the break outputs of the MDF are connected.

Table 278. MDF break connections

Trigger name	Trigger source
mdf_break0	tim1_brk_cmp[8]
mdf_break1	tim1_brk2_cmp[8]
mdf_break2	tim8_brk_cmp[8]
mdf_break3	tim8_brk2_cmp[8]

35.4.3 Serial input interfaces (SITF)

The SITFx input interfaces allow the connection of the external sensors to the digital filters, via the bitstream matrix (BSMX). The SITFx serial interface can be configured in the following modes:

- LF_MASTER SPI mode (low-frequency)
- normal SPI mode
- Manchester mode

The amount of SITFx instances is equal to the amount of filters.

The data from each serial interface can be routed to any filter in order to perform:

- the PDM to PCM conversion
- the out-off limit detection
- the short detection

The serial interfaces are enabled by setting the corresponding SITFEN bit to 1. Once the interface is enabled, it receives serial data from the external ΣΔ modulator.

Note: Before enabling the serial interface, the user must insure that the mdf_proc_ck is already enabled (see Section 35.4.5 for details).

The SITFx are controlled via the *MDF serial interface control register x (MDF_SITFxCR)*.

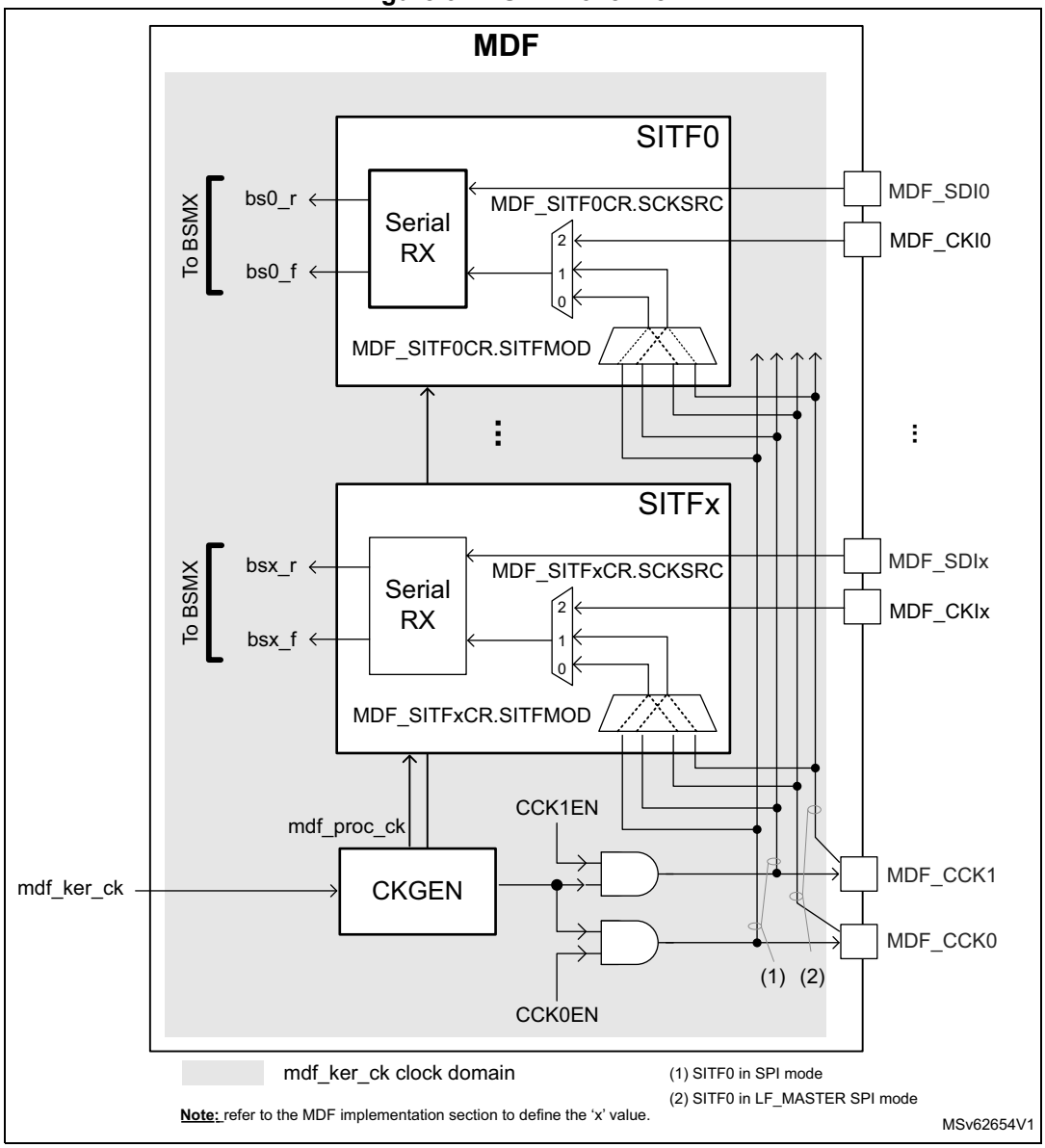
As shown in *Figure 322*, for each SITF, there is a large choice of clocking possibilities:

- If the serial interface is programmed in SPI mode, the selected clock source is a copy of the clock present on MDF_CCK0 or MDF_CCK1 or MDF_CKIx pin (see 2 in *Figure 322*).
- If the serial interface is programmed in LF_MASTER SPI mode, the selected clock source must be the clock directly provided by the CCKDIV to the MDF_CCK0 or MDF_CCK1 pin (see 1 in *Figure 322*). In this case MDF_CKIx must not be selected.

See *Table 279* for additional information.

Note: Using the common clock (MDF_CCK0 or MDF CCK1) can be helpful to share the same clock between several SITFx.

Figure 322. SITFx overview



LF_MASTER and normal SPI modes

The LF_MASTER SPI mode is a special mode allowing the usage of a mdf_proc_ck clock frequency, only two times higher than the sensor clock. This mode is dedicated to low-power use-cases, using low-speed sensors.

In LF_MASTER SPI mode, the MDF must provide the bitstream clock to the external sensors via MDF_CCK0 and MDF_CCK1 pins, and receives the bitstream data via the serial data input MDF_SDIx.

For each SITFx, the application must select the same clock than the one provided to the external sensor (MDF_CCK0 or MDF_CCK1), in order to guarantee optimal timing performances. This selection is done via SCKSRC[1:0].

Warning: The MDF_CKIx pin cannot be used in LF_MASTER SPI mode.

The normal SPI interface is a more flexible interface than the LF_MASTER SPI, but the mdf_proc_ck frequency must be at least four times higher than the sensor clock.

The application can select MDF_CCK0, MDF_CCK1 or MDF_CKIx clock for the capture of the data received via the MDF_SDIx pin.

The MDF can generate a clock to the sensors via MDF_CCK0 or MDF_CCK1 if needed.

For all SPI modes, all SITFs can share the same clock input (MDF_CCK0 or MDF_CCK1), in order to optimize the amount of requested I/Os.

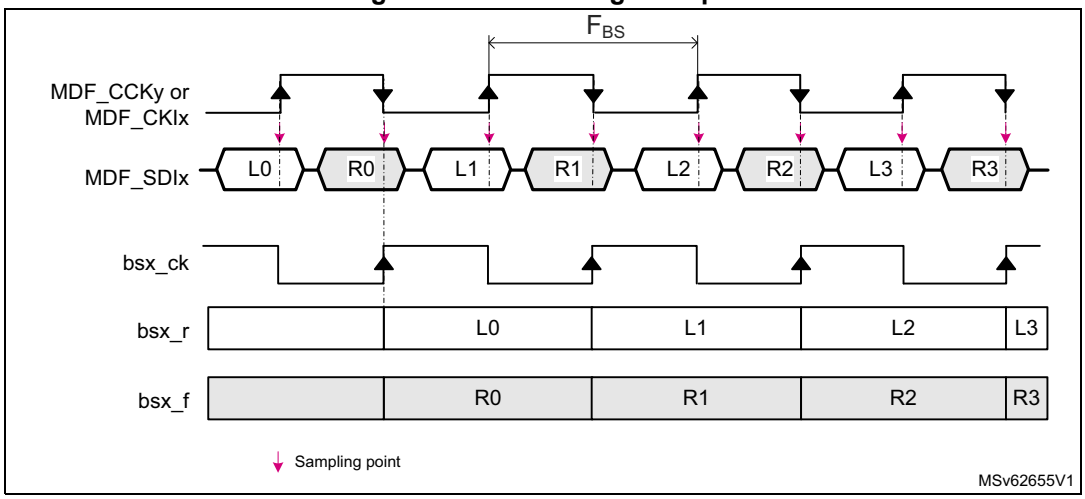
For all SPI modes, the serial data is captured using the rising and the falling edge of the selected clock. The SITFx always provides the following bitstreams:

- bitstream received using the bitstream clock falling edge (bsx_f)
- bitstream received using the bitstream clock rising edge (bsx_r)

According to the sensors connected, one of the two bitstreams may not be available.

The application can select the wanted stream via the BSMX matrix.

Figure 323. SPI timing example



To properly synchronize/receive the data stream, the frequency of the mdf_proc_ck clock must be adjusted according to the constraints listed in *Table 280*.

Clock absence detection

A no-clock-transition period may be detected when the serial interface works in normal SPI mode. This feature can be used to detect a clock failure in the SPI link.

The application can program a timeout value via the STH[4:0] bitfield of the corresponding SITFx. If the MDF does not detect clock transitions for a duration of STH[4:0] x $T_{mdf_proc_ck}$, then the CKABF flag is set.

An interrupt can be generated if CKABIE is set to 1. The STH[4:0] bitfield is in the *MDF serial interface control register x (MDF_SITFxCR)*.

When the serial interface is enabled, the CKABF flag remains to 1 until a first clock transition is detected.

To avoid spurious clock absence detection, the following sequence must be respected:

1. Configure the serial interface in normal SPI mode and enable it.
2. Clear the CKABF flag by writing CKABF bit to 1.

   If no clock transition is detected on the serial interface, the hardware immediately sets the CKABF flag to 1.
3. Read the CKABF flag:
   - If CKABF = 1, go back to step 2.
   - If CKABF i= 0, a clock has been detected. The CKABIE bit can be set to 1 if the application wants an interrupt on detection of a clock absence.

Note: The clock absence detection feature is not available in the LF_MASTER SPI mode.

Manchester mode

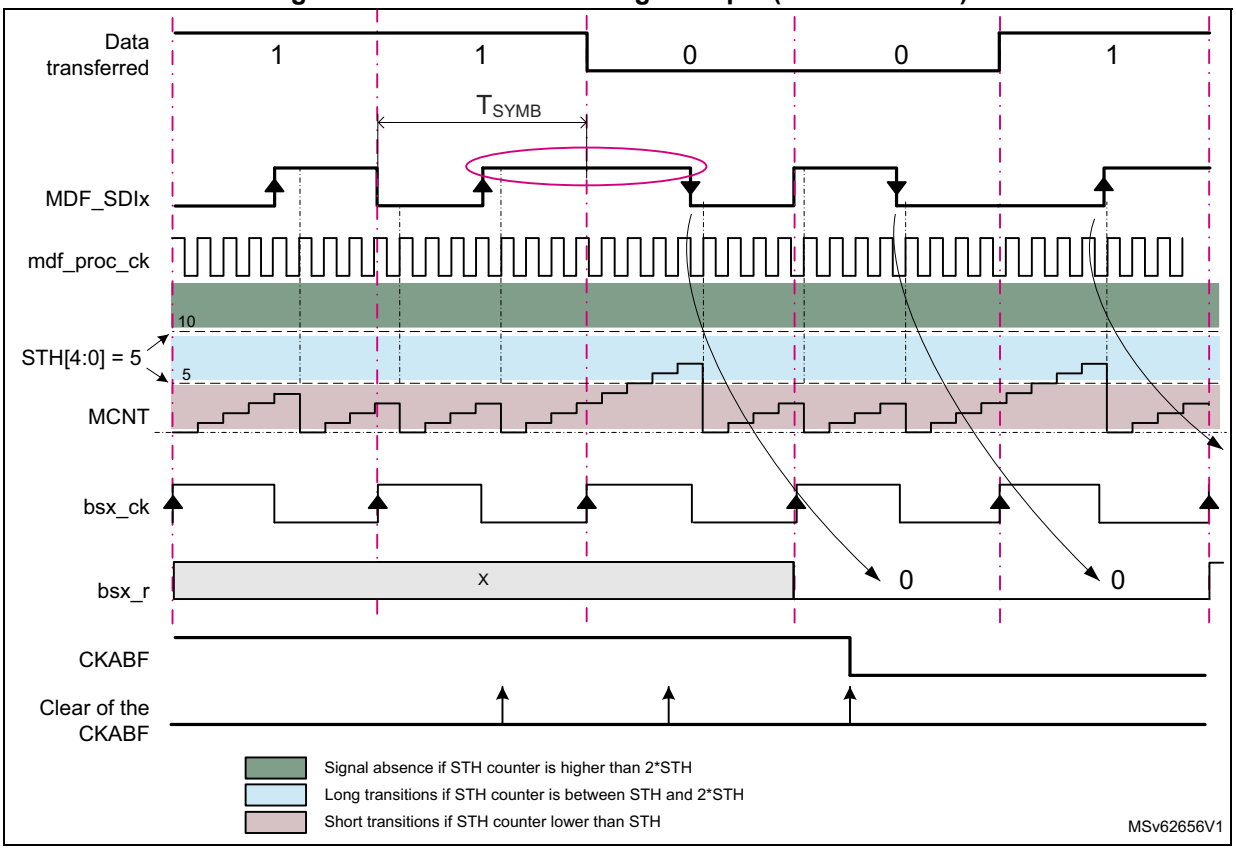
In Manchester coded format, the MDF receives data stream from the external sensor via the MDF_SDIx pin only.

The MDF_CKIx pins are not needed in this mode.

Decoded data and clock signals are recovered from serial stream after Manchester decoding. They are available on bsx_r. There are two possible settings of Manchester codings:

- signal rising edge decoded as 0 and signal falling edge decoded as 1
- signal rising edge decoded as 1 and signal falling edge decoded as 0

Figure 324. Manchester timing example (SITFMOD = 11)



To decode the incoming Manchester stream, the user must program the STH[4:0] bitfield in the *MDF serial interface control register x (MDF_SITFxCR)*. The STH[4:0] bitfield is used by the SITFx to estimate the Manchester symbol length and to detect a clock absence. An internal counter (MCNT) is restarted every time a transition is detected in the MDF_SDIx input. It is used to detect short transitions, long transitions or clock absence. A long transition indicates that the data value changed. *Figure 324* shows a case where the OVR is around height and STH[4:0] = 5.

The estimated Manchester symbol rate (T_{SYMB}) must respect the following formula:

$$(STH + 1)\times T_{mdf_proc_ck} < T_{SYMB} < (2 \times STH \times T_{mdf_proc_ck})$$

It is recommended to compute STH as follows:

$$STH[4:0] = round\left(\frac{(2 \times OVR) - 1}{3}\right)$$

where OVR represents the ratio between the mdf_proc_ck frequency and the expected Manchester symbol frequency. OVR must be higher than five, and the mdf_proc_ck clock must be adjusted according to the constraints listed in *Table 280*.

The clock absence flag CKABF is set to 1 when no transition is detected during more than 2 x STH[4:0] x $T_{mdf_proc_ck}$, or when the SITFx is not yet synchronized to the incoming Manchester stream. In addition, an interrupt can be generated if the bit CKABIE is set to 1.

When the serial interface is enabled, the MDF must first be synchronized to the incoming Manchester stream. The synchronization ends when a data transition from 0 to 1 or from 1 to 0 (pink circle in the *Figure 324*) is detected.

The end of the synchronization phase can be checked by following the software sequence:

1. Clear the CKABF flag in the *MDF DFLTx interrupt status register x (MDF_DFLTxISR)* by writing CKABF bit to 1. If the serial interface is not yet synchronized, the hardware immediately sets the CKABF flag to 1.
2. Read the CKABF flag.
   – If CKABF = 1, go back to step 1.
   – If CKABF = 0, the Manchester interface is synchronized and provides valid data.

Programming example

In the following example, the MDF kernel clock frequency ($F_{mdf_ker_ck}$) is 100 MHz and the received Manchester stream is at about 6 MHz (F_{SYMB}).

1. Provide a valid mdf_proc_ck to the SITFx.

   The mdf_proc_ck frequency must be at least six times higher than the Manchester symbol frequency (means at least 36 MHz).

   PROCDIV is programmed to 1 to perform a division by two of the kernel clock. In that case, $F_{mdf_proc_ck}$ = 50 MHz (8.33 times higher than the Manchester symbol frequency).
2. Compute STH.

   OVR is given by: OVR = $F_{mdf_proc_ck}$ / F_{SYMB} = 50 MHz / 6 MHz = 8.33.

   Then STH[4:0] = $\text{round}\left(\frac{(2 \times 8.33) - 1}{3}\right) = 5$

   The minimum allowed frequency for the Manchester stream is then:

   1 / (2 × STH × $T_{mdf_proc_ck}$) = 1 / (10 x 20 ns) = 5 MHz

   The maximum allowed frequency for the Manchester stream is then:

   1 / ((STH+1) × $T_{mdf_proc_ck}$) = 1 / (6 x 20 ns) = 8.33 MHz

35.4.4 ADC slave interface (ADCITF)

The ADCs are not always connected to the MDF. Refer to *Section 35.3* to check the situation for this product.

The MDF allows the connection of up to two ADCs to the filter path. For each filter, the DATSRC[1:0] bitfield in the *MDF digital filter configuration register x (MDF_DFLTxCICR)* allows the application to select either data from the ADCs.

Warning: The MDF does not support receiving interleaved data from one of the ADCITF input.

35.4.5 Clock generator (CKGEN)

The RCC (reset and clock controller) provides the following clocks to the MDF:

- AHB clock (mdf_hclk) used for the register interface
- kernel clock (mdf_ker_ck) mainly used by all other parts of the circuit via the CKGEN

Those clocks are not supposed to be phase locked, so all signals crossing those clock domains are resynchronized.

The clock generator (CKGEN) is responsible of the generation of the processing clock, and the clock provided to the MDF_CCK0 and MDF_CCK1 pins. All those clocks are generated from the mdf_ker_ck.

The processing clock (mdf_proc_ck) is used to run all the signals processing and to re-sample the incoming serial or parallel stream.

Note: The reshape filter (RSFLT) needs up to 24 cycles of mdf_proc_ck clock to process a sample.

To adapt the kernel clock frequency provided by the RCC, the following dividers are available:

- PROCDIV[6:0] used to adapt the kernel clock frequency to the constraints of the parallel and serial interfaces, and to the processing blocks
- CCKDIV[3:0] used to adapt the frequency of the MDF_CCK0 and MDF_CCK1 clocks

PROCDIV[6:0] and CCKDIV[3:0] must be programmed when no clock is provided to the dividers (CKGDEN = 0).

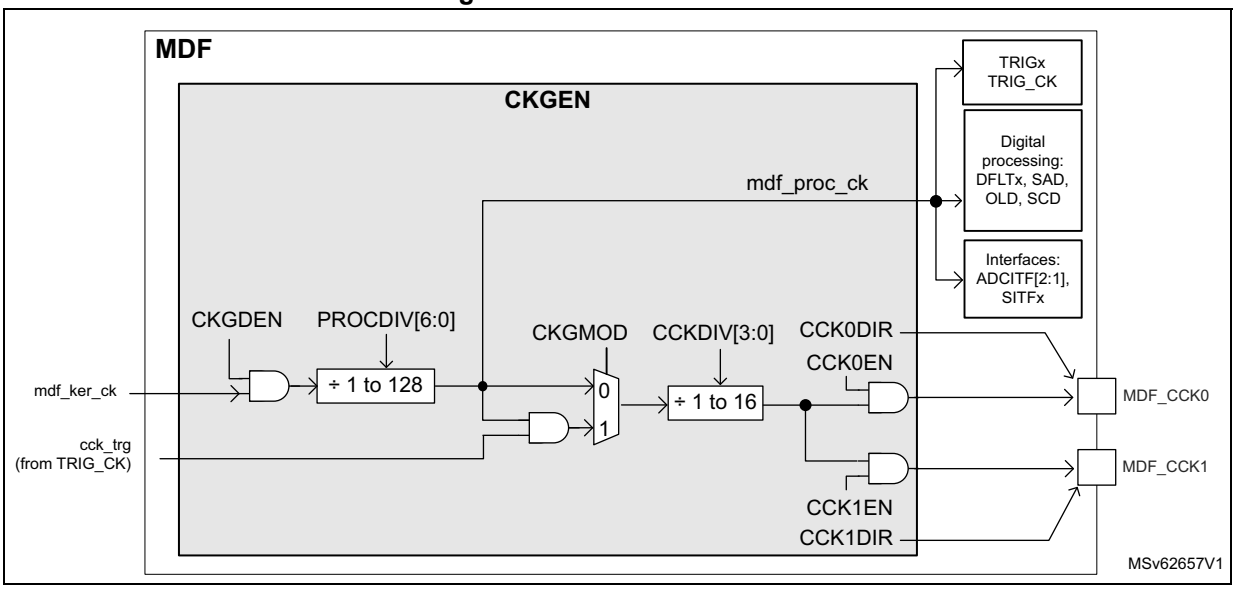
The mdf_proc_ck generation is controlled by CKGDEN.

In addition, the CKGMOD bit allows the application to define the way to trigger the CCKDIV divider:

- When CKGMOD = 0, the CCKDIV divider is started as soon as CKGDEN is set to 1.
- When CKGMOD = 1, the CCKDIV divider is started when CKGDEN is set to 1 and the programmed trigger condition occurred.

All bits and fields controlling the CKGEN are in MDF_CKGCR.

Figure 325. CKGEN overview



The trigger logic for CKGEN is handled by the TRG_CK block. As shown in *Figure 334*, the CCKDIV divider can be triggered on the rising or falling edge of one of the 16 trigger sources. When the proper trigger condition occurs, the cck_trg signal goes to high, allowing the CCKDIV divider to start. The TRG_CK logic is reset when CKGDEN is set to 0.

This feature can be helpful to synchronize the MDF_CCKy (y = 0,1) clock of several MDF instances, or to synchronize the clock generation to a timer event.

The application can control the activation of the MDF_CCK0 or MDF_CCK1 pin CCK0EN/CCK1EN and CCK0DIR/CCK1DIR bits:

- CCKyEN is used to enable the CCKDIV, and thus generates a clock for the external sensors.
- CCKyDIR is used to control the direction of the ADF_CCKy pin (input or output)

Table 279. Control of the common clock generation(1)

CCKyEN	CCKyDIR	Description
0	0	The MDF_CCKy pin is in input. An external clock can be connected to the MDF_CCKy pin and used by the SITFx in order to decode the serial stream
0	1	The MDF_CCKy pin is in output. No clock is generated, thus the MDF_CCKy pin is driven low.
1	1	The MDF_CCKy pin is in output. A clock is generated on the MDF_CCKy pin. The SITFx can use this pin as clock source in order to decode the serial stream

1. The configuration with CCKyEN = 1 and CCKyDIR = 0 must be avoided (no interest).

When CCKyDIR = 1, as soon as the CCKyEN bit is set to 1, a clock is generated to the corresponding output without any spurs.

Note: The mdf_proc_ck must be enabled (by CKGDEN = 1) before enabling other blocks (such as SITFx or DFLTx).

CKGEN activation sequence example

1. Set CKGDEN to 0.
2. Wait for CKGACTIVE = 0. If CKGDEN was previously enabled, this phase can take two periods of mdf_hclk and two periods of mdf_proc_ck.
3. Program PROCDIV[6:0], CKGMOD, CCKDIV[3:0], TRGSRC[3:0] and TRGSENS.
4. Set CKGDEN to 1.
5. Set CCKxDIR to 1 (optional).
6. Set CCKxEN to 1 (optional).

When needed, at any moment, the CCK0EN or CCK1EN value can be changed without disabling the clock generator.

Clock frequency constraints

The table below shows the frequency constraints to receive and process properly the samples.

Table 280. Clock constraints with respect to the incoming stream(1)

SITFx mode	MDF clock constraints	
	With RSFLT disabled	With RSFLT enabled
LF_MASTER SPI	F_{MDF_CCKy} max frequency limited to 5 MHz	
	$F_{mdf_proc_ck} > 2 * F_{MDF_CCKy}$ and $F_{mdf_hclk} \geq F_{mdf_proc_ck}$	$F_{mdf_proc_ck} > 24 * F_{MDF_CCKy} / (MCICD+1)$ and $F_{mdf_proc_ck} > 2 * F_{MDF_CCKy}$ and $F_{mdf_hclk} \geq F_{mdf_proc_ck}$
MASTER SPI SLAVE SPI	F_{MDF_CKx} max frequency limited to 25 MHz	
	$F_{mdf_proc_ck} > 4 * F_{MDF_CKKy}$ and F_{mdf_hclk} higher or equal to $F_{mdf_proc_ck}$	$F_{mdf_proc_ck} > 24 * F_{MDF_CKKy} / (MCICD+1)$ and $F_{mdf_proc_ck}$ higher than 4 x F_{MDF_CKKy} and F_{mdf_hclk} higher or equal to $F_{mdf_proc_ck}$
Manchester	F_{SYMB} max frequency limited to 20 MHz	
	$F_{mdf_proc_ck}$ higher than 6 * F_{SYMB} and $F_{mdf_hclk} \geq F_{mdf_proc_ck}$	$F_{mdf_proc_ck}$ higher than 24 * $F_{MDF_CKKy} / (MCICD+1)$ and $F_{mdf_proc_ck} > 6 * F_{SYMB}$ and $F_{mdf_hclk} \geq F_{mdf_proc_ck}$

1. F_{MDF_CKKy} represents the frequency of the clock received via MDF_CKIx and MDF_CCKy, or generated via MDF_CCKy.F_{SYMB} represents the frequency of the received symbol rate for Manchester mode.

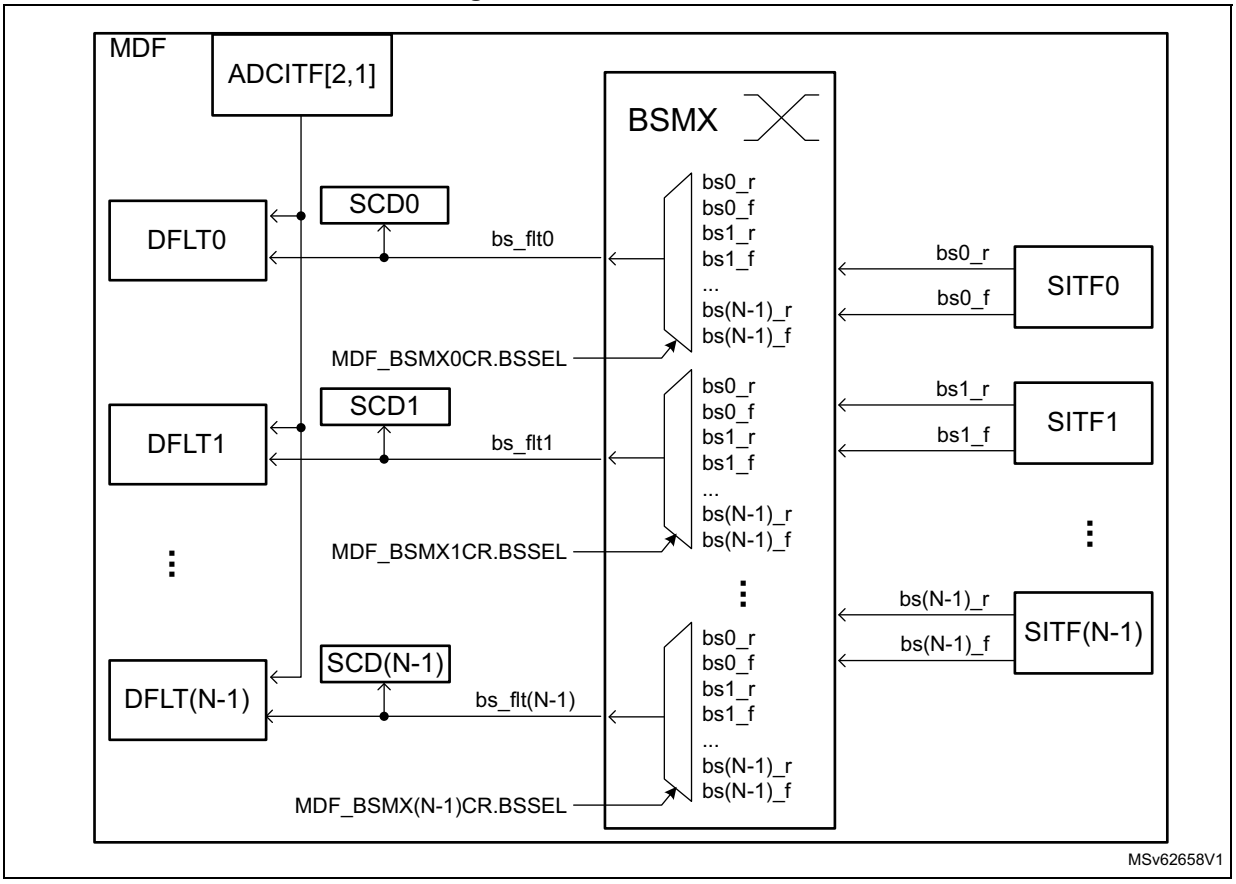
35.4.6 Bitstream matrix (BSMX)

The BSMX receives the bitstreams from all serial interfaces (SITFx) and provides the selected input to the digital filters (DFLTx) and to the short-circuit detectors (SCDx). For each filter path, any of the bitstream input can be selected.

As shown in the *Figure 322*, each SITFx block provides two bitstreams (bsx_r and bsx_f) to the BSMX.

The application to select the wanted stream via the *MDF bitstream matrix control register x (MDF_BSMXxCR)*. This selection is intended to be static.

Figure 326. BSMX overview



BSMX programming sequence example

The BSSEL[4:0] bitfield cannot be changed if the corresponding SCDx, OLDx or DFLTx is enabled. The following steps are needed to change the value of BSMX, for filter path x:

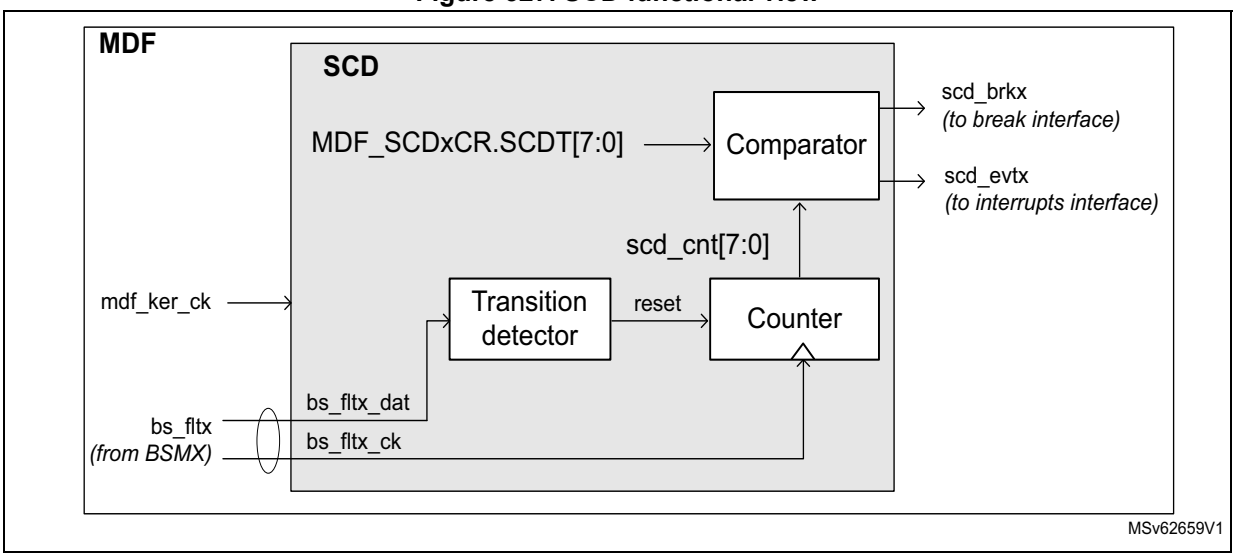
1. Set SCDEN of SCDx to 0.
2. Set DFLTEN of DFLTx to 0.
3. Set OLDEN of OLDx to 0.
4. Wait for BSMXACTIVE = 0.
5. Program BSSEL[4:0] of filter path x.
6. Set SCDEN of SCDx to 1 (optional).
7. Set DFLTEN of DFLTx to 1 (optional).
8. Set OLDEN of OLDx block to 1 (optional).

35.4.7 Short-circuit detectors (SCD)

The SCDx detects, with a very fast response time, if an analog signal reached saturated values (out-of-full scale ranges) and remained on this value for a given time.

This behavior can detect short-circuit or open-circuit errors (such as over current or over voltage). An interrupt event or/and a break signal can be generated.

Figure 327. SCD functional view

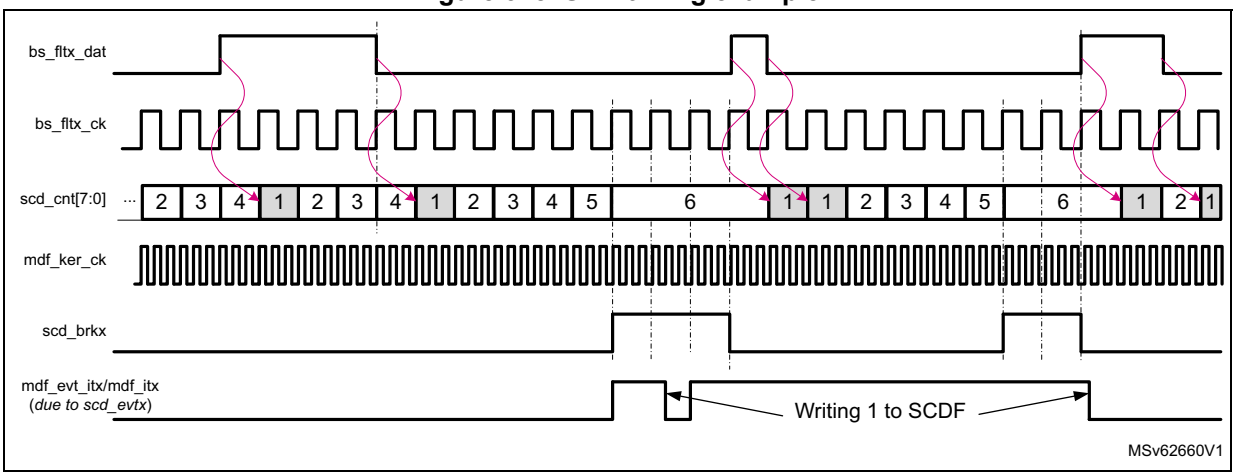


The SCDx is counting consecutive zeros or consecutive ones received from the serial interface (SITFx). A counter is restarted if there is a change in the data stream received. If this counter reaches a short-circuit threshold value (SCDT[7:0] in the *MDF short circuit detector control register x (MDF_SCDxCR)*), then a short-circuit event is invoked. Each BSMX output has its own short-circuit detector. Any BSMX output can be continuously monitored by setting the corresponding SCDEN bit to 1 in MDF_SCDxCR. Each SCD has its own threshold settings (SCDT) and its own status bit (SCDF).

The figure below shows an example where SCDT[7:0] is set to six. The break signal remains high as long as the short circuit condition is present.

No overrun event is generated when the interrupt event is pending while a new short-circuit condition is detected.

Figure 328. SCD timing example



The SCDx event generated by the SCDx block can be assigned to break output signals mdf_break[3:0]. The break signal assignment to a given short-circuit detector event is done by BKSCD[3:0] in MDF_SCDxCR. The break outputs are shared with the over-current function.

Note: SCDs cannot be used to monitor the ADC data interface (ADCITF).

SCD activation sequence example

1. Enable and configure CKGEN to generate the mdf_proc_ck.
2. Set SCDEN to 0.
3. Wait for SCDACTIVE = 0. If SCDEN was previously enabled, this phase can take two periods of mdf_hclk, and two periods of mdf_proc_ck.
4. Program BKSCD[3:0] and SCDT[7:0].
5. Set SCDEN to 1.

Note: BKSCD[3:0] and SCDT[7:0] must not be changed when SCDACTIVE = 1.

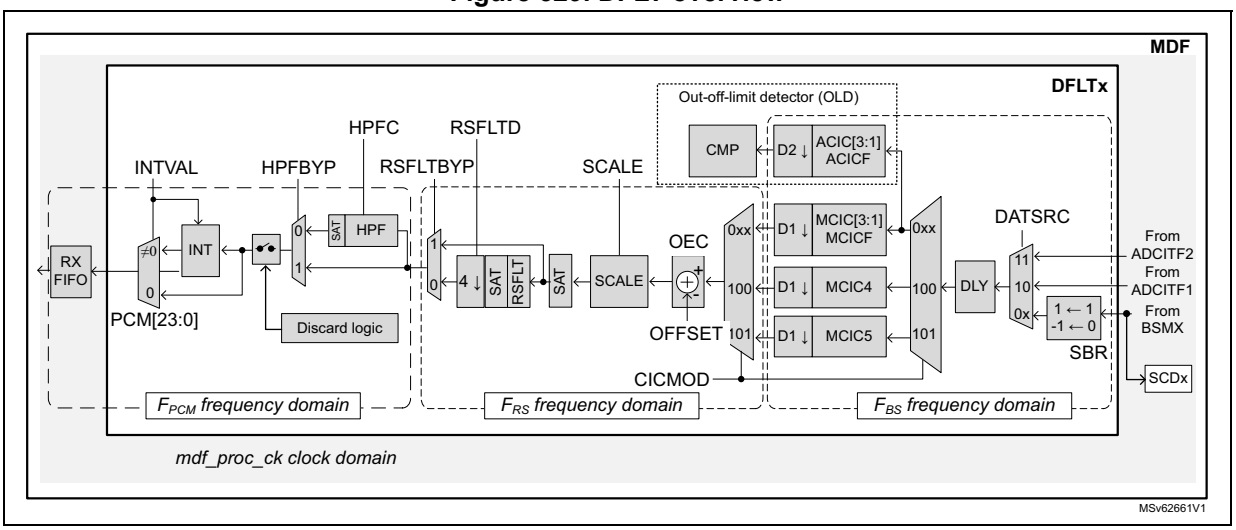
35.4.8 Digital filter processing (DFLT)

The digital filter processing includes the following sub-blocks:

- symbol remap (SBR)
- source selector
- clock skipper delay (DLY)
- CIC decimation filter that can be configured:
  - in single $Sinc^4$ or $Sinc^5$ order CIC (MCIC)
  - in two CIC filters:
    - a main filter (MCIC), high resolution
    - an auxiliary filter (ACIC), low-latency, for the out-off-limit detector (OLD)
    - both MCIC and ACIC can be configured as $Sinc^1$, $Sinc^2$, $Sinc^3$ or FastSinc
- gain control (SCALE)
- signal saturation (SAT)
- reshape filter (RSFLT)
- high-pass filter (HPF)
- integrator (INT)
- receive RXFIFO

Figure 329 shows the filter-path configuration according to CICMOD[2:0]. Several configuration bits are available to configure the digital filter to the application needs.

Figure 329. DFLT overview



Symbol remap and source selection

The symbol remap (SBR) converts the bitstream selected by the BSMX into data usable by the filter path. More especially:

- The high levels are converted into a 16-bit signed number + 1.
- The low levels are converted into a 16-bit signed number - 1.

The signal source of the digital filter can be selected via DATSRC[1:0] between the two following:

- data coming from the BSMX
- data coming from one of the ADC interfaces (ADCITF2 or 1)

Programmable micro-delay control (DLY)

The digital filter has a delay line that allows the timing adjustment of each stream with the resolution of the bitstream clock.

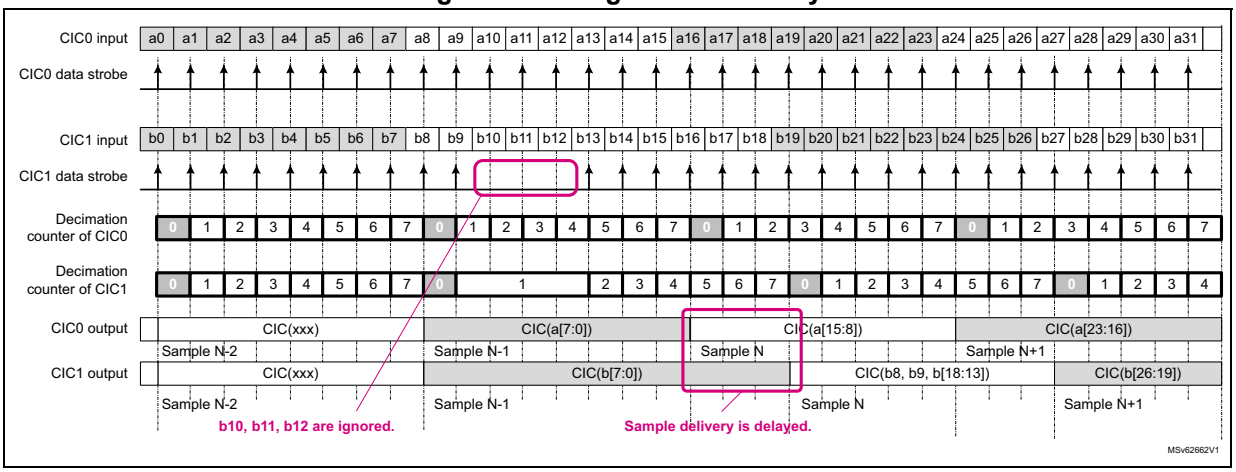
This feature is particularly helpful in the case of microphone beam forming applications where delays smaller than the final sampling rate, must be applied to the incoming stream.This feature is helpful when the MDF is synchronized with other ADF or MDF blocks for a beam forming application for example.

The delay is performed by discarding a given number of samples from the selected input stream, before samples enter into the CIC filter. This data discarding is performed by skipping a given number of data strobe, preventing the CIC filter to take into account those data.

When the wanted amount of data strobe has been skipped, the next incoming samples are strobed normally.

Figure 330 shows an example of how to apply dynamically small delay to an incoming stream. For simplification, the CIC filter performs a decimation by height in this example. CIC1 represents the CIC included in the ADF and CIC0 represents a filter from another ADF or MDF instance.

Figure 330. Programmable delay



The CIC of filter 1 (CIC1) receives a command in order to skip three incoming samples. So the input samples named b10, b11 and b12 are not processed by CIC1. As a consequence, the output sample N+1 generated by CIC0 is build from input samples a[23:16] while the sample N+1 of CIC1 is build from input samples b[26:19].

Finally, the non-skipped data stream looks delayed by three bitstream periods.

Note: When the input data strobes are skipped, the decimation counter remains frozen. As a consequence, the samples delivered by the CIC1 are a bit delayed.

Warning: It is not recommended to apply a delay bigger than the programmed decimation ratio (CIC + RSFLT decimation), especially if the MDF is programmed in interleaved acquisition mode. There is a strong risk of data misalignment inside the FIFOs.

The following steps are needed to program the amount of bitstream clock periods to be skipped:

1. Wait for SKPBF equal to 0.
2. Write SKPDLY[6:0] to the wanted number of bitstream clock periods to be skipped. The SKPBF flag goes immediately to 1, indicating that the delay value entered into SKPDLY[6:0] is under process.
   - If the corresponding DFLTx is not yet enabled (DFLTEN = 0), then the DLY logic waits for DFLTEN = 1. When the application sets DFLTEN to 1, the DLY logic starts to skip the amount of wanted data strobes.
   - If the corresponding digital filter is already enabled (DFLTEN = 1), then the DLY logic immediately starts to skip the amount of wanted data strobes.

   When the MDF skipped the amount of wanted data strobes, then SKPBF goes back to 0.
3. If the application needs to skip more data strobes, then the operation must be restarted from step 1.

The effect of the delay performed with this mechanism is cumulative as long as the MDF is enabled. In a given filter, if the application performs a D1 delay, followed by a D2 delay, then all other active filters are delayed by D1 + D2.

Data coming from ADCs can also benefit of this feature.

In interleaved acquisition mode, it is up to the application to insure that the delay applied on the different microphones is in line with the depth of the RXFIFO buffers. If the relative delay between each activated filter is less than the decimation ratio, then it costs one FIFO location.

If the interleaved acquisition mode is not used, then the delay value has no impact on the RXFIFO buffering.

Note: If SKPDLY[6:0] is written when SKPBF = 1, the write operation is ignored.

Cascaded-integrator-comb (CIC) filter

The CIC digital filters are an efficient implementation of low-pass filters, often used for decimation and interpolation. The CIC frequency response is equal to a $Sinc^N$ function, this is why they are often called Sinc filters.

The $Sinc^N$ digital filter embedded into the MDF is configurable according to the application targeted.

- For audio applications, such as speech capture from digital microphones, the application can select a high-resolution low-pass decimation filter by setting CICMOD to 100 or 101.
- If the targeted application is motor control or any other sensor capture, then the application can configure the CIC in order to offer a main filter path for acquisition (MCIC) and an auxiliary filter for fast-event generation (ACIC). This auxiliary filter can be used for example, to detect over-current conditions. This mode is selected by setting CICMOD[2:0] = 0xx.

When CICMOD[2:0] = 0xx, the following CIC filters are available:

- main filter (MCIC)
- auxiliary CIC filter (ACIC)

Both of them are configurable in FastSinc, $Sinc^1$ to $Sinc^3$.

When CICMOD[2:0] = 100, the CIC is configured into a single $Sinc^4$ and when CICMOD[2:0] = 101, the CIC is configured into a single $Sinc^5$ filter.

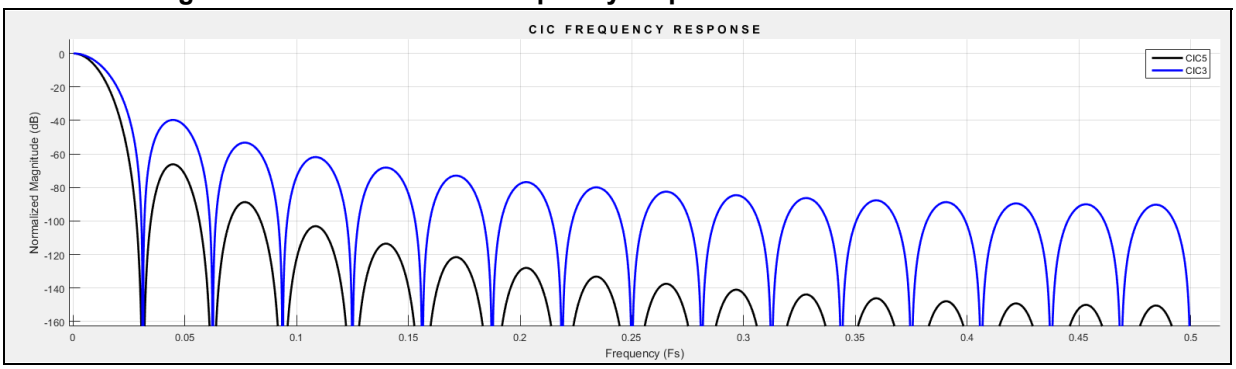
The filters have the following transfer function (impulse response in z domain):

- $Sinc^N$ filter type: $$H(z) = \left(\frac{1 - z^{-D}}{1 - z^{-1}}\right)^N$$
- FastSinc filter type: $$H(z) = \left(\frac{1 - z^{-D}}{1 - z^{-1}}\right)^2 \cdot (1 + z^{-(2 \cdot D)})$$

where N can be 1, 2, 3, 4 or 5, and D is the decimation ratio.

D is equal to MCICD+1 or ACICD+1.

Figure 331. CIC3 and CIC5 frequency response with decimation ratio = 32



CIC output data size

The size of samples delivered by the CIC (DS_{CIC}) depends on the following parameters:

- CIC order (N)
- CIC decimation ratio (D)
- data size of the input stream (DSIN)

The CIC order and decimation ratio must be programmed in order to insure that the data size does not exceed the 26-bit CIC capability.

The following formula gives the output data size (DS_{CIC}) according to the parameters above.

$$DS_{CIC} = \left(\frac{N \times Ln(D)}{Ln(2)}\right) + DS_{IN}$$

and the CIC gain is given by this formula:

$$G_{CIC} = (D)^{N}$$

The decimation ratio can be adjusted from 2 to 512 for the main CIC filter and from 1 to 32 for the auxiliary CIC filter.

The table below gives some data output size in bits for some decimation values, when the data source is a full-scale signal coming from the serial interface or from a 12-bit ADC.

Note: DS_{IN} = 1 bit for a serial bitstream but can be up to 16 bits when coming from the ADCITF.

Table 281. Data size according to CIC order and CIC decimation values

Decimation	Data size (bits) when DS_{IN} = 1 bit (data from SITFx)						Data size (bits) when DS_{IN} = 12 bits (data from ADCITF)					
	$Sinc^1$	$Sinc^2$	FastSinc	$Sinc^3$	$Sinc^4$	$Sinc^5$	$Sinc^1$	$Sinc^2$	FastSinc	$Sinc^3$	$Sinc^4$	$Sinc^5$
4	3	5	6	7	9	11	14	16	17	18	20	22
8	4	7	8	10	13	16	15	18	19	21	24	-
12	5	9	9	12	16	19	16	20	21	23	-	-
16	5	9	10	13	17	21	16	20	21	24	-	-
24	6	11	12	15	20	24	17	22	23	-	-	-
32	6	11	12	16	21	26	17	22	23	-	-	-

Table 281. Data size according to CIC order and CIC decimation values

Decimation	Data size (bits) when DS_{IN} = 1 bit (data from SITFx)						Data size (bits) when DS_{IN} = 12 bits (data from ADCITF)					
	Sinc^1	Sinc^2	FastSinc	Sinc^3	Sinc^4	Sinc^5	Sinc^1	Sinc^2	FastSinc	Sinc^3	Sinc^4	Sinc^5
64	7	13	14	19	25	-	18	24	25	-	-	-
128	8	15	16	22	-	-	19	26	-	-	-	-
256	9	17	18	25	-	-	20	-	-	-	-	-
512	10	19	20	-	-	-	21	-	-	-	-	-

The LSB part of the data provided by the CIC is not necessarily significant: it depends on the sensor performances and the ability of the CIC to reject the out-off-band noise.

The sample size at CIC output can be adjusted thanks to the SCALE block.

The table below shows the maximum allowed decimation ratio for the CIC filter, depending on the input data size. Bigger decimation ratio causes a wrap-around of the signal at CIC output, for strong input signals.

Note: The MDF cannot detect or prevent a CIC wrap-around.

Table 282. Maximum decimation ratio versus order and input data size

Filter order	Max. decimation ratio when DS_{IN} = 1 bit (SITFx)	Max. decimation ratio when DS_{IN} = 12 bits (ADCITF)	Max. decimation ratio when DS_{IN} = 16 bits (ADCITF)
Sinc^1	512	512	512
Sinc^2	512	128	32
FastSinc	512	90	22
Sinc^3	322	25	10
Sinc^4	76	11	5
Sinc^5	32	6	4

Scaling (SCALE) and saturation (SAT)

The SCALE block allows the application to adjust the amplitude of the signal provided by the CIC, by steps of 3 dB (± 0.5 dB).

The signal amplitude can be decreased by up to 8 bits (- 48.2 dB) and can be increased by up to 12 bits (+ 72.2 dB).

The gain is adjusted by the SCALE[5:0] bitfield in the *MDF digital filter configuration register x (MDF_DFLTxCICR)*.

SCALE[5:0] can be changed even if the corresponding DLFTx is enabled. During the gain transition, the signal provided by the filter is disturbed.

Due to internal resynchronization, there is a delay of some cycles of mdf_proc_ck clock between the moment where the application writes the new gain, and the moment where the gain is effectively applied to the samples. If the application attempts to write a new gain value while the previous one is not yet applied, this new gain value is ignored. Reading back SCALE[5:0] informs the application on the current gain value.

The table below shows the possible gain values.

Table 283. Possible gain values

SCALE[5:0]	Gain (dB)	SCALE[5:0]	Gain (dB)	SCALE[5:0]	Gain (dB)	SCALE[5:0]	Gain (dB)
0x20	- 48.2	0x2B	- 14.5	0x06	+ 18.1	0x11	+ 51.7
0x21	- 44.6	0x2C	- 12.0	0x07	+ 21.6	0x12	+ 54.2
0x22	- 42.1	0x2D	- 8.5	0x08	+ 24.1	0x13	+ 57.7
0x23	- 38.6	0x2E	- 6.0	0x09	+ 27.6	0x14	+ 60.2
0x24	- 36.1	0x2F	- 2.5	0x0A	+ 30.1	0x15	+ 63.7
0x25	- 32.6	0x00	0.0	0x0B	+ 33.6	0x16	+ 66.2
0x26	- 30.1	0x01	+ 3.5	0x0C	+ 36.1	0x17	+ 69.7
0x27	- 26.6	0x02	+ 6.0	0x0D	+ 39.6	0x18	+ 72.2
0x28	- 24.1	0x03	+ 9.5	0x0E	+ 42.1	-	-
0x29	- 20.6	0x04	+ 12.0	0x0F	+ 45.7	-	-
0x2A	- 18.1	0x05	+ 15.6	0x10	+ 48.2	-	-

The SAT blocks avoid having a wrap-around of the binary code when the code exceeds its maximal or minimal value.

The MDF performs saturation operations at the following levels:

- after the SCALE block (performed by the SAT block). The signal is saturated at 24 bits.
- inside the RSFLT, to insure a good filter behavior
- at the output of the HPF, to insure that the output signal does not exceed 24 bits

The SATF bit informs the application that a saturation occurred either after the SCALE, inside the RSFLT or after the HPF. In addition, an interrupt can be generated if SATIE is set to 1. As soon as a saturation is detected, the SATF flag is set to 1. It is up to the application to clear this flag in order to be able to detect a new saturation.

Those bits are in the *MDF DFLTx interrupt enable register x (MDF_DFLTxIER)* and *MDF DFLTx interrupt status register x (MDF_DFLTxISR)*.

Gain adjustment policy

To get the best MDF performances, it is important to properly adjust the gain value via SCALE[5:0].

A usual way to adjust the gain is to select the SCALE[5:0] value that gives a final signal amplitude as close as possible to the 24-bit full scale, for the maximum input signal.

A way to select the optimal gain is detailed below:

1. Check that, for the expected input signal, the data size into the CIC filter does not exceed 26 bits. This can be checked using this formula:

$$\frac{LN(SIN_{pp} \cdot D^N)}{LN(2)} < 26$$

where N represents the CIC order, D the decimation ratio and SIN_{pp} the maximum peak-to-peak amplitude of the input signal.

SIN_{pp}can take:

– a maximum peak-to-peak amplitude of 2 (± 1), for samples coming from SITF

– a maximum peak-to-peak amplitude of 4095 (+ 2047, - 2048) for samples coming from a 12-bit ADC

Example: a $Sinc^4$ can be used with a decimation ratio of 96, if the maximum input signal does not exceed ± 0.35. Indeed:

$$\frac{LN(0.7 \cdot 96^4)}{LN(2)} \sim 25.82 \text{ bits} < 26 \text{ bits}$$

2. Adjust the SCALE value.

To select the most appropriate SCALE value, the user must check if the RSFLT is used or not. If the RSFLT is used, the data size at SCALE output must not exceed 22 bits, otherwise the data size can be up to 24 bits.

The SCALE value in dB is selected using this formula:

$$SCALE_{dB} < 20 \cdot \log 10\left(\frac{2^{NB}}{SIN_{pp} \cdot D^N}\right)$$

where NB is equal to 22 if RSFLT is enabled, or 24 if RSFLT is bypassed. $SCALE_{dB}$ represents the gain value selected by SCALE[5:0].

Example: for a $Sinc^4$ with a decimation ratio of 96 and a SIN_{pp} of 0.7:

– If the RSFLT is bypassed:

$$20 \cdot \log 10\left(\frac{2^{24}}{0.7 \cdot 96^4}\right) \sim 11 \text{ dB}$$

$SCALE_{dB}$ value must be lower than - 11 dB. The closest lower value is - 12dB (SCALE[5:0] = 0x2C).

– If the RSFLT is enabled:

$$20 \cdot \log 10\left(\frac{2^{22}}{0.7 \cdot 96^4}\right) \sim 23 \text{ dB}$$

$SCALE_{dB}$ value must be lower than - 23 dB, the closest lower value is - 24.1 dB (SCALE[5:0] = 0x28).

If SCALE[5:0] is set to a higher value, then a saturation may occur. An event flag informs the user if a saturation occurred.

The table below proposes gain values for different filter configurations, when the data comes from the SITFx, according to the CIC order, and the CIC decimation ratio. This table is not exhaustive, and considers a full-scale input signal (see *Section 35.7.5* for details).

Table 284. Recommended maximum gain values versus CIC decimation ratios

CIC decimation ratio	Gain settings (dB) for configuration SITF + CICx + RSFLT (+ HPF)					Gain settings (dB) for configuration SITF + CICx (+ HPF)				
	CIC5	CIC4	CIC3	CIC2	CIC1	CIC5	CIC4	CIC3	CIC2	CIC1
8	33.6	51.7	69.7	72.2	72.2	45.7	63.7	72.2	72.2	72.2
12	18.1	39.6	60.2			30.1	51.7			
16	3.5	27.6	51.7			15.6	39.6	63.7		
20	- 6.0	21.6	48.2			6.0	33.6	60.2		
24	- 12.0	15.6	42.1	69.7		0	27.6	54.2		
28	- 20.6	9.5	36.1	66.2		- 8.5	21.6	48.2		
32	-26.6	3.5	33.6	63.7		- 14.5	15.6	45.7		
48	-	- 8.5	24.1	57.7		-	3.5	31.6	69.7	
64	-	- 20.6	15.6	51.7		-	-8.5	27.6	63.7	
128	-	-	- 2.5	39.6		-	-	9.5	51.7	
256	-	-	- 20.6	27.6		-	-	-8.5	39.6	

Reshaping filter (RSFLT)

In addition to the CIC, the MDF offers a reshaping IIR filter mainly dedicated to the audio application but also usable in other applications.

When the RSFLT is used, the sample size at its input must not exceed 22 bits.

The samples at the RSFLT output can be decimated by four or not according to the RSFLTD bit in the *MDF reshape filter configuration register x (MDF_DFLTxRSFR)*.

The RSFLT can be bypassed by setting RSFBYP to 1 in MDF_DFLTxRSFR.

The table below shows which sampling rate must be provided to the RSFLT in order to process the most common audio streams.

The RSFLT cutoff frequency (F_C) depends on the sample rates at its input (F_{RS}), and is given by the following formula:

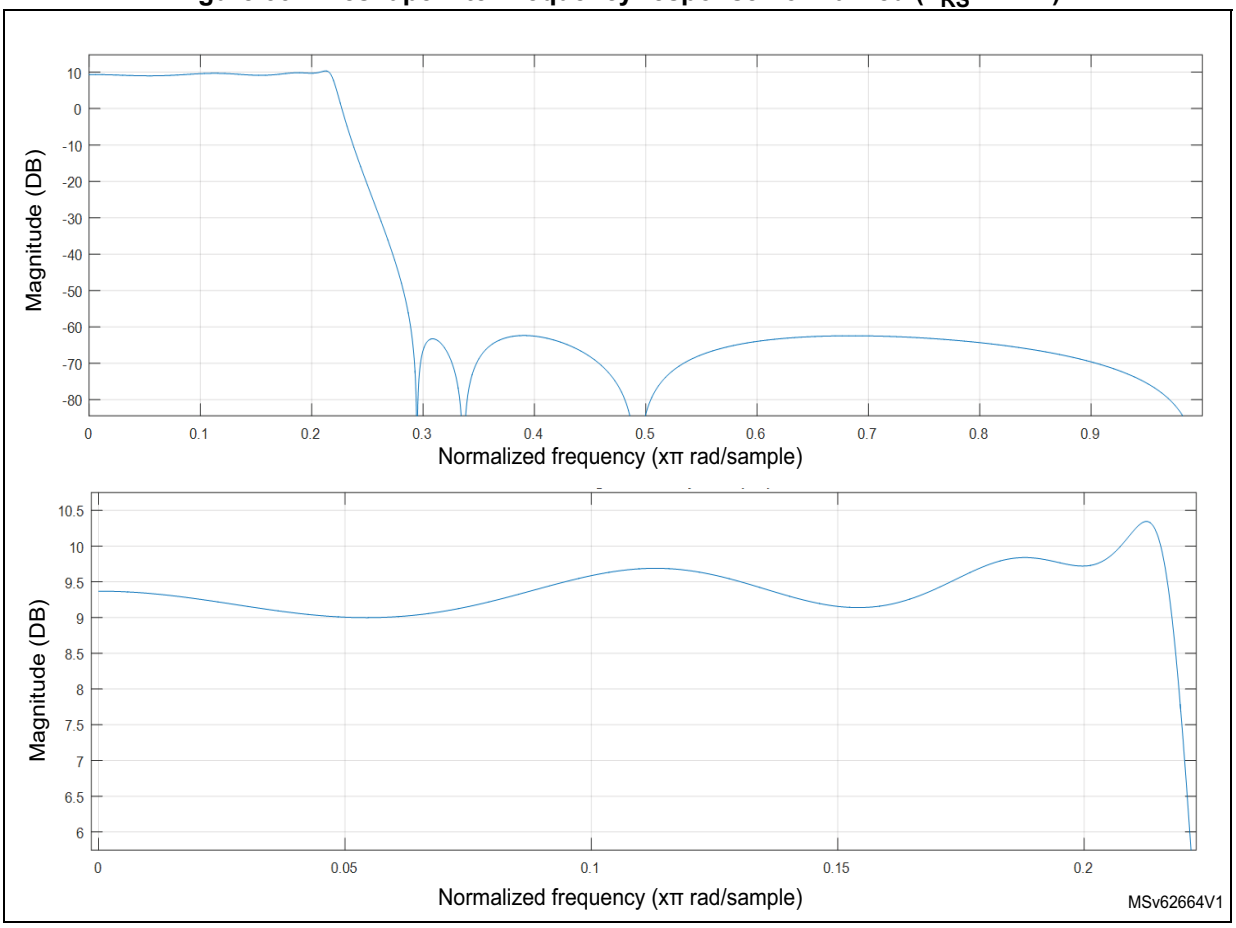
$$F_C = 0.111 \times F_{RS}$$

Table 285. Most common microphone settings

Sample rate (kHz) at RSFLT (F_{RS})	Pass band (kHz)	D2	PCM sampling rate (kHz)
32	3.55	4	8
64	7.1	4	16
128	14.2	4	32
192	21.3	4	48

The figure below shows the frequency response of the reshape filter.

Figure 332. Reshape filter frequency response normalized (F_{RS} / 2 = 1)



The RSFLT gain is about 9.3 dB, so the output data size is a little bit lower than 24 bits for a 22-bit wide input signal.

The RSFLT takes 24 clock cycles of mdf_proc_ck clock to process one sample at F_{RS}. When the RSFLT is enabled, the application must insure that the mdf_proc_ck is at least 24 times faster F_{RS}.

The RSFLT generates an event (rfovr_evt) and sets the RFOVRF flag, if the RSFLT receives a new sample while the previous one is still under processing.

When RFOVRF is set, the samples provided by the RSFLT are invalid. The application must then stop the data acquisition and provide x a faster mdf_proc_ck clock to the RSFLT.

High-pass filter (HPF)

The high-pass filter suppresses the low-frequency content from the final output data stream in case of continuous conversion mode. The high-pass filter can be enabled or disabled via HPFBYP in the *MDF reshape filter configuration register x (MDF_DFLTxRSFR)*.

The HPF is useful when there is parasitic low-frequency noise (or DC signal) in the input data source and it must be removed from the final data.

The HPF is a first order IIR filter and the cut-off frequency can be selected via HPFC[1:0] in the *MDF reshape filter configuration register x (MDF_DFLTxRSFR)*, among the following values:

- 0.000625 x F_{PCM}
- 0.00125 x F_{PCM}
- 0.00250 x F_{PCM}
- 0.00950 x F_{PCM}

Table 286. HPF 3 dB cut-off frequencies examples

HPFC	3 dB cut-off frequency for common F_{PCM} frequencies (Hz)		
	F_{PCM} = 8 kHz	F_{PCM} = 16 kHz	F_{PCM} = 48 kHz
0	5	10	30
1	10	20	60
2	20	40	120
3	76	152	456

The HPF output is saturated at 24 bits. The SATF flag is set if a sample is saturated.

Offset error compensation (OEC)

Each digital filter has its own OEC. The offset correction is performed by subtracting to the signal provided by the CIC, the OFFSET[25:0] in the *MDF offset error compensation control register x (MDF_OECxCR)*.

Due to the internal resynchronization, there is a delay of some cycles of mdf_proc_ck clock between the moment where the application writes the new offset, and the moment where the new offset value is effectively applied to the samples. If the application attempts to write a new offset value while the previous one is not yet applied, this new offset value is ignored. Reading back OFFSET[25:0] informs the application on the current offset value.

Integrator (INT)

The INT performs additional decimation and a resolution increase of data coming from the digital filter. The INT simply performs the sum of data from a digital filter for a given number of data samples from a filter.

The INT is enabled by setting INTVAL[6:0] to a value different from 0.

The amount of integrated values can be defined by INTVAL[6:0] in the range of 2 to 128.

In order to control the data width at the integrator output, the resulting data can be divided by 1, 4, 32 or 128. This feature is controlled by INTDIV[1:0].

35.4.9 Out-of-limit detector (OLD)

The OLD triggers an event when a signal reaches or crosses given maximum and minimum threshold values. The generated event can drive an interrupt or break signals (mdf_break[3:0]) when conditions are met.

The OLD can be used only if the CIC filter is configured in motor and sensing mode (CICMOD[2:0] = 0xx).

The OLD takes the input signal selected by the main filter, process it using a $Sinc^N$ or FastSinc filter (ACIC), and compares the resulting signal to programmed thresholds.

The OLD is enabled via the OLDEN bit in MDF_OLD0CR. Once enabled, the input data are continuously monitored. There is no need to have the DFLTx enabled for using the OLD function.

The MDF offers a high- and low-threshold register that are compared with given data values.

The application can generate an event if the signal is inside or outside the boundary defined by those two thresholds. This behavior is controlled via THINB in the MDF_OLD0CR.

If the application only wants to generate an event when the input signal is higher than OLDTHH, then THINB and OLDTHL must both be cleared to 0.

If the application only wants to generate an event when the input signal is lower than OLDTHH, then THINB must be set to 1 and OLDTHL must be cleared to 0.

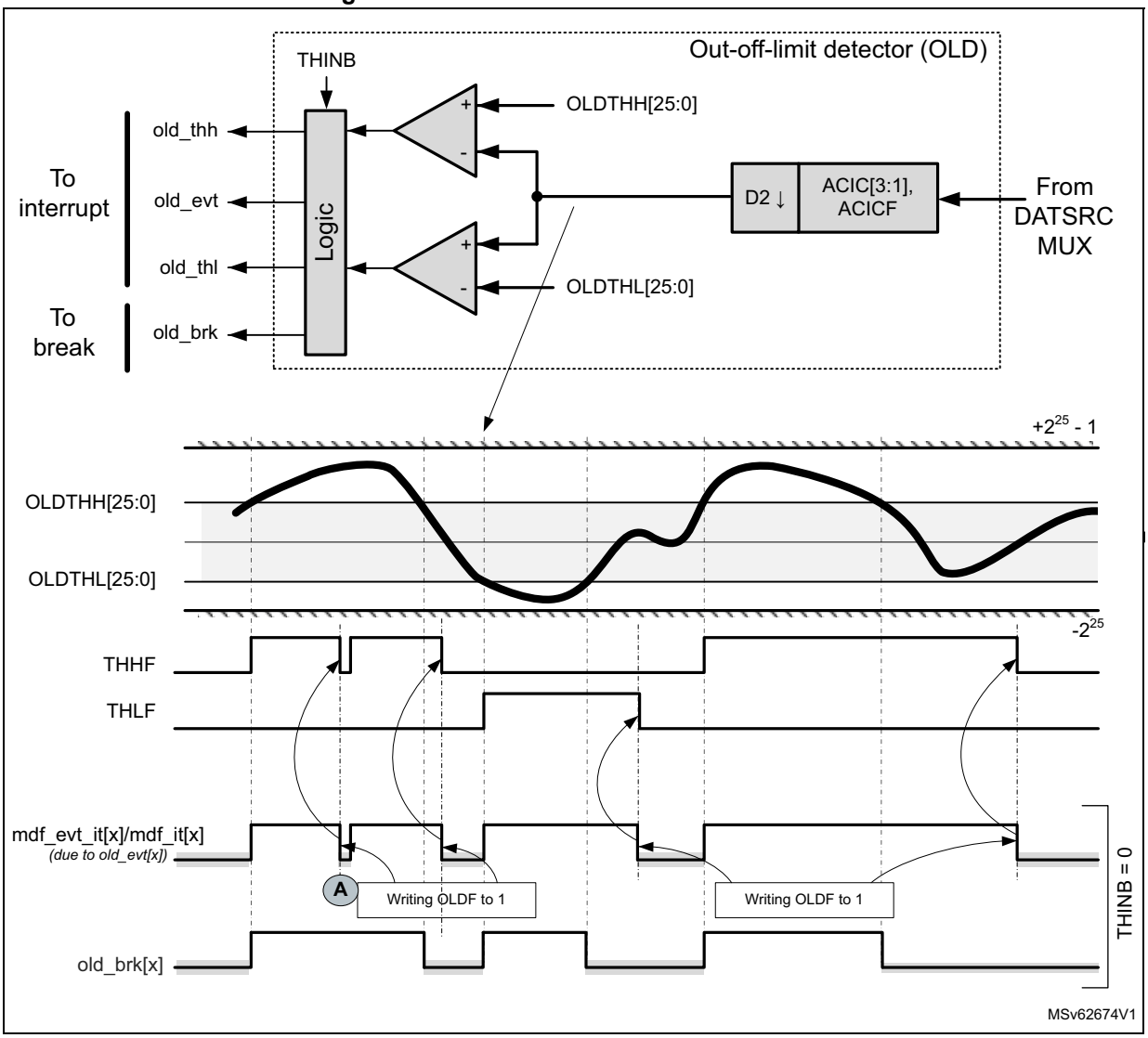
Note: It is not recommended to set a OLDTHL to a value bigger than OLDTHH.

The response time of the OLD depends on several parameters, the most important are listed hereafter:

- sampling rate frequency used by the external sensor
- ACIC decimation ratio
- ACIC order

The OLD can be used for over-current detection but also as current limiter if the PWM signal is generated by a timer receiving a old_brk signal. Generally, to get a fast response time for over-current detection, it is recommended to use the ACIC with lowest order as possible and the minimum decimation ratio. FastSinc is also a good choice for over-current detection. The application must perform a trade-off between the filter resolution and the response time.

Figure 333. Out-of-limit detector thresholds



Status flags are available in order to inform the application that an OLD event is detected. Latched events are cleared by writing 1 into the OLDF bit.

As shown in *Figure 333*, when THINB = 0, the interrupt signal remains active as long as the signal is outside the gray area. At position A, the application clears the interrupt, but the interrupt is re-asserted because the signal is still outside the gray area. When the application clears OLDF, THHF and THLF are cleared as well.

An OLD event can be assigned to one or several break output signals (mdf_break[3:0]). The break signal assignment to a given OLD event is done by BKOLD[3:0] in the *MDF out-of limit detector control register x (MDF_OLDxCR)*.

Note: The generation of break signals is independent from the interrupts generation.

OLD activation sequence example

1. Enable and configure CKGEN.
2. Set OLDEN to 0.

3. Wait for OLDACTIVE = 0. If OLDEN was previously enabled, this phase can take two periods of mdf_hclk and two periods of mdf_proc_ck.
4. Program BKOLD[3:0], ACICN[1:0], ACICD[4:0], THINB,OLDTHL[25:0] and OLDTHH[25:0].
5. Set OLDEN to 1.

35.4.10 Digital filter acquisition modes

The MDF offers the following modes to perform a data capture:

- asynchronous continuous acquisition mode
- asynchronous single-shot acquisition mode
- synchronous continuous acquisition mode
- synchronous single-shot acquisition mode
- window continuous acquisition mode
- synchronous snapshot acquisition mode

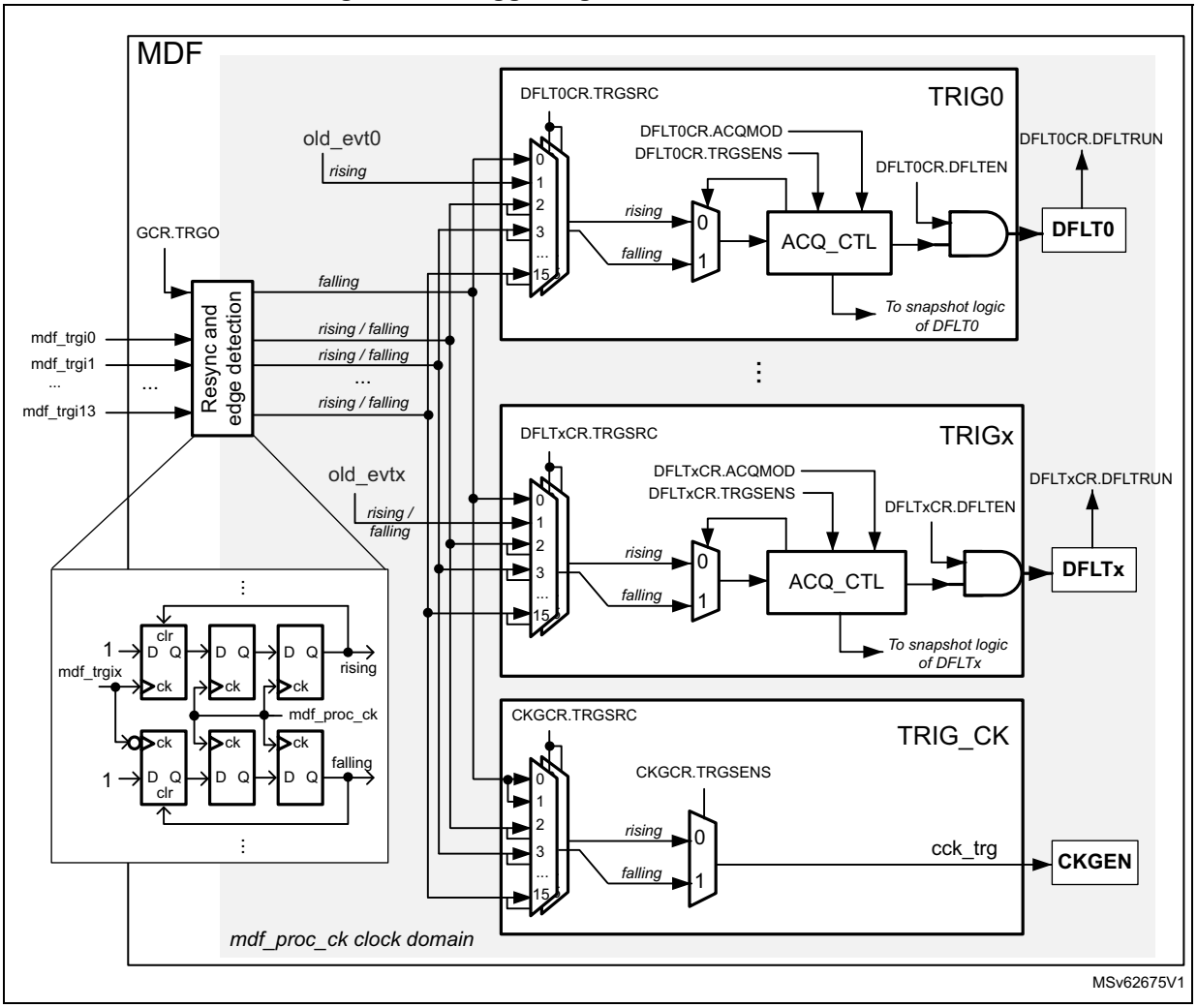
For each filter, one of these modes can be selected independently.

Note: To perform a data capture, the filters, the interfaces providing the data (SITFx or ADCITF) and the CKGEN must be enabled. If needed, MDF_CCK0 or MDF_CCK1 must be enabled as well.

The filter can be stopped immediately when DFLTEN is set to 0. This resets the filters and flushes the RXFIFO of the corresponding filter path. The DFLTACTIVE flag also goes back to 0 when the RXFIFO and the filters are reset.

The figure below shows a simplified view of the trigger logic available for each filter and for the clock generator.

Figure 334. Trigger logic for DFLT and CKGEN



A block common to all TRIG blocks performs the rising and falling edges detection and the resynchronization of the input triggers to the mdf_ker_ck clock domain. This implementation allows the application to use triggers with pulse width smaller than the mdf_ker_ck period.

In synchronous modes, the TRIG block offers the possibility to select one of the following trigger sources:

- mdf_trgi[13:0] signals (refer to *Table 277* for the trigger connections)
- TRGO bit in MDF_GCR
- OLD event of each filter path

The edge sensitivity can also be selected, except for TRGO and OLD events.

Asynchronous continuous acquisition mode

This mode allows the application to start a continuous acquisition on one or several filters by simply writing their DFLTEN bits to 1.

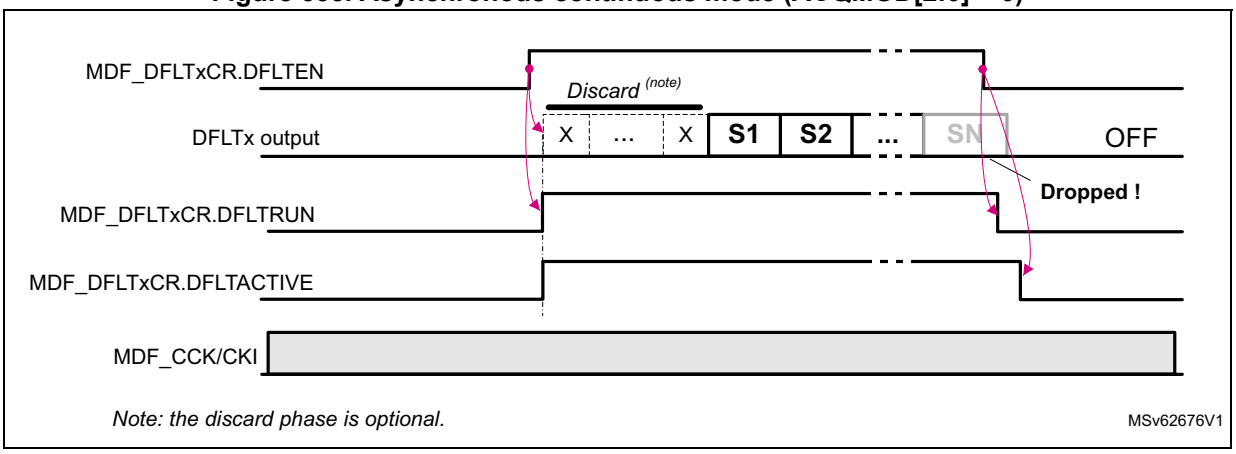
The Asynchronous continuous acquisition mode is selected when ACQMOD[2:0] = 0.

The sequence below shows the most important programming steps (assuming that DFLTEN bits of the filters are set to 0):

1. Configure and enable the clock generator (CKGEN) so that the mdf_proc_ck frequency is compatible with the targeted application (see examples in *Table 290*).
2. Enable the CKGEN (CKGDEN = 1) and, if needed, enable the MDF_CCK0 and MDF_CCK1 clocks.
3. Program the filter configuration and set the ACQMOD[2:0] to 0.
4. Set to 1 the SITFEN bit of the requested data interfaces.
5. Before setting DFLTEN to 1, wait for DFLTACTIVE = 0: it insures that the previous filter deactivation sequence terminated properly.
6. When DFLTEN is set to 1 for the filters to enable, the acquisition sequence starts immediately.

The figure below shows a simplified example of the samples generated by the DFLTx.

Figure 335. Asynchronous continuous mode (ACQMOD[2:0] = 0)



Note: The acquisition can be stopped by setting DFLTEN back to 0. This resets the filter and flushes the RXFIFO, so the samples located into the RXFIFO are lost. The ongoing DMA transfer is properly terminated. DFLTACTIVE goes back to 0 when the filter chain is reset and the RXFIFO flushed.

Asynchronous single-shot acquisition mode

This mode allows the application to start the acquisition of one sample on one or several filters by simply writing their DFLTEN bits to 1.

The Asynchronous single-shot acquisition mode is selected when ACQMOD[2:0] = 001.

The sequence below shows the most important programming steps (assuming that DFLTEN bits of the filters are cleared to 0):

1. Configure and enable the clock generator (CKGEN), so that the mdf_proc_ck frequency is compatible with the targeted application (see examples in *Table 286*).
2. Enable the CKGEN (CKGDEN = 1) and, if needed, enable the MDF_CCK0 and MDF_CCK1 clocks.
3. Program the filter configuration and set the ACQMOD[2:0] to 001.
4. Set to 1 the SITFEN bit of the requested data interfaces.

5. Before setting DFLTEN to 1, wait for DFLTACTIVE = 0: it insures that the previous filter deactivation sequence terminated properly.
6. When DFLTEN is set to 1 for the filters to enable, each selected filter provides one data to the RXFIFO and stops the acquisition.

To trigger a new acquisition, for each filter, the application must:

1. Check that the previous acquisition is completed, by waiting that DFLTRUN = 0.
2. Set again DFLTEN to 1.

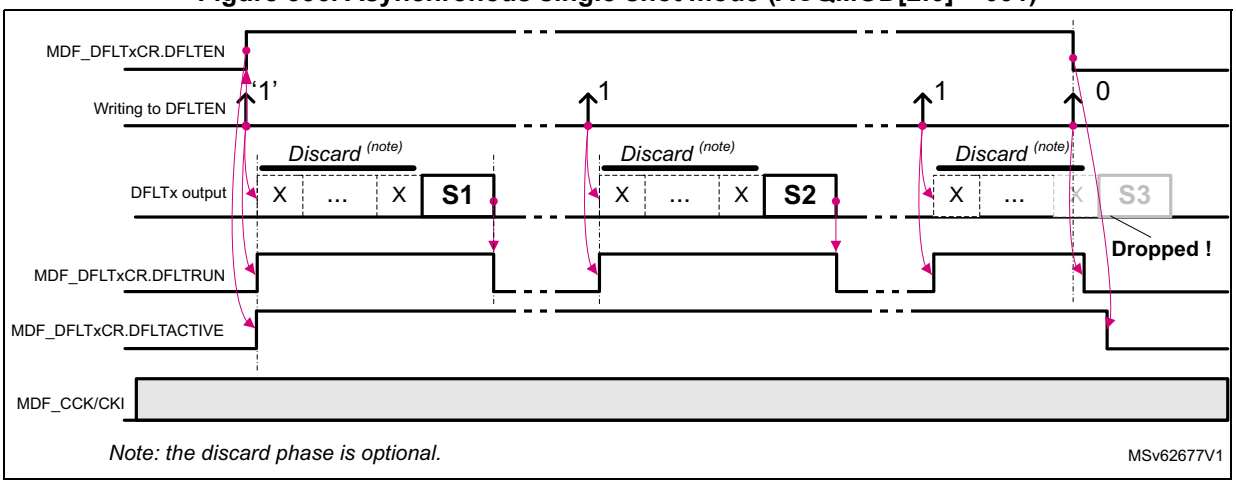
This sequence can be repeated every time a new data must be converted.

As shown in the *Figure 336*, every time DFLTEN is set to 1, an acquisition sequence is triggered. The first samples provided by the filter can be discarded if needed. At the end of each conversion, the decimation counters and filter taps are reset, and the filter is ready to start a new conversion.

If the DFLTEN is set to 0 while an acquisition is ongoing, the ongoing conversion is stopped (in the example, S3 is lost). This situation can be avoided with the following steps:

1. Wait for DFLTRUN = 0.
2. Read the sample from the RXFIFO.
3. Set DFLTEN to 0.

Figure 336. Asynchronous single-shot mode (ACQMOD[2:0] = 001)



Note: The acquisition can be stopped by setting DFLTEN back to 0. This resets the filter and flushes the RXFIFO, so the samples located into the RXFIFO are lost. The ongoing DMA transfer is properly terminated. DFLTACTIVE goes back to 0 when the filter chain is reset and the RXFIFO flushed.

Synchronous continuous acquisition mode

This mode allows the application to start a continuous acquisition on one or several filters by using the following trigger sources:

- one of the mdf_trg[13:0] signals
- OLD event of the corresponding filter
- TRGO bit

The Synchronous continuous acquisition mode is selected when ACQMOD[2:0] = 010.

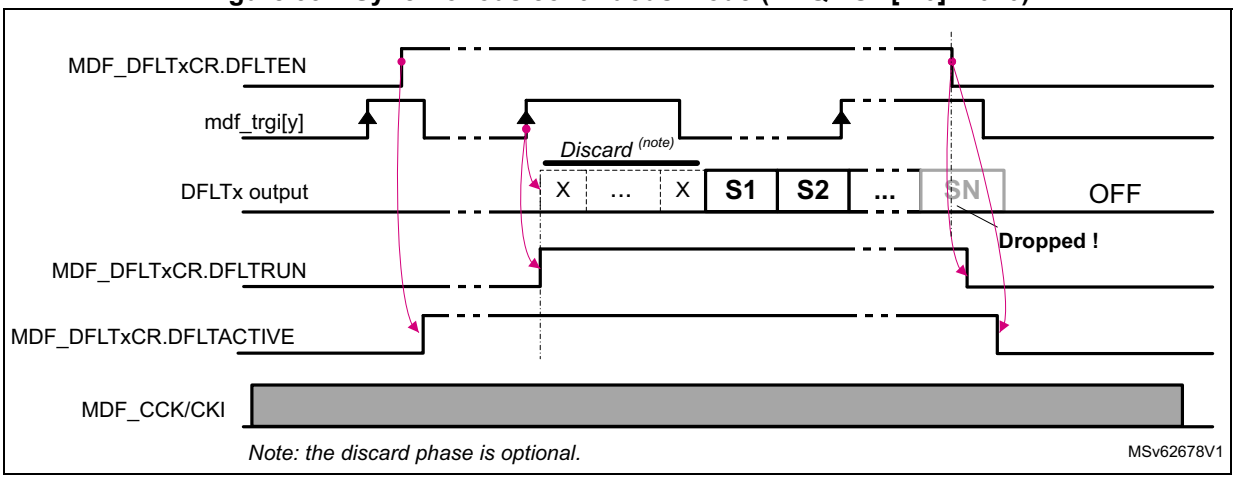
The sequence below shows the most important programming steps (assuming that DFLTEN bits of the filters are set to 0):

1. Configure and enable the clock generator (CKGEN), so that the frequency of mdf_proc_ck clock is compatible with the targeted application (see examples in *Table 286*).
2. Enable the CKGEN (CKGDEN = 1) and, if needed, enable the MDF_CCK0 and MDF_CCK1 clocks.
3. Program the filter configuration, and set the ACQMOD[2:0] to 010.
4. Set to 1 the SITFEN bit of the requested data interfaces.
5. Select the proper trigger source and sensitivity for each filter.
6. Before setting DFLTEN to 1, wait for DFLTACTIVE = 0: it insures that the previous filter deactivation sequence terminated properly.
7. Set DFLTEN to 1 for the filters to enable.
8. When the trigger condition is met, the filters start the acquisition.

The TRGSENS bit allows the selection of the trigger edge (rising or falling). The trigger is ignored if an acquisition is ongoing or if DFLTEN is set to 0.

The figure below shows a simplified example where the trigger logic is sensitive to a rising edge trigger (TRGSENS = 0). The first rising edge of the trigger signal is ignored because DFLTEN = 0. The next rising edge is taken into account and starts the acquisition. All other rising edges are ignored. The trigger logic is re-initialized when DFLTRUN goes back to 0.

Figure 337. Synchronous continuous mode (ACQMOD[2:0] = 010)



Note: The acquisition can be stopped by setting DFLTEN back to 0. This resets the filter and flushes the RXFIFO, so the samples located into the RXFIFO are lost. The ongoing DMA transfer is properly terminated. DFLTACTIVE goes back to 0 when the filter chain is reset and the RXFIFO flushed.

Synchronous single-shot acquisition mode

This mode allows the application to start a single acquisition on one or several filters by using the following trigger sources:

- one of the mdf_trg[13:0] signals
- OLD event of the corresponding filter
- TRGO bit

The Synchronous single-shot acquisition mode is selected when ACQMOD[2:0] = 011.

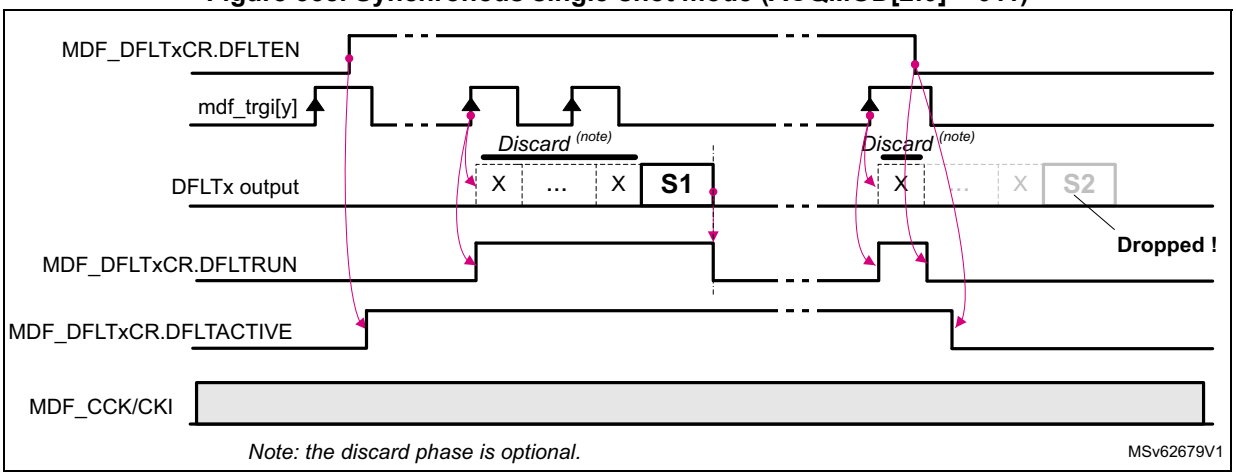
The sequence below shows the most important programming steps (assuming that DFLTEN bits of the filters are set to 0):

1. Configure and enable the clock generator (CKGEN), so that the frequency of mdf_proc_ck clock is compatible with the targeted application (see examples in *Table 286*).
2. Enable the CKGEN and, if needed, enable the MDF_CCK0 and MDF_CCK1 clocks.
3. Program the wanted filter configuration and set the ACQMOD[2:0] to 011.
4. Set to 1 the SITFEN bit of the requested data interfaces.
5. Select the proper trigger source and sensitivity for each filter.
6. Before setting DFLTEN to 1, wait for DFLTACTIVE = 0: it insures that the previous filter deactivation sequence has been properly terminated.
7. Set DFLTEN to 1 for the filters to enable.
8. Then when the trigger condition is met, the filters start the acquisition and provide one data to the RXFIFO, then the filters are ready to accept a new trigger.

TRGSENS allows the selection of the trigger edge (rising or falling). The trigger is ignored if an acquisition is ongoing or if DFLTEN is set to 0.

The figure below shows a simplified example where the trigger logic is sensitive to a rising edge trigger (TRGSENS = 0). Every time a trigger rising edge is detected with DFLTEN = 1, an acquisition sequence is triggered. The first samples provided by the filter can be discarded if needed. At the end of each conversion, the decimation counters and filter taps are reset. DFLTRUN is set to 0 and the filter is ready to start a new conversion.

Figure 338. Synchronous single-shot mode (ACQMOD[2:0] = 011)



Note: The acquisition can be stopped by setting DFLTEN back to 0. This resets the filter and flushes the RXFIFO, so the samples located into the RXFIFO are lost. The ongoing DMA transfer is properly terminated. DFLTACTIVE goes back to 0 when the filter chain is reset and the RXFIFO flushed.

Figure 338 shows a case where DFLTEN is cleared to 0 while an acquisition is ongoing: thus the sample S2 is lost. This situation can be avoided with the following steps:

1. Wait for DFLTRUN = 0.
2. Read the sample from the RXFIFO.
3. Clear DFLTEN to 0.

Note: The ongoing DMA transfer is properly terminated.

Window continuous acquisition mode

This mode allows the application to start or stop a continuous acquisition on one or several filters controlled by consecutive edges of one of the following trigger sources:

- one of the mdf_trg[13:0] signals
- TRGO bit

The window continuous acquisition mode is selected when ACQMOD[2:0] = 100.

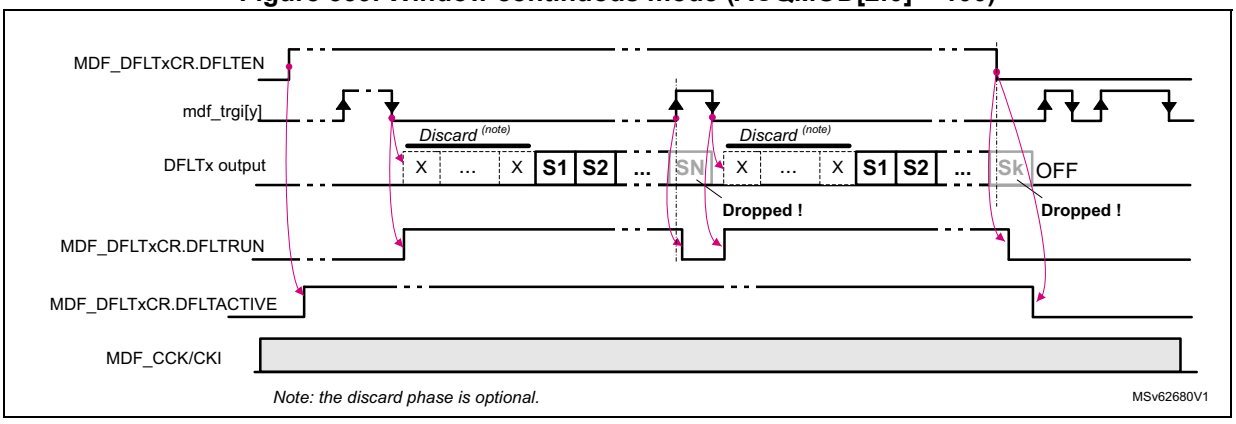
The sequence below shows the most important programming steps (assuming that DFLTEN bits of the filters are set to 0):

1. Configure and enable the clock generator (CKGEN), so that the frequency of mdf_proc_ck clock is compatible with the targeted application (see examples in *Table 286*).
2. Enable the CKGEN and, if needed, enable the MDF_CCK0 and MDF_CCK1 clocks.
3. Program the wanted filter settings and set the ACQMOD[2:0] to 100.
4. Set to 1 the SITFEN bit of the requested data interfaces.
5. Select the proper trigger source and sensitivity for each filter.
6. Before setting DFLTEN to 1, wait for DFLTACTIVE = 0: it insures that the previous filter deactivation sequence has been properly terminated.
7. Set DFLTEN to 1 for the filters to enable.
8. If TRGSENS = 0, the acquisition starts on trigger rising edge and stops on trigger falling edge. If TRGSENS = 1, the acquisition starts on trigger falling edge and stops on trigger rising edge.

Note: The acquisition may restart if the trigger condition becomes again active.

Figure 339 shows a simplified example of window continuous acquisition mode, with TRGSENS = 1. Once DFLTEN is set to 1, the MDF waits for a falling edge on the selected trigger input. When the trigger condition is met, DFLTRUN goes to 1 and the acquisition starts. The acquisition stops if the MDF detects a rising edge on the selected trigger input. If DFLTEN is still set to 1, the MDF waits again for a falling edge on the selected trigger input.

Figure 339. Window continuous mode (ACQMOD[2:0] = 100)



Note: The acquisition can be stopped by setting DFLTEN back to 0. This resets the filter and flushes the RXFIFO, so the samples located into the RXFIFO are lost. The ongoing DMA

transfer is properly terminated. DFLTACTIVE goes back to 0 when the filter chain is reset and the RXFIFO flushed.

Synchronous snapshot acquisition mode

In the acquisition modes described on previous sections, the application must wait for filters settlings time or the end of conversion before getting a new valid sample. For applications very critical in latency, the last valid sample can be get immediately, with an information on the age of this sample by using the snapshot acquisition mode.

In snapshot mode, the DFLTx is continuously acquiring the samples, but the processed samples are not stored into the RXFIFO. When a trigger occurs, the last valid sample, the current decimation value of MCIC and the sample counter of the INT are stored in MDF_SNPS0DR.

The possible trigger sources are the following:

- one of the mdf_trg[13:0] signals
- OLD event of the corresponding filter
- TRGO bit

The Synchronous snapshot acquisition mode is selected when ACQMOD[2:0] = 101.

The sequence below shows the most important programming steps (assuming that DFLTEN bits of the filters are set to 0):

1. Configure and enable the clock generator (CKGEN), so that the frequency of mdf_proc_ck clock is compatible with the targeted application (see examples in *Table 286*).
2. Enable the CKGEN and, if needed, enable the MDF_CCK0 and MDF_CCK1 clocks.
3. Program the wanted filter settings and set the ACQMOD[2:0] to 101. FTH is usually cleared to 0 to receive an interrupt as soon as the RXFIFO is not empty.
4. Set the SITFEN bit of the requested data interfaces to 1.
5. Select the proper trigger source and sensitivity for each filter.
6. Before setting DFLTEN to 1, wait for DFLTACTIVE = 0: it insures that the previous filter deactivation sequence has been properly terminated,
7. Set DFLTEN to 1 for the filters to enable.

When the trigger condition is met,

1. The MDF stores the last valid data provided by the corresponding filter, the value of the MCIC decimation counter, and the counter value of INT if selected by SNPSFMT into a shadow register.
2. The snps_evt is activated, indicating that a data is ready. If SSDRIE is set to 1, the MDF requests the AHB clock.
3. When the AHB clock is available the shadow register is stored in *MDF snapshot data register x (MDF_SNPSxDR)* and an interrupt is generated.
4. In the interrupt sub-routine, the application must do the following:
   a) Read the data located in MDF_SNPSxDR.
   b) Clear the SSDRF flag by writing it to 1.

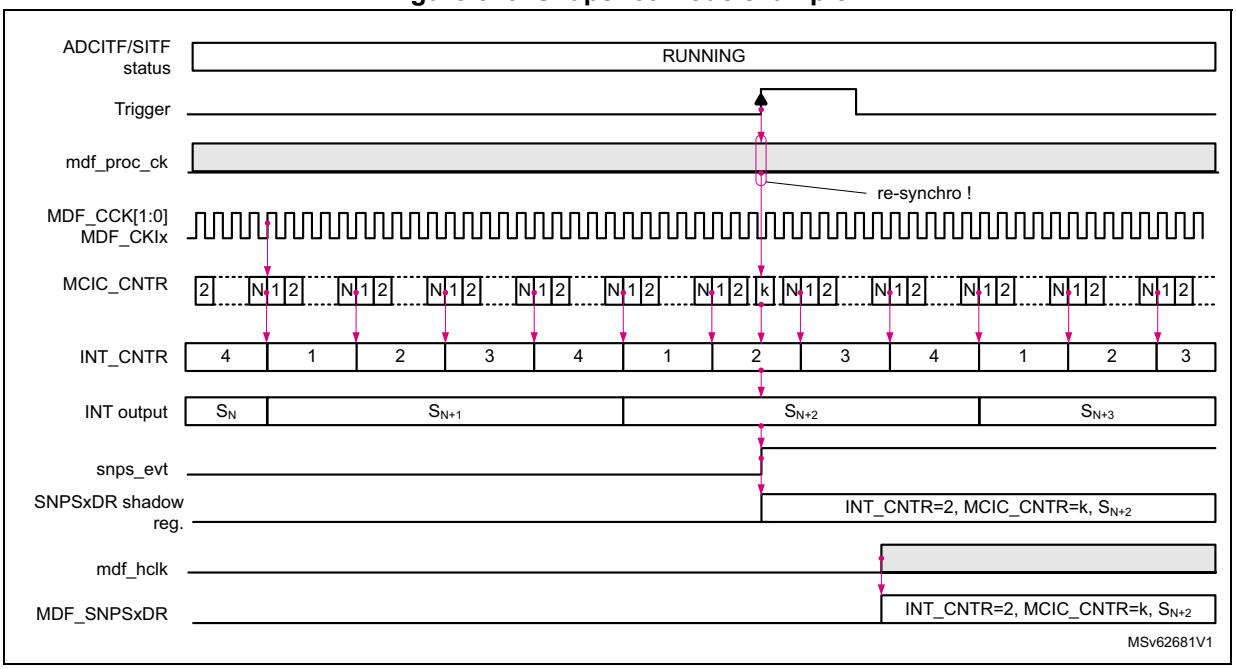
As shown in the *Figure 340*, in snapshot mode, when the trigger event occurs, it is resynchronized with the processing clock (mdf_proc_ck), the decimation counter of the MCIC (MCIC_CNTR), the integrator counter (INT_CNTR) and the last valid sample (S_{N+2}), are stored into a shadow register, and the snps_evt event is generated. When the AHB

clock is available, an interrupt is generated and the shadow register is copied in MDF_SNPSxDR. The application can read immediately the sample and the timing information.

The SSOVRF overrun flag is also available, allowing the application to check if an overrun condition occurs. A snapshot overrun condition is detected when the SSDRF flag is set to 1 while a new trigger event occurs.

In that case, the new trigger event is not taken into account. The application must clear the SSOVRF flag.

Figure 340. Snapshot mode example



Note: When an activated filter did not receive a data strobe, the filter is frozen. If the interface providing the stream is not enabled (SITF or ADCITF), the filter is frozen. Once a DFLT x is activated, CIC, HPF, IIR and INT are reinitialized. For each filter a status bit indicates if the filter is currently running or not.

Starting several filters synchronously

To start the acquisition of several filters synchronously, the following sequence must be performed (assuming that DFLTEN is set to 0):

1. Enable the CKGEN and, if needed, enable the MDF_CCK0 and MDF_CCK1 clocks.
2. Set the SITFEN bit of the requested data interfaces to 1.
3. For each filter, set the acquisition mode to synchronous (ACQMOD[2:0] = 01x).
4. For each filter, set TRGSRC[3:0] to 0 (TRGO is selected).
5. For each filter, set TRGSENS to 0 (rising edge).
6. For each filter, set DFLTEN to 1.
7. Read TRGO bit until it is read to 0.
8. Set TRGO to 1. Then the acquisition sequence for all selected filters starts immediately.

To trigger a new acquisition (in case of single-shot), the application must do the following:

1. Check that the previous acquisition is completed, by waiting DFLTRUN = 0.
2. Read TRGO until it is read to 0.
3. Set again TRGO to 1.

Discarded samples

The MDF offers the possibility to program the amount of samples to be discarded after each restart:

- to avoid capturing samples affected by the impulse response of the filter
- to delay the acquisition of filters by a specific amount of samples

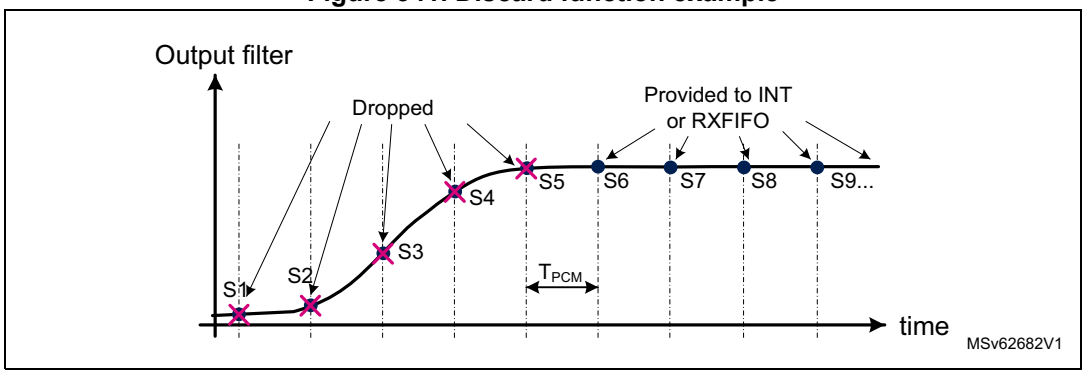
The discard function is controlled via NBDIS[7:0] as follows:

- When NBDIS[7:0] = 0, the discard function is disabled.
- When NBDIS[7:0] ≠ 0, the discard function is activated in one of the following condition:
  - when the DFLTEN bit goes to 1
  - every time an acquisition is started in (A)synchronous single-shot modes

Refer to *Figure 335* to *Figure 339*, *Figure 342* and *Figure 343*.

In the example shown in the figure below, the discard function is used to drop the first five samples provided by the digital filter (S1 to S5). The first sample transfered to the RXFIFO (or INT block if enabled) is S6.

Figure 341. Discard function example



Warning: All filters working in interleaved DMA mode must have the same NBDIS[7:0] value.

Data interface activation

The data interfaces are enabled by setting the corresponding SITFEN or ADCITFEN bit to 1. Once the digital filter is enabled, it receives the serial data from the external ΣΔ modulator or parallel internal data sources (ADC).

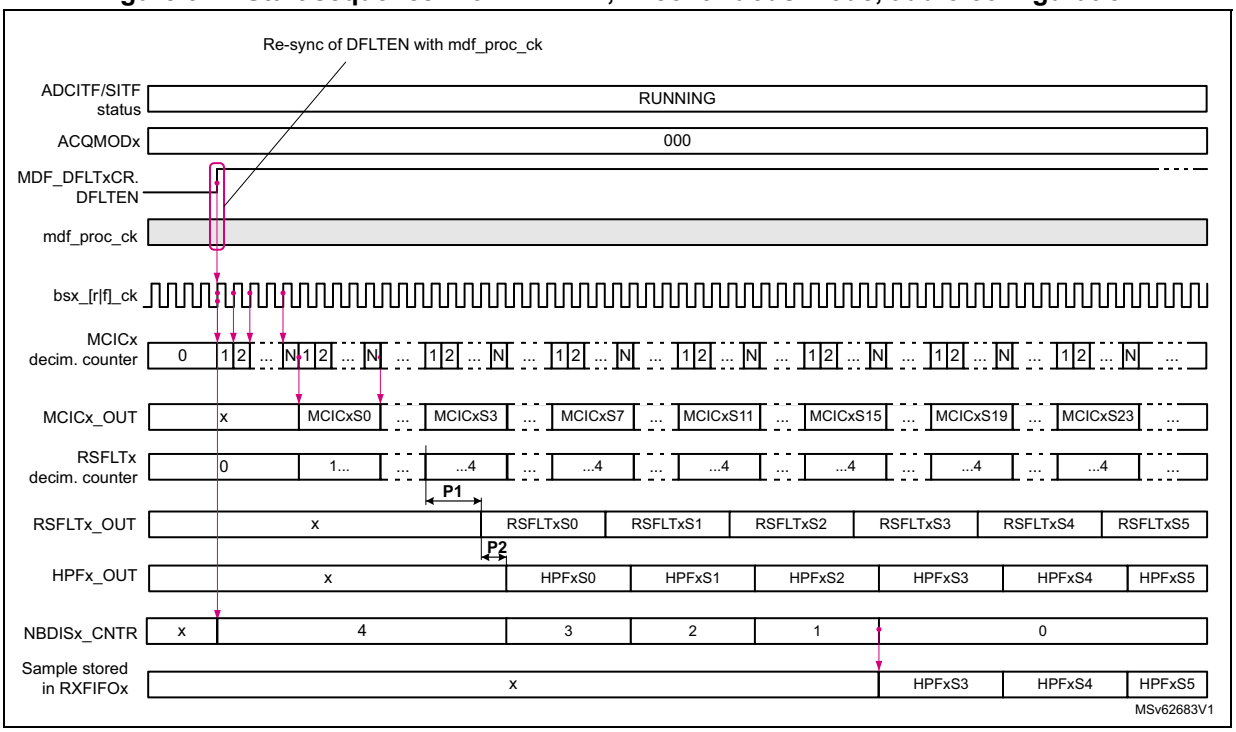
35.4.11 Start-up sequence examples

Figure 342 details an acquisition sequence start of a digital filter triggered by DFLTEN (ACQMOD[2:0] = 0), with NBDIS[7:0] = 3 (three samples to discard before acquisition).

The DFLTx is configured for audio application: MCIC, RSFLT and HPF activated.The data interface (SITFx or ADCITF) is assumed to be already activated.

Note: NBDIS[7:0] is set on purpose to a low value to simplify the drawing.

Figure 342. Start sequence with DFLTEN, in continuous mode, audio configuration



The DFLTEN bit is re-sampled into the MDF processing clock domain. When DFLTEN is detected high, the filter chain is enabled and the decimation counter of the MCIC filter is incremented at the rate of the bitstream clock.

When the MCIC decimation counter reaches its programmed value N, a sample is available for the RSFLT.

The RSFLT processes all the samples provided by the MCIC and delivers a sample to the HPF every time it processes four samples (decimation by 4) The RSFLT needs up to 24 cycles of mdf_proc_ck clock before delivering a sample (P1).

The HPF processes all the samples provided by the RSFLT, but the NBDIS function prevents the data writing in the RXFIFO as long as NBDIS_CNTR does not reach 0. This counter is decremented every time the HPF delivers a sample.

When NBDIS_CNTR reaches 0, the samples provided by the HPF are stored into the RXFIFO.

The example shown in *Figure 343* is based in a motor-control filter configuration:

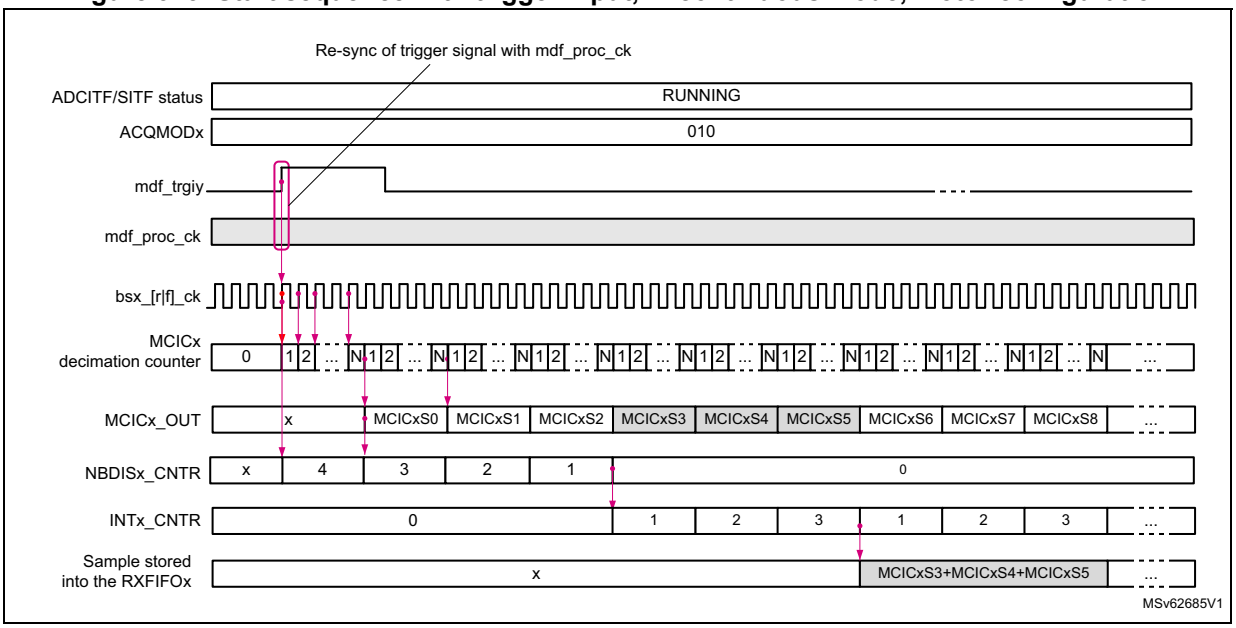
- MCIC and INT are enabled.
- SFLT and HPF are disabled.
- ACQMOD[2:0] is set to 1 and uses a mdf_trgiy input as trigger.
- NBDIS[7:0] is set to 3.
- INTVAL[6:0] is set to 2.

The mdf_trgiy input signal is re-sampled into the MDF processing clock domain to avoid any metastability issues. When the trigger condition is met (rising edge), the decimation counter of the MCIC filter is incremented at the rate of the bitstream clock.

When the MCIC decimation counter reaches its programmed value N, a sample is available. The NBDIS function prevents the samples writing in the INT as long as NBDIS_CNTR does not reach 0. This counter is decremented every time the MCIC delivers a sample.

When NBDIS_CNTR reaches 0, the new samples are provided to the INT, that performs the integration of three consecutive samples and stores them into the RXFIFO.

Figure 343. Start sequence with trigger input, in continuous mode, motor configuration

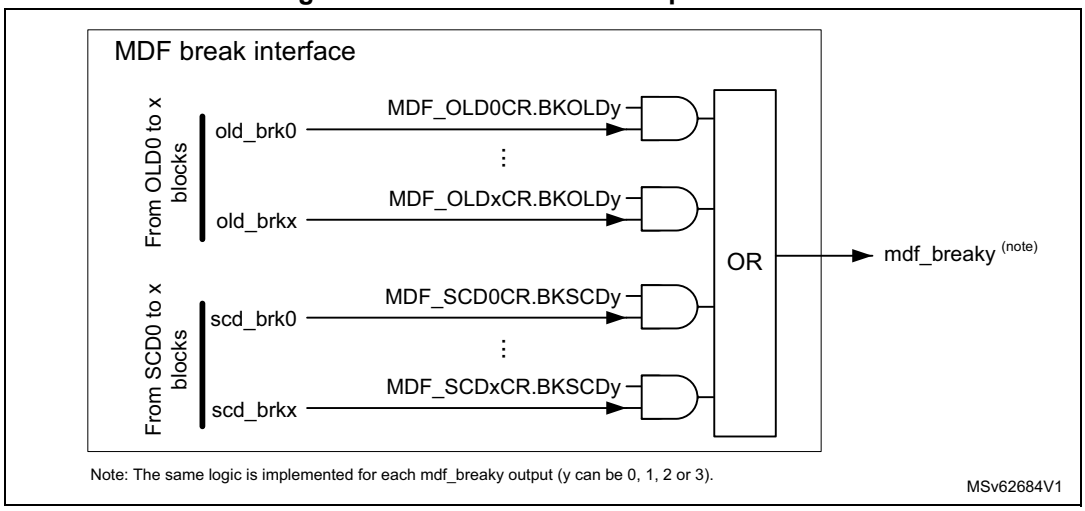


35.4.12 Break interface

The break interface merges the break events coming from the OLDx and SDCx blocks into four break signals connected to various peripherals of the product (see *Table 278* for details).

As shown in *Figure 344*, several blocks can share the same break line. A same block can also drive several break lines. The break interface is controlled via MDF_OLDxCR and MDF_SCDxCR.

Figure 344. Break interface simplified view



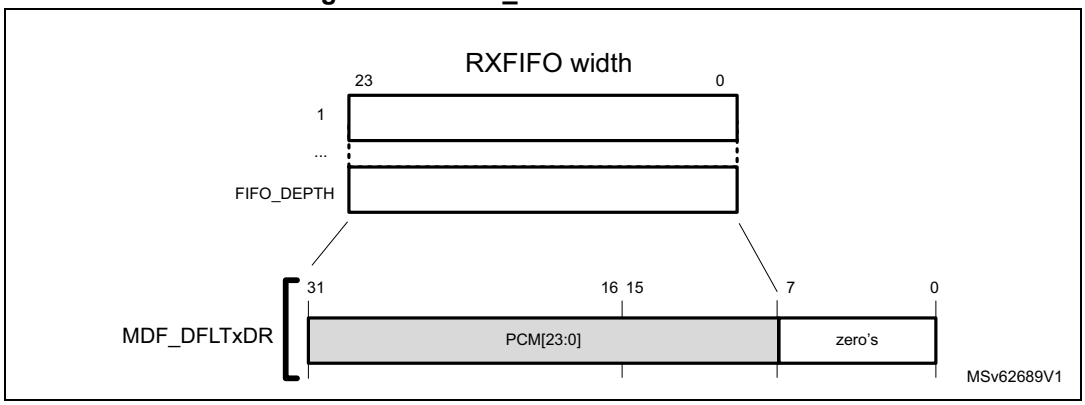
35.4.13 Data transfer to memory

Data format

The samples processed by DFLTx are stored into a RXFIFO. The application can read the samples stored into the FIFOs via the *MDF digital filter data register x (MDF_DFLTxDR)*. The samples inside this register are signed and left aligned. The bit 31 always represents the sign.

The MDF provides 24-bit left-aligned data. Performing a 16-bit access to MDF_DFLTxDR allows the application to get the 16 most significant bits. Performing a 32-bit access to MDF_DFLTxDR allows the application to get a 24-bit data size.

Figure 345. MDF_DFLTxDR data format



Data resynchronization

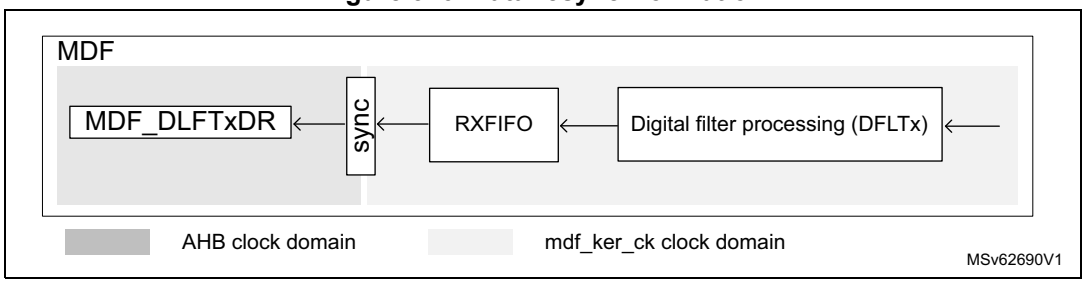
The samples stored into the RXFIFOs can be transferred into the memory by using either DMA requests or interrupt signaling.

Note: The RXFIFOs are located into the mdf_ker_ck clock domain, while MDF_DFLTxDR *are located into the mdf_hclk (AHB) clock domain.*

When the AHB clock is available, if MDF_DFLTxDR is empty and if a sample is available into the RXFIFO, this sample is transferred into MDF_DFLTxDR.

The sample transfer from the RXFIFO to MDF_DFLTxDR takes two periods of the AHB clock (mdf_hclk) and two periods of the mdf_ker_ck clock. The MDF inserts automatically wait-states if the application performs a read operation of MDF_DFLTxDR while the transfer of the new sample from the RXFIFO to MDF_DFLTxDR is not yet completed.

Figure 346. Data resynchronization



The MDF can also combine two transfer types: the independent-transfer and the interleaved-transfer modes.

Independent-transfer mode

In this mode, each RXFIFO has its own DMA channel and its own FTHF flag event.

Both single and burst DMA transfers are supported in this mode, but the application must care about the following points:

- The RXFIFO must contain at least the same amount of samples than the burst size.
- The burst mode efficiency may be reduced due to the data resynchronization explained in the previous section.

In addition, the application can select the RXFIFO threshold (FTH bit) to trigger the data transfer: a data transfer can be triggered as soon as the RXFIFO is not empty, or when the RXFIFO is half-full (containing depth / 2 samples).

For the DMA transfer, as soon as one of the RXFIFO reaches the threshold level, the corresponding DMA request is asserted in order to ask for data transfer. Successive DMA requests are performed as long as the corresponding RXFIFO is not empty.

The DMA mode of the RXFIFOx is enabled via the corresponding DMAEN bit in the *MDF digital filter control register x (MDF_DFLTxCR)*.

For the interrupt signaling, the following cases must be considered:

- If FTH = 0, as soon as a data is available in MDF_DFLTxDR, the corresponding FTHF is set, allowing the generation of an interrupt. FTHF is released as soon as MDF_DFLTxDR is read.
- If FTH = 1, as soon as one of the RXFIFO reaches the threshold level and a data is available into MDF_DFLTxDR, the corresponding FTHF is set, allowing the generation of an interrupt. The FTHF flag is released as soon as one data is read. FTHF is set again if the threshold condition is met again. In this mode, every time an interrupt occurs, the application is supposed to read FIFO_SIZE / 2 data.

The independent-transfer mode must be used when the sample rates provided by each filter paths are not perfectly synchronous, or when the streams are independent. This situation may occur for example:

- when SITF is in SLAVE mode and each external sensor provides its own sampling clock

- when the decimation ratios of the DFLTx are not the same
- when all the streams are not starting at the same time

Interleaved-transfer mode

This mode optimizes the DMA request resources by sharing a single DMA request with several RXFIFOs.

Only single DMA transfers are supported in this mode.

In interleaved-transfer mode, FTH cannot be used and only the FTHF of RXFIFO0 is available for the interrupt signaling.

For the DMA transfer, when all the RXFIFOs working in interleaved-transfer mode are not empty, some DMA requests are generated in order to read sequentially one sample of each RXFIFOs, via MDF_DFLT0DR.

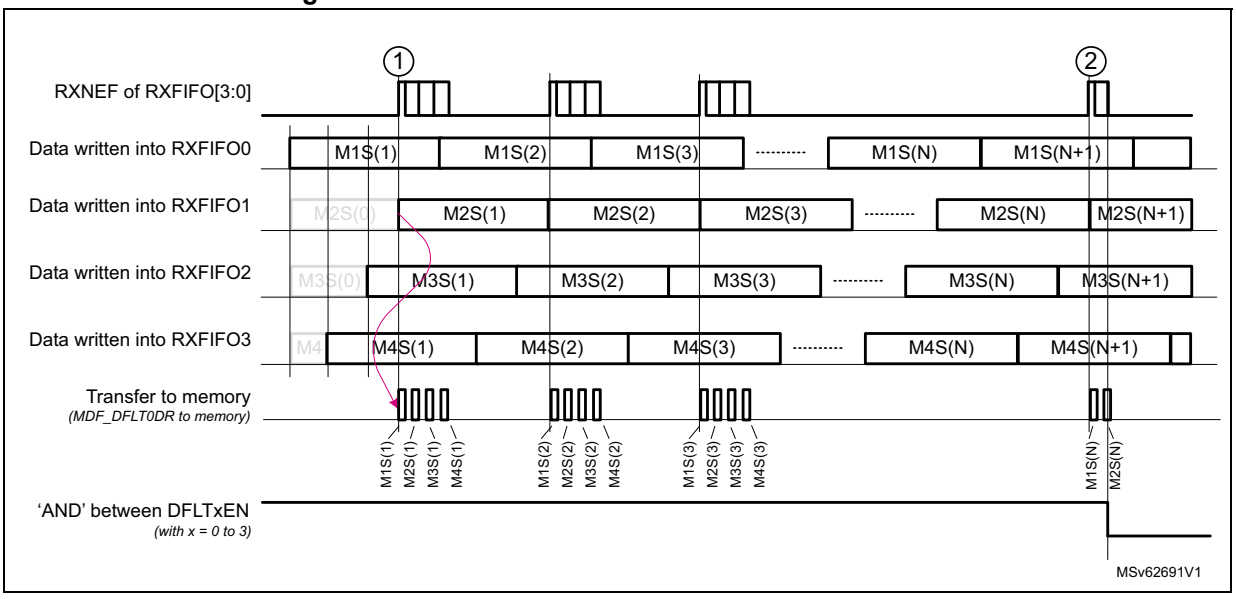
For the RXFIFOs working in interleaved-transfer mode, the DMA channel is enabled by setting to 1 DMAEN in MDF_DFLT0DR and by defining the amount of RXFIFOs working in interleaved mode via ILVNB[3:0]. DMAEN for the other RXFIFOs working in interleaved-transfer mode (RXFIFO 1 to ILVNB[3:0]) are not taken into account.

Figure 347 shows four RXFIFOs working in interleaved-transfer mode. Each RXFIFO has a delay due to the programming of the DLY block.

When all the RXFIFOs have at least one sample available (in this example the last FIFO receiving a sample is RXFIFO1), the MDF requests the data transfer to the memory (see 1 in *Figure 347*). In this figure, the DMA is assumed to be used for the data transfer to memory.

The acquisition can be simply stopped by setting DFTLEN to 0 for one of the filter configured in interleaved-transfer mode. In the example, when a DFLTEN is set to 0 (see 2 in *Figure 347*), the transfer to memory is immediately stopped. Samples M2S(N), M3S(N) and M4S(N) may be lost.

Figure 347. Data transfer in interleaved-transfer mode



In interrupt mode, when all the RXFIFOs working in interleaved-transfer mode have a data ready, the FTHF flag of RXFIFO0 is asserted to allow the interrupt generation. The application is then supposed to read sequentially one sample of each RXFIFOs, via MDF_DFLT0DR. The FTHF event is released when the first data is read.

For the RXFIFOs working in interleaved-transfer mode, the interrupt mode is enabled by setting FTHIE to 1 in MDF_DFLT0DR and by defining the amount of RXFIFOs working in interleaved-transfer mode via ILVNB[3:0]. FTHIE of the other RXFIFOs working in interleaved-transfer mode (RXFIFO1 to ILVNB[3:0]) are not taken into account.

The interleaved-transfer mode can only be applied starting at RXFIFO0 up to RXFIFOk. The number of RXFIFOs working in interleaved-transfer mode are defined by ILVNB[3:0].

When ILVNB[3:0] = 0, all RXFIFOs work in independent-transfer mode.

The interleaved-transfer mode can be used only for digital filters delivering samples perfectly synchronous each other. This is typically the case of sound capture with an array of digital microphones sharing the same bitstream clock.

Caution: To make the MDF working properly in interleaved-transfer mode, the following rules must be respected:

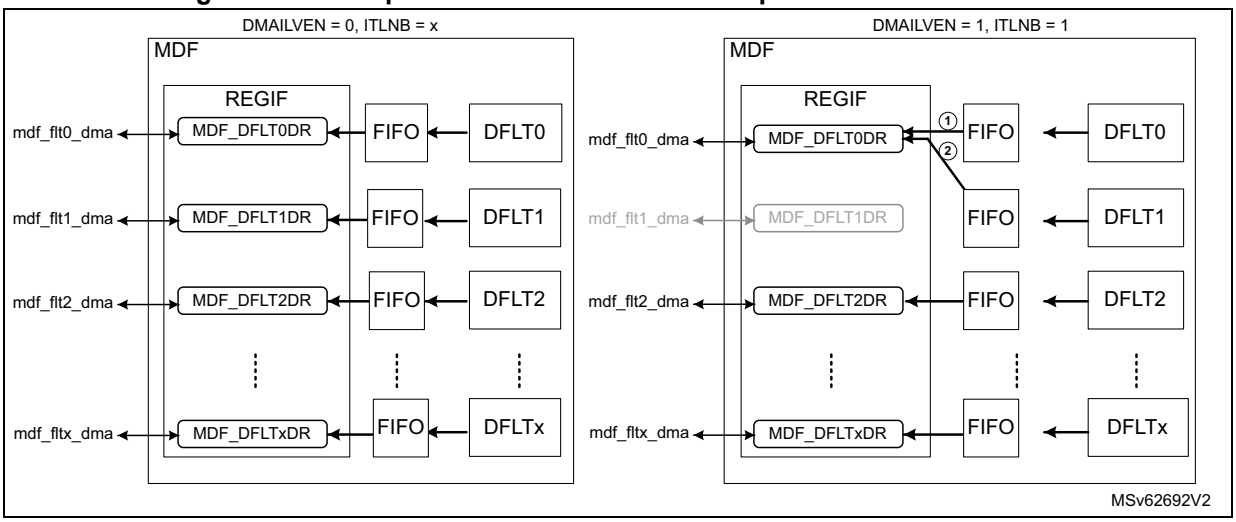
- All filters working in interleaved-transfer mode must be all synchronized together, meaning that all filters must select the same ACQMOD[2:0] (must be equal to 010 or 011), the same trigger source and the same trigger sensitivity.
- All filters working in interleaved-transfer mode must have the same configuration.
- All sensors providing the data for the filters working in interleaved-transfer mode must use the same bitstream clock frequency.
- The maximum delay difference applied with SPKDLY[6:0] for the filters working in interleaved-transfer mode must remain strictly lower than the RXFIFO depth. Normally this delay does not need to exceed one T_{PCM} period.

Note: Both independent- and interleaved-transfer modes can work in parallel: for example, the first RXFIFOs (RXFIFO0 to RXFIFOk) may work in interleaved-transfer mode when the others (RXFIFO(k+1) to RXFIFO(N)) works in independent-transfer mode.

If the AHB clock is not present when a data transfer must be performed, the MDF first requests the AHB clock (refer to Section 35.4.5: Clock generator (CKGEN) for details).

The figure below shows the data path for a configuration in independent-transfer mode (left-hand figure) and a configuration mixing both independent- and interleaved-transfer modes (right-hand figure).

Figure 348. Data path for interleaved- and independent-transfer modes



In the right-hand figure, the DFLT0 and DFLT1 filters are configured in interleaved-transfer mode. For those filters, FTH is no longer taken into account by the hardware. The samples provided by DFLT0 and DFLT1 are read from MDF_DFLT0DR. When both RXFIFOs are no longer empty, the MDF generates two DMA requests to read the sample from DFLT0 and the sample from DFLT1. If both RXFIFOs are not empty again, the same sequence is triggered.

Note: When one of the filter working in interleaved-transfer mode is disabled, the data transfer to memory of all filters in interleaved-transfer mode is stopped immediately as well.

RXFIFO overrun

A RXFIFO overrun condition is detected when the RXFIFO is full and a new sample from the DFLTx must be written.

In this case, DOVRF is set and the new sample is dropped. When the RXFIFO has at least one location available, the new incoming sample is written into the RXFIFO.

Figure 349 shows an example based on a RXFIFO depth of four words and FTH set to 1, so that FTHF goes to 1 when the RXFIFO is half-full.

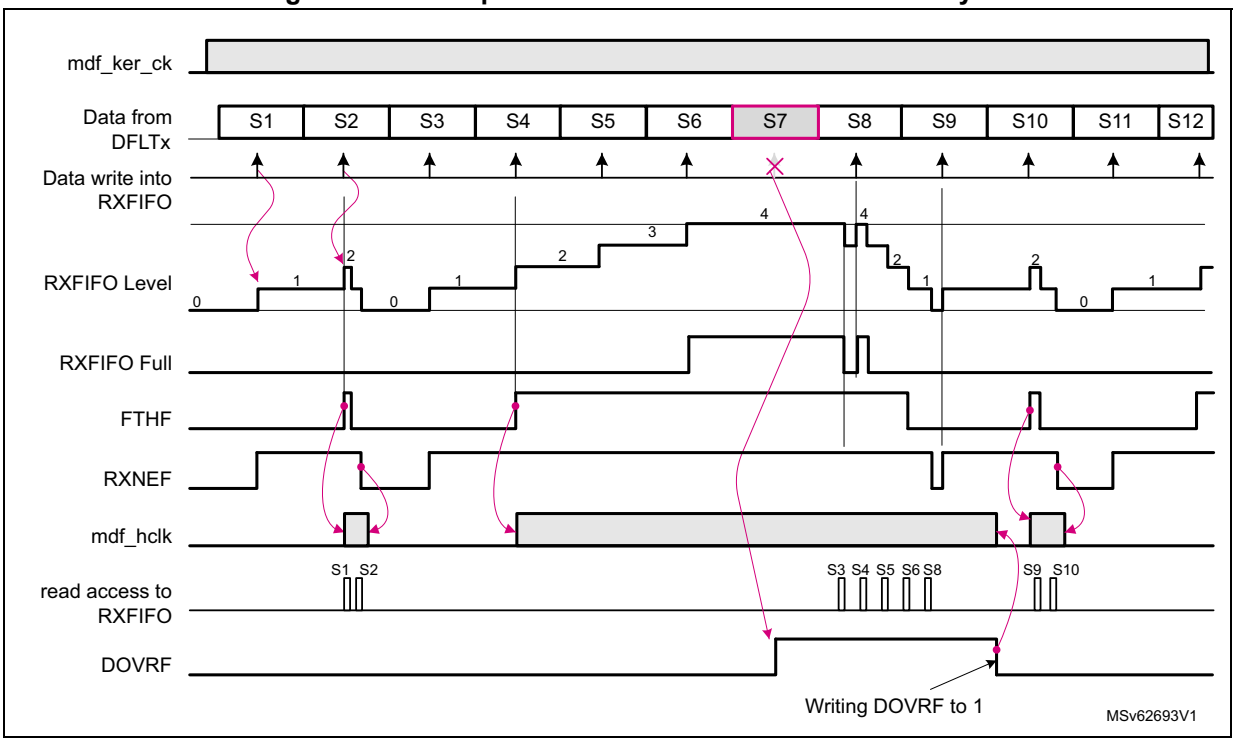
The S7 sample is lost due to an overrun: the RXFIFO is full while S7 must be written into the RXFIFO. The S7 write operation is not performed. DOVRF is set to 1 at the moment where the write operation was expected. The overflow event remains to 1 as long as it is not cleared by the application.

In this example, DOVRIE is set to 1 to have an interrupt if an overrun condition is detected.

After the S7 sample, the application manages to read data from the RXFIFO and the MDF can write the S8 sample and consecutive. Later, the application clears DOVR, allowing the detection of a new overrun situation.

In the mdf_hclk line, the gray boxes indicate that the MDF requested the AHB clock. The figure below shows the AHB clock available only when the MDF requests it. In real applications, the AHB clock may also be present if the MDF does not request it.

Figure 349. Example of overflow and transfer to memory



Note: If the MDF works in interleaved-transfer mode, the application must check the overrun status of all RXFIFOs that works in interleaved-transfer mode.

35.4.14 Autonomous mode

The MDF can work even if the AHB bus clock is not available (Stop modes). The MDF uses the AHB clock only for the register interface. All the processing part is clocked with the kernel clock.

In Stop mode, the MDF receives a kernel clock if the selected clock source is an oscillator available in Stop mode

In Stop mode, the MDF receives the AHB clock if the following conditions are met:

- The MDF autonomous mode is enabled in the RCC.
- The MDF requests the AHB clock, in the following situations:
  - when the MDF must transfer data into the memory via the DMA

    The data are directly transferred to the SRAM thanks to the DMA while the product remains in Stop mode. The AHB clock request is maintained until the DMA transfer is completed.
  - when the MDF must generate an interrupt

    An interrupt generally wakes up the device from Stop mode, as an action from the application is needed. Once the AHB clock is available, the interrupt is generated.The AHB clock request is maintained as long as an enabled interrupt

flag is active. More precisely, when an interrupt must be generated, the MDF requests the AHB clock.

35.4.15 Register protection

The MDF embeds some hardware protection to prevent invalid situations. The table below shows the list of write-protected and unprotected fields.

Table 287. Register protection summary

Registers	Unprotected fields	Write-protected fields	Write-protection condition
MDF global control register (MDF_GCR)	TRGO	ILVNB[3:0]	DFLTACTIVE0 = 1
MDF clock generator control register (MDF_CKGCR)	CKGDEN CCK0EN CCK1EN	PROCDIV[6:0], CCKDIV[3:0], CKGMOD, TRGSRC[3:0], TRGSENS, CCK[1:0]DIR	CKGACTIVE = 1
MDF serial interface control register x (MDF_SITFxCR)	SITFEN	STH[4:0], SITFMOD[1:0], SCKSRC[1:0]	SITFACTIVEx = 1
MDF bitstream matrix control register x (MDF_BSMXxCR)	-	BSSEL[4:0]	DFLTACTIVEx = 1 or SCDACTIVEx = 1 or OLDACTIVEx = 1
MDF digital filter control register x (MDF_DFLTxCR)	DFLTEN	NBDIS[7:0], TRGSRC[3:0], TRGSENS, FTH, DMAEN, SNPSFMT, ACQMOD[2:0]	DFLTACTIVEx = 1
MDF digital filter configuration register x (MDF_DFLTxCICR)	SCALE[5:0]	MCICD[8:0], CICMOD[2:0], DATSRC[1:0]	
MDF reshape filter configuration register x (MDF_DFLTxRSFR)	-	All fields	
MDF integrator configuration register x (MDF_DFLTxINTR)	-	All fields	
MDF out-of limit detector control register x (MDF_OLDxCR)	OLDEN	ACICD[4:0], ACICN[1:0],THINB BKOLD[3:0]	OLDACTIVEx = 1
MDF OLDx low threshold register x (MDF_OLDxTHLR)	-	All fields	
MDF OLDx high threshold register x (MDF_OLDxTHHR)	-	All fields	
MDF delay control register x (MDF_DLYxCR)	-	SKPDLY[6:0]	SKPBF = 1
MDF short circuit detector control register x (MDF_SCDxCR)	SCDEN	BKSCD[3:0], SCDT[7:0]	SCDACTIVEx = 1
MDF DFLTx interrupt enable register x (MDF_DFLTxIER)	All fields	-	-

Table 287. Register protection summary (continued)

Registers	Unprotected fields	Write-protected fields	Write-protection condition
MDF DFLTx interrupt status register x (MDF_DFLTxISR)	All fields	-	-
MDF offset error compensation control register x (MDF_OECxCR)	OFFSET[25:0]	-	-

All the MDF processing is performed in the mdf_proc_ck clock domain. For that reason, enabling or disabling a MDF sub-block may take some time due to the resynchronization between the AHB clock domain and the mdf_proc_ck clock domain. XXXACTIVE flags are available to allow the application to check that the synchronization between the two clock domains is completed.

To change a write-protected bitfield, the application must follow this sequence:

1. Set the enable bit of the sub-block to 0.
2. Wait for corresponding flag XXXACTIVE = 0.
3. Modify the wanted fields.
4. Set the enable bit of the sub-block to 1.

Refer to the description of each sub-block for more details.

35.5 MDF low-power modes

Table 288. Effect of low-power modes on MDF

Mode	Description
Sleep	No effect. MDF interrupts cause the device to exit Sleep mode.
Stop(1)	The MDF registers content is kept. If the MDF is clocked by an internal oscillator available in Stop mode, the MDF remains active. The interrupts cause the device to exit Stop mode.
Standby	The MDF is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 35.3: MDF implementation* for details about Stop modes supported by the MDF.

35.6 MDF interrupts

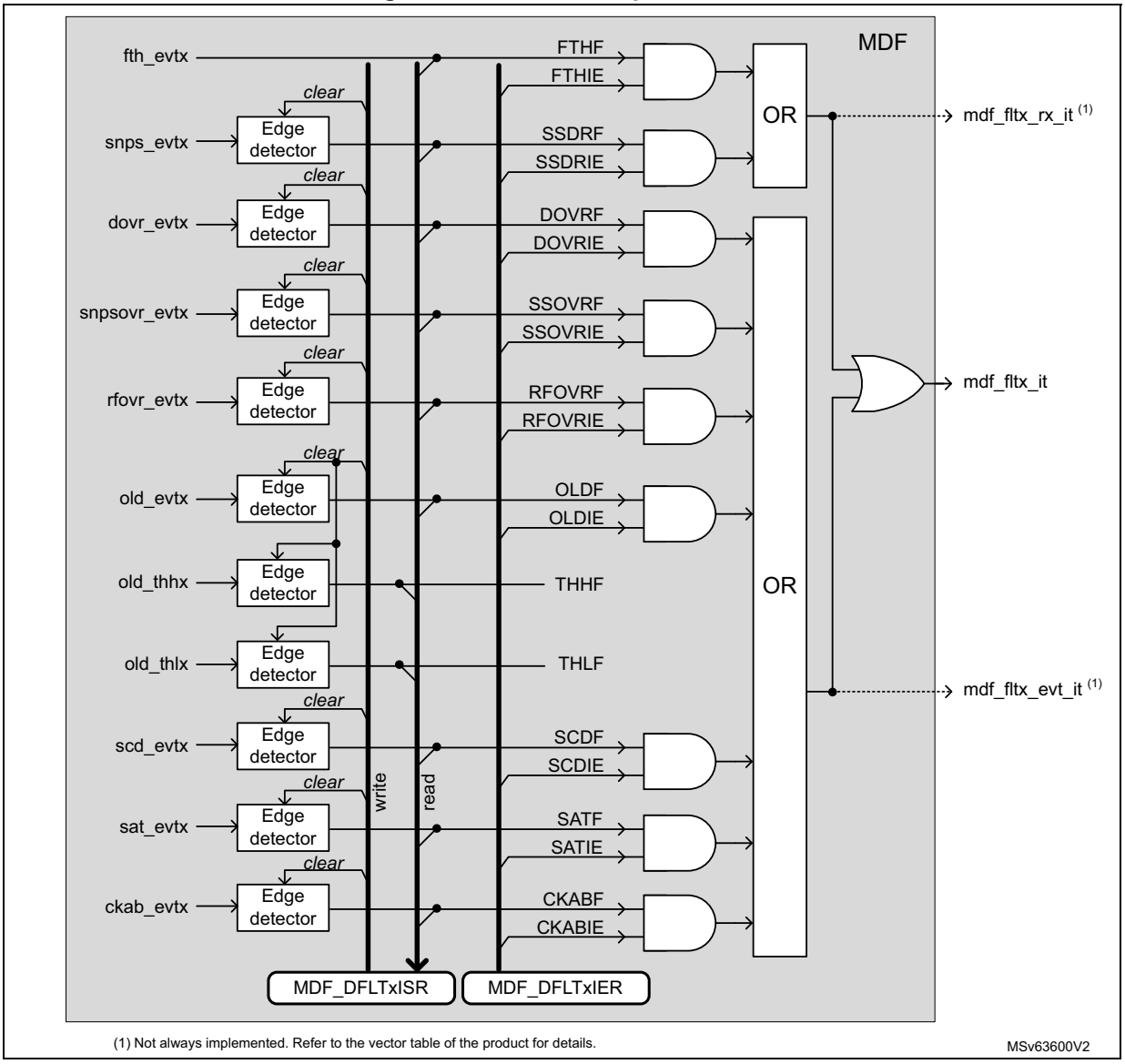
To increase the CPU performance, the MDF offers the following interrupt lines per digital filter:

- receive interrupt MDF_FLTx_RX (mdf_fltx_rx_it)
- event interrupt MDF_FLTx_EVT (mdf_fltx_evt_it)
- a combination of both interrupts MDF_FLTx (mdf_fltx_it)

Note: Interrupts are not always connected to the device (see Section 35.3: MDF implementation for more details).
The status flags are available even if the corresponding interrupt enable flag is not enabled.

The interrupt interface is controlled via the *MDF DFLTx interrupt enable register x (MDF_DFLTxIER)* and the *MDF DFLTx interrupt status register x (MDF_DFLTxISR)*.

Figure 350. MDF interrupt interface



(1) Not always implemented. Refer to the vector table of the product for details.

MSv63600V2

The table below shows which interrupt line is affected by which event, and how to clear and activate each interrupt/event.

Table 289. MDF interrupt requests

Interrupt vectors		Interrupt event	Event flag	Event/interrupt clearing method	Exit Sleep mode	Exit Stop modes[1]	Exit Standby mode
MDF_FLTx[2]	MDF_FLTx_RX[3]	RXFIFO threshold reached	FTHF	Read MDF_DFLTxDR until RXFIFO level is lower than the threshold.	Yes	Yes	No
		Snapshot data ready	SSDRF	Write SSDRF to 1.			
	MDF_FLTx_EVT[4]	Snapshot data overrun	SSOVRF	Write SSOVRF to 1.			
		RXFIFO overrun	DOVRF	Write DOVRF to 1.			
		RSFLT overrun	RFOVRF	Write RFOVRF to 1.			
		Short-circuit detector	SCDF	Write SCDF to 1.			
		Saturation detection	SATF	Write SATF to 1.			
		Channel clock absence detection	CKABF	Write CKABF to 1.			
		Out-of-limit detector	OLDF	Write OLDF to 1.			
-	-		THHF	Write OLDF to 1.	-	-	-
-	-		THLF	Write OLDF to 1.	-	-	-

1. Refer to *Section 35.3: MDF implementation* for details.
2. MDF_FLTx vector corresponds to the assertion of mdf_fltx_it signal
3. MDF_FLTx_RX vector corresponds to the assertion of mdf_fltx_rx signal.
4. MDF_FLTx_EVT vector corresponds to the assertion of mdf_fltx_evt signal.

35.7 MDF application informations

35.7.1 MDF configuration examples for audio capture

Table 290 gives some examples of the MDF settings for the digital microphones, focusing on 16 and 48 kHz output data rate. In these examples, the following is expected:

- The INT block is bypassed (INTVAL = 0).
- The offset error compensation is disabled (OFFSET = 0).

Configurations #1 and #2 are for very low-power use-cases and have a reduced signal-to-noise ratio. The user must also insure that the selected digital microphone can work properly at 512 kHz. These configurations can be used for sound detection. The RSFLT is not used to reduce as much as possible the frequency of the kernel clock (mdf_ker_ck).

Configurations #3, #4, #9, #10, #11 give signal-to-noise ratios around 115 dB, with an ideal microphone model, with a sinus signal of 997 Hz. Using the RSFLT allows a good control on the in-band ripple and a good image rejection.

Configurations #7, #8, #10 give signal-to-noise ratio around 120 dB, with an ideal microphone model, using a sinus signal of 997 Hz.

Table 290. Examples of MDF settings for microphone capture

Configuration	mdf_ker_ck (MHz)	PROCDIV + 1	CCKDIV + 1	CIC order (1)	MCICD + 1	SCALE	RSFLTBYP	RSFLTD	HPFBYP	-	mdf_proc_ck (MHz)	Total dec. ratio	F_{RS} (kHz)	F_{MDF_CCKx} (MHz)	F_{PCM} (kHz)
#1	1.024	1	2	4	64	0x2D (- 8.5 dB)	1	x	x	=>	1.024	64	-	0.512	8
#2		1	2	5	32	0x2B (- 14.5 dB)						32	-		16
#3		1	2	5	16	0x01 (+ 3.5 dB)	0	0				64	32		8
#4	2.048	1	2	5	16	0x01 (+ 3.5 dB)	0	0			2.048	64	64	1.024	16
#5	3.072	1	6	5	8	0x0B (+ 33.6 dB)	0	0			3.072	32	32	0.512	8
#6		2	2	5	12	0x06 (+ 18.1 dB)						48	64	0.768	16
#7		1	2	5	24	0x2C (- 12 dB)						96		1.536	
#8	4.096	1	2	5	32	0x27 (- 26.6 dB)	0	0			4.096	128	64	2.048	16
#9	6.144	3	2	5	16	0x02 (+ 6.0 dB)	0	0			2.048	64	64	1.024	16
#10		2	2	5	24	0x2C (- 12 dB)					3.072	96		1.536	
#11		1	2	5	16	0x01 (+ 3.5 dB)					6.144	64	192	3.072	48
#12	7.680	1	2	5	20	0x2E (- 6.0 dB)	0	0			7.680	80	192	3.840	48

1. CICMOD = 100 for CIC order equal to 4. CICMOD = 101 for CIC order equal to 5.

35.7.2 Programming examples

This example describes how to capture sound from four microphones, assuming that each microphone pair shares the same data line. The MDF_SD0 and MDF_SD2 data lines are used. The microphone clock is provided by the MDF via MDF_CCK0 pin.

Table 291. Programming sequence

Operations	Comments
Adjust the proper kernel clock frequency via the RCC block	Assuming that the RCC is programmed to provide a kernel clock (mdf_ker_ck) of 12.288 MHz
Select the proper MDF kernel clock source via the RCC block	Refer to the RCC of the product.
Enable the MDF clocks via the RCC block	Refer to the RCC of the product.
Reset the MDF via the RCC block	Refer to the RCC of the product.
AFMUX programming	Program the AFMUX to select MDF_SD0, MDF_SD2 and MDF_CCK0 function.

Table 291. Programming sequence (continued)

Operations	Comments
Enable MDF processing clock: MDF_CKGCR = 0x0103 0023	PROCDIV = 1 (division by 2): mdf_proc_ck frequency is 6.144 MHz. CCKDIV = 3 (division by 4): MDF_CCK0 clock frequency is 1.536 MHz. The MDF_CCK0 pad is set in output and generates a clock so that the microphones can exit from low-power mode.
Serial interfaces configuration: MDF_SITF0CR = 0x0000 1F01 MDF_SITF2CR = 0x0000 1F01	SCKSRC = 0 to select MDF_CCK0 as serial clock. SIFTMOD = 0 to select LF_MASTER mode. Clock absence feature is not working in this mode. The serial interfaces are enabled.
Bitstream matrix configuration: MDF_BSMX0CR = 0x0000 0000 MDF_BSMX1CR = 0x0000 0001 MDF_BSMX2CR = 0x0000 0004 MDF_BSMX3CR = 0x0000 0005	DFLT0 filter takes the bitstream of SITF0, sampled on rising edge. DFLT1 filter takes the bitstream of SITF0, sampled on falling edge. DFLT2 filter takes the bitstream of SITF2, sampled on rising edge. DFLT3 filter takes the bitstream of SITF2, sampled on falling edge.
Filters configuration (CIC): MDF_DFLT0CICR = 0x02C0 1750 MDF_DFLT1CICR = 0x02C0 1750 MDF_DFLT2CICR = 0x02C0 1750 MDF_DFLT3CICR = 0x02C0 1750	SCALE = 0x2C (- 12 dB) to avoid any saturation MCICD = 0x17 (decimation by 24) CICMOD = 5 to select a $Sinc^5$ DATSCR = 0 to select data coming from BSMX
Filters configuration (RSFLT and HPF): MDF_DFLT0RSFR = 0x0000 0100 MDF_DFLT1RSFR = 0x0000 0100 MDF_DFLT2RSFR = 0x0000 0100 MDF_DFLT3RSFR = 0x0000 0100	HPFC = 1: cut-off frequency of 16 kHz * 0.00125 = 20 Hz HPFBYP = 0: HPF not bypassed RSFLTD = 0: RSFLT decimates by 4 RSFLTBYP = 0: RSFLT is not bypassed
Filters configuration (INT): MDF_DFLT0INTR = 0x0000 0000 MDF_DFLT1INTR = 0x0000 0000 MDF_DFLT2INTR = 0x0000 0000 MDF_DFLT3INTR = 0x0000 0000	INTVAL = 0: INT filter not used Other parameter is not significant.
Micro delay adjust: MDF_DLY0CR = 0x0000 0005 MDF_DLY0CR = 0x0000 0012 MDF_DLY0CR = 0x0000 0023 MDF_DLY0CR = 0x0000 0000	Initial micro-delay for each microphone, values just given as example
Offset error correction: MDF_OEC0CR = 0x0000 0000 MDF_OEC1CR = 0x0000 0000 MDF_OEC2CR = 0x0000 0000 MDF_OEC3CR = 0x0000 0000	No correction. DC offset is removed by HPF.

Table 291. Programming sequence (continued)

Operations	Comments
Short circuit detection: MDF_SCD0CR = 0x0000 0000 MDF_SCD1CR = 0x0000 0000 MDF_SCD2CR = 0x0000 0000 MDF_SCD3CR = 0x0000 0000	SCD function for each filter is disabled.
Enable interrupt events: MDF_DFLT0IER = 0x0000 0202 MDF_DFLT1IER = 0x0000 0202 MDF_DFLT2IER = 0x0000 0202 MDF_DFLT3IER = 0x0000 0202	Enable the interrupt event the application wants to handle. In this example, the SATIE and DOVRIE bits are set to 1 to have an interrupt if a saturation or an data overflow occurs.
Digital filter control: MDF_DFLT0CR = 0x0000 0027 MDF_DFLT1CR = 0x0000 0027 MDF_DFLT2CR = 0x0000 0027 MDF_DFLT3CR = 0x0000 0027 Wait for DFLTACTIVE = 0 for the filters	NBDIS = 0: no samples discarded TRGSRC = 0: TRGO selected as trigger source TRGSENS = 0: Trigger on rising edge ACQMOD = 2: Synchronous continuous acquisition mode DMAEN = 1: DMA interface enabled DFLTEN = 1: digital filter enabled
Clear status flags: MDF_DFLT0ISR = 0x0000 0FFF MDF_DFLT1ISR = 0x0000 0FFF MDF_DFLT2ISR = 0x0000 0FFF MDF_DFLT3ISR = 0x0000 0FFF	Clear all the status flags before running the filters.
Program the DMA Enable the DMA	The DMA must be programmed in order to read the data inside MDF_DFLT0DR every time a DMA request is generated. Note that when the MDF is in interleaved acquisition mode, data of filters 0, 1, 2, and 3 are read via MDF_DFLT0DR register.
Start acquisition: MDF_GCR = 0x0000 0031	ILVNB = 3: interleaved mode with DFLT0,1,2, and 3 TRGO = 1 to trigger the acquisition of all filters waiting for TRGO rising edge event

35.7.3 Connection examples

Figure 351 shows simple connection examples of the MDF to external sensors:

- Picture on the left: four digital microphones connected to the MDF

  In this connection, DMIC1 is intended to be used alone. This is why it uses a dedicated clock (MDF_CCK0). DMIC1 is also connected to a dedicated data line in case the application intends to power-down DMIC2/3/4, when only DMIC1 is used. If the application keeps DMIC2/3/4 powered, DMIC2 and DMIC1 can share the same data line, reducing the connection to four I/Os.

  If the power consumption is not an issue, when a single microphone is used, all the microphones can be connected to the same common clock (MDF_CCK0) and each

microphone pair can share the same data line. In this case, only three I/Os are required.

In this example, the data transfer to memory can be performed either using the interleaved- or the independent-transfer mode.

- Picture in the center: sensor providing several data lines with a common clock

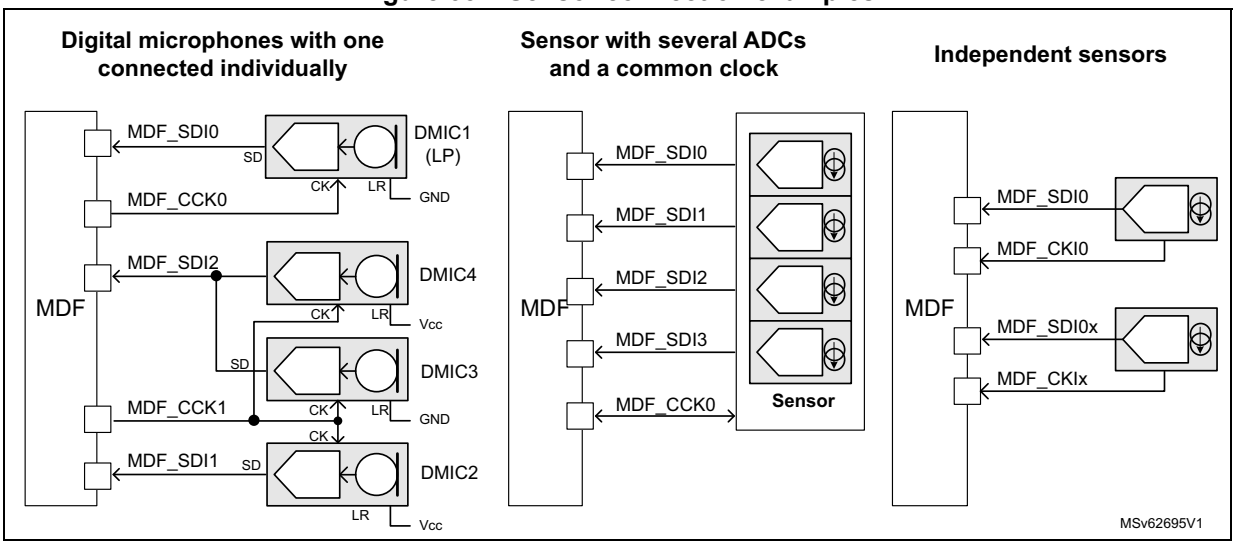
  Each data line can represent different parameters (such as current or voltage). The common clock can be provided either by the sensor or by the MDF. The data line can be shared or not by two sensors if the sensor allows it. In the figure below, the sensor does not allow the sharing of the data lines.

  In this example, the data transfer to memory can be performed either using the interleaved- or independent-transfer mode.

- Picture on the right: two independent sensors connected to the MDF

  Each of them has its dedicated clock and data lines. In this case, the data transfer to memory must use the independent-transfer mode.

Figure 351. Sensor connection examples



35.7.4 Global frequency response

Figure 352 shows the global frequency response for a 16 kHz audio signal with a digital microphone working at 1.024 MHz. The filter configuration is the following:

- CIC order 4 or 5, with a decimation ratio of 16
- RSFLT enabled, with a decimation ratio of 4
- HPF enable with a cut-off frequency of 40 Hz

The figure below shows the theoretical frequency response using a CIC4 and a CIC5.

Figure 352. Global frequency response

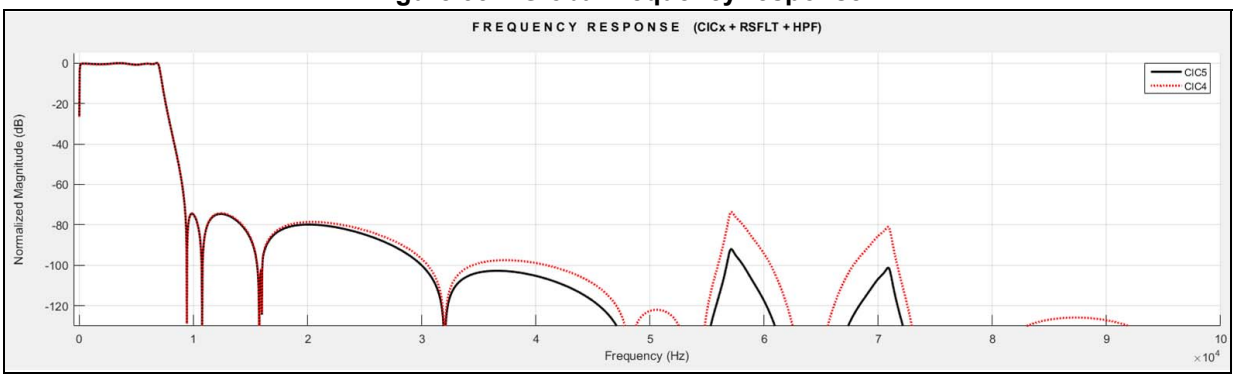


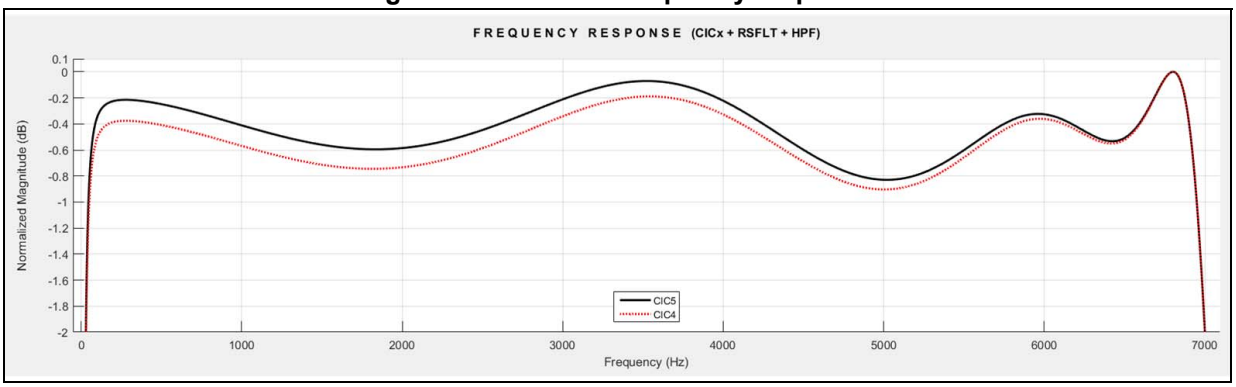
Figure 353 shows the in-band ripple for a 16 kHz audio signal with a digital microphone working at 1.024 MHz. The filter configuration is the following:

- CIC order 4 or 5, with a decimation ratio of 16
- RSFLT enabled, with a decimation ratio of 4
- HPF enable with a cut-off frequency of 20 Hz

The resulting in-band ripple is ± 0.41 dB for CIC5, and ± 0.45 for CIC4.

The -3 dB cut-off frequency is 7061 Hz.

Figure 353. Detailed frequency response



35.7.5 Total MDF gain

This section details how to compute the signal level provided by the MDF according to the filter settings. The formula does not take into account the filters transfer function.

A signal level may be expressed in dBFS (decibel full scale). A 0 dBFS level is assigned to the maximum possible digital level. For example, a signal that reaches 50 % of the maximum level, has a - 6 dBFS level (6 dB below full scale).

For example, for the MDF offering a final data width of 24 bits, a signal having an amplitude of 2 * 10^6 LSB has a level of:

$$20 \times \log_{10}\left(\frac{2 \times 10^6}{2^{(24-1)}}\right) = -12.45 \text{ dBFS}$$

In addition, the data size of a signal having an amplitude (Amp) expressed in LSB is given by:

$$DS = \left(\frac{\ln(Amp)}{\ln(2)} + 1\right) \text{ bits}$$

One bit need to be added for negative values.

So a signal having an amplitude of 2 * 10^6 LSB, has a data size of 21.9 bits.

CIC gain

The CIC gain (G_{CIC} and GdB_{CIC}) can be deduced from the following formula giving data size in bits (DS_{CIC}).

$$DS_{CIC} = (N \times \log_2(D1)) + DS_{IN}$$

And the bit growth is:

$$BG_{CIC} = (N \times \log_2(D1))$$

where N represents the CIC order (selected by CICMOD[2:0]), and D1 is the decimation ratio (given by MCICD).

DS_{IN} represents the data size (in bits) of the input signal.

Warning: DS_{CIC} is very important for CIC filters. In order to work fine, DS_{CIC} must not exceed 26 bits.

$$G_{CIC} = 2^{(BG_{CIC})} = (D1)^N$$

which gives, in decibels:

$$GdB_{CIC} = 20 \times \log_{10}((D1)^N)$$

Note: The same formulas are valid for the ACIC.

Data size at SCALE output

The data size at SCALE output (including the CIC gain) is a key information as the RSFLT starts to have some saturations if the peak-to-peak signal amplitude at SCALE output is higher than 22 bits.

If the RSFLT is bypassed, then a peak-to-peak signal amplitude of 24 bits is accepted. The resulting data size is given by:

$$DS_{SCALE} = N \times \log_2(D1) + \log_2\left(10^{\frac{GDB_{SCALE}}{20}}\right) + DS_{IN}$$

The data size at SCALE output (DS_{SCALE}) is expressed in bits and GdB_{SCALE} represents the gain selected by SCALE[5:0], in dB.

RSFLT gain

The RSFLT gain in the useful bandwidth is typically 9.5 dB, but due to ripple a margin of about ± 0.5 dB must be considered. Typically, the RSFLT increases the bit size by BG_{RSFLT}:

$$BG_{RSFLT} = 10^{\frac{9.5\ dB}{20}} = 2.98 = 1.6\ bits$$

INT gain

The INT block can also introduce a gain if the rescaling value is different from the integration value.

$$G_{INT} = \frac{IVAL}{IDIV}$$

and:

$$GdB_{INT} = 20 \times \log_{10}\left(\frac{IVAL}{IDIV}\right)$$

The bit growth of the INT is then given by the following formula:

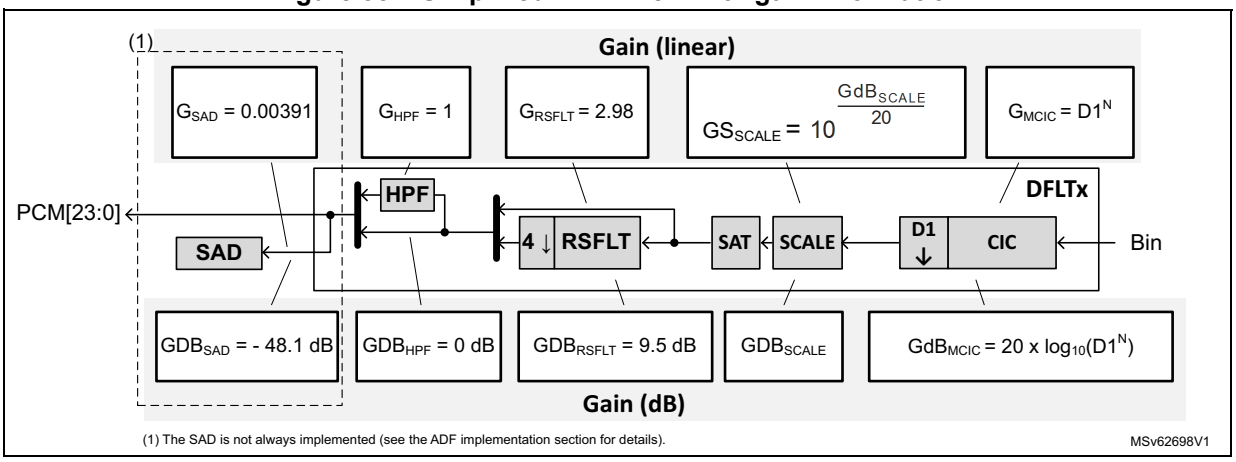
$$BG_{INT} = \log_2\left(\frac{IVAL}{IDIV}\right)$$

IVAL represents the integration value selected by INTVAL[6:0], and IDIV represents the integrator output division selected by INTDIV[1:0].

Note: The HPF filter has a gain of 0 dB.

The figure below shows a simplified view of the filter path, and gives for each significant component the expression of the bit growth and the gain.

Figure 354. Simplified DFLT view with gain information



The table below summarizes the final data size for different filter configurations.

Table 292. Output signal levels

Filter configurations	Final signal level (dBFS)	Final signal size (bits)
CIC + RSFLT (+ HPF)	$SdB_{OUT} = 20 \times \log_{10}\left(\frac{2^{DS_{OUT}}}{2^{24}}\right)$	$DS_{OUT} = DS_{SCALE} + 1.6$ bits DS_{SCALE} must be lower than 22 bits.
CIC + RSFLT (+ HPF) + INT		$DS_{OUT} = DS_{SCALE} + BG_{INT} + 1.6$ bits DS_{SCALE} must be lower than 22 bits.
CIC (+ HPF) + INT		$DS_{OUT} = DS_{SCALE} + BG_{INT}$ DS_{SCALE} must be lower than 24 bits.

Example using the main filter chain

If the MDF filter is programmed as follows:

- The input signal is coming from a serial interface (DS_{IN} = 1 bit).
- CIC order = 5 (N), with a decimation value of 24 (D1).
- SCALE[5:0] is set to -12 dB.
- RSFLT is enabled and the decimation by four is enabled.
- HPF is enabled.
- INT is disabled.

Check first the data size at CIC output:

$$DS_{CIC} = (5 \times \log_2(24)) + 1 \text{ bit} = 23.92 \text{ bits}$$

The size is lower than 26 bits, so the CIC works in good conditions.

The data size at CIC output is very close to 24 bits, so the SCALE must be adjusted in order to provide a signal 22 bits max to the RSFLT. An attenuation of 12 dB is needed.

Then the signal level provided to the RSFLT is:

$$DS_{SCALE} = DS_{CIC} + \log_2\left(10^{\frac{-12}{20}}\right) = 23.92 - 1.99 = 21.93 \text{ bits}$$

At the end, the final signal amplitude is:

$$DS_{OUT} = DS_{SCALE} + 1.6 \text{ bits} = (21.93 + 1.6) = 23.52 \text{ bits}$$

or

$$SDB_{OUT} = 20 \times \log_{10}\left(\frac{2^{23.52}}{2^{24}}\right) = -2.84 \text{ dBFS}$$

The RSFLT ripple a margin of about ± 0.41 dB (in this configuration) must be considered.

Example using the OLD filter chain

In the following example, the application wants to trigger an OLD event when the voltage coming from a shunt resistor reaches ± 200 mV.

Hypothesis:

- The ΣΔ sensor is specified to support a full-scale signal of ± 350 mV.
- The ACIC decimation ratio (D2) is fixed to 32.
- The ACIC order (N) is fixed to 3.

The ΣΔ sensor provides a full-scale digital signal (amplitude of one bit) for a signal higher or equal to ± 350 mV. If the input signal is equal 200 mV, then the digital signal amplitude provided by the sensor is 200 / 350 = 0.571.

The gain of the ACIC filter is:

$G_{CIC} = (D2)^N = 32^3 = 32768$

A signal having an amplitude of 200 mV at sensor input has then an amplitude of about 32768 * 0.571 = 18725 LSB at ACIC output.

OLDTHH must be set to 18725 and OLDTHL must be set to -18725.

35.8 MDF registers

All the MDF registers must be accessed either in word (32-bit) or half-word (16-bit) formats.

The registers are described for the MDF instance having the biggest number of filters. For registers related to filters, the number of registers is equal to the amount of filters. Refer to *Section 35.3* for details.

35.8.1 MDF global control register (MDF_GCR)

Address offset: 0x000

Reset value: 0x0000 0000

This register is used for controls common to all digital filters.

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

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **ILVNB[3:0]**: Interleaved number

This bitfield is set and reset by software. it enables or disables the interleaved-transfer mode and defines how many digital filters work in this mode.
This bitfield cannot be changed when DFLTEN = 1 in MDF_DFLT0CR.
0000: Interleaved-transfer mode disabled
0001: Data from DFLT0 and DFLT1 are interleaved.
0010: Data from DFLT0, DFLT1 and DFLT2 are interleaved.
...
1111: Data from DFLT0 to DFLT15 are interleaved.

Note: This bitfield can be write-protected (refer to Section 35.4.15 for details).

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **TRGO**: Trigger output control

This bit is set by software and reset by hardware. It is used to start the acquisition of several filters synchronously. It is also able to synchronize several MDF together by controlling the mdf_trgo signal.

0: Write 0 has no effect. Read 0 means that the trigger can be set again to 1.

1: Write 1 generates a positive pulse on mdf_trgo signal and triggers the acquisition on the enabled filters having ACQMOD[2:0] = 0x1 and selecting TRGO as trigger. Read 1 means that the trigger pulse is still active.

35.8.2 MDF clock generator control register (MDF_CKGCR)

Address offset: 0x004

Reset value: 0x0000 0000

This register is used to control the clock generator. The mdf_proc_ck clock must be enabled before enabling other MDF parts.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CKGACTIVE	PROCDIV[6:0]							Res.	Res.	Res.	Res.	CCKDIV[3:0]			
r	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRGSRC[3:0]				Res.	Res.	Res.	TRGSENS	Res.	CCK1DIR	CCK0DIR	CKGMOD	Res.	CCK1EN	CCK0EN	CKGDEN
rw	rw	rw	rw				rw		rw	rw	rw		rw	rw	rw

Bit 31 **CKGACTIVE**: Clock generator active flag

This bit is set and cleared by hardware. This flag must be used by the application to check if the clock generator is effectively enabled (active) or not. The protected fields of this function can only be updated when CKGACTIVE = 0 (refer to *Section 35.4.15* for details). The delay between a transition on CKGDEN and a transition on CKGACTIVE is two periods of AHB clock and two periods of mdf_proc_ck.

0: The clock generator is not active and can be configured if needed.

1: The clock generator is active and protected fields cannot be configured.

Bits 30:24 **PROCDIV[6:0]**: Divider to control the serial interface clock

This bitfield is set and reset by software. It is used to adjust the frequency of the clock provided to the SITF.

$$F_{mdf_itf_ck} = \frac{F_{mdf_ker_ck}}{(PROCDIV + 1)}$$

This bitfield must not be changed if one of the filters is enabled (DFTEN = 1).

0: mdf_ker_ck provided to the SITF

1: mdf_ker_ck/2 provided to the SITF

2: mdf_ker_ck/3 provided to the SITF

...

127: mdf_ker_ck/128 provided to the SITF

Note: This bitfield can be write-protected (refer to Section 35.4.15 for details).

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **CCKDIV[3:0]**: Divider to control the MDF_CCK clock

This bitfield is set and reset by software. It is used to adjust the frequency of the MDF_CCK clock. The input clock of this divider is the clock provided to the SITF. More globally, the frequency of the MDF_CCK is given by the following formula:

$$F_{MDF_CCK} = \frac{F_{mdf_ker_ck}}{(PROCDIV + 1) \times (CCKDIV + 1)}$$

This bitfield must not be changed if one of the filters is enabled (DFTEN = 1).

0000: The MDF_CCK clock is mdf_proc_ck.

0001: The MDF_CCK clock is mdf_proc_ck / 2.

0010: The MDF_CCK clock is mdf_proc_ck / 3.

...

1111: The MDF_CCK clock is mdf_proc_ck / 16.

Note: This bitfield can be write-protected (refer to Section 35.4.15 for details).

Bits 15:12 **TRGSRC[3:0]**: Digital filter trigger signal selection

This bitfield is set and cleared by software.It is used to select which external signals trigger for the corresponding filter. This bitfield is not significant if CKGMOD = 0.

000x: TRGO selected

0010: mdf_trg[0] selected

0011: mdf_trg[1] selected

...

1111: mdf_trg[13] selected

Note: This bitfield can be write-protected (refer to Section 35.4.15 for details.)

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **TRGSENS**: CKGEN trigger sensitivity selection

This bit is set and cleared by software. It is used to select the trigger sensitivity of the trigger signals. This bit is not significant if the CKGMOD = 0.

0: A rising edge event triggers the activation of CKGEN dividers.

1: A falling edge even triggers the activation of CKGEN dividers.

Note: When the trigger source is TRGO, TRGSENS value is not taken into account. When TRGO is selected, the sensitivity is forced to falling edge. This bit can be write-protected (refer to Section 35.4.15 for details).

Bit 7 Reserved, must be kept at reset value.

Bit 6 **CCK1DIR**: MDF_CCK1 direction

This bit is set and reset by software.It is used to control the direction of the MDF_CCK1 pin.

0: MDF_CCK1 pin direction is in input.

1: MDF_CCK1 pin direction is in output.

Note: This bit can be write-protected (refer to Section 35.4.15 for details).

Bit 5 **CCK0DIR**: MDF_CCK0 direction

This bit is set and reset by software.It is used to control the direction of the MDF_CCK0.

0: MDF_CCK0 pin direction is in input.

1: MDF_CCK0 pin direction is in output.

Note: This bit can be write-protected (refer to Section 35.4.15 for details).

Bit 4 **CKGMOD**: Clock generator mode

This bit is set and reset by software. It is used to define the way the clock generator is enabled. This bit must not be changed if one of the filters is enabled (DFTEN = 1).

0: The kernel clock is provided to the dividers as soon as CKGDEN is set to 1.

1: The kernel clock is provided to the dividers when CKGDEN is set to 1 and the trigger condition met.

Note: This bit can be write-protected (refer to Section 35.4.15 for details).

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CCK1EN**: MDF_CCK1 clock enable

This bit is set and reset by software. It is used to control the generation of the bitstream clock on the MDF_CCK1 pin.

0: Bitstream clock not generated

1: Bitstream clock generated on the MDF_CCK1 pad

Bit 1 **CCK0EN**: MDF_CCK0 clock enable

This bit is set and reset by software.It is used to control the generation of the bitstream clock on the MDF_CCK0 pin.

0: Bitstream clock not generated

1: Bitstream clock generated on the MDF_CCK0 pad

Bit 0 **CKGDEN**: CKGEN dividers enable

This bit is set and reset by software. It is used to enable/disable the clock dividers of the CKGEN: PROCDIV and CCKDIV.

0: CKGEN dividers disabled

1: CKGEN dividers enabled

35.8.3 MDF serial interface control register x (MDF_SITFxCR)

Address offset: 0x080 + 0x80 * x, (x = 0 to 5)

Reset value: 0x0000 1F00

This register is used to control the serial interfaces (SITFx). The number of registers is equal to the amount of filters. Refer to *Section 35.3* for details.

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

Bit 31 **SITFACTIVE**: Serial interface active flag

This bit is set and cleared by hardware. It is used by the application to check if the serial interface is effectively enabled (active) or not. The protected fields of this function can only be updated when the SITFACTIVE is set to 0 (refer to *Section 35.4.15* for details).

The delay between a transition on SITFEN and a transition on SITFACTIVE is two periods of AHB clock and two periods of mdf_proc_ck.

0: The serial interface is not active and can be configured if needed.

1: The serial interface is active and protected fields cannot be configured.

Bits 30:13 Reserved, must be kept at reset value.

Bits 12:8 **STH[4:0]**: Manchester symbol threshold/SPI threshold
This bitfield is set and cleared by software. It is used for Manchester mode to define the expected symbol threshold levels (refer to *Manchester mode* for details on computation).
In addition this bitfield is used to define the timeout value for the clock absence detection in Normal SPI mode. STH[4:0] values lower than four are invalid.
Note: This bitfield can be write-protected (refer to Section 35.4.15 for details).

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **SITFMOD[1:0]**: Serial interface type
This bitfield is set and cleared by software. it is used to define the serial interface type.
00: LF_MASTER SPI mode
01: Normal SPI mode
10: Manchester mode: rising edge = logic 0, falling edge = logic 1
11: Manchester mode: rising edge = logic 1, falling edge = logic 0
Note: This bitfield can be write-protected (refer to Section 35.4.15 for details).

Bit 3 Reserved, must be kept at reset value.

Bits 2:1 **SCKSRC[1:0]**: Serial clock source
This bitfield is set and cleared by software. it is used to select the clock source of the serial interface.
00: Serial clock source is MDF_CCK0.
01: Serial clock source is MDF_CCK1.
1x: Serial clock source is MDF_CKIx (not allowed in LF_MASTER SPI mode).
Note: This bitfield can be write-protected (refer to Section 35.4.15 for details).

Bit 0 **SITFEN**: Serial interface enable
This bit is et and cleared by software. It is used to enable/disable the serial interface.
0: Serial interface disabled
1: Serial interface enabled

35.8.4 MDF bitstream matrix control register x (MDF_BSMXxCR)

Address offset: 0x084 + 0x80 * x, (x = 0 to 5)

Reset value: 0x0000 0000

This register is used to select the bitstream to be provided to the corresponding digital filter and to the SCD. The number of registers is equal to the amount of filters. Refer to *Section 35.3* for details.

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

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

Bit 31 **BSMXACTIVE**: BSMX active flag

This bit is set and cleared by hardware. It is used by the application to check if the BSMX is effectively enabled (active) or not. BSSEL[4:0] can only be updated when the BSMXACTIVE is set to 0. Th is BSMXACTIVE flag is a logical OR between OLDACTIVE, DFLTACTIVE, and SCDACTIVE flags. Both of them must be set to 0 in order update BSSEL[4:0] bitfield.

0: BSMX is not active and can be configured if needed.

1: BSMX is active and protected fields cannot be configured.

Bits 30:5 Reserved, must be kept at reset value.

Bits 4:0 **BSSEL[4:0]**: Bitstream Selection

This bitfield is set and cleared by software. It is used to select the bitstream to be processed for DFLTx and SCDx. The size of this bitfield depends on the number of DFLTx instantiated. If this bitfield selects a not instantiated input, the MDF selects the valid stream bsx_f having the higher index number.

00000: bs0_r provided to DFLTx and SCDx

00001: bs0_f provided to DFLTx and SCDx

00010: bs1_r provided to DFLTx and SCDx (if instantiated)

00011: bs1_f provided to DFLTx and SCDx (if instantiated)

...

11110: bs15_r provided to DFLTx and SCDx (if instantiated)

11111: bs15_f provided to DFLTx and SCDx (if instantiated)

Note: This bitfield can be write-protected (refer to Section 35.4.15 for details).

35.8.5 MDF digital filter control register x (MDF_DFLTxCR)

Address offset: 0x088 + 0x80 * x, (x = 0 to 5)

Reset value: 0x0000 0000

The number of registers is equal to the amount of filters. Refer to *Section 35.3* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DFLTACTIVE	DFLTRUN	Res.	Res.	NBDIS[7:0]								Res.	Res.	Res.	SNPSFMT
r	r			rw	rw	rw	rw	rw	rw	rw	rw				rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRGSRC[3:0]				Res.	Res.	Res.	TRGSENS	Res.	ACQMOD[2:0]			Res.	FTH	DMAEN	DFLTEN
rw	rw	rw	rw				rw		rw	rw	rw		rw	rw	w

Bit 31 **DFLTACTIVE**: Digital filter active flag

This bit is set and cleared by hardware. It indicates if the digital filter is active: can be running or waiting for events.

0: Digital filter not active (can be re-enabled again, via DFLTEN, if needed)

1: Digital filter active

Bit 30 **DFLTRUN**: Digital filter run status flag

This bit is set and cleared by hardware. It indicates if the digital filter is running or not.

0: Digital filter not running and ready to accept a new trigger event

1: Digital filter running

Bits 29:28 Reserved, must be kept at reset value.

Bits 27:20 **NBDIS[7:0]**: Number of samples to be discarded

This bitfield is set and cleared by software. it is used to define the number of samples to be discarded every time the DFLTx is re-started.

0: No sample discarded

1: 1 sample discarded

2: 2 samples discarded

...

255: 255 samples discarded

Note: This bitfield can be write-protected (refer to Section 35.4.15 for details).

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **SNPSFMT**: Snapshot data format

This bit is set and cleared by software. It is used to select the data format for the snapshot mode.

0: Integrator counter (INT_CNT) not inserted into MDF_SNPSxDR, leaving a data resolution of 23 bits

1: Integrator counter (INT_CNT) inserted at position [15:9] of MDF_SNPSxDR, leaving a data resolution of 16 bits.

Note: This bit can be write-protected (refer to Section 35.4.15 for details).

Bits 15:12 **TRGSRC[3:0]**: Digital filter trigger signal selection

This bitfield is set and cleared by software. It is used to select which external signals trigger the corresponding filter.

0000: TRGO selected

0001: OLDx event selected

0010: mdf_trg[0] selected

...

1111: mdf_trg[13] selected

Note: This bitfield can be write-protected (refer to Section 35.4.15 for details).

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **TRGSENS**: Digital filter trigger sensitivity selection

this bit is set and cleared by software. It is used to select the trigger sensitivity of the external signals

0: A rising edge event triggers the acquisition.

1: A falling edge even triggers the acquisition.

Note: When the trigger source is TRGO or OLDx event, TRGSENS value is not taken into account. When TRGO is selected, the sensitivity is forced to falling edge, when OLDx event is selected, the sensitivity is forced to rising edge.
This bit can be write-protected (refer to Section 35.4.15 for details).

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ACQMOD[2:0]**: Digital filter trigger mode

This bitfield is set and cleared by software. It is used to select the filter trigger mode.

000: Asynchronous continuous acquisition mode

001: Asynchronous single-shot acquisition mode

010: Synchronous continuous acquisition mode

011: Synchronous, single-shot acquisition mode

100: Window continuous acquisition mode

101: Synchronous snapshot mode

Others: same a 000

Note: This bitfield can be write-protected (refer to Section 35.4.15 for details).

Bit 3 Reserved, must be kept at reset value.

Bit 2 **FTH**: RXFIFO threshold selection

This bit is set and cleared by software. It is used to select the RXFIFO threshold. This bit is not significant for RXFIFOs working in interleaved-transfer mode (see *Interleaved-transfer mode* for details).

0: RXFIFO threshold event generated when the RXFIFO is not empty

1: RXFIFO threshold event generated when the RXFIFO is half-full

Note: This bit can be write-protected (refer to Section 35.4.15 for details).

Bit 1 **DMAEN**: DMA requests enable

This bit is set and cleared by software. It is used to control the generation of DMA request to transfer the processed samples into the memory.

0: DMA interface for the corresponding digital filter disabled

1: DMA interface for the corresponding digital filter enabled

Note: This bit can be write-protected (refer to Section 35.4.15 for details).

Bit 0 **DFLTEN**: Digital filter enable

This bit is set and cleared by software. It is used to control the start of acquisition of the corresponding digital filter path. This bit behavior depends on ACQMOD[2:0]and external events. The serial or parallel interface delivering the samples must be enabled as well.

0: Acquisition immediately stopped

1: Acquisition immediately started if ACQMOD[2:0] = 00x or 101, or acquisition started when the proper trigger event occurs if ACQMOD[2:0] = 01x or 100.

35.8.6 MDF digital filter configuration register x (MDF_DFLTxCICR)

Address offset: 0x08C + 0x80 * x, (x = 0 to 5)

Reset value: 0x0000 0000

This register is used to control the main CIC filter.The number of registers is equal to the amount of filters. Refer to *Section 35.3* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	SCALE[5:0]						Res.	Res.	Res.	MCICD8
						rw	rw	rw	rw	rw	rw				rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCICD[7:0]								Res.	CICMOD[2:0]			Res.	Res.	DATSRC[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw			rw	rw

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

Bits 25:20 **SCALE[5:0]**: Scaling factor selection

This bitfield is set and cleared by software. It is used to select the gain to be applied at CIC output (refer to *Table 283* for details). If the application attempts to write a new gain value while the previous one is not yet applied, this new gain value is ignored. Reading back this bitfield informs the application on the current gain value.

000000: 0 dB
000001: + 3.5 dB
000010: + 6 dB or shift left by 1 bit
...
011000: + 72 dB or shift left by 12 bits
100000: - 48.2 dB or shift right by 8 bits (default value)
100001: - 44.6 dB
100010: - 42.1 dB or shift right by 7 bits
100011: - 38.6 dB
...
101110: - 6 dB or shift right by 1 bit
101111: - 2.5 dB
Others: Reserved

Bits 19:17 Reserved, must be kept at reset value.

Bits 16:8 **MCICD[8:0]**: CIC decimation ratio selection

This bitfield is set and cleared by software. It is used to select the CIC decimation ratio. A decimation ratio smaller than two is not allowed. The decimation ratio is given by (CICDEC+1).

0: Decimation ratio is 2.
1: Decimation ratio is 2.
2: Decimation ratio is 3..
3: Decimation ratio is 4
...
511: Decimation ratio is 512.

Note: This bitfield can be write-protected (refer to Section 35.4.15 for details).

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **CICMOD[2:0]**: Select the CIC mode

This bitfield is set and cleared by software.It is used to select the configuration and the order of the MCIC. When CICMOD[2:0] = 0xx, the CIC is split into two filters: the main CIC (MCIC) and the auxiliary CIC (ACIC, used for the out-off limit detector).

000: CIC split in two filters and MCIC configured in FastSinc filter
001: CIC split in two filters and MCIC configured in $Sinc^1$ filter
010: CIC split in two filters and MCIC configured in $Sinc^2$ filter
011: CIC split in two filters and MCIC configured in $Sinc^3$ filter
100: CIC configured in single $Sinc^4$ filter
others: CIC configured in single $Sinc^5$ filter

Note: This bitfield can be write-protected (refer to Section 35.4.15 for details).

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **DATSRC[1:0]**: Source data for the digital filter

This bitfield is set and cleared by software.
0x: Stream coming from the BSMX selected
10: Stream coming from the ADCITF1 selected
11: Stream coming from the ADCITF2 selected

Note: This bitfield can be write-protected (refer to Section 35.4.15 for details).

35.8.7 MDF reshape filter configuration register x (MDF_DFLTxRSFR)

Address offset: 0x090 + 0x80 * x, (x = 0 to 5)

Reset value: 0x0000 0000

This register is used to control the reshape and HPF filters. The number of registers is equal to the amount of filters. Refer to *Section 35.3* for details.

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

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

Bits 9:8 **HPFC[1:0]**: High-pass filter cut-off frequency

This bitfield is set and cleared by software. It is used to select the cut-off frequency of the high-pass filter. F_{PCM} represents the sampling frequency at HPF input.

00: Cut-off frequency = 0.000625 x F_{PCM}

01: Cut-off frequency = 0.00125 x F_{PCM}

10: Cut-off frequency = 0.00250 x F_{PCM}

11: Cut-off frequency = 0.00950 x F_{PCM}

Note: This bitfield can be write-protected (refer to Section 35.4.15 for details).

Bit 7 **HPFBYP**: High-pass filter bypass

This bit is set and cleared by software. It is used to bypass the high-pass filter.

0: HPF not bypassed (default value)

1: HPF bypassed

Note: This bit can be write-protected (refer to Section 35.4.15 for details).

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **RSFLTD**: Reshaper filter decimation ratio

This bit is set and cleared by software. It is used to select the decimation ratio for the reshape filter.

0: Decimation ratio is 4 (default value).

1: Decimation ratio is 1.

Note: This bit can be write-protected (refer to Section 35.4.15 for details).

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **RSFLTBYP**: Reshaper filter bypass

This bit is set and cleared by software. It is used to bypass the reshape filter and its decimation block.

0: Reshape filter not bypassed (default value)

1: Reshape filter bypassed

Note: This bit can be write-protected (refer to Section 35.4.15 for details).

35.8.8 MDF integrator configuration register x (MDF_DFLTxINTR)

Address offset: 0x094 + 0x80 * x, (x = 0 to 5)

Reset value: 0x0000 0000

This register is used to the integrator (INT) settings. The number of registers is equal to the amount of filters. Refer to *Section 35.3* for details.

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

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:4 **INTVAL[6:0]**: Integration value selection

This bitfield is set and cleared by software. It is used to select the integration value.
0: The integration value is 1, meaning bypass mode (default after reset).
1: The integration value is 2.
2: The integration value is 3.
...
127: The integration value is 128.

Note: This bitfield can be write-protected (refer to Section 35.4.15 for details).

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **INTDIV[1:0]**: Integrator output division

This bitfield is set and cleared by software. It is used to rescale the signal at the integrator output in order keep the data width lower than 24 bits.
00: The integrator data outputs are divided by 128 (default value).
01: The integrator data outputs are divided by 32.
10: The integrator data outputs are divided by 4.
11: The integrator data outputs are not divided.

Note: This bitfield can be write-protected (refer to Section 35.4.15 for details).

35.8.9 MDF out-of limit detector control register x (MDF_OLDxCR)

Address offset: 0x098 + 0x80 * x, (x = 0 to 5)

Reset value: 0x0000 0000

This register is used to configure the OLDx.The number of registers is equal to the amount of filters. Refer to *Section 35.3* for details.

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

Bit 31 **OLDACTIVE**: OLDx active flag

This bit is set and cleared by hardware. It is used to check if the OLDx is effectively enabled (active) or not. The protected fields and registers of this function can only be updated when the OLDACTIVE is set to 0 (refer to *Section 35.4.15* for details).

The delay between a transition on OLDEN and a transition on OLDACTIVE is two periods of AHB clock and two periods of mdf_proc_ck.

0: OLDx not active and can be configured if needed

1: OLDx active and protected fields cannot be configured

Bits 30:22 Reserved, must be kept at reset value.

Bits 21:17 **ACICD[4:0]**: OLDx CIC decimation ratio selection

This bitfield is set and cleared by software. It is used to select the decimation ratio of the ACIC. It is only taken into account by the MDF when CICMOD[2:0] = 0xx. The decimation ratio is given by (ACICD + 1).

0: Decimation ratio is 1.

1: Decimation ratio is 2.

2: Decimation ratio is 3.

3: Decimation ratio is 4.

...

31: Decimation ratio is 32.

Note: This bitfield can be write-protected (refer to Section 35.4.15 for details).

Bits 16:14 Reserved, must be kept at reset value.

Bits 13:12 **ACICN[1:0]**: OLDx CIC order selection

This bitfield is set and cleared by software. It is used to select the ACIC type and order. It is only taken into account by the MDF when CICMOD[2:0] = 0xx.

00: FastSinc filter type

01: $Sinc^1$ filter type

10: $Sinc^2$ filter type

11: $Sinc^3$ filter type

Note: This bitfield can be write-protected (refer to Section 35.4.15 for details).

Bits 11:8 Reserved, must be kept at reset value.

Bits 7:4 **BKOLD[3:0]**: Break signal assignment for out-of limit detector

This bitfield is set and cleared by software.

BKOLD[i] = 0: Break signal (mdf_break[i]) not assigned to threshold event

BKOLD[i] = 1: Break signal (mdf_break[i]) assigned to threshold event

Note: This bitfield can be write-protected (refer to Section 35.4.15 for details).

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **THINB**: Threshold In band

This bit is set and cleared by software.

0: The OLDx generates an event if the signal is lower than OLDTHL or higher than OLDTHH (default value).

1: The OLDx generates an event if the signal is lower than OLDTHH and higher than OLDTHL.

Note: This bit can be write-protected (refer to Section 35.4.15 for details).

Bit 0 **OLDEN**: OLDx enable

This bit is set and cleared by software.

0: OLDx disabled (default value)

1: OLDx enabled, including the ACIC filter working in continuous mode

35.8.10 MDF OLDx low threshold register x (MDF_OLDxTHLR)

Address offset: 0x09C + 0x80 * x, (x = 0 to 5)

Reset value: 0x0000 0000

This register is used for the adjustment of the out-off-limit low threshold. The number of registers is equal to the amount of filters. Refer to *Section 35.3* for details.

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

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

Bits 25:0 **OLDTHL[25:0]**: OLD low threshold value

This bitfield is set and cleared by software. OLDTHL represents a 26-bit signed value. The real threshold compared to the signal provided by the filter is OLDTHL.

Note: This bitfield can be write-protected (refer to Section 35.4.15 for details).

35.8.11 MDF OLDx high threshold register x (MDF_OLDxTHHR)

Address offset: 0x0A0 + 0x80 * x, (x = 0 to 5)

Reset value: 0x0000 0000

This register is used for the adjustment of the OLDx high threshold. The number of registers is equal to the amount of filters. Refer to *Section 35.3* for details.

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

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

Bits 25:0 **OLDTHH[25:0]**: OLDx high threshold value

This bitfield is set and cleared by software. OLDTHH represents a 26-bit signed value. The real threshold compared to the signal provided by the filter is OLDTHH.

Note: This bitfield can be write-protected (refer to Section 35.4.15 for details).

35.8.12 MDF delay control register x (MDF_DLYxCR)

Address offset: 0x0A4 + 0x80 * x, (x = 0 to 5)

Reset value: 0x0000 0000

This register is used for the adjustment stream delays. The number of registers is equal to the amount of filters. Refer to *Section 35.3* for details.

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

Bit 31 **SKPBF**: Skip busy flag

This bit is set and cleared by hardware. It is used to control if the delay sequence is completed.

0: MDF ready to accept a new value into SKPDLY[6:0]

1: Last valid SKPDLY[6:0] still under processing

Bits 30:7 Reserved, must be kept at reset value.

Bits 6:0 **SKPDLY[6:0]**: Delay to apply to a bitstream

This bitfield is set and cleared by software. It defines the number of input samples that are skipped. Skipping is applied immediately after writing to this bitfield, if SKPBF = 0 and the corresponding DFLTEN = 1. If SKPBF = 1, the value written into the register is ignored by the delay state machine.

0: No input sample skipped

1: 1 input sample skipped

...

127: 127 input samples skipped

35.8.13 MDF short circuit detector control register x (MDF_SCDxCR)

Address offset: 0x0A8 + 0x80 * x, (x = 0 to 5)

Reset value: 0x0000 0000

This register is used for the adjustment stream delays. The number of registers is equal to the amount of filters. Refer to *Section 35.3* for details.

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

Bit 31 **SCDACTIVE**: SCDx active flag

This bit is set and cleared by hardware. It is used to check if the SCDx is effectively enabled (active) or not. The protected fields of this function can only be updated when the SCDACTIVE is set to 0 (refer to *Section 35.4.15* for details).

The delay between a transition on SCDEN and a transition on SCDACTIVE is two periods of AHB clock and two periods of mdf_proc_ck.

0: SCDx not active and can be configured if needed

1: SCDx active and protected fields cannot be configured

Bits 30:20 Reserved, must be kept at reset value.

Bits 19:12 **SCDT[7:0]**: SCDx threshold

This bitfield is set and cleared by software. These bits are written by software to define the threshold counter for SCDx. If this value is reached, a short-circuit detector event occurs on a given input stream.

0: 2 consecutive 1's or 0's generate an event.

1: 2 consecutive 1's or 0's generate an event.

2: 3 consecutive 1's or 0's generate an event.

...

255: 256 consecutive 1's or 0's generate an event.

Note: This bitfield can be write-protected (refer to Section 35.4.15 for details).

Bits 11:8 Reserved, must be kept at reset value.

Bits 7:4 **BKSCD[3:0]**: Break signal assignment for short circuit detector

This bitfield is set and cleared by software.

BKSCD[i] = 0: Break signal (mdf_break[i]) not assigned to this SCD event

BKSCD[i] = 1: Break signal (mdf_break[i]) assigned to this SCD event

Note: This bitfield can be write-protected (refer to Section 35.4.15 for details).

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **SCDEN**: SCDx enable

This bit is set and cleared by software.

0: SCDx disabled

1: SCDx enabled,

35.8.14 MDF DFLT0 interrupt enable register 0 (MDF_DFLT0IER)

Address offset: 0x0AC

Reset value: 0x0000 0000

This register is used for allowing or not the events to generate an interrupt.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	RFOVRIE	CKABIE	SATIE	SCDIE	SSOVRIE	Res.	Res.	OLDIE	Res.	SSDRIE	DOVRIE	FTHIE
				rw	rw	rw	rw	rw			rw		rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **RFOVRIE**: Reshape filter overrun interrupt enable
This bit is set and cleared by software.
0: Reshape filter overrun interrupt disabled
1: Reshape filter overrun interrupt enabled

Bit 10 **CKABIE**: Clock absence detection interrupt enable
This bit is set and cleared by software.
0: Clock absence interrupt disabled
1: Clock absence interrupt enabled

Bit 9 **SATIE**: Saturation detection interrupt enable
This bit is set and cleared by software.
0: Saturation interrupt disabled
1: Saturation interrupt enabled

Bit 8 **SCDIE**: SCD0 interrupt enable
This bit is set and cleared by software.
0: SCD0 interrupt disabled
1: SCD0 interrupt enabled

Bit 7 **SSOVRIE**: Snapshot overrun interrupt enable
This bit is set and cleared by software.
0: Snapshot overrun interrupt disabled
1: Snapshot overrun interrupt enabled

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **OLDIE**: OLD0 interrupt enable
This bit is set and cleared by software.
0: OLD0 event interrupt disabled
1: OLD0 event interrupt enabled

Bit 3 Reserved, must be kept at reset value.

Bit 2 **SSDRIE**: Snapshot data ready interrupt enable
This bit is set and cleared by software.
0: Snapshot data ready interrupt disabled
1: Snapshot data ready interrupt enabled

Bit 1 **DOVRIE**: Data overflow interrupt enable
This bit is set and cleared by software.
0: Data overflow interrupt disabled
1: Data overflow interrupt enabled

Bit 0 **FTHIE**: RXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: RXFIFO threshold interrupt disabled
1: RXFIFO threshold interrupt enabled

35.8.15 MDF DFLTx interrupt enable register x (MDF_DFLTxIER)

Address offset: 0x12C + 0x80 * (x - 1), (x = 1 to 5)

Reset value: 0x0000 0000

This register is used for allowing or not, the events to generate an interrupt. The number of registers is equal to the amount of filters. Refer to *Section 35.3* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	RFOVRIE	CKABIE	SATIE	SCDIE	SSOVRIE	Res.	Res.	OLDIE	Res.	SSDRIE	DOVRIE	FTHIE
				rw	rw	rw	rw	rw			rw		rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **RFOVRIE**: Reshape filter overrun interrupt enable
This bit is set and cleared by software.
0: Reshape filter overrun interrupt disabled
1: Reshape filter overrun interrupt enabled

Bit 10 **CKABIE**: Clock absence detection interrupt enable
This bit is set and cleared by software.
0: Clock absence interrupt disabled
1: Clock absence interrupt enabled

Bit 9 **SATIE**: Saturation detection interrupt enable
This bit is set and cleared by software.
0: Saturation interrupt disabled
1: Saturation interrupt enabled

Bit 8 **SCDIE**: SCDx interrupt enable
This bit is set and cleared by software.
0: SCDx interrupt disabled
1: SCDx interrupt enabled

Bit 7 **SSOVRIE**: Snapshot overrun interrupt enable
This bit is set and cleared by software.
0: Snapshot overrun interrupt disabled
1: Snapshot overrun interrupt enabled

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **OLDIE**: OLDx interrupt enable
This bit is set and cleared by software.
0: OLDx event interrupt disabled
1: OLDx event interrupt enabled

Bit 3 Reserved, must be kept at reset value.

Bit 2 **SSDRIE**: Snapshot data ready interrupt enable
This bit is set and cleared by software.
0: Snapshot data ready interrupt disabled
1: Snapshot data ready interrupt enabled

Bit 1 **DOVRIE**: Data overflow interrupt enable
This bit is set and cleared by software.
0: Data overflow interrupt disabled
1: Data overflow interrupt enabled

Bit 0 **FTHIE**: RXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: RXFIFO threshold interrupt disabled
1: RXFIFO threshold interrupt enabled

35.8.16 MDF DFLT0 interrupt status register 0 (MDF_DFLT0ISR)

Address offset: 0x0B0

Reset value: 0x0000 0000

This register contains the status flags for each digital filter path.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	RFOVRF	CKABF	SATF	SCDF	SSOVRF	THHF	THLF	OLDF	RXNEF	SSDRF	DOVRF	FTHF
				rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	r	r	rc_w1	r	rc_w1	rc_w1	r

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **RFOVRF**: Reshape filter overrun detection flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no reshape filter overrun is detected. Write 0 has no effect.
1: Read 1 means that reshape filter overrun is detected. Write 1 clears this flag.

Bit 10 **CKABF**: Clock absence detection flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no clock absence is detected. Write 0 has no effect.
1: Read 1 means that a clock absence is detected. Write 1 clears this flag.

Bit 9 **SATF**: Saturation detection flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no saturation is detected. Write 0 has no effect.
1: Read 1 means that a saturation is detected. Write 1 clears this flag.

Bit 8 **SCDF**: Short-circuit detector flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no SCD0 event is detected. Write 0 has no effect.
1: Read 1 means that a SCD0 event is detected. Write 1 clears this flag.

Bit 7 **SSOVRF**: Snapshot overrun flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no snapshot overrun event is detected. Write 0 has no effect.
1: Read 1 means that a snapshot overrun event is detected. Write 1 clears this flag.

Bit 6 **THHF**: High-threshold status flag

This bit is set by hardware and cleared by software by writing this bit to 1. It indicates the status of the high-threshold comparator when the last OLD0 event occurred. This bit gives additional information on the conditions triggering the last OLD0 event. It can be cleared by writing OLDF flag to 1.

0: The signal was lower than OLDTHH when the last OLD0 event occurred.

1: The signal was higher than OLDTHH when the last OLD0 event occurred.

Bit 5 **THLF**: Low-threshold status flag

This bit is set by hardware and cleared by software by writing this bit to 1. It indicates the status of the low-threshold comparator when the last OLD0 event occurred. This bit gives additional information on the conditions triggering the last OLD0 event. It can be cleared by writing OLDF flag to 1.

0: The signal was higher than OLDTHL when the last OLD0 event occurred.

1: The signal was lower than OLDTHL when the last OLD0 event occurred.

Bit 4 **OLDF**: OLD0 flag

This bit is set by hardware and cleared by software by writing this bit to 1.

0: Read 0 means that no OLD0 event is detected. Write 0 has no effect.

1: Read 1 means that an OLD0 event is detected Write 1 clears THHF, THLF and OLDF.

Bit 3 **RXNEF**: RXFIFO not-empty flag

this bit is set and cleared by hardware according to the RXFIFO level.

0: RXFIFO empty

1: RXFIFO not empty

Bit 2 **SSDRF**: Snapshot data ready flag

This bit is set by hardware and cleared by software by writing this bit to 1.

0: Read 0 means that no data is available. Write 0 has no effect.

1: Read 1 means that a new data is available. Write 1 clears this flag.

Bit 1 **DOVRF**: Data overflow flag

This bit is set by hardware and cleared by software by writing this bit to 1.

0: Read 0 means that no overflow is detected. Write 0 has no effect.

1: Read 1 means that an overflow is detected; Write 1 clears this flag.

Bit 0 **FTHF**: RXFIFO threshold flag

This bit is set by hardware and cleared by hardware when the RXFIFO level is lower than the threshold.

0: RXFIFO threshold not reached

1: RXFIFO threshold reached

35.8.17 MDF DFLTx interrupt status register x (MDF_DFLTxISR)

Address offset: 0x130 + 0x80 * (x - 1), (x = 1 to 5)

Reset value: 0x0000 0000

This register contains the status flags for each digital filter path. The number of registers is equal to the amount of filters. Refer to *Section 35.3* for details.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	RFOVRF	CKABF	SATF	SCDF	SSOVRF	THHF	THLF	OLDF	RXNEF	SSDRF	DOVRF	FTHF
				rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	r	r	rc_w1	r	rc_w1	rc_w1	r

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **RFOVRF**: Reshape filter overrun detection flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no reshape filter overrun is detected. Write 0 has no effect.
1: Read 1 means that reshape filter overrun is detected. Write 1 clears this flag.

Bit 10 **CKABF**: Clock absence detection flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no clock absence is detected. Write 0 has no effect.
1: Read 1 means that a clock absence is detected. Write 1 clears this flag.

Bit 9 **SATF**: Saturation detection flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no saturation is detected. Write 0 has no effect.
1: Read 1 means that a saturation is detected. Write 1 clears this flag.

Bit 8 **SCDF**: Short-circuit detector flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no SCD event is detected. Write 0 has no effect.
1: Read 1 means that a SCD event is detected. Write 1 clears this flag.

Bit 7 **SSOVRF**: Snapshot overrun flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no snapshot overrun event is detected. Write 0 has no effect.
1: Read 1 means that a snapshot overrun event is detected. Write 1 clears this flag.

Bit 6 **THHF**: High-threshold status flag
This bit is set by hardware and cleared by software by writing this bit to 1. It indicates the status of the high-threshold comparator when the last OLDx event occurred. This bit gives additional information on the conditions triggering the last OLDx event. It can be cleared by writing OLDF flag to 1.
0: The signal was lower than OLDTHH when the last OLDx event occurred.
1: The signal was higher than OLDTHH when the last OLDx event occurred.

Bit 5 **THLF**: Low-threshold status flag

This bit is set by hardware and cleared by software by writing this bit to 1. It indicates the status of the low-threshold comparator when the last OLDx event occurred. This bit gives additional information on the conditions triggering the last OLDx event. It can be cleared by writing OLDF flag to 1.

0: The signal was higher than OLDTHL when the last OLDx event occurred.

1: The signal was lower than OLDTHL when the last OLDx event occurred.

Bit 4 **OLDF**: OLDx flag

This bit is set by hardware and cleared by software by writing this bit to 1.

0: Read 0 means that no OLDx event is detected. Write 0 has no effect.

1: Read 1 means that an OLDx event is detected Write 1 clears THHF, THLF and OLDF.

Bit 3 **RXNEF**: RXFIFO not-empty flag

this bit is set and cleared by hardware according to the RXFIFO level.

0: RXFIFO empty

1: RXFIFO not empty

Bit 2 **SSDRF**: Snapshot data ready flag

This bit is set by hardware and cleared by software by writing this bit to 1.

0: Read 0 means that no data is available. Write 0 has no effect.

1: Read 1 means that a new data is available. Write 1 clears this flag.

Bit 1 **DOVRF**: Data overflow flag

This bit is set by hardware and cleared by software by writing this bit to 1.

0: Read 0 means that no overflow is detected. Write 0 has no effect.

1: Read 1 means that an overflow is detected; Write 1 clears this flag.

Bit 0 **FTHF**: RXFIFO threshold flag

This bit is set by hardware and cleared by hardware when the RXFIFO level is lower than the threshold.

0: RXFIFO threshold not reached

1: RXFIFO threshold reached

35.8.18 MDF offset error compensation control register x (MDF_OECxCR)

Address offset: 0x0B4 + 0x80 * x, (x = 0 to 5)

Reset value: 0x0000 0000

This register contains the offset compensation value. The number of registers is equal to the amount of filters. Refer to *Section 35.3* for details.

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

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

Bits 25:0 **OFFSET[25:0]**: Offset error compensation
This bitfield is set and cleared by software. If the application attempts to write a new offset value while the previous one is not yet applied, this new offset value is ignored. Reading back this bitfield informs the application on the current offset value.
This bitfield represents the value to be subtracted to the signal before going to the SCALE.

35.8.19 MDF snapshot data register x (MDF_SNPSxDR)

Address offset: 0x0EC + 0x80 * x, (x = 0 to 5)

Reset value: 0x0000 0000

This register is used to read the data processed by each digital filter in snapshot mode. The number of registers is equal to the amount of filters. Refer to *Section 35.3* for details.

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

Bits 31:16 **SDR[15:0]**: Contains the 16 MSB of the last valid data processed by the digital filter.

Bits 15:9 **EXTSDR[6:0]**: Extended data size
If SNPSFMT = 0, this bitfield contains the bits 7 to 1 of the last valid data processed by the digital filter.
If SNPSFMT = 1, this bitfield contains the INT accumulator counter value when the last trigger event occurs (INT_CNT).

Bits 8:0 **MCICDC[8:0]**: Contains the MCIC decimation counter value when the last trigger event occurs (MCIC_CNT)

35.8.20 MDF digital filter data register x (MDF_DFLTxDR)

Address offset: 0x0F0 + 0x80 * x, (x = 0 to 5)

Reset value: 0x0000 0000

This register is used to read the data processed by each digital filter. The number of registers is equal to the amount of filters. Refer to *Section 35.3* for details.

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

Bits 31:8 **DR[23:0]**: Data processed by digital filter

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

35.8.21 MDF register map

Table 293. MDF register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	MDF_GCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ILVNB [3:0]				Res.	Res.	Res.	TRGO
	Reset value																									0	0	0	0				0
0x004	MDF_CKGCR	CKGACTIVE	PROCDIV[6:0]							Res.	Res.	Res.	Res.	CCKDIV [3:0]			TRGSRC [3:0]					Res.	Res.	Res.	TRGSENS	Res.	CCK1DIR	CCK0DIR	CKGMOD	Res.	CCK1EN	CCK0EN	CKGDEN
	Reset value	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0				0		0	0	0		0	0	0
0x008-0x07C	Reserved																																
0x080 + 0x80 * x (x = 0 to 5)	MDF_SITFxCR	SITFACTIVE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STH[4:0]				Res.	Res.	SITFMOD [1:0]		Res.	SCKSRC [1:0]		SITFEN
	Reset value	0																			1	1	1	1	1			0	0		0	0	0
0x084 + 0x80 * x (x = 0 to 5)	MDF_BSMXxCR	BSMXACTIVE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BSSEL[4:0]				
	Reset value	0																											0	0	0	0	0
0x088 + 0x80 * x (x = 0 to 5)	MDF_DFLTxCR	DFLTACTIVE	DFLTRUN	Res.	Res.	NBDIS[7:0]								Res.	Res.	Res.	SNPSFMT	TRGSRC [3:0]				Res.	Res.	Res.	TRGSENS	Res.	ACQMOD [2:0]			Res.	FTH	DMAEN	DFLTEN
	Reset value	0	0			0	0	0	0	0	0	0	0				0	0	0	0	0				0		0	0	0		0	0	0
0x08C + 0x80 * x (x = 0 to 5)	MDF_DFLTxCICR	Res.	Res.	Res.	Res.	Res.	Res.	SCALE[5:0]						Res.	Res.	Res.	MCICD[8:0]									Res.	CICMOD [2:0]			Res.	Res.	DATSRC [1:0]	
	Reset value							0	0	0	0	0	0				0	0	0	0	0	0	0	0	0		0	0	0			0	0
0x090 + 0x80 * x (x = 0 to 5)	MDF_ DFLTxRSFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HPFC[1:0]		HPFBYP	Res.	Res.	RSFLTD	Res.	Res.	Res.	RSFLTBYP
	Reset value																							0	0	0			0				0
0x094 + 0x80 * x (x = 0 to 5)	MDF_DFLTxINTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTVAL[6:0]							Res.	Res.	INTDIV [1:0]	
	Reset value																						0	0	0	0	0	0	0			0	0
0x098 + 0x80 * x (x = 0 to 5)	MDF_OLDxCR	OLDACTIVE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACICD[4:0]					Res.	Res.	Res.	ACICN [1:0]		Res.	Res.	Res.	Res.	BKOLD [3:0]				Res.	Res.	THINB	OLDEN
	Reset value	0										0	0	0	0	0				0	0					0	0	0	0			0	0

Table 293. MDF register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x09C + 0x80 * x (x = 0 to 5)	MDF_OLDxTHLR	Res.	Res.	Res.	Res.	Res.	Res.	OLDTHL[25:0]																									
	Reset value							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A0 + 0x80 * x (x = 0 to 5)	MDF_OLDxTHHR	Res.	Res.	Res.	Res.	Res.	Res.	OLDTHH[25:0]																									
	Reset value							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A4 + 0x80 * x (x = 0 to 5)	MDF_DLYxCR	SKPBF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SKPDLY[6:0]						
	Reset value	0																									0	0	0	0	0	0	0
0x0A8 + 0x80 * x (x = 0 to 5)	MDF_SCDxCR	SCDACTIVE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SCDT[7:0]								Res.	Res.	Res.	Res.	BKSCD [3:0]				Res.	Res.	Res.	SCDEN
	Reset value	0												0	0	0	0	0	0	0	0					0	0	0	0				0
0x0AC	MDF_DFLT0IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFOVRIE	CKABIE	SATIE	SCDIE	SSOVRIE	Res.	Res.	OLDIE	Res.	SSDRIE	DOVRIE	FTHIE
	Reset value																					0	0	0	0	0			0		0	0	0
0x0B0	MDF_DFLT0ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFOVRF	CKABF	SATF	SCDF	SSOVRF	THHF	THLF	OLDF	RXNEF	SSDRF	DOVRF	FTHF
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x0B4 + 0x80 * x (x = 0 to 5)	MDF_OECxCR	Res.	Res.	Res.	Res.	Res.	Res.	OFFSET[25:0]																									
	Reset value							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B8-0x0E8	Reserved																																
0x0EC + 0x80 * x (x = 0 to 5)	MDF_SNPSxDR	SDR[15:0]																EXTSDR[6:0]							MCICDC[8:0]								
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0F0 + 0x80 * x (x = 0 to 5)	MDF_DFLTxDR	DR[23:0]																							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								
0x12C + 0x80*(x-1) (x=1 to 5)	MDF_DFLTxIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFOVRIE	CKABIE	SATIE	SCDIE	SSOVRIE	Res.	Res.	OLDIE	Res.	SSDRIE	DOVRIE	FTHIE
	Reset value																					0	0	0	0	0			0		0	0	0
0x130 + 0x80*(x-1) (x=1 to 5)	MDF_DFLTxISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFOVRF	CKABF	SATF	SCDF	SSOVRF	THHF	THLF	OLDF	RXNEF	SSDRF	DOVRF	FTHF
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

36 Audio digital filter (ADF)

36.1 ADF introduction

The audio digital filter (ADF) is a high-performance module dedicated to the connection of external sigma-delta (∑Δ) modulators. It is mainly targeted for the following applications:

- audio capture signals
- metering

The ADF features one digital serial interface (SITF0) and one digital filter (DFLT0) with flexible digital processing options in order to offer up to 24-bit final resolution.

The ADF serial interface supports several standards allowing the connection of various ∑Δ modulator sensors:

- SPI interface
- Manchester coded 1-wire interface
- PDM interface

The ADF converts an input data stream into clean decimated digital data words. This conversion is done thanks to low-pass digital filters and decimation blocks. In addition, it is possible to insert a high-pass filter.

The conversion speed and resolution are adjustable according to configurable parameters for digital processing: filter bypass, filter order, decimation ratio. The maximum output data resolution is up to 24 bits. There are two conversion modes: single conversion and continuous modes. The data can be automatically stored in a system RAM buffer through DMA, thus reducing the software overhead.

A sound activity detector (SAD) is available for the detection of sounds or voice signals. The SAD is connected at the output of the DFLT0. Several parameters can be programmed in order to adjust properly the SAD to the sound environment. The SAD strongly reduces the power consumption by preventing the storage of samples into the system memory, as long as the observed signal does not match the programmed criteria.

The digital processing is performed using only the kernel clock. The ADF requests the bus interface clock (AHB clock) only when data must be transfered or when a specific event requests the attention of the system processor.

36.2 ADF main features

- AHB Interface
- 1 serial digital input:
  - configurable SPI interface to connect various digital sensors
  - configurable Manchester coded interface support
  - compatible with PDM interface to support digital microphones
- 2 common clocks input/output for ∑Δ modulators
- 1 flexible digital filter path including:
  - A MCIC filter configurable in $Sinc^4$ or $Sinc^5$ filter with an adjustable decimation ratio
  - A reshape filter to improve the out-off band rejection and in-band ripple

  - A high-pass filter to cancel the DC offset
  - Gain control
  - Saturation blocks
- Clock absence detector
- Sound activity detector
- 24-bit signed output data resolution
- Continuous or single conversion
- Possibility to delay the selected bitstream
- One trigger input
- Autonomous functionality in Stop mode(s)
- DMA can be used to read the conversion data
- Interrupts services

36.3 ADF implementation

The devices embed one MDF instance and one ADF instance, both being digital filters with common features.

Table 294. ADF features (1)

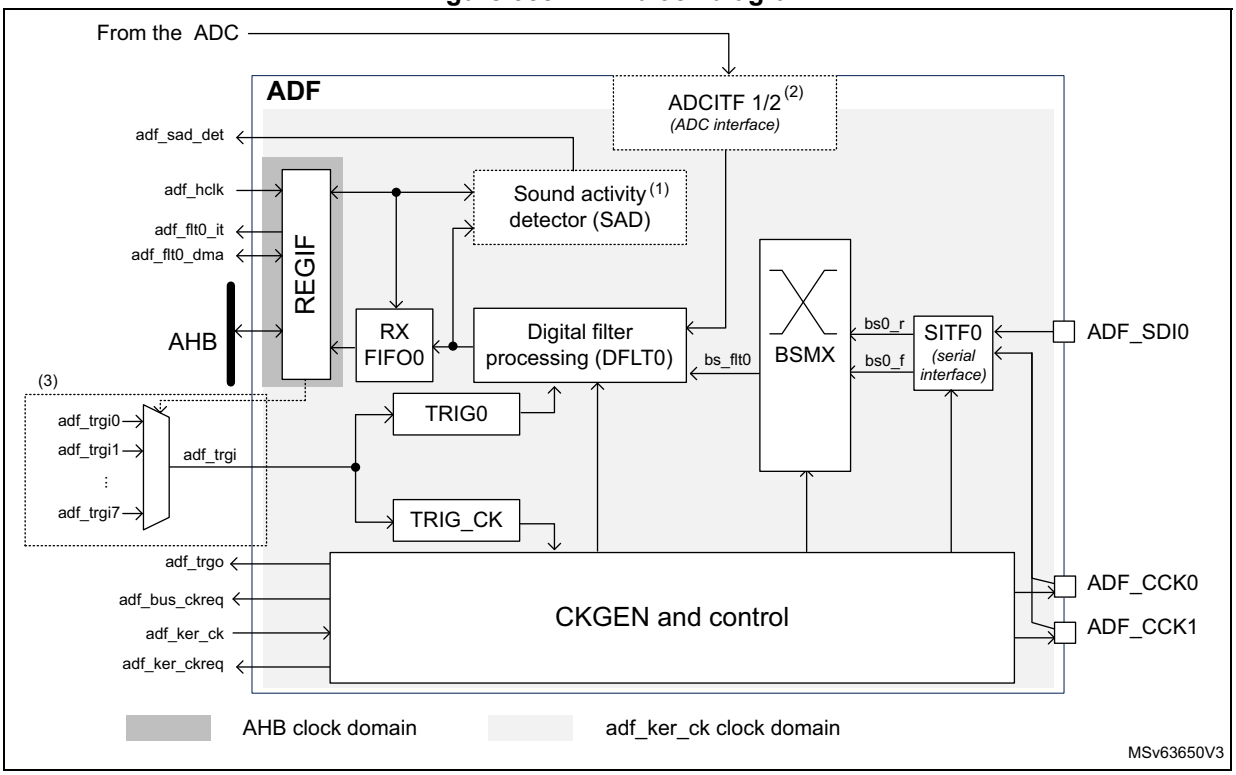
ADF modes/features	ADF
Number of filters (DFLT) and serial interfaces (SITF)	1
ADF_CKI[x] connected to pins	-
Sound activity detection (SAD)	X
RXFIFO depth (number of 24-bit words)	4
Number of trigger inputs	1
ADC connected to ADCITF1	-
ADC connected to ADCITF2	-
Motor dedicated features (SCD, OLD, OEC, INT, snapshot, break)	-
Main path with CIC4, CIC5	X
Main path with CIC1,2, 3 or FastSinc	-
RSFLT, HPF, SAT, SCALE, DLY, Discard functions	X
Global interrupts ADF_FLT[0:0] (adf_flt_it[0:0])	X
Receive interrupts ADF_RX[0:0] (adf_rx_it[0:0])	-
Event interrupts ADF_EVT[0:0] (adf_evt_it[0:0])	-
Autonomous in Stop mode	X(2)
Only DMA of filter 0 connected	-
All DMA channels connected	X

1. ‘X’ = supported, ‘-’ = not supported.
2. Only in Stop SVOS high mode.

36.4 ADF functional description

36.4.1 ADF block diagram

Figure 355. ADF block diagram



1. Refer to *Section 36.3: ADF implementation* to check if the SAD is available.
2. Refer to *Section 36.3: ADF implementation* to check if the ADCITF is available, and which ADCs are connected.
3. The number of trigger inputs depends on the product. Refer to *Table 297* for details.

36.4.2 ADF pins and internal signals

Table 295. ADF external pins

Pin name	Pin type	Remarks
ADF_SDI0	Input	Data signal from external sensors
ADF_CCKy (y = 0,1)	Input/output	Clock outputs for external sensor, or common clock input from external sensors

Table 296. ADF internal signals

Signal name	Signal type	Remarks
adf_trgi	Input	Trigger inputs to control the acquisition (see *Table 297: ADF trigger connections* for details)
adf_trgo	Output	Trigger output for synchronizing with other MDF instances
adf_flt0_dma	Input/output	DMA request/acknowledge signals for the ADF processing chain
adf_flt0_it	Output	Global interrupt signals
adf_bus_ckreq	Output	Bus interface clock request output
adf_ker_ckreq	Output	Kernel clock request output
adf_ker_ck	Input	Kernel clock input
adf_hclk	Input	AHB bus interface clock input
adf_sad_det	Output	SAD sound detection: 1 means that detecting sound
adf_adcitf1_dat[15:0]	Input	ADCITF1 data input
adf_adcitf2_dat[15:0]	Input	ADCITF2 data input

Table 297. ADF trigger connections

Trigger name	Direction	Trigger source/destination
adf_trgi	Input	From exti15
adf_trgo	Output	mdf_trgi13
adf_sad_det	Output	mdf_trgi12

36.4.3 Serial input interface (SITF)

The SITF0 input interface allows the connection of the external sensor to the digital filter via the bitstream matrix (BSMX). The SITF0 can be configured in the following modes:

- LF_MASTER SPI mode (low-frequency)
- normal SPI mode
- Manchester mode

The data from the serial interface is routed to the filter in order to perform the PDM to PCM conversion and the sound activity detection.

The serial interface is enabled by setting the SITFEN bit to 1. Once the interface is enabled, it receives serial data from the external ∑Δ modulator.

Note: Before enabling the *serial interface, the user must insure that the adf_proc_ck is already enabled (see Section 36.4.5: Clock generator (CKGEN) for details).*

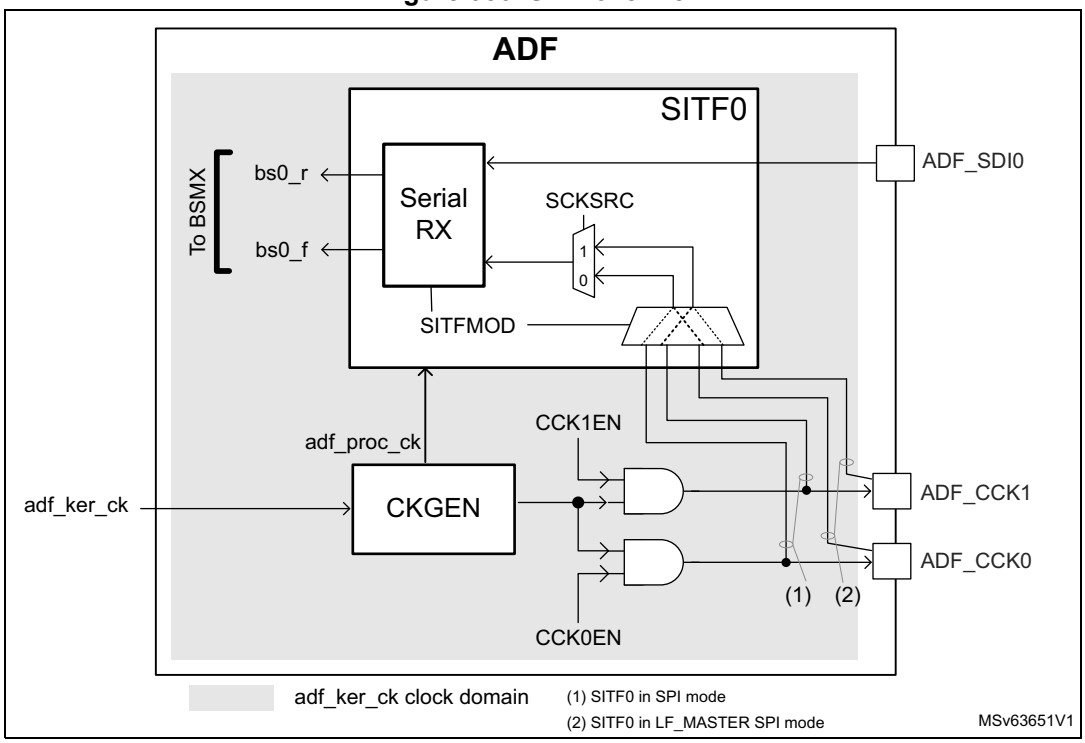
The SITF0 is controlled via the *ADF serial interface control register 0 (ADF_SITF0CR)*.

As shown in the *Figure 356*, ADF_CCK0 or ADF_CCK1 can be selected as clock source, in order to sample the incoming bitstream:

- If the serial interface is programmed in SPI mode, the selected clock source is a copy of the clock present on the ADF_CCK0 or ADF_CCK1 pin.
- If the serial interface is programmed in LF_MASTER SPI mode, the selected clock source is the clock directly provided by the CCKDIV to the ADF_CCK0 or ADF_CCK1 pin.

See *Table 298* for additional information.

Figure 356. SITF overview



LF_MASTER and normal SPI modes

The LF_MASTER SPI mode is a special mode allowing the use of an adf_proc_ck clock frequency, only two times bigger than the sensor clock. This mode is dedicated to low-power use-cases, using low-speed sensors.

In LF_MASTER SPI mode, the ADF must provide the bitstream clock to the external sensors via ADF_CCK0 and ADF_CCK1 pins. The ADF receives the bitstream data via the serial data input ADF_SDI0.

For the SITF0, the application must select the same clock than the one provided to the external sensor (ADF_CCK0 or ADF_CCK1), in order to guarantee optimal timing performances. This selection is done via SCKSRC[1:0].

The normal SPI interface is a more flexible interface than the LF_MASTER SPI, but the adf_proc_ck frequency must be at least four times higher than the sensor clock.

The application can select ADF_CCK0 or ADF_CCK1 clock for the capture of the data received via the ADF_SDI0 pin.

The ADF can generate a clock to the sensors via ADF_CCK0 or ADF_CCK1 if needed.

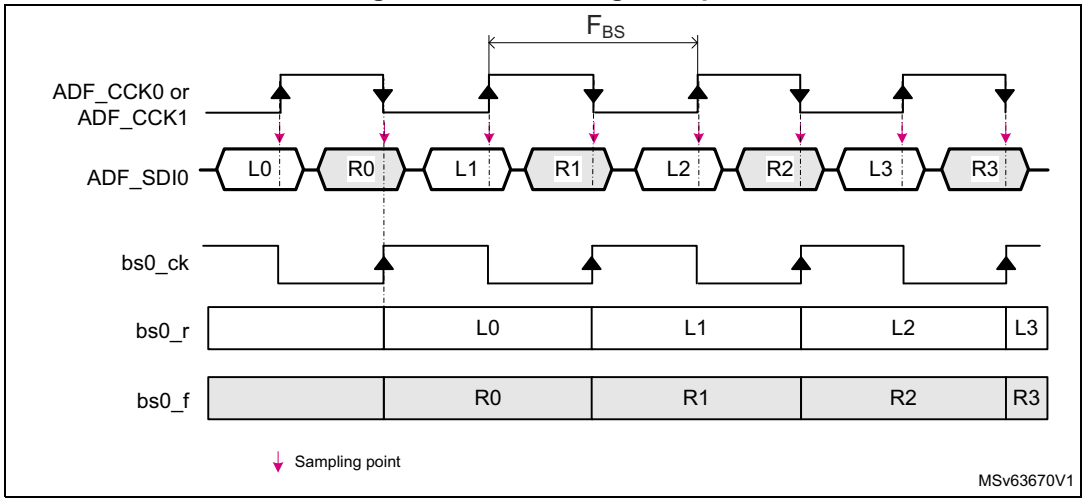
For all SPI modes, the serial data is captured using the rising and the falling edge of the selected clock. The SITF0 always provides the following bitstreams:

- bitstream received using the bitstream clock falling edge (bs0_f)
- bitstream received using the bitstream clock rising edge (bs0_r)

According to the sensors connected, one of the two bitstreams may not be available.

The application can select the wanted stream via the BSMX matrix.

Figure 357. SPI timing example



To properly synchronize/receive the data stream, the adf_proc_ck frequency must be adjusted according to the constraints listed in *Table 299*.

Clock absence detection

A no-clock-transition period may be detected when the serial interface works in normal SPI mode. This feature can be used to detect a clock failure in the SPI link.

The application can program a timeout value via the STH[4:0] bitfield of the SITF0. If the ADF does not detect clock transitions for a duration of STH[4:0] x $T_{adf_proc_ck}$, then the CKABF flag is set.

An interrupt can be generated if CKABIE is set to 1. The STH[4:0] bitfield is in the *ADF serial interface control register 0 (ADF_SITF0CR)*.

When the serial interface is enabled, the CKABF flag remains to 1 until a first clock transition is detected.

To avoid spurious clock absence detection, the following sequence must be respected:

1. Configure the serial interface in normal SPI mode and enable it.
2. Clear the CKABF flag by writing CKABF bit to 1.

   If no clock transition is detected on the serial interface, the hardware immediately sets the CKABF flag to 1.

3. Read the CKABF flag:
   – If CKABF = 1, go back to step 2.
   – If CKABF = 0, a clock has been detected. The CKABIE bit can be set to 1 if the application wants an interrupt on detection of a clock absence.

Note: The clock absence detection feature is not available in the LF_MASTER SPI mode.

Manchester mode

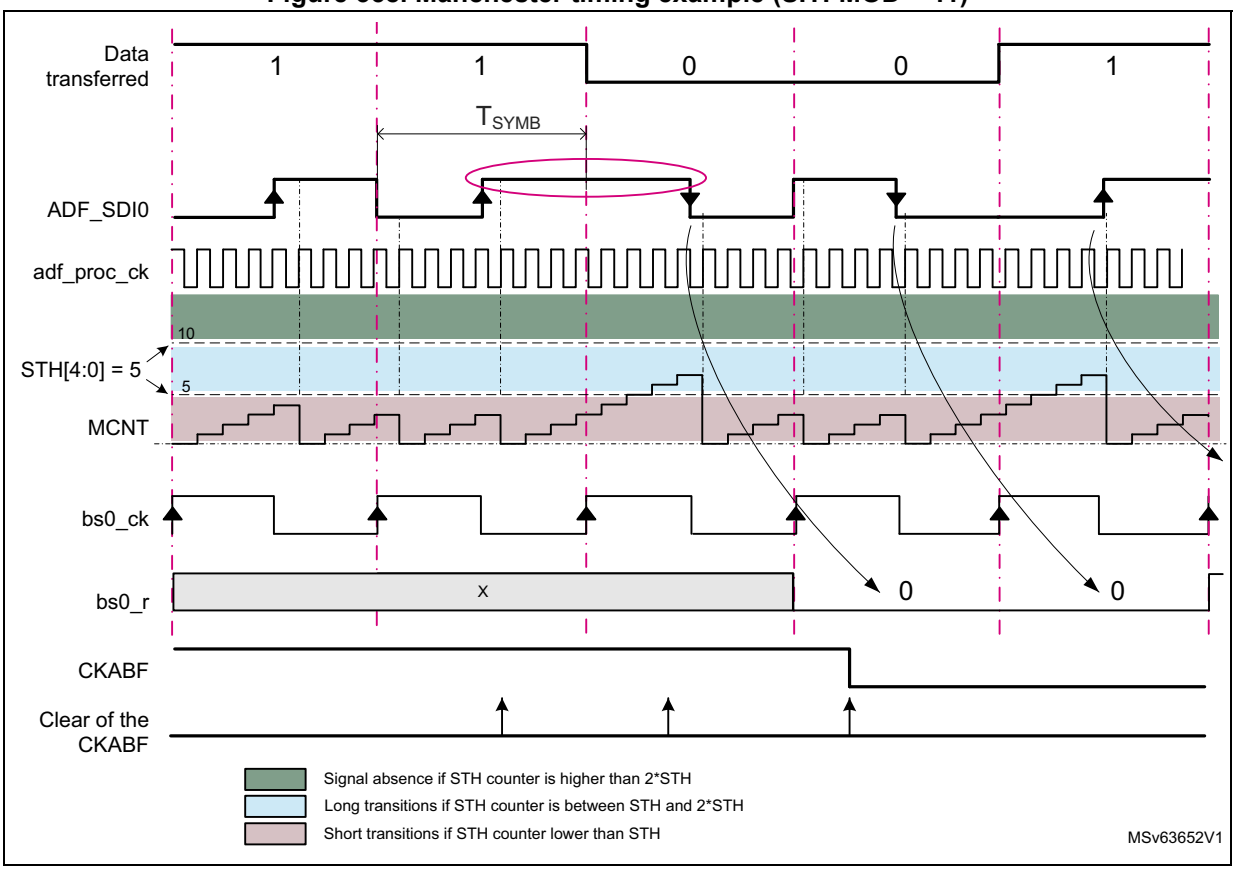
In Manchester coded format, the ADF receives data stream from the external sensor via the ADF_SDI0 pin only.

The ADF_CCK0 and ADF_CCK1 pins are not needed in this mode.

Decoded data and clock signals are recovered from serial stream after Manchester decoding. They are available on bs0_r. There are two possible settings of Manchester codings:

- signal rising edge decoded as 0 and signal falling edge decoded as 1
- signal rising edge decoded as 1 and signal falling edge decoded as 0

Figure 358. Manchester timing example (SITFMOD = 11)



To decode the incoming Manchester stream, the user must program STH[4:0] in the *ADF serial interface control register 0 (ADF_SITF0CR)*. The STH[4:0] bitfield is used by the SITF0 to estimate the Manchester symbol length and to detect a clock absence. An internal counter (MCNT) is restarted every time a transition is detected in the ADF_SDI0 input. It is used to detect short transitions, long transitions or clock absence. A long transition indicates

that the data value changed. *Figure 358* shows a case where the OVR is around height and STH[4:0] = 5.

The estimated Manchester symbol rate (T_{SYMB}) must respect the following formula:

$$(STH + 1) \times T_{adf_proc_ck} < T_{SYMB} < (2 \times STH \times T_{adf_proc_ck})$$

It is recommended to compute STH as follows:

$$STH[4:0] = round\left(\frac{(2 \times OVR) - 1}{3}\right)$$

where OVR represents the ratio between the adf_proc_ck frequency and the expected Manchester symbol frequency. OVR must be higher than five, and the adf_proc_ck clock must be adjusted according to the constraints listed in *Table 299*.

The clock absence flag CKABF is set to 1 when no transition is detected during more than 2 x STH[4:0] x $T_{adf_proc_ck}$, or when the SITF0 is not yet synchronized to the incoming Manchester stream. In addition, an interrupt can be generated if the bit CKABIE is set to 1.

When the serial interface is enabled, the ADF must first be synchronized to the incoming Manchester stream. The synchronization ends when a data transition from 0 to 1 or from 1 to 0 (pink circle in the *Figure 358*) is detected.

The end of the synchronization phase can be checked by following the software sequence:

1. Clear the CKABF flag in the *ADF DFLT0 interrupt status register 0 (ADF_DFLT0ISR)* by writing CKABF bit to 1. If the serial interface is not yet synchronized the hardware immediately set the CKABF flag to 1.
2. Read the CKABF flag:
   – If CKABF= 1, go back to step 1.
   – If CKABF = 0, the Manchester interface is synchronized and provides valid data.

Programming example

In the following example, the ADF kernel clock frequency ($F_{adf_ker_ck}$) is 100 MHz and the received Manchester stream is at about 6 MHz (F_{SYMB}):

1. Provide a valid adf_proc_ck to the SITF0.

   The adf_proc_ck frequency must be at least six times higher than the Manchester symbol frequency (means at least 36 MHz).

   PROCDIV is programmed to 1 to perform a division by two of the kernel clock. In that case, $F_{adf_proc_ck}$ = 50 MHz (8.33 times higher than the Manchester symbol frequency).
2. Compute STH.

   OVR is given by: OVR = $F_{adf_proc_ck}$ / F_{SYMB} = 50 MHz / 6 MHz = 8.33.

   Then $STH[4:0] = round\left(\frac{(2 \times 8.33) - 1}{3}\right) = 5$

   The minimum allowed frequency for the Manchester stream is then:

   1 / (2 × STH × $T_{adf_proc_ck}$) = 1 / (10 x 20 ns) = 5 MHz

   The maximum allowed frequency for the Manchester stream is then:

   1 / ((STH+1) × $T_{adf_proc_ck}$) = 1 / (6 x 20 ns) = 8.33 MHz

36.4.4 ADC slave interface (ADCITF)

The ADCs are not always connected to the ADF. Refer to *Section 36.3* to check the situation for this product.

The ADF allows the connection of up to two ADCs to the filter path. For the filter, the DATSRC[1:0] bitfield in the *ADF digital filer configuration register 0 (ADF_DFLT0CICR)* allows the application to select data from the ADCs.

Warning: The ADF does not support receiving interleaved data from one of the ADCITF input.

36.4.5 Clock generator (CKGEN)

The RCC (reset and clock controller) provides the following clocks to the ADF:

- AHB clock (adf_hclk) used for the register interface
- kernel clock (adf_ker_ck) mainly used by all other parts of the circuit via the CKGEN

These clocks are not supposed to be phase locked, so all signals crossing those clock domains are resynchronized.

The clock generator (CKGEN) is responsible of the generation of the processing clock, and the clock provided to the ADF_CCK0 and ADF_CCK1 pins. All those clocks are generated from the adf_ker_ck.

The processing clock (adf_proc_ck) is used to run all the signal processing and to re-sample the incoming serial or parallel stream.

To adapt the kernel clock frequency provided by the RCC, the following dividers are available:

- PROCDIV[6:0] used to adapt the kernel clock frequency to the constraints of the parallel and serial interfaces, and to the processing blocks
- CCKDIV[3:0] used to adapt the frequency of the ADF_CCK0 and ADF_CCK1 clocks

PROCDIV[6:0] and CCKDIV[3:0] must be programmed when no clock is provided to the dividers (CKGDEN = 0).

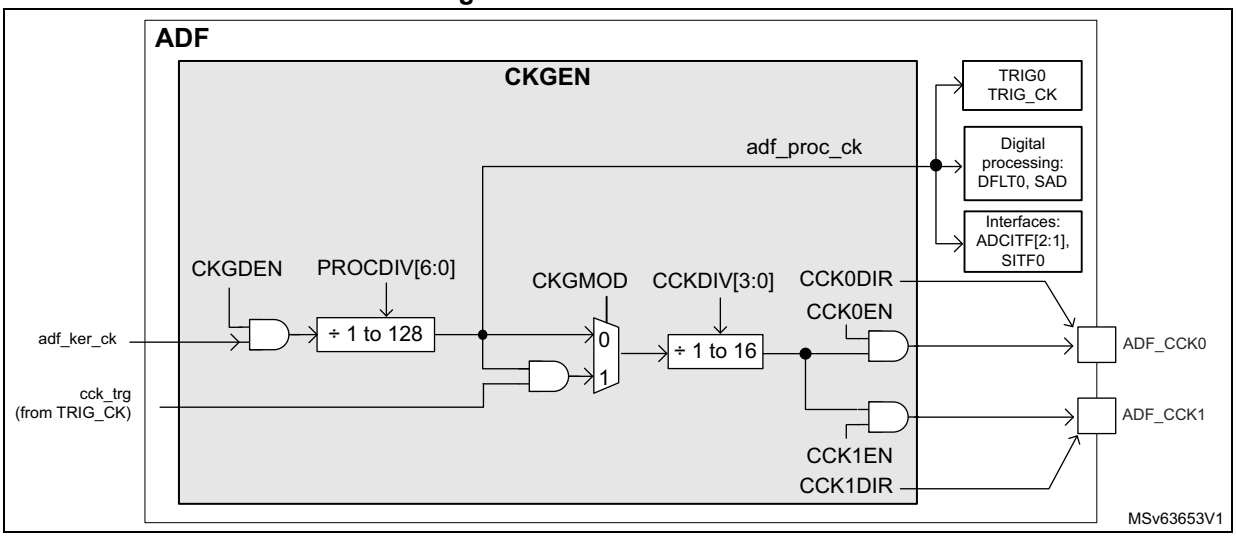
The adf_proc_ck generation is controlled by CKGDEN.

In addition, the CKGMOD bit allows the application to define the way to trigger the CCKDIV divider:

- When CKGMOD = 0, the CCKDIV divider is started as soon as CKGDEN is set to 1.
- When CKGMOD = 1, the CCKDIV divider is started when CKGDEN is set to 1 and the programmed trigger condition occurred.

All the bits and fields controlling the CKGEN are in the *ADF clock generator control register (ADF_CKGCR)*.

Figure 359. CKGEN overview



The trigger logic for CKGEN is handled by the block TRG_CK. As shown in *Figure 364*, the CCKDIV divider can be triggered on the rising or falling edge of an external trigger source. When the proper trigger condition occurs, the cck_trg signal goes to high, allowing the CCKDIV divider to start. The TRG_CK logic is reset when CKGDEN is set to 0.

This feature can be helpful to synchronize the ADF_CCKy (y = 0,1) clock of several ADF instances, or to synchronize the clock generation to a timer event.

The application can control the activation of the ADF_CCK0 or ADF_CCK1 pin thanks to CCK0EN/CCK1EN and CCK0DIR/CCK1DIR bits:

- CCKyEN is used to enable the CCKDIV, and thus generates a clock for the external sensors.
- CCKyDIR is used to control the direction of the ADF_CCKy pin (input or output)

Table 298. Control of the common clock generation(1)

CCKyEN	CCKyDIR	Description
0	0	The ADF_CCKy pin is in input. An external clock can be connected to the ADF_CCKy pin and used by the SITF0 in order to decode the serial stream
0	1	The ADF_CCKy pin is in output. No clock is generated. The ADF_CCKy pin is driven low.
1	1	The ADF_CCKy pin is in output. A clock is generated on the ADF_CCKy pin. The SITF0 can use this pin as clock source in order to decode the serial stream

1. The configuration with CCKyEN = 1 and CCyDIR = 0 is not shown must be avoided (no interest).

Note: The adf_proc_ck must be enabled (by CKGDEN = 1) before enabling other blocks (such as SITF0 or DFLT0).

CKGEN activation sequence example

1. Set CKGDEN to 0.
2. Wait for CKGACTIVE = 0. If CKGDEN was previously enabled, this phase can take two periods of adf_hclk, and two periods of adf_proc_ck.
3. Program PROCDIV[6:0], CKGMOD, CCKDIV[3:0], TRGSRC[3:0], TRGSENS, CCK1EN and CCK0EN.
4. Set CKGDEN to 1.

When needed, at any moment, CCK[1:0]EN bitfield value can be changed without disabling the clock generator.

Clock frequency constraints

Table 299 shows the frequency constraints to receive and process properly the samples.

Note: The reshape filter (RSFLT) needs up to 24 cycles of adf_proc_ck clock to process one sample.

Table 299. Clock constraints with respect to the incoming stream(1)

SITF0 mode	ADF clock constraints	
	With RSFLT disabled	With RSFLT enabled
LF_MASTER SPI	F_{ADF_CCKy} max frequency limited to 5 MHz	
	$F_{adf_proc_ck} > 2 \times F_{ADF_CCKy}$ and $F_{adf_hclk} \geq F_{adf_proc_ck}$	$F_{adf_proc_ck} > 24 \times F_{ADF_CCKy} / (MCICD+1)$ and $F_{adf_proc_ck} > 2 \times F_{ADF_CCKy}$ and $F_{adf_hclk} \geq F_{adf_proc_ck}$
MASTER SPI SLAVE SPI	F_{ADF_CKx} max frequency limited to 25 MHz	
	$F_{adf_proc_ck} > 4 \times F_{ADF_CCKy}$ and $F_{adf_hclk} \geq F_{adf_proc_ck}$	$F_{adf_proc_ck} > 24 \times F_{ADF_CCKy} / (MCICD+1)$ and $F_{adf_proc_ck} > 4 \times F_{ADF_CCKy}$ and $F_{adf_hclk} \geq F_{adf_proc_ck}$
Manchester	F_{SYMB} max frequency limited to 20 MHz	
	$F_{adf_proc_ck} > 6 \times F_{SYMB}$ and $F_{adf_hclk} \geq F_{adf_proc_ck}$	$F_{adf_proc_ck} > 24 \times F_{ADF_CCKy} / (MCICD+1)$ and $F_{adf_proc_ck} > 6 \times F_{SYMB}$ and $F_{adf_hclk} \geq F_{adf_proc_ck}$

1. F_{ADF_CCKy} represents the frequency of clock received via ADF_CCKy, or generated via ADF_CCKy. F_{SYMB} represents the frequency of the received symbol rate for Manchester mode.

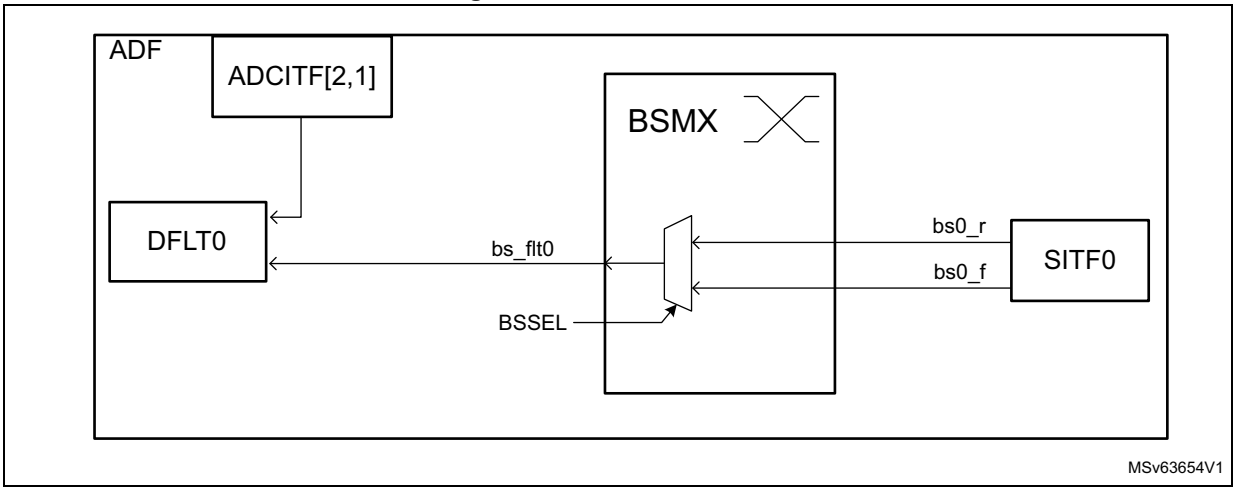
36.4.6 Bitstream matrix (BSMX)

The BSMX receives the bitstreams from the serial interface SITF0 and provides the selected stream to the digital filter DFLT0.

As shown in *Figure 355*, the SITF0 provides two bitstreams (bs0_r and bs0_f) to the BSMX.

The application to select the wanted stream via the *ADF bitstream matrix control register 0 (ADF_BSMX0CR)*. This selection is intended to be static.

Figure 360. BSMX overview



BSMX programming sequence example

The BSSEL[4:0] bitfield cannot be changed if the DFLT0 is enabled. The following steps are needed to change the value of BSMX:

1. Set DFLTEN of DFLT0 to 0.
2. Wait for BSMXACTIVE = 0.
3. Program BSSEL[4:0].
4. Set DFLTEN of DFLT0 to 1.

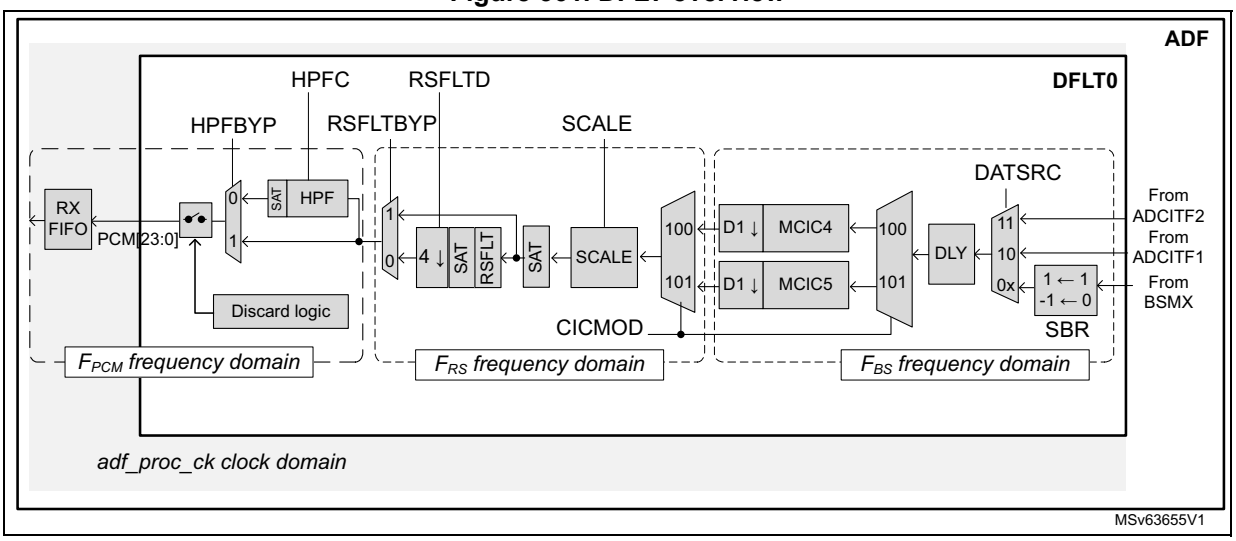
36.4.7 Digital filter processing (DFLT)

The digital filter processing includes the following sub-blocks:

- symbol remap (SBR)
- clock skipper delay (DLY)
- MCIC decimation filter that can be configured in $Sinc^4$ or $Sinc^5$
- gain control (SCALE)
- signal saturation (SAT)
- reshape filter (RSFLT)
- high-pass filter (HPF)
- receive RXFIFO

Figure 361 shows the filter path configuration according to CICMOD[2:0]. Several configuration bits are available to configure the digital filter to the application needs.

Figure 361. DFLT overview



Symbol remap and source selection

The symbol remap (SBR) converts the bitstream selected by the BSMX into data usable by the filter path. More especially:

- The high levels are converted into a 16-bit signed number + 1.
- The low levels are converted into a 16-bit signed number - 1.

The signal source of the digital filter can be selected via DATSRC[1:0] between the two following:

- data coming from the BSMX
- data coming from one of the ADC interfaces (ADCITF2 or 1)

Programmable micro-delay control (DLY)

The digital filter has a delay line that allows the timing adjustment of each stream with the resolution of the bitstream clock.

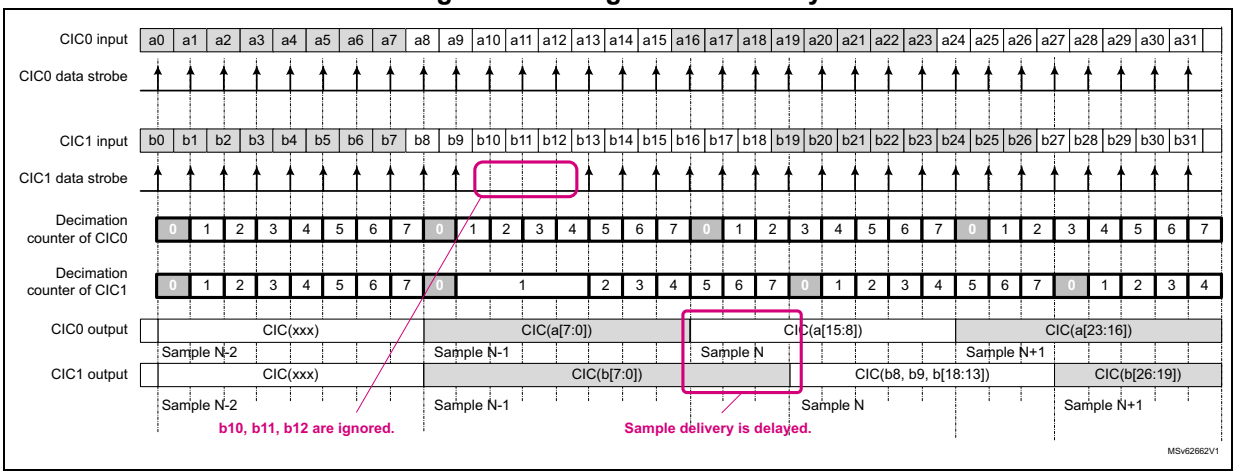
This feature is particularly helpful in the case of microphone beam forming applications where delays smaller than the final sampling rate must be applied to the incoming stream. This feature can be used when the ADF is synchronized with another MDF instance (if present in the product) for a beam forming application for example.

The delay is performed by discarding a given number of samples from the selected input stream, before samples enter into the CIC filter. This data discarding is done by skipping a given number of data strobe, preventing the CIC filter to take into account those data.

When the wanted amount of data strobe has been skipped, the next incoming samples are strobed normally.

Table 362 shows an example on how to apply dynamically small delay to an incoming stream. For simplification, the CIC filter performs a decimation by height in this example. CIC1 represents the CIC included in the ADF and CIC0 represents a filter from another MDF instance (if present in the product).

Figure 362. Programmable delay



The CIC of the ADF (CIC1) receives a command in order to skip three incoming samples. So the input samples named b10, b11 and b12 are not processed by CIC1. As a consequence, the output sample N+1 generated by CIC0 is built from input samples a[23:16] while the sample N+1 of CIC1 is built from input samples b[26:19].

Finally, the non-skipped data stream looks delayed by three bitstream periods.

Note: When the input data strobes are skipped, the decimation counter remains frozen. As a consequence, the samples delivered by the CIC1 are a bit delayed.

The following steps are needed to program the amount of bitstream clock periods to be skipped:

1. Wait for SKPBF equal to 0.
2. Write SKPDLY[6:0] to the wanted number of bitstream clock periods to be skipped. The SKPBF flag goes immediately to 1, indicating that the delay value entered into SKPDLY[6:0] is under process.
   - If the DFLT0 is not yet enabled (DFLTEN = 0), then the DLY logic waits for DFLTEN = 1. When the application sets DFLTEN to 1, the DLY logic starts to skip the amount of wanted data strobes.
   - If the DFLT0 is already enabled (DFLTEN = 1), then the DLY logic immediately starts to skip the amount of wanted data strobes.

   When the ADF skipped the amount of wanted data strobes, then SKPBF goes back to 0.
3. If the application needs to skip more data strobes, then the operation must be restarted from step 1.

The effect of the delay performed with this mechanism is cumulative as long as the ADF is enabled. If the application performs a D1 delay followed by a D2 delay, then all other active filters are delayed by D1 + D2.

Note: If SKPDLY[6:0] is written when SKPBF = 1, the write operation is ignored.

Cascaded-integrator-comb (CIC) filter

The CIC digital filters are an efficient implementation of low-pass filters, often used for decimation and interpolation. The CIC frequency response is equal to a $Sinc^N$ function, this is why they are often called Sinc filters.

The Sinc^N digital filter embedded into the ADF can be configurable in Sinc^4 or Sinc^5, according to CICMOD:

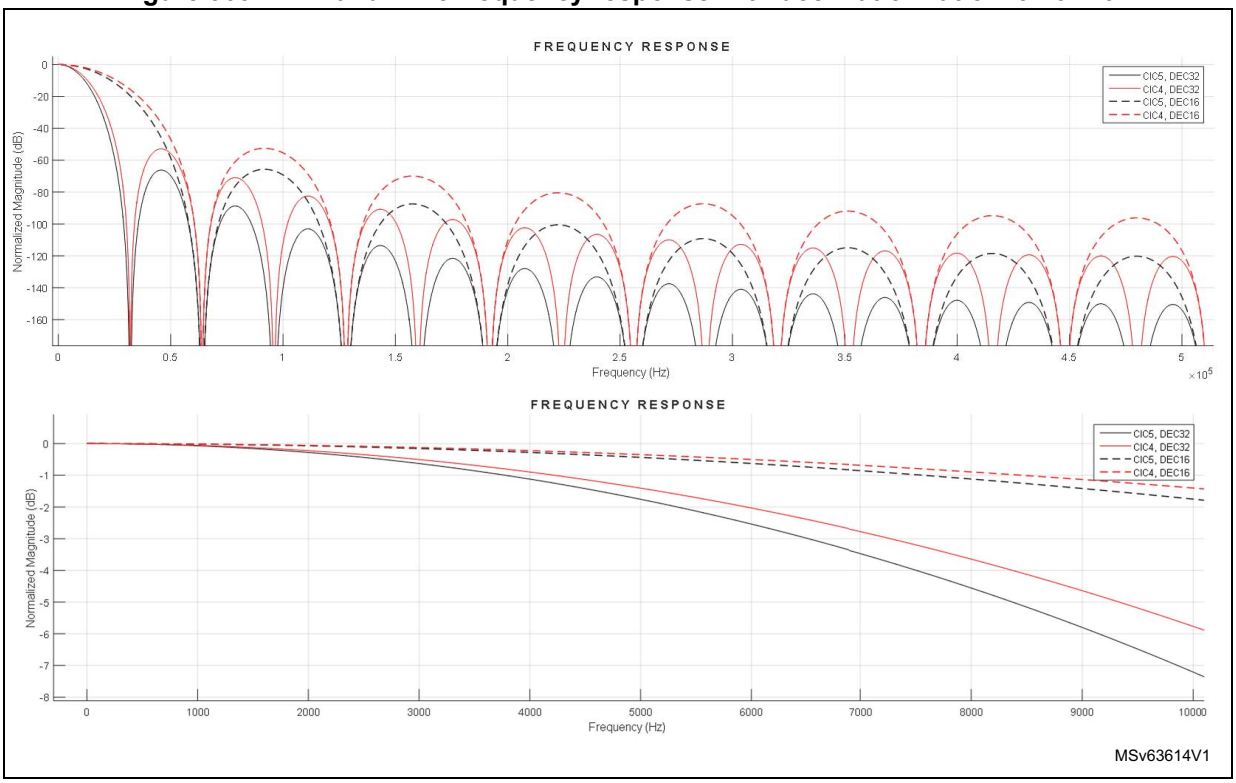
- If CICMOD[2:0] = 4, Sinc^4 is selected.
- If CICMOD[2:0] = 5, Sinc^5 is selected.

The filters have the following transfer function:

$$H(z) = \left(\frac{1 - z^{-D}}{1 - z^{-1}}\right)^N$$

where N can be 4 or 5, and D is the decimation ratio. D is equal to MCICD+1.

Figure 363. CIC4 and CIC5 frequency response with decimation ratio = 32 or 16



CIC output data size

The size of samples delivered by the CIC (DS_{CIC}), depends on the following parameters:

- CIC order (N)
- CIC decimation ratio (D)
- data size of the input stream (DSIN)

The CIC order and decimation ratio must be programmed in order to insure that the data size does not exceed the 26-bit CIC capability.

The following formula gives the output data size (DS_{CIC}) according to the parameters above.

$$DS_{CIC} = \left(\frac{N \times Ln(D)}{Ln(2)}\right) + DS_{IN}$$

and the CIC gain is given by this formula:

$$G_{CIC} = (D)^N$$

The decimation ratio can be adjusted from 2 to 512 for the CIC filter.

Table 300 gives some data output size in bits for some decimation values, when the data source is a full-scale signal coming from the serial interface or from a 12-bit ADC.

Note: DS_{IN} = 1 bit for a serial bitstream, but can be up to 16 bits when coming from the ADCITF.

Table 300. Data size according to CIC order and CIC decimation values

Decimation	Data size (bits) when DS_{IN} = 1 bit (data from SITF0)		Data size (bits) when DS_{IN} = 12 bits (data from ADCITF)	
	$Sinc^4$	**$Sinc^5$**	**$Sinc^4$**	**$Sinc^5$**
4	9	11	20	22
8	13	16	24	-
10	15	18	26	-
12	16	19	-	-
16	17	21	-	-
20	19	23	-	-
24	20	24	-	-
32	21	26	-	-
48	24	-	-	-
64	25	-	-	-
76	26	-	-	-

Note: For a full-scale input signal, the decimation ratio must not exceed 76 for a $Sinc^4$ and 32 for a $Sinc^5$.

The LSB parts of the data provided by the CIC is not necessarily significant: it depends on the sensor performances and the ability of the CIC to reject the out-off band noise.

The sample size at CIC output can be adjusted thanks to the SCALE block.

Scaling (SCALE) and saturation (SAT)

The SCALE block allows the application to adjust the amplitude of the signal provided by the CIC, by steps of 3 dB (± 0.5 dB).

The signal amplitude can be decreased by up to 8 bits (- 48.2 dB), and can be increased by up to 12 bits (+ 72.2 dB).

The gain is adjusted by the SCALE[5:0] bitfield in the *ADF digital filer configuration register 0 (ADF_DFLT0CICR)*.

SCALE[5:0] can be changed even if the DFLT0 is enabled. During the gain transition, the signal provided by the filter is disturbed.

Due to internal resynchronization, there is a delay of some cycles of adf_proc_ck clock between the moment where the application writes the new gain, and the moment where the gain is effectively applied to the samples. If the application attempts to write a new gain value while the previous one is not yet applied, this new gain value is ignored. Reading back SCALE[5:0] informs the application on the current gain value.

Table 301 shows the possible gain values.

Table 301. Possible gain values

SCALE[5:0]	Gain (dB)	SCALE[5:0]	Gain (dB)	SCALE[5:0]	Gain (dB)	SCALE[5:0]	Gain (dB)
0x20	- 48.2	0x2B	- 14.5	0x06	+ 18.1	0x11	+ 51.7
0x21	- 44.6	0x2C	- 12.0	0x07	+ 21.6	0x12	+ 54.2
0x22	- 42.1	0x2D	- 8.5	0x08	+ 24.1	0x13	+ 57.7
0x23	- 38.6	0x2E	- 6.0	0x09	+ 27.6	0x14	+ 60.2
0x24	- 36.1	0x2F	- 2.5	0x0A	+ 30.1	0x15	+ 63.7
0x25	- 32.6	0x00	0.0	0x0B	+ 33.6	0x16	+ 66.2
0x26	- 30.1	0x01	+ 3.5	0x0C	+ 36.1	0x17	+ 69.7
0x27	- 26.6	0x02	+ 6.0	0x0D	+ 39.6	0x18	+ 72.2
0x28	- 24.1	0x03	+ 9.5	0x0E	+ 42.1	-	-
0x29	- 20.6	0x04	+ 12.0	0x0F	+ 45.7	-	-
0x2A	- 18.1	0x05	+ 15.6	0x10	+ 48.2	-	-

The SAT blocks avoid having a wrap-around of the binary code when the code exceeds its maximal or minimal value.

The ADF performs saturation operations at the following levels:

- after the SCALE block (performed by the SAT block): The signal is saturated at 24 bits.
- inside the RSFLT, to insure a good filter behavior
- at the output of the HPF, to insure that the output signal does not exceed 24 bits

The SATF bit informs the application that a saturation occurred either after the SCALE, inside the RSFLT or after the HPF. In addition, an interrupt can be generated if SATIE is set to 1. As soon as a saturation is detected, the SATF flag is set to 1. It is up to the application to clear this flag in order to be able to detect a new saturation.

Those bits are in the *ADF DFLT0 interrupt enable register (ADF_DFLT0IER)* and *ADF DFLT0 interrupt status register 0 (ADF_DFLT0ISR)*.

Gain adjustment policy

To get the best ADF performances, it is important to properly adjust the gain value via SCALE[5:0].

A usual way to adjust the gain is to select the SCALE[5:0] value that gives a final signal amplitude as close as possible to the 24-bit full-scale, for the maximum input signal.

A way to select the optimal gain is detailed below:

1. Check that, for the expected input signal, the data size into the CIC filter does not exceed 26 bits. This can be checked using this formula:

$$\frac{LN(SIN_{pp} \cdot D^N)}{LN(2)} < 26$$

where N represents the CIC order, D the decimation ratio and SIN_{pp} the maximum peak-to-peak amplitude of the input signal.

SIN_{pp}can take:

– a maximum peak-to-peak amplitude of 2 (± 1), for samples coming from SITF0

– A maximum peak-to-peak amplitude of 4095 (+ 2047, - 2048), for samples coming from a 12-bit ADC

Example: a $Sinc^4$ can be used with a decimation ratio of 96, if the maximum input signal does not exceed ± 0.35. Indeed:

$$\frac{LN(0.7 \cdot 96^4)}{LN(2)} \sim 25.82 \text{ bits} < 26 \text{ bits}$$

2. Adjust the SCALE value.

   To select the most appropriate SCALE value, the user must check if the RSFLT is used or not. If the RSFLT is used, the data size at SCALE output must not exceed 22 bits, otherwise the data size can be up to 24 bits.

   The SCALE value in dB must be selected using this formula:

$$SCALE_{dB} < 20 \cdot \log 10\left(\frac{2^{NB}}{SIN_{pp} \cdot D^{N}}\right)$$

   where NB is equal to 22 if RSFLT is enabled, or 24 if RSFLT is bypassed. $SCALE_{dB}$ represents the gain value selected by SCALE[5:0].

   Example: For a $Sinc^4$ with a decimation ratio of 96 and a SIN_{pp} of 0.7.

   – If the RSFLT is bypassed:

$$20 \cdot \log 10\left(\frac{2^{24}}{0.7 \cdot 96^{4}}\right) \sim -11 \text{ dB}$$

     $SCALE_{dB}$ value must be lower than - 11 dB, the closest lower value is - 12dB (SCALE[5:0] = 0x2C).

   – If the RSFLT is enabled:

$$20 \cdot \log 10\left(\frac{2^{22}}{0.7 \cdot 96^{4}}\right) \sim -23 \text{ dB}$$

     $SCALE_{dB}$ value must be lower than - 23 dB, the closest lower value is - 24.1 dB (SCALE[5:0] = 0x28).

   If SCALE[5:0] is set to a higher value, then a saturation may occur. An event flag informs the user if a saturation occurred.

Table 302 proposes gain values for different filter configurations, when the data comes from the SITF0, according to the MCIC order, and the MCIC decimation ratio. This table is not exhaustive and considers a full-scale input signal (see *Section 36.7.5: Total ADF gain* for details).

Table 302. Recommended maximum gain values versus CIC decimation ratios

CIC decimation ratio	Gain settings (dB) for configuration SITF + CICx + RSFLT (+ HPF)		Gain settings (dB) for configuration SITF + CICx (+ HPF)	
	CIC5	CIC4	CIC5	CIC4
8	33.6	51.7	45.7	63.7
12	18.1	39.6	30.1	51.7
16	3.5	27.6	15.6	39.6
20	- 6.0	21.6	6.0	33.6
24	- 12.0	15.6	0	27.6
28	- 20.6	9.5	- 8.5	21.6

Table 302. Recommended maximum gain values versus CIC decimation ratios (continued)

CIC decimation ratio	Gain settings (dB) for configuration SITF + CICx + RSFLT (+ HPF)		Gain settings (dB) for configuration SITF + CICx (+ HPF)	
	CIC5	CIC4	CIC5	CIC4
32	- 26.6	3.5	- 4.5	15.6
48	-	- 8.5	-	3.5
64	-	- 20.6	-	- 8.5

Reshaping filter (RSFLT)

In addition to the CIC, the ADF offers a reshaping IIR filter mainly dedicated to the audio application, but also usable in other applications.

When the RSFLT is used, the sample size at its input must not exceed 22 bits.

The samples at the RSFLT output can be decimated by four or not according to the RSFLTD bit in the *ADF reshape filter configuration register 0 (ADF_DFLT0RSFR)*.

The RSFLT can be bypassed by setting RSFBYP to 1 in the *ADF reshape filter configuration register 0 (ADF_DFLT0RSFR)*.

Table 303 shows which sampling rate must be provided to the RSFLT in order to process the most common audio streams.

The RSFLT cutoff frequency (F_C) depends on the sample rates at its input (F_{RS}), and is given by the following formula:

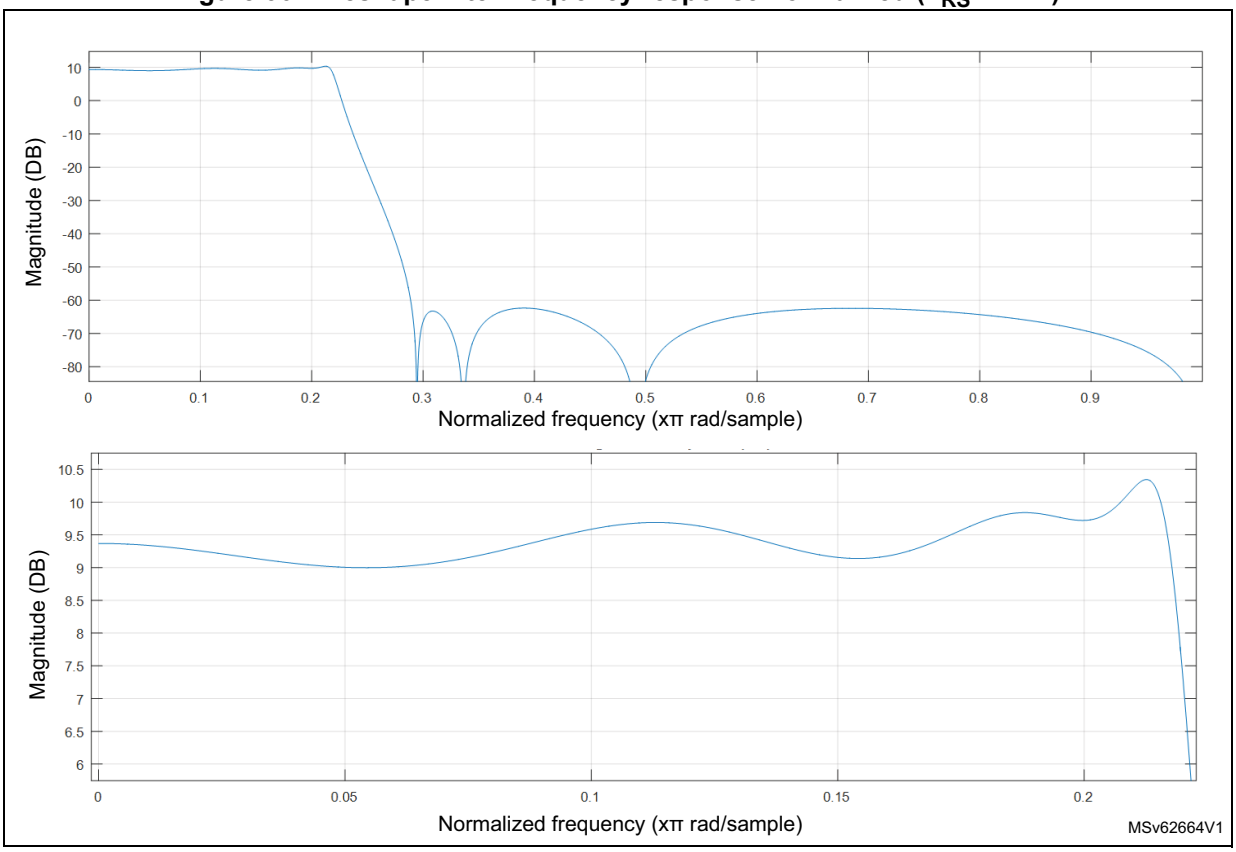
$$F_C = 0.111 \times F_{RS}$$

Table 303. Most common microphone settings

Sample rate (kHz) at RSFLT (F_{RS})	Pass band (kHz)	D2	PCM sampling rate (kHz)
32	3.55	4	8
64	7.1	4	16
128	14.2	4	32
192	21.3	4	48

Table 364 shows the frequency response of the reshape filter.

Figure 364. Reshape filter frequency response normalized (F_{RS} / 2 = 1)



The RSFLT gain is close to 9.3 dB, so the output data size is a little bit lower than 24 bits for a 22-bit wide input signal.

The RSFLT takes 24 clock cycles of adf_proc_ck clock to process one sample at F_{RS}. When the RSFLT is enabled, the application must insure that the adf_proc_ck is at least 24 times faster F_{RS}.

The RSFLT generates an event (rfovr_evt) and sets the RFOVRF flag, if the RSFLT receives a new samples while the previous one is still under processing.

When RFOVRF is set, the samples provided by the RSFLT are invalid. The application must then stop the data acquisition and provides a faster adf_proc_ck clock to the RSFLT.

High-pass filter (HPF)

The high-pass filter suppresses the low-frequency content from the final output data stream in case of continuous conversion mode. The high-pass filter can be enabled or disabled via HPFBYP in the *ADF reshape filter configuration register 0 (ADF_DFLT0RSFR)*.

The HPF is useful when there is parasitic low-frequency noise (or DC signal) in the input data source that must be removed from the final data.

The HPF is a first order IIR filter and the cut-off frequency can be selected via HPFC[1:0] in the *ADF reshape filter configuration register 0 (ADF_DFLT0RSFR)*, among the following values:

- 0.000625 x F_{PCM}
- 0.00125 x F_{PCM}
- 0.00250 x F_{PCM}
- 0.00950 x F_{PCM}

Table 304. HPF 3 dB cut-off frequency examples

HPFC	3 dB cut-off frequency for common F_{PCM} frequencies (Hz)		
	F_{PCM} = 8 kHz	F_{PCM} = 16 kHz	F_{PCM} = 48 kHz
0	5	10	30
1	10	20	60
2	20	40	120
3	76	152	456

The HPF output is saturated at 24 bits. The SATF flag is set if a sample is saturated.

36.4.8 Digital filter acquisition modes

The ADF offers the following modes to perform a data capture:

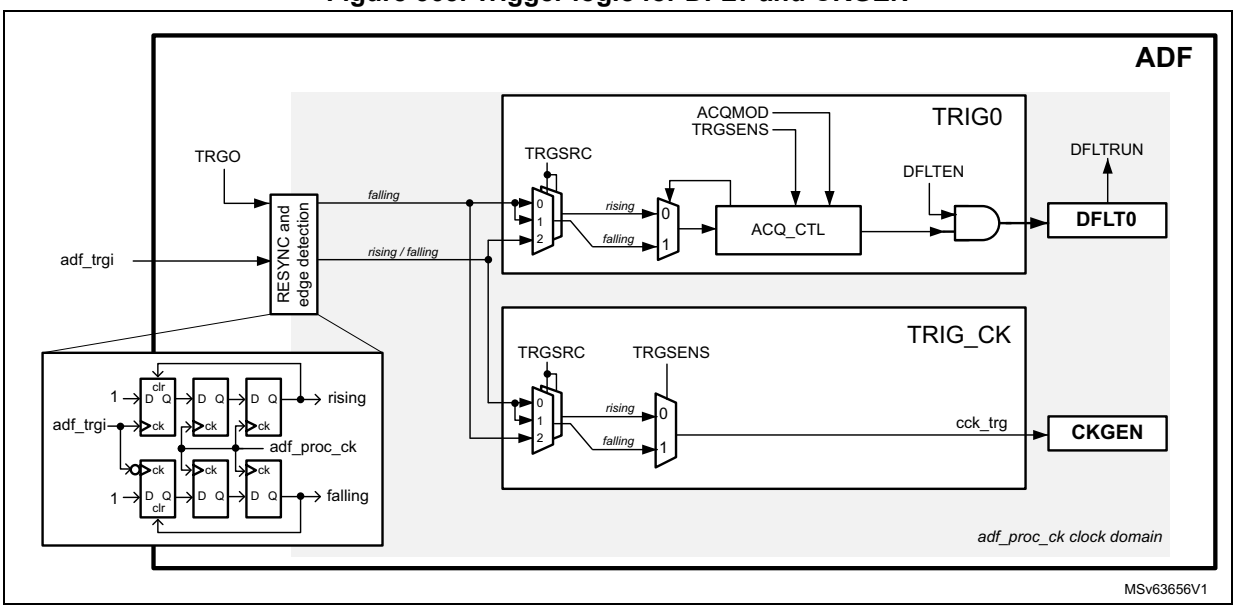
- asynchronous continuous acquisition mode
- asynchronous single-shot acquisition mode
- synchronous continuous acquisition mode
- synchronous single-shot acquisition mode
- window continuous acquisition mode

Note: To perform a data capture, the filter, the interface providing the data (SITF0 or ADCITF) and the CKGEN must be enabled. If needed, the ADF_CCK0 or ADF_CCK1 must be enabled as well.

The filter can be stopped immediately when DFLTEN is set to 0. This action resets the filter and flushes the RXFIFO. The DFLTACTIVE flag also goes back to 0 when the RXFIFO and the filter is reset.

Table 365 shows a simplified view of the trigger logic available for each filter and for the clock generator.

Figure 365. Trigger logic for DFLT and CKGEN



A block common to all TRIG blocks performs the rising and falling edges detection and the resynchronization of the input trigger to the adf_ker_ck clock domain. This implementation allows the application to use triggers with pulse width smaller than the adf_ker_ck period.

In synchronous modes, the TRIG block offers the possibility to select adf_trgi or TRGO bit as trigger sources. The TRGO bit is in the *ADF global control register (ADF_GCR)*.

The edge sensitivity can also be selected.

Asynchronous continuous acquisition mode

This mode allows the application to start a continuous acquisition by simply writing the DFLTEN bit to 1.

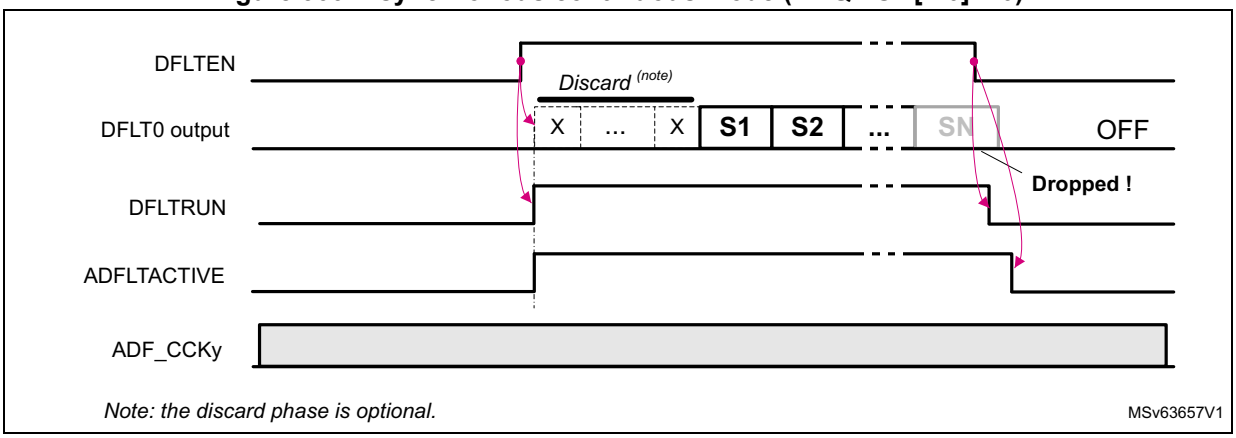
The asynchronous continuous acquisition mode is selected when ACQMOD[2:0] = 0.

The sequence below shows the most important programming steps (assuming that DFLTEN is set to 0):

1. Configure and enable the clock generator (CKGEN) so that the adf_proc_ck frequency is compatible with the targeted application (see examples in *Table 308*).
2. Enable the CKGEN (CKGDEN = 1) and, if needed, enable the ADF_CCK0 and ADF_CCK1 clocks.
3. Program the filter configuration and set the ACQMOD[2:0] to 0.
4. Set to 1 the SITFEN bit of the serial data interface.
5. Before setting DFLTEN to 1, wait for DFLTACTIVE = 0: it insures that the previous filter deactivation sequence terminated properly.
6. When DFLTEN is set to 1, the acquisition sequence starts immediately.

Figure 366 shows a simplified example of the samples generated by the DFLT0.

Figure 366. Asynchronous continuous mode (ACQMOD[2:0] = 0)



Note: The acquisition can be stopped by setting DFLTEN back to 0. This resets the filter and flushes the RXFIFO, so the samples located into the RXFIFO are lost. The ongoing DMA transfer is properly terminated. DFLTACTIVE goes back to 0 when the filter chain is reset and the RXFIFO flushed.

Asynchronous single-shot acquisition mode

This mode allows the application to start the acquisition of one sample by simply writing the DFLTEN bit to 1.

The asynchronous single-shot acquisition mode is selected when ACQMOD[2:0] = 001.

The sequence below shows the most important programming steps (assuming that DFLTEN is set to 0):

1. Configure and enable the clock generator (CKGEN), so that the adf_proc_ck frequency is compatible with the targeted application (see examples in *Table 308*).
2. Enable the CKGEN (CKGDEN = 1) and, if needed, enable the ADF_CCK0 and ADF_CCK1 clocks.
3. Program the filter configuration, and set the ACQMOD[2:0] to 001.
4. Set to 1 the SITFEN bit.
5. Before setting DFLTEN to 1, wait for DFLTACTIVE = 0: it insures that the previous filter deactivation sequence terminated properly.
6. When DFLTEN is set to 1, the filter provides one data to the RXFIFO and stops the acquisition.

To trigger a new acquisition, the application must:

1. Check that the previous acquisition is completed, by waiting that DFLTRUN = 0.
2. Set again DFLTEN to 1.

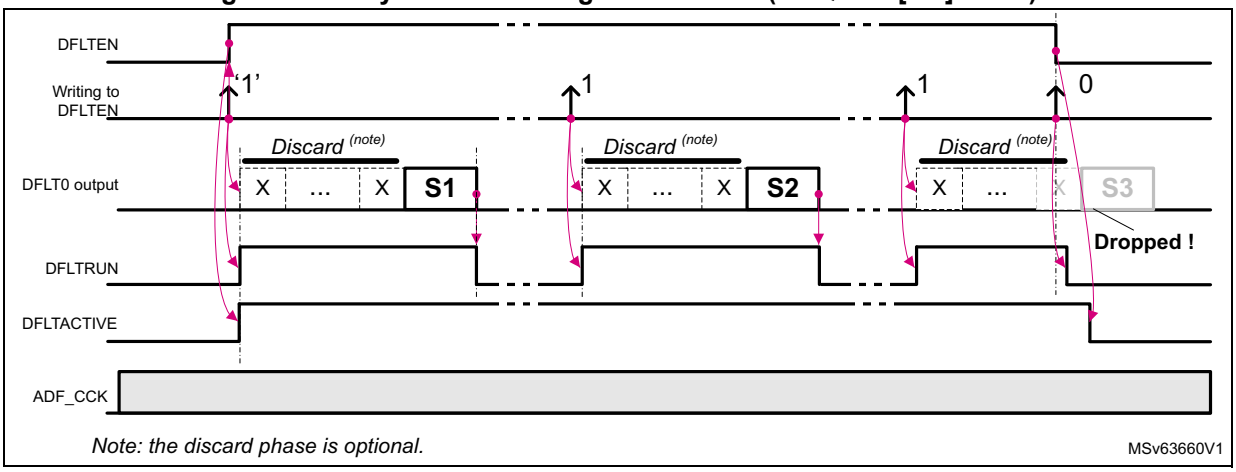
This sequence can be repeated every time a new data must be converted.

As shown in *Figure 366*, every time DFLTEN is set to 1, an acquisition sequence is triggered. The first samples provided by the filter can be discarded if needed. At the end of each conversion, the decimation counters and filter taps are reset, and the filter is ready to start a new conversion.

If DFLTEN is set to 0 while an acquisition is ongoing, the ongoing conversion is stopped (in the example, S3 is lost). This situation can be avoided with the following steps:

1. Wait for DFLTRUN = 0.
2. Read the sample from the RXFIFO.
3. Set DFLTEN to 0.

Figure 367. Asynchronous single-shot mode (ACQMOD[2:0] = 001)



Note: The acquisition can be stopped by setting DFLTEN back to 0. This resets the filter and flushes the RXFIFO, so the samples located into the RXFIFO are lost. The ongoing DMA transfer is properly terminated. DFLTACTIVE goes back to 0 when the filter chain is reset and the RXFIFO flushed.

Synchronous continuous acquisition mode

This mode allows the application to start a continuous acquisition by using one of the following trigger sources:

- adf_trgi signal
- TRGO bit

The Synchronous continuous acquisition mode is selected when ACQMOD[2:0] = 010.

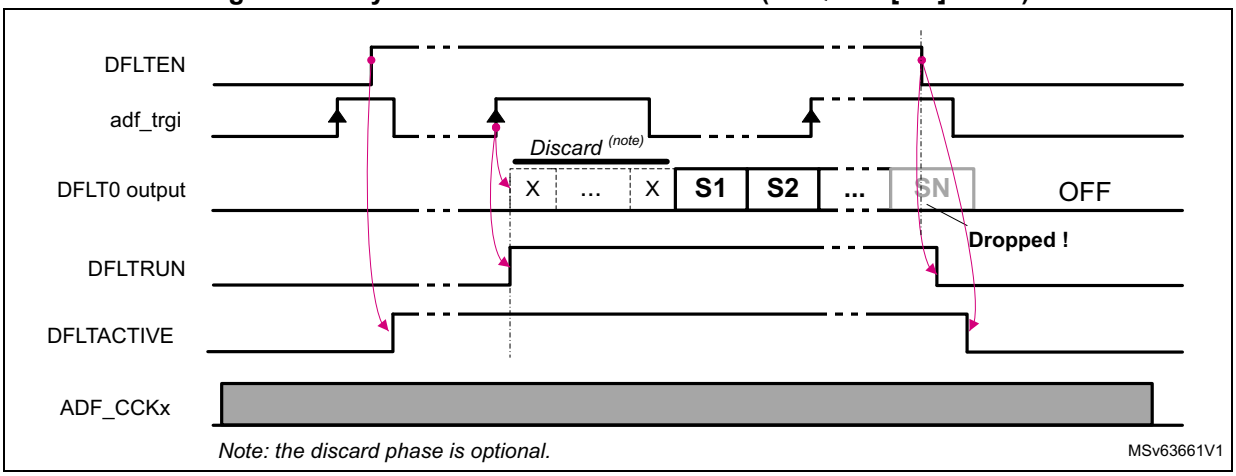
The sequence below shows the most important programming steps (assuming that DFLTEN is set to 0):

1. Configure and enable the clock generator (CKGEN), so that the frequency of adf_proc_ck clock is compatible with the targeted application (see examples in *Table 308*).
2. Enable the CKGEN (CKGDEN = 1) and, if needed, enable the ADF_CCK0 and ADF_CCK1 clocks.
3. Program the filter configuration and set the ACQMOD[2:0] to 010.
4. Set to 1 the bit SITFEN.
5. Select the proper trigger source and sensitivity.
6. Before setting DFLTEN to 1, wait for DFLTACTIVE = 0: it insures that the previous filter deactivation sequence terminated properly.
7. Set DFLTEN to 1.
8. When the trigger condition is met, the filter starts the acquisition.

The TRGSENS bit allows the selection of the trigger edge (rising or falling). The trigger is ignored if an acquisition is ongoing or if DFLTEN is set to 0.

Figure 368 shows a simplified example where the trigger logic is sensitive to a rising edge trigger (TRGSENS = 0). The first rising edge of the trigger signal is ignored because DFLTEN = 0. Then the next rising edge is taken into account and starts the acquisition. All other rising edges are ignored. The trigger logic is re-initialized when DFLTRUN goes back to 0.

Figure 368. Synchronous continuous mode (ACQMOD[2:0] = 010)



Note: The acquisition can be stopped by setting DFLTEN back to 0. This resets the filter and flushes the RXFIFO, so the samples located into the RXFIFO are lost. The ongoing DMA transfer is properly terminated. DFLTACTIVE goes back to 0 when the filter chain is reset and the RXFIFO flushed.

Synchronous single-shot acquisition mode

This mode allows the application to start a single acquisition by using one of the following trigger sources:

- adf_trgi signal
- TRGO bit

The Synchronous single-shot acquisition mode is selected when ACQMOD[2:0] = 011.

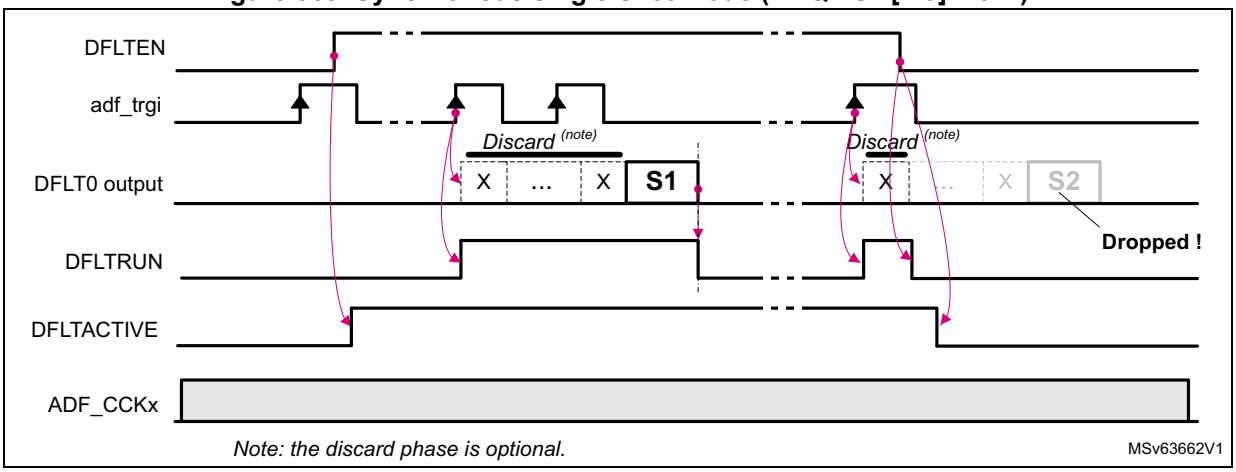
The sequence below shows the most important programming steps (assuming that DFLTEN is set to 0):

1. Configure and enable the clock generator (CKGEN), so that the frequency of adf_proc_ck clock is compatible with the targeted application (see examples in *Table 308*).
2. Enable the CKGEN and, if needed, enable the ADF_CCK0 and ADF_CCK1 clocks.
3. Program the filter configuration, and set the ACQMOD[2:0] to 011.
4. Set to 1 the SITFEN bit.
5. Select the proper trigger source and sensitivity.
6. Before setting DFLTEN to 1, wait for DFLTACTIVE = 0: it insures that the previous filter deactivation sequence terminated properly.
7. Set DFLTEN to 1.
8. When the trigger condition is met, the filter starts the acquisition and provides one data to the RXFIFO, then the filter is ready to accept a new trigger.

TRGSENS allows the selection of the trigger edge (rising or falling). The trigger is ignored if an acquisition is ongoing, or if DFLTEN is set to 0.

Figure 369 shows a simplified example where the trigger logic is sensitive to a rising edge trigger (TRGSENS = 0). Every-time a trigger rising edge is detected with DFLTEN = 1, an acquisition sequence is triggered. The first samples provided by the filter can be discarded if needed. At the end of each conversion, the decimation counters and filter taps are reset. DFLTRUN is set to 0 and the filter is ready to start a new conversion.

Figure 369. Synchronous single-shot mode (ACQMOD[2:0] = 011)



Note: The acquisition can be stopped by setting DFLTEN back to 0. This resets the filter and flushes the RXFIFO, so the samples located into the RXFIFO are lost. The ongoing DMA transfer is properly terminated. DFLTACTIVE goes back to 0 when the filter chain is reset and the RXFIFO flushed.

Figure 368 shows a case where the DFLTEN is set to 0 while an acquisition is ongoing (the sample S2 is lost). This situation can be avoided with the following steps:

1. Wait for DFLTRUN = 0.
2. Read the sample from the RXFIFO.
3. Clear DFLTEN to 0.

Window continuous acquisition mode

This mode allows the application to start or stop a continuous acquisition controlled by consecutive edges of one of the following trigger sources:

- adf_trgi signal
- TRGO bit

The window continuous acquisition mode is selected when ACQMOD[2:0] = 100.

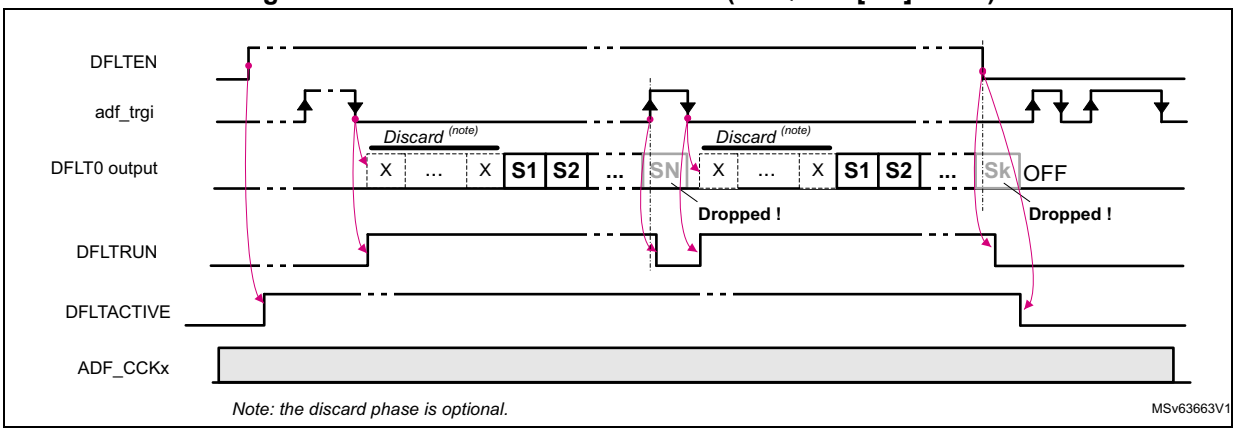
The sequence below shows the most important programming steps (assuming that DFLTEN is set to 0):

1. Configure and enable the clock generator (CKGEN), so that the frequency of adf_proc_ck clock is compatible with the targeted application (see examples in *Table 308*).
2. Enable the CKGEN and, if needed, enable the ADF_CCK0 and ADF_CCK1 clocks.
3. Program the filter settings and set the ACQMOD[2:0] to 100.
4. Set to 1 the SITFEN bit.
5. Select the proper trigger source and sensitivity.
6. Before setting DFLTEN to 1, wait for DFLTACTIVE = 0: it insures that the previous filter deactivation sequence terminated properly.
7. Set DFLTEN to 1.
8. If TRGSENS = 0, the acquisition starts on trigger rising edge and stops on trigger falling edge. If TRGSENS = 1, the acquisition starts on trigger falling edge and stops on trigger rising edge.

Note: The acquisition may restart if the trigger condition becomes again active.

Figure 370 shows a simplified example of window continuous acquisition mode, with TRGSENS = 1. Once DFLTEN is set to 1, the ADF waits for a falling edge on the selected trigger input. When the trigger condition is met, DFLTRUN goes to 1 and the acquisition starts. The acquisition stops if the ADF detects a rising edge on the selected trigger input. If DFLTEN is still set to 1, the ADF waits again for a falling edge on the selected trigger input.

Figure 370. Window continuous mode (ACQMOD[2:0] = 100)



Note: The acquisition can be stopped by setting DFLTEN back to 0. This resets the filter and flushes the RXFIFO, so the samples located into the RXFIFO are lost. The ongoing DMA transfer is properly terminated. DFLTACTIVE goes back to 0 when the filter chain is reset and the RXFIFO flushed.

Starting several filters synchronously

If the ADF is used with MDF instances (if present in the product), it is possible to start simultaneously the acquisition of all the filters. This synchronization capability depends on the way the triggers are connected in the product. Generally, an ADF is able to trigger MDF instances, if its adf_trgo signal is connected as trigger input to those blocks (see *Section 36.4.2: ADF pins and internal signals* to check trigger capabilities).

In the following programming example, one ADF has its adf_trgo signal connected to some MDFs. To start the acquisition of several filters synchronously, the following sequence must be performed (assuming that DFLTEN bits of the filters are set to 0):

On MDFs receiving the adf_trgo trigger:

1. Enable the CKGEN (CKGDEN = 1) and, if needed, enable the ADF_CCK0 and ADF_CCK1 clocks.
2. Set to 1 the SITFEN bit of the requested data interfaces.
3. For each filter, set the acquisition mode to synchronous (ACQMOD[2:0] = 01x).
4. For each filter, set TRGSRC[3:0] in order to select the adf_trgo trigger input.
5. For each filter, set TRGSENS to 0 (rising edge).
6. For each filter, set DFLTEN to 1.

On the ADF generating the adf_trgo trigger:

1. Enable the CKGEN (CKGDEN = 1) and, if needed, enable the ADF_CCK[1:0] clocks.
2. Set to 1 the SITFEN bit of the requested data interfaces.
3. Set the acquisition mode to synchronous (ACQMOD[2:0] = 01x).
4. Set TRGSRC[3:0] to 0 (TRGO selected).
5. Set TRGSENS to 0 (rising edge).
6. Set DFLTEN to 1.
7. Read TRGO bit until it is read to 0.
8. Set TRGO to 1. Then the acquisition sequence for all selected filters starts immediately.

To trigger a new acquisition (in case of single-shot) the application must do the following:

1. Check that the previous acquisition is completed, by waiting DFLTRUN = 0.
2. Read TRGO until it is read to 0.
3. Set again the bit TRGO to 1.

Discarded samples

The ADF offers the possibility to program the amount of samples to be discarded after each restart:

- to avoid capturing samples affected by the impulse response of the filter
- to delay the acquisition of filters by a specific amount of samples

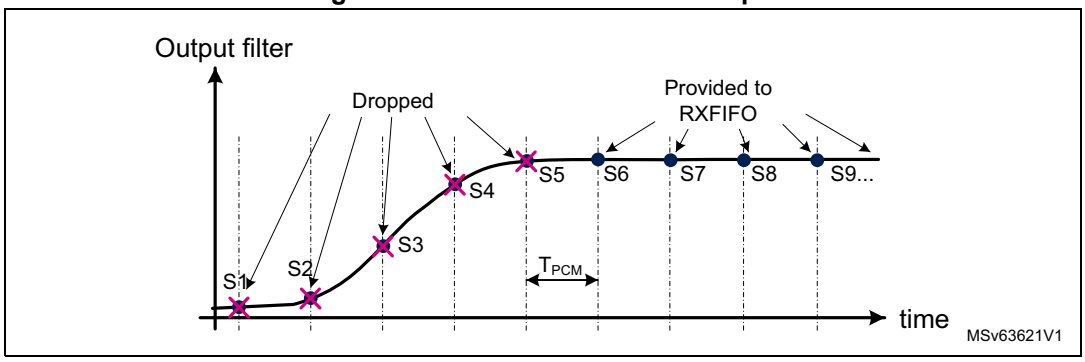
The discard function is controlled via NBDIS[7:0] as follows:

- When NBDIS[7:0] = 0, the discard function is disabled.
- When NBDIS[7:0] ≠ 0, the discard function is activated in one of the following condition:
  - when the DFLTEN bit goes to 1
  - every time an acquisition is started in (A)synchronous single-shot modes

Refer to *Figure 365* to *Figure 369*, and *Figure 371*.

In the example shown in *Figure 371*, the discard function is used to drop the first five samples provided by the digital filter (S1 to S5). The first sample transfered to the RXFIFO is S6.

Figure 371. Discard function example



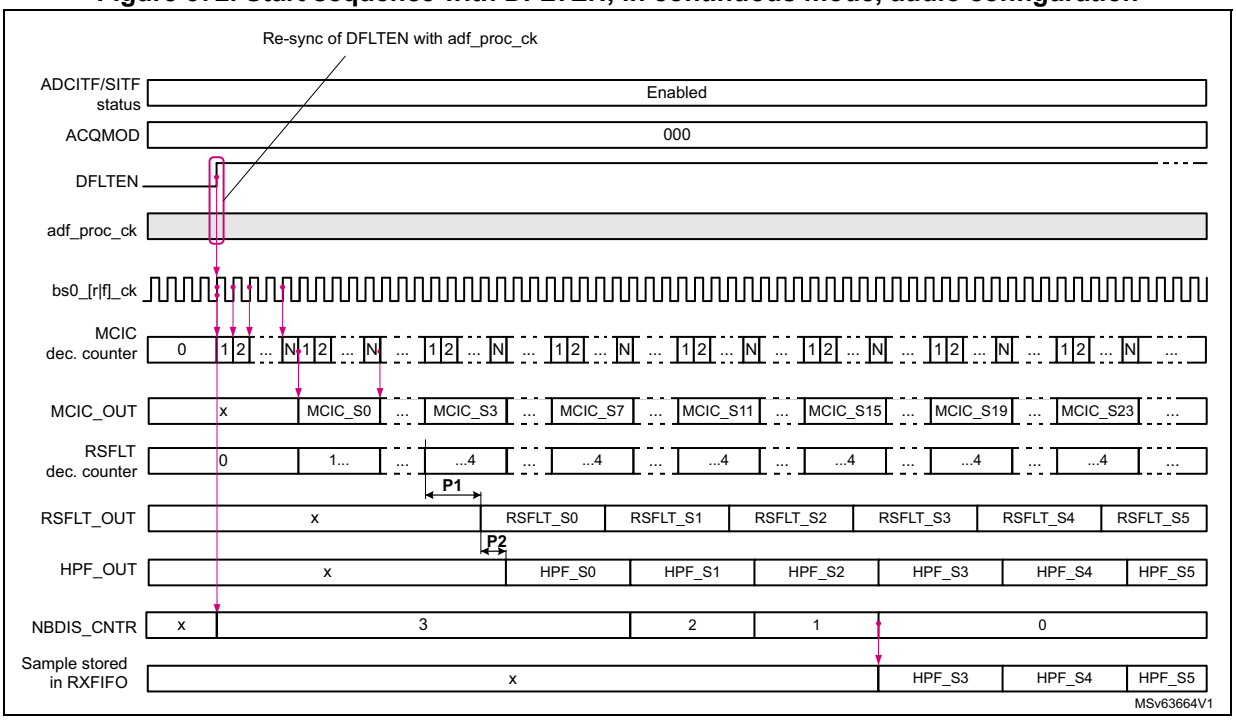
36.4.9 Start-up sequence examples

Figure 372 details a start of acquisition sequence of a digital filter triggered by DFLTEN (ACQMOD[2:0] = 0), with NBDIS[7:0] = 3 (three samples to discard before acquisition).

The DFLT0 is configured for audio application: MCIC, RSFLT and HPF activated.The data interface (SITF0 or ADCITF) is assumed to be already activated.

Note: NBDIS[7:0] is set on purpose to a small value to simplify the drawing.

Figure 372. Start sequence with DFLTEN, in continuous mode, audio configuration



The DFLTEN bit is re-sampled into the ADF processing clock domain. When DFLTEN is detected high, the filter chain is enabled, and the decimation counter of the MCIC filter is incremented at the rate of the bitstream clock.

When the MCIC decimation counter reached its programmed value N, a sample is available for the RSFLT.

The RSFLT processes all the samples provided by the MCIC, and delivers a sample to the HPF every-time it processes four samples (decimation by 4). The RSFLT needs up to 24 cycles of adf_proc_ck clock before delivering a sample (P1).

The HPF processes all the samples provided by the RSFLT, but the NBDIS function prevents the data writing in the RXFIFO as long as NBDIS_CNTR does not reach 0.

When NBDIS_CNTR reaches 0, the samples provided by the HPF are stored into the RXFIFO.

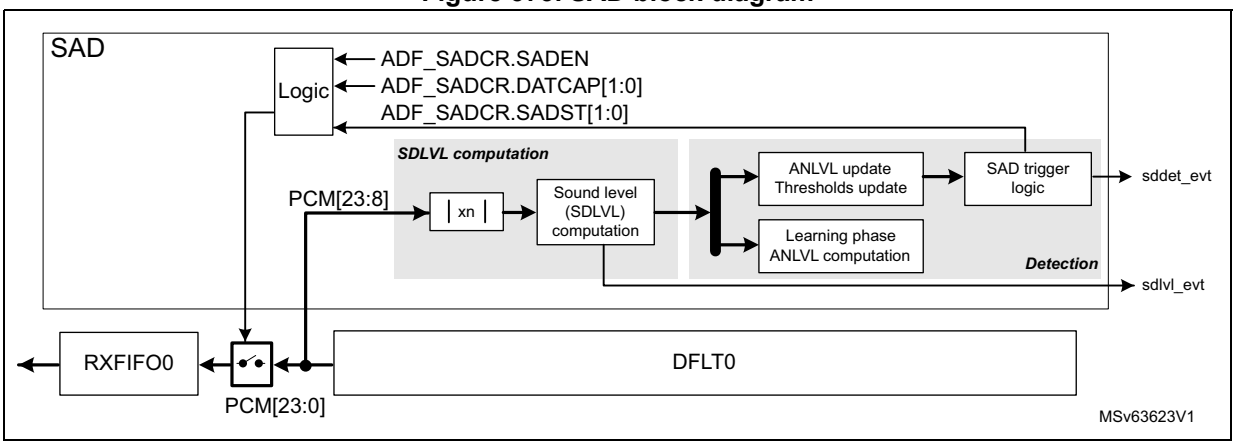
36.4.10 Sound activity detection (SAD)

The SAD is based on the computation of the ambient noise level (ANLVL) and of the short-term sound level (SDLVL). The SAD offers the following ways to detect a sound:

- when the SDLVL reaches a threshold referenced to the ambient noise level
- when the SDLVL reaches a fixed threshold
- when the ANLVL reaches a fixed threshold

As shown in *Figure 373*, the SAD takes the 16 MSB samples from the DFLT0 output.

Figure 373. SAD block diagram



The SAD is highly configurable, and the application can adjust several parameters:

- SAD detection behavior (SADMOD)
- number of samples used to compute the sound level (FRSIZE)
- number of frames used to compute the ambient noise level during the learning phase (LFRNB)
- slope of the ambient noise estimator (ANSLP)
- minimum expected ambient noise level (ANMIN)
- threshold level (SNTHR)
- threshold hysteresis (HYSTEN)
- hangover window in order to filter spurious transitions between DETECT and MONITOR states (HGOVR)
- data capture mode (DATCAP)

SAD detection behavior

The SAD can use the following ways to detect a sound, selected by SADMOD[1:0]:

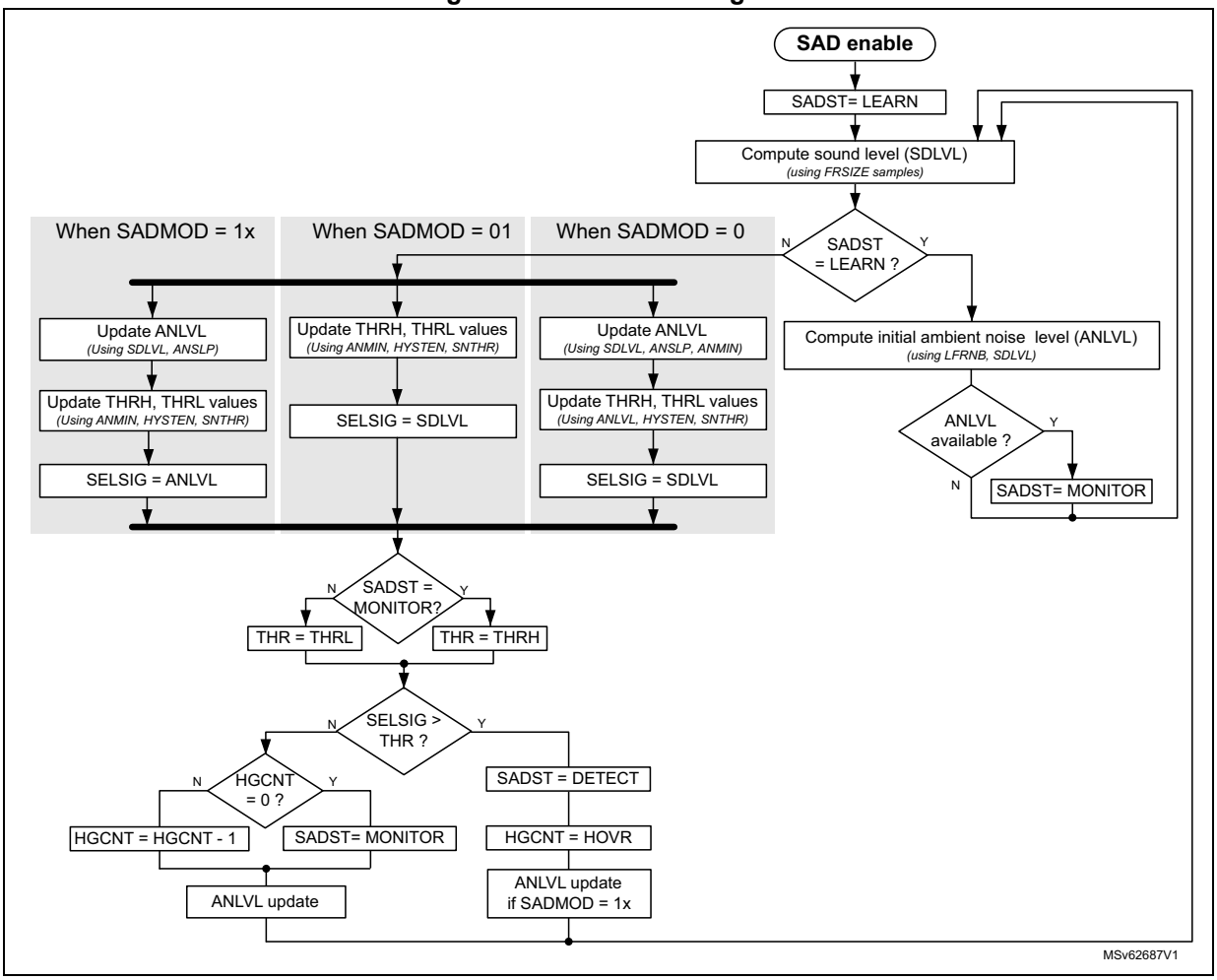
- When SADMOD[1:0] = 0, the SAD works like a voice-activity detection. In this mode, the SAD estimates the ambient noise level according to the computed sound level values. The threshold of the trigger is elaborated from the estimated ambient noise. Finally the current sound level is compared to this threshold. In a first approximation, the SAD triggers if the peak-to-average value of the input signal reaches a level defined by SNTHR[3:0].
- When SADMOD[1:0] = 01, the SAD compares the current sound level (SDLVL) to a fixed trigger value defined by the application via SNTHR[3:0] and ANMIN[12:0]. This mode allows a fast SAD reaction as the amount of samples used to compute the sound level can be configured via FRSIZE[2:0].
- When SADMOD[1:0] = 1x, the SAD compares the estimated ambient noise level (ANLVL) to a fixed trigger value defined by the application via SNTHR[3:0] and ANMIN[12:0]. This mode avoids unwanted triggers, due to peak levels, but the SAD reacts more slowly to an input signal variation. It is nevertheless possible to adjust the reaction time via FRSIZE[2:0] and ANSLP[2:0].

SAD states

As shown in *Figure 373*, the SAD works as follows:

1. When enabled (SADEN = 1), the SAD is first in LEARN state to perform a first estimation of the ambient noise level.
2. The SAD continuously computes the short-term sound level (SDLVL) using the samples provided by the DFLT0. The amount of samples used to compute the sound level is given by FRSIZE[2:0]. The samples processed by the DFLT0 can be transferred into the memory or not depending on DATCAP[1:0] value.
3. The initial ambient noise level (ANLVL) is computed using the consecutive sound level values. The application can define how much sound level values are used to perform the computation of this initial ambient noise estimation (LFRNB).
4. When the initial ambient noise level (ANLVL) is computed, the SAD switches to the MONITOR state.
5. Every time a new short-term sound level value is available, the SAD updates the ambient noise level and the thresholds according to the selected detection mode.
6. If the SAD triggers, then the following happens:
   – The SAD switches to DETECT state.
   – The sddet_evt event is asserted.
   – The adf_sad_det signal is set to high.
7. The hangover function insures that the DETECT state is maintained even if the sound level goes below the threshold level for a time given by HGOVR.

Figure 374. SAD flow diagram



Sound level computation (SDLVL)

Once enabled, the SAD computes continuously the sound level value. The sound level represents the average of the absolute value of an amount of PCM samples given by FRSIZE[2:0].

$$SDLVL = \frac{1}{N_{FRSIZE}} \times \sum_{n=1}^{N_{FRSIZE}} |PCM(n)|$$

where N_{FRSIZE} is the amount of PCM samples given by FRSIZE[2:0].

Ambient noise estimation (ANLVL)

The ambient noise level (ANLVL) is computed when SADMOD[1:0] is 00 or 10.

The ambient noise level is computed differently according to the state of the SAD as detailed below:

- ANLVL computation during the LEARN state

  Every time the SAD is enabled, a learning phase is initiated in order to estimate a first value of the ambient noise level. During this phase, the SAD cannot trigger.

  During the LEARN phase, the ambient noise level is computed as follows:

$$ANLVL = \frac{1}{N_{LFRNB}} \times \sum_{n=1}^{N_{LFRNB}} |SDLVL(n)|$$

  where N_{LFRNB} is the amount of frames given by LFRNB[2:0] bitfield.

- ANLVL computation during the MONITOR or DETECT state

  When the learning phase is completed, the SAD updates the ambient noise level in the following way:

  – The SAD computes the new possible values for the ambient noise level:

    ANLVL_UP = ANLVL * (1+2^(ANSLP-12))

    ANLVL_DN = ANLVL * (1-2^(ANSLP-10))

  – The ANLVL takes the ANLVL_DN value if the current sound level is lower than ANLVL_DN, otherwise ANLVL takes the value of ANLVL_UP.

    The ANLVL is not updated if the current sound level is higher than the threshold level, except if SADMOD[1:0] = 10.

  – When SADMOD[1:0] = 0, if the new ANLVL value is lower than ANMIN[12:0], ANLVL is replaced by ANMIN.

  The slope of the noise estimator can be adjusted to optimize the detection of the wanted signal. This slope is adjusted via ANSLP[2:0] in the *ADF SAD configuration register (ADF_SADCFGR)*.

Table 305 shows the allowed values according to the frame size and the sampling rate of the data observed by the SAD. The recommended values when the SADMOD[1:0] = 0 are the ones into the gray shaded cells.

Table 305. ANSLP values versus FRSIZE and sampling rates

FRSIZE	ANSLP values for Fs = 8 kHz			ANSLP values for Fs = 16 kHz		
	Slow(1)	typical(2)	Fast(3)	Slow(1)	typical(2)	Fast(3)
0 (8 samples)	0	1	2	-	0	1
1 (16 samples)	1	2	3	0	1	2
2 (32 samples)	2	3	4	1	2	3
3 (64 samples)	3	4	5	2	3	4
4 (128 samples)	4	5	6	3	4	5
5 (256 samples)	5	6	7	4	5	6
6 (512 samples)	6	7	-	5	6	7

1. The slow slope is equal to - 8.5 dB/s for the negative slope and + 2.1 dB/s for the positive slope.
2. The typical slope is equal to - 17.1 dB/s for the negative slope and + 4.2 dB/s for the positive slope.
3. The fast slope is equal to - 34.2 dB/s for the negative slope and + 8.5 dB/s for the positive slope.

The slopes can also be computed using the following formulas:

$$SLC_{UP} = 20 \times \frac{F_s}{F_{SIZE}} \times \log 10(1 + 2^{(ANSLP - 12)})$$

$$SLC_{DN} = 20 \times \frac{F_s}{F_{SIZE}} \times \log 10(1 - 2^{(ANSLP - 10)})$$

where F_S is the sampling rate of the stream observed by the SAD and F_{SIZE} is the frame size defined by FRSIZE[2:0].

Threshold computation

The way the threshold value is computed depends on SADMOD[1:0]:

- If SADMOD[1:0] = 0, THRH is obtained by multiplying the current ANLVL value with the gain defined in SNTHR[3:0].

  $$THR_H = ANLVL \times 10^{\frac{GdB_{SNTHR}}{20}}$$

  This threshold value is then compared to the current sound level (SDLVL).
- If SADMOD[1:0] = 01, THRH is obtained by multiplying the current ANMIN[12:0] with the gain defined by SNTHR[3:0].

  $$THR_H = ANMIN \times 10^{\frac{GdB_{SNTHR}}{20}}$$

  This threshold value is then compared to the current sound level (SDLVL).

- If SADMOD[1:0] = 1x, THRH is obtained by multiplying the current ANMIN[12:0] by 4.

$$THR_H = ANMIN \times 4$$

This threshold value is then compared to:

$$ANLVL \times 10^{\frac{GdB_{SNTHR}}{20}}$$

The hysteresis mode can be enabled to reduce the spurious transitions between MONITOR and DETECT states. In hysteresis mode (HYSTEN = 1), the following threshold values are used:

- THRH when the SAD is in MONITOR state.
- THRL when the SAD is in DETECT state.

Table 306 shows the thresholds values according to SNTHR.

Table 306. Threshold values according SNTHR(1)

SNTHR[3:0]	THR_H		THR_L		Comments
0	LVL + 3.5 dB	LVL x 1.5	LVL + 3.5 dB	LVL x 1.5	No hysteresis
1	LVL + 6.0 dB	LVL x 2	LVL + 3.5 dB	LVL x 1.5	Hysteresis of 2.5 dB
2	LVL + 9.5 dB	LVL x 3	LVL + 6.0 dB	LVL x 2	Hysteresis of 3.5 dB
3	LVL + 12.0 dB	LVL x 4	LVL + 9.5 dB	LVL x 3	Hysteresis of 2.5 dB
4	LVL + 15.6 dB	LVL x 6	LVL + 12.0 dB	LVL x 4	Hysteresis of 3.5 dB
5	LVL + 18.1 dB	LVL x 8	LVL + 15.6 dB	LVL x 6	Hysteresis of 2.5 dB
6	LVL + 21.6 dB	LVL x 12	LVL + 18.1 dB	LVL x 8	Hysteresis of 3.5 dB
7	LVL + 24.1 dB	LVL x 16	LVL + 21.6 dB	LVL x 12	Hysteresis of 2.5 dB
8	LVL + 27.6 dB	LVL x 24	LVL + 24.1dB	LVL x 16	Hysteresis of 3.5 dB
9	LVL + 30.1 dB	LVL x 32	LVL + 27.6dB	LVL x 24	Hysteresis of 2.5 dB

1. LVL must be replaced by ANLVL when SADMOD[1:0] = 0 and by ANMIN for other SADMOD[1:0] values.

When the hysteresis function is disabled, the SAD always use THRH.

Note: The hysteresis mode must not be used when SADMOD[1:0] = 1x.

Trigger logic

The signal compared to this threshold depends also on SADMOD[1:0].

The trigger condition is reached when the selected signal (SELSIG) is bigger than the threshold level.

If the trigger condition is met, the following happens:

- The SAD switches to DETECT state.
- The SAD refreshes the hangover counter with HGOVR.
- The sddet_evt event is asserted if the SAD transits from MONITOR to DETECT.
- The adf_sad_det signal is set to high.

The SAD remains in DETECT state as long as the trigger condition is met or the hangover down-counter is different from 0.

The sddet_evt event indicates when the SAD enters and/or exits the DETECT state. This event is used to generate an interrupt when a sound is detected or when a sound is no longer detected:

- When DETCFG = 0, the application receives an event only when the SAD enters the DETECT state.
- When DETCFG = 1, the application receives an event when the SAD enters or exits the DETECT state.

The adf_sad_det signal remains high as long as the SAD is in DETECT state.

The SAD also provides a flag indicating that a new sound level value is available (SDLVLF). The last computed sound level (SDLVL[14:0]) is available in the *ADF SAD sound level register (ADF_SADSDLVR)*, and the last computed ambient noise level (ANLVL[14:0]), in the *ADF SAD ambient noise level register (ADF_SADANLVR)*.

Note: The SAD can work even when the AHB clock is not present. In that case, the SAD does not update SDLVL[14:0] and ANLVL[14:0].

To get the latest valid SDLVL[14:0] and ANLVL[14:0] values, the application must read the ADF_SADSDLVR, and ADF_SADANLVR registers, when the SDLVLF flag goes high. This can be done in the following ways:

- by polling the SDLVLF flag:
  a) Clear the SDLVLF flag by writing SDLVLF to 1.
  b) Wait for SDLVLF = 1, by reading ADF_DFLTxISR.
  c) Read ADF_SADSDLVR and ADF_SADANLVR.
  d) Clear SDLVLF by writing it to 1.
  e) Go to step 2 if other values must to be read.
- by generating an interrupt:
  a) Read ADF_DFLTxISR.
  b) If SDLVLF = 1, read ADF_SADSDLVR and ADF_SADANLVR, and clear SDLVLF by writing it to 1.
  c) Handle other status flags and exit from ISR.

Sample transfer to memory

The SAD offers the following options to control the samples transfer from DFLT0 to the system memory:

- If DATCAP[1:0] = 1x, the samples are transferred into the system memory as soon as DFLT0 and SAD are enabled. The transfer does not depend on the SAD state.
- If DATCAP[1:0] = 01, the samples are transferred into the system memory when the SAD detects a sound (when the SAD is in DETECT state), assuming that DFLT0 and SAD are enabled.
- If DATCAP[1:0] = 0, the samples are not transferred into the memory. This mode can be used if the application only wants to observe but does not need samples for other processing.

Note: DATCAP[1:0] is taken into account only when the SADEN = 1. For example, if the SAD configuration is DATCAP[1:0] = 0, SADEN = DFLTEN = 1, and if the application sets now SADEN to 0, the samples provided by the DFLT0 are transferred to the RXFIFO.

Programming recommendations

To make the SAD function working properly, the ADF must be programmed as follows:

1. Provide the proper kernel clock (adf_ker_ck) to the ADF.
2. Configure the CKGEN and enable it.
3. Configure the SITF and enable it (note that microphones have a settling time of several milliseconds).
4. Configure the DFLT0. A typical setting is the following:
   – CIC5 with a decimation ratio of 12, 16 or 24
   – RSFLT with a decimation ratio of 4
   – HPF with HPFC = 2 or 3

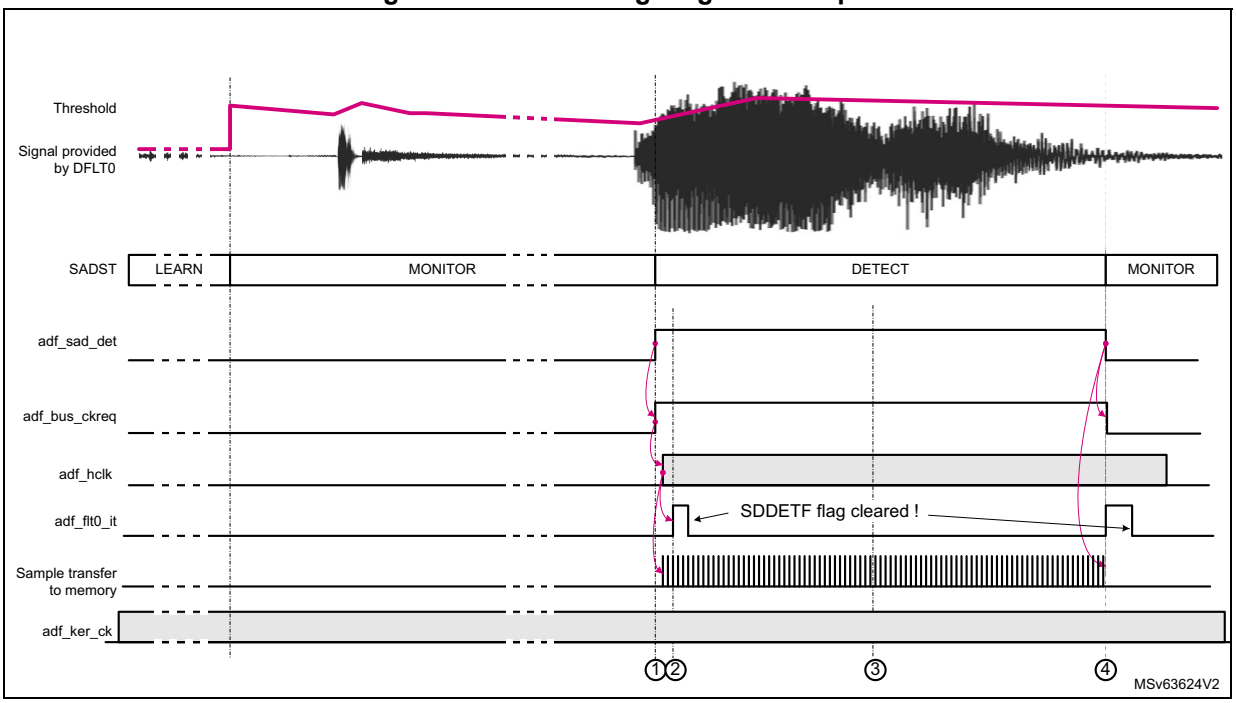
   For a very-low power implementation, the RSFLT can be bypassed.
5. Set SADEN to 0.
6. Wait for SADACTIVE = 0. If SADEN was previously enabled, this phase can take two periods of adf_hclk, and two periods of adf_proc_ck.
7. Configure the SAD as follows:
   – Set DATCAP[1:0] to 0, if the application does not want to store the samples into the system memory.
   – Set DATCAP[1:0] to 01, if the application wants to store the samples into the system memory only when the SAD detects a sound.
   – Set DATCAP[1:0] to 11, if the application wants to store the samples into the system memory continuously.
8. Configure the DMA (optional).
9. Enable the SAD.
10. Enable the DFLT0.

Figure 374 shows a simplified timing diagram when the SAD works with DATCAP[1:0] = 01.

Thanks to the kernel clock (adf_ker_ck), the SAD continuously monitors the audio signal provided by the DFLT0. The threshold is also continuously updated according to the ambient noise level estimation.

- When the SAD detects a sound higher than the programmed threshold (1), the ADF requests the bus clock (adf_bus_ckreq asserted).
- When the bus clock is available (see 2 in *Figure 374*) then:
  – The data transfer to the memory is triggered.
  – The event interrupt (adf_evt_it) can be generated.
- In this example, the event interrupt (adf_evt_it) is used to wake up the application. The interrupt line is released by clearing SDDETF by writing 1 to it.
- As long as the SAD remains in DETECT state, the application waits to get enough samples and calls, for example the keyword recognition algorithm (see 3 in *Figure 374*).
- In the case shown in *Figure 375*, the SAD state (SADST) goes back to MONITOR before the keyword is recognized. If DETCFG is set to 1, an event signals when the SAD goes back to MONITOR state. The SAD stops the transfer of samples into the memory and the application can clean up the receive buffer for the next detection (see 4 in *Figure 374*).

Figure 375. SAD timing diagram example



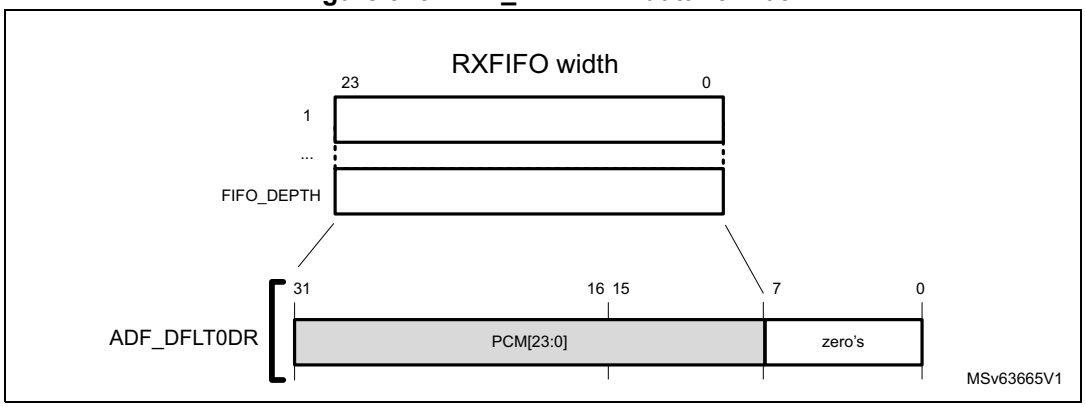
36.4.11 Data transfer to memory

Data format

The samples processed by DFLT0 are stored into a RXFIFO. The application can read the samples stored into these FIFOs via the *ADF digital filter data register 0 (ADF_DFLT0DR)*. The samples inside this register are signed and left aligned. The bit 31 always represents the sign.

The ADF provides 24-bit left-aligned data. Performing a 16-bit access to ADF_DFLT0DR allows the application to get the 16 most significant bits. Performing a 32-bit access to ADF_DFLT0DR allows the application to get a 24-bit data size.

Figure 376. ADF_DFLTxDR data format



Data resynchronization

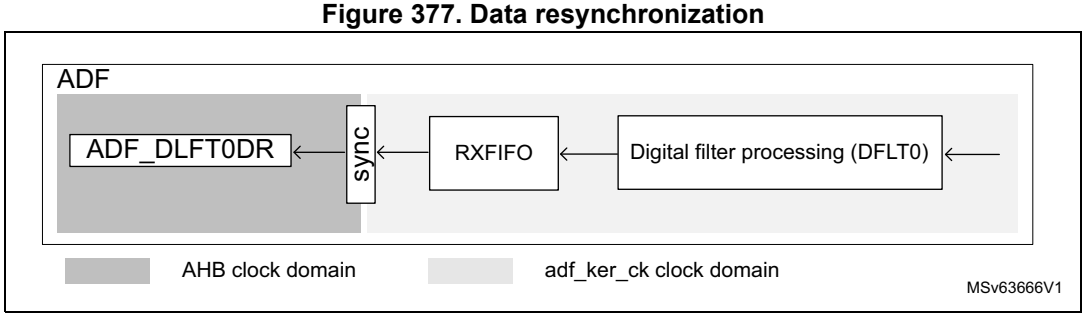
The samples stored into the RXFIFO can be transferred into the memory by using either DMA requests or interrupt signaling.

Note: The RXFIFO is located into the adf_ker_ck clock domain, while ADF_DFLT0DR *is located into the adf_hclk (AHB) clock domain.*

When the AHB clock is available, if ADF_DFLT0DR is empty and if a sample is available into the RXFIFO, this sample is transferred into ADF_DFLT0DR.

The sample transfer from the RXFIFO to ADF_DFLT0DR takes two periods of the AHB clock (adf_hclk) and two periods of the adf_ker_ck clock. The ADF inserts automatically wait-states if the application performs a read operation of ADF_DFLT0DR while the transfer of the new sample from the RXFIFO to ADF_DFLT0DR is not yet completed.

Figure 377. Data resynchronization



Data transfer

The content of the RXFIFO can be transfered to the memory either by using a DMA channel or interrupt services.

Both single and burst, DMA transfers are supported by the ADF, but the application has to care about the following points:

- The RXFIFO must contain at least the same amount of samples than the burst size.
- The burst mode efficiency may be reduced due to the data resynchronization explained in the previous section.

Note: The burst mode is not available in all products (see the DMA section to check if the product supports it).

In addition, the application can select the RXFIFO threshold (FTH bit) in order to trigger the data transfer: a data transfer can be triggered as soon as the RXFIFO is not empty, or when the RXFIFO is half-full (containing depth/2 samples).

For the DMA transfer, as soon as one of the RXFIFO reaches the threshold level, the DMA request is asserted in order to ask for data transfer. Successive DMA requests are performed as long as the RXFIFO is not empty.

The DMA mode of the RXFIFO is enabled via the DMAEN bit in ADF_DFLT0DR.

For the interrupt signaling, the following cases must be considered:

- If FTH = 0, as soon as a data is available in ADF_DFLT0DR, the FTHF is set, allowing the generation of an interrupt. FTHF is released as soon as ADF_DFLT0DR is read.
- If FTH = 1, as soon as the RXFIFO reaches the threshold level and a data is available in ADF_DFLT0DR, the FTHF is set, allowing the generation of an interrupt. FTHF is released as soon as one data is read. FTHF is set again if the threshold condition is

met again. In this mode, every time an interrupt occurs, the application is supposed to read FIFO_SIZE/2 data.

RXFIFO overrun

A RXFIFO overrun condition is detected when the RXFIFO is full, and a new sample from the DFLT0 must be written.

In this case, DOVRF is set and the new sample is dropped. When the RXFIFO has at least one location available, the new incoming sample is written into the RXFIFO.

Figure 378 shows an example based on a RXFIFO depth of four words and FTH set to 1, so that FTHF goes to 1 when the RXFIFO is half-full.

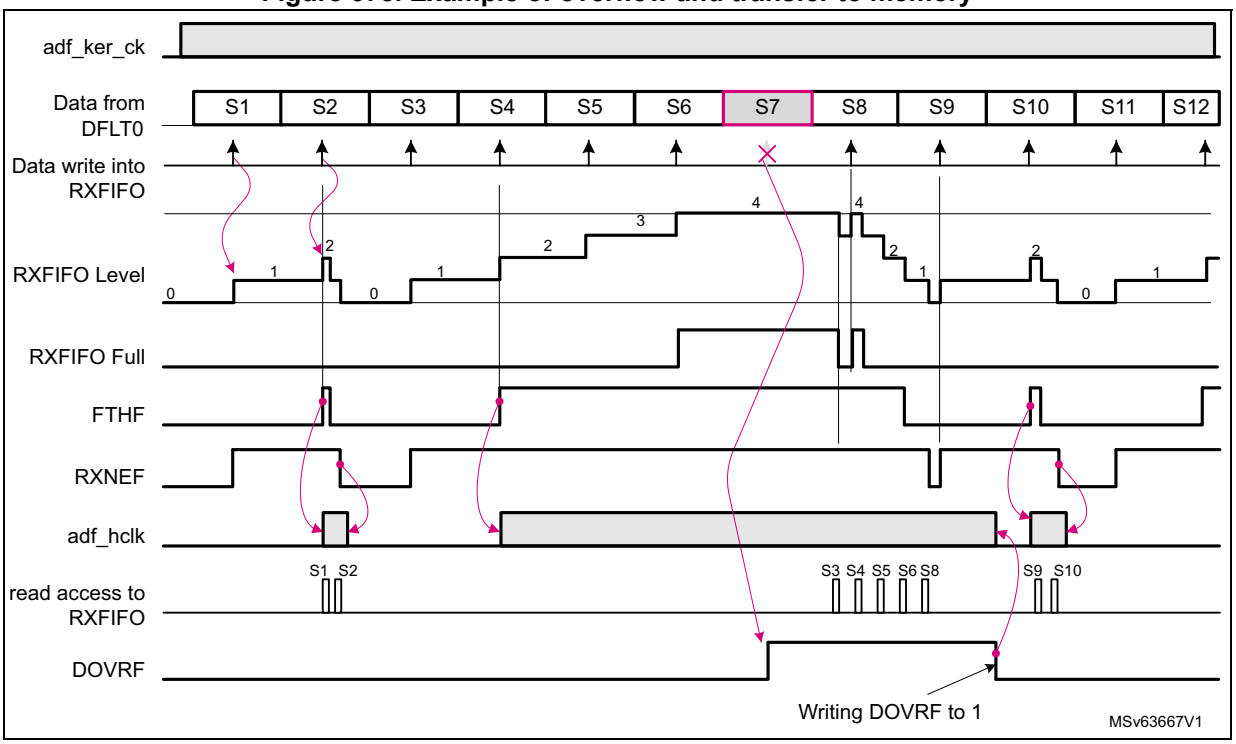
The S7 sample is lost due to an overrun: the RXFIFO is full while S7 must be written into the RXFIFO. The S7 write operation is not performed. DOVRF is set to 1 at the moment where the write operation was expected. The overflow event remains to 1 as long as it is not cleared by the application.

In this example, DOVRIE is set to 1 to have an interrupt if an overrun condition is detected.

After the S7 sample, the application manages to read data from the RXFIFO, and the ADF can write the S8 sample and consecutive. Later, the application clears DOVR, allowing the detection of a new overrun situation.

In the adf_hclk line, the gray boxes indicate that the ADF requested the AHB clock. *Figure 378* shows the AHB clock available only when the ADF requests it. In real applications, the AHB clock may also be present if the ADF does not request it.

Figure 378. Example of overflow and transfer to memory



36.4.12 Autonomous mode

The ADF can work even if the AHB bus clock is not available (Stop modes). The ADF uses the AHB clock only for the register interface. All the processing part is clocked with the kernel clock.

In Stop mode, the ADF receives a kernel clock if the following conditions are met:

- The ADF autonomous mode is enabled in the RCC.
- The selected kernel clock source is taken from an oscillator available in Stop mode.

In Stop mode, the ADF receives the AHB clock if the following conditions are met:

- The ADF autonomous mode is enabled in the RCC.
- The ADF requests the AHB clock in the following situations:
  - when the ADF must transfer data into memory via the DMA

    The data is directly transferred to the SRAM thanks to the DMA while the product remains in Stop mode. The AHB clock request is maintained until the DMA transfer is completed.
  - when the ADF needs to generate an interrupt

    An interrupt generally wakes up the device from Stop mode, as an action from the application is needed. Once the AHB clock is available, the interrupt is generated.The AHB clock request is maintained as long as an enabled interrupt flag is still active.

36.4.13 Register protection

The ADF embeds some hardware protection to prevent invalid situations.
Table 307 shows the list of write-protected and unprotected fields.

Table 307. Register protection summary

Registers	Unprotected fields	Write-protected fields	Write-protection condition
ADF global control register (ADF_GCR)	TRGO	-	DFLTACTIVE0 = 1
ADF clock generator control register (ADF_CKGCR)	CKGDEN CCK0EN CCK1EN	PROCDIV[6:0], CCKDIV[3:0], CKGMOD, TRGSRC[3:0], TRGSENS, CCK[1:0]DIR	CKGACTIVE = 1
ADF serial interface control register 0 (ADF_SITF0CR)	SITFEN	STH[4:0], SITFMOD[1:0], SCKSRC[1:0]	SITFACTIVEx = 1
ADF bitstream matrix control register 0 (ADF_BSMX0CR)	-	BSSEL[4:0]	DFLTACTIVEx = 1
ADF digital filter control register 0 (ADF_DFLT0CR)	DFLTEN	NBDIS[7:0], TRGSRC[3:0], TRGSENS, FTH, DMAEN, SNPSFMT, ACQMOD[2:0]	DFLTACTIVEx = 1
ADF digital filer configuration register 0 (ADF_DFLT0CICR)	SCALE[5:0]	MCICD[8:0], CICMOD[2:0], DATSRC[1:0]	
ADF reshape filter configuration register 0 (ADF_DFLT0RSFR)	-	All fields	
ADF delay control register 0 (ADF_DLY0CR)	-	SKPDLY[6:0]	SKPBF = 1

Table 307. Register protection summary (continued)

Registers	Unprotected fields	Write-protected fields	Write-protection condition
ADF DFLT0 interrupt enable register (ADF_DFLT0IER)	All fields	-	-
ADF DFLT0 interrupt status register 0 (ADF_DFLT0ISR)	All fields	-	-
ADF SAD control register (ADF_SADCR)	SADEN	FRSIZE[2:0], DETCFG, SADMOD[1:0], HYSTEN, DATCAP[1:0]	SADACTIVE = 1
ADF SAD configuration register (ADF_SADCFGR)	-	ANMIN[12:0], SNTHR[3:0], HGOVR[3:0], LFRNB[2:0], ANSLP[2:0]	SADACTIVE = 1

All the ADF processing is performed in the adf_proc_ck clock domain. For that reason, enabling or disabling an ADF sub-block may take some time due to the resynchronization between the AHB clock domain and the adf_proc_ck clock domain. XXXACTIVE flags are available to allow the application to check that the synchronization between the two clock domains is completed.

To change a write-protected bitfield, the application must follow this sequence:

1. Set the enable bit of the sub-block to 0.
2. Wait for corresponding flag XXXACTIVE = 0.
3. Modify the wanted fields.
4. Set the enable bit of the sub-block to 1.

Refer to the description of each sub-block for details.

36.5 ADF low-power modes

Table 308. Effect of low-power modes on ADF

Mode	Description
Sleep	No effect. ADF interrupts cause the device to exit the Sleep mode.
Stop(1)	The ADF registers content is kept. Thanks to the kernel clock request feature, if the ADF is clocked by an internal oscillator available in Stop mode, the ADF remains active. The interrupts cause the device to exit Stop mode.
Standby	The ADF is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 36.3: ADF implementation* for details about Stop modes supported by the ADF.

36.6 ADF interrupts

To increase the CPU performance, the ADF offers an interrupt line (adf_flt0_it), sensitive to several events.

Note: The status flags are available even if the corresponding interrupt enable flag is not enabled.

The interrupt interface is controlled via the *ADF DFLT0 interrupt enable register (ADF_DFLT0IER)* and the *ADF DFLT0 interrupt status register 0 (ADF_DFLT0ISR)*.

Figure 379. ADF interrupt interface

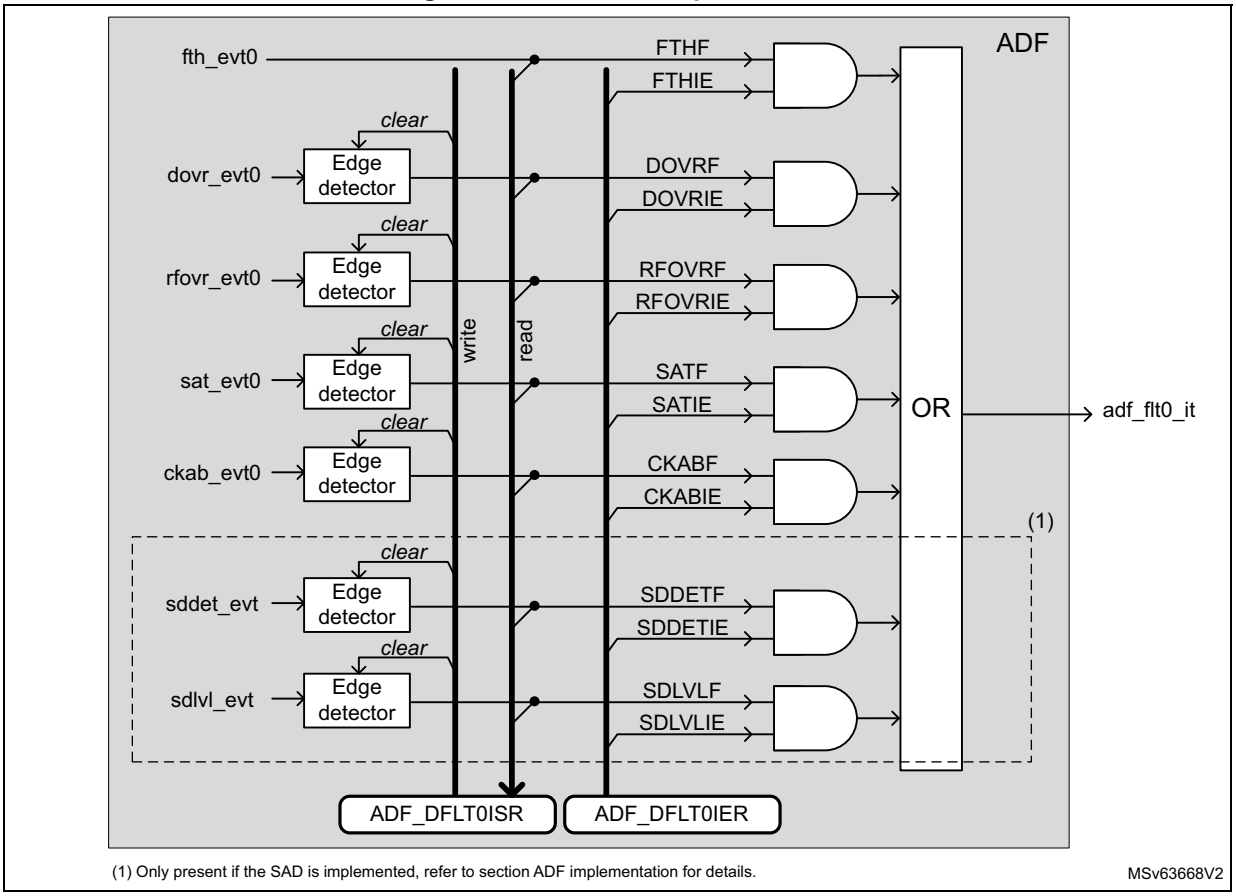


Table 309 shows which interrupt line is affected by which event, and how to clear and activate each interrupt/event.

Table 309. ADF interrupt requests

Interrupt vector	Interrupt event	Event flag	Event/interrupt clearing method	Exit Sleep mode	Exit Stop modes(1)	Exit Standby mode
ADF_FLT0(2)	RXFIFO threshold reached	FTHF	Read ADF_DFLT0DR until RXFIFO level is lower than the threshold.	Yes	Yes	No
	RXFIFO overrun	DOVRF	Write DOVRF to 1.			
	RSFLT overrun	RFOVRF	Write RFOVRF to 1.			
	Saturation detection	SATF	Write SATF to 1.			
	Channel clock absence detection	CKABF	Write CKABF to 1.			
	SAD: sound detected	SDDETF	Write SDDETF to 1.			
	SAD: sound level value available	SDLVLF	Write SDLVLF to 1.			

1. Refer to *Section 36.3: ADF implementation* for details.
2. ADF_FLT0 vector corresponds to the assertion of adf_flt0_it signal.

36.7 ADF application information

36.7.1 ADF configuration examples for audio capture

Table 310 gives some examples of the ADF settings for the digital microphones, focusing on 16 and 48 kHz output data rate.

Configurations #1 and #2 are for very low-power use-cases and have a reduced signal-to-noise ratio. The user must also insure that the selected digital microphone can work properly at 512 kHz. These configurations can be used for sound detection. The RSFLT is not used to reduce as much as possible the frequency of the kernel clock (adf_ker_ck).

Configurations #3, #4, #9, #10, #11 give signal-to-noise ratios around 115 dB, with an ideal microphone model, with a sinus signal of 997 Hz. Using the RSFLT allows a good control on the in-band ripple, and a good image rejection.

Configurations #7, #8, #10 give signal-to-noise ratio around 120 dB, with an ideal microphone model, using a sinus signal of 997 Hz.

Table 310. Examples of ADF settings for microphone capture

Configuration	adf_ker_ck (MHz)	PROCDIV + 1	CCKDIV + 1	CIC order (1)	MCICD + 1	SCALE	RSFLTBYP	RSFLTD	HPFBYP	-	adf_proc_ck (MHz)	Total dec. ratio	F_{RS} (kHz)	F_{ADF_CCKx} (MHz)	F_{PCM} (kHz)
#1	1.024	1	2	4	64	0x2D (- 8.5 dB)	1	x	x	=>	1.024	64	-	0.512	8
#2		1	2	5	32	0x2B (- 14.5 dB)						32	-		16
#3		1	2	5	16	0x01 (+ 3.5 dB)	0	0				64	32		8
#4	2.048	1	2	5	16	0x01 (+ 3.5 dB)	0	0			2.048	64	64	1.024	16
#5	3.072	1	6	5	8	0x0B (+ 33.6 dB)	0	0			3.072	32	32	0.512	8
#6		2	2	5	12	0x06 (+ 18.1 dB)						48	64	0.768	16
#7		1	2	5	24	0x2C (- 12 dB)						96		1.536	
#8	4.096	1	2	5	32	0x27 (- 26.6 dB)	0	0			4.096	128	64	2.048	16
#9	6.144	3	2	5	16	0x02 (+ 6.0 dB)	0	0			2.048	64	64	1.024	16
#10		2	2	5	24	0x2C (- 12 dB)					3.072	96		1.536	
#11		1	2	5	16	0x01 (+ 3.5 dB)					6.144	64	192	3.072	48
#12	7.680	1	2	5	20	0x2E (- 6.0 dB)	0	0			7.680	80	192	3.840	48

1. CICMOD = 100 for CIC order equal to 4. CICMOD = 101 for CIC order equal to 5.

36.7.2 Programming examples

Example 1

This example describes the programming of ADF for the capture of a signal coming from a digital microphone, using only the CIC4, with a decimation of 48, assuming that the kernel clock is 1.536 MHz. Typically, this configuration can be used to detect sound using the SAD.

Table 311. Programming sequence (CIC4)

Operations	Comments
Adjust the proper kernel clock frequency via the RCC	Assuming that the RCC is programmed to provide a kernel clock (adf_ker_ck) of 1.536 MHz coming from a RC oscillator.
Select the proper ADF kernel clock source via the RCC	Refer to the RCC of the product.
Enable the ADF clocks via the RCC	Refer to the RCC of the product.
Reset the ADF via the RCC	Refer to the RCC of the product.
AFMUX programming	Program the AFMUX to select ADF_SD0 and ADF_CCK0 functions.
Enable ADF processing clock: ADF_CKGCR = 0x0001 0023	PROCDIV = 0 (bypass): adf_proc_ck frequency is 1.536 MHz. CCKDIV = 1 (division by 2): ADF_CCK0 clock frequency is 768 kHz. The ADF_CCK0 pin is set in output and generates a clock so that the microphone can exit from low-power mode.
Serial interfaces configuration: ADF_SITF0CR = 0x0000 1F01	SCKSRC = 0 to select ADF_CCK0 as serial clock. SIFTMOD = 0 to select LF_MASTER SPI mode. Clock absence feature is not working in this mode. The serial interface is enabled.
Bitstream matrix configuration: ADF_BSMX0CR = 0x0000 0000	DFLT0 filter takes the bitstream of SITF0, sampled on rising edge
Filters configuration (CIC): ADF_DFLT0CICR = 0x0040 2F40	SCALE = 0x04 (12 dB): RSFLT is not enabled. Note that the gain is 8.5 dB, higher than recommended in order to improve signal accuracy for the SAD. Saturation is not an issue in this case, as only the detection of a signal much lower than the full scale is needed. MCICD = 0x2F (decimation by 48) CICMOD = 4 to select a $Sinc^4$ DATSCR = 0 to select data coming from BSMX
Filters configuration (RSFLT and HPF): ADF_DFLT0RSFR = 0x0000 0301	HPFC = 3: cut-off frequency of 16kHz * 0.0095 = 152 Hz HPFBYP = 0: HPF not bypassed RSFLTBYP = 1: RSFLT bypassed
Micro delay adjust: ADF_DLY0CR = 0x0000 0000	Not used in this example
Enable interrupt events: ADF_DFLT0IER = 0x0000 1000	Enable the interrupt events the application wants to handle. In this example, SDDETIE is set to 1 to have an interrupt if a sound is detected.

Table 311. Programming sequence (CIC4) (continued)

Operations	Comments
SAD setting: ADF_SADCR = 0x0000 1400	SADMOD = 1: Trigger based on sound level values FRSIZE = 4: Sound level computed on 128 samples HYSTEN = 0: No hysteresis DETCFG = 0: interrupt generated when the SAD enters in DETECT state DATCAP = 0: samples not transferred to memory SADEN = 0: SAD not yet enabled
SAD setting: ADF_SADCFGR = 0x0025 0001	ANMIN = 0x25: Sound threshold of 37 * 10^SNTHR/20 = 74 LSB(1) HGOVR = 0: four frames (4 x 128 = 512 samples) LFRNB = 0: Learning phase set to two frames. Not useful in this mode. ANSLP = 0: Not used in this mode SNTHR = 1: Threshold set to 6 dB (see ANMIN)
Clear status flags: ADF_DFLT0ISR = 0x0000 0FFF	Clear all the status flags before running the ADF
Enable the SAD: ADF_SADCR = 0x0000 1401 Wait for SADACTIVE flag = 1	Enable first the SAD, in order to be sure that data transfer to memory are blocked by the SAD.
Enable the filter: ADF_DFLT0CR = 0x0000 0001	NBDIS = 0: no samples discarded ACQMOD = 0: Asynchronous continuous acquisition mode DMAEN = 0: DMA interface not used DFLTEN = 1: Filter enabled

1. SNTHR and ANMIN values are computed using the same approach than in *Threshold programming with SADMOD = 01* (detection of a sound higher than 63 dBSPL).

Example 2

This example describes the programming of ADF for the capture of a signal coming from a digital microphone, using the CIC5, and the RSFLT, with a total decimation of 64.

Table 312. Programming sequence (CIC5)

Operations	Comments
Adjust the proper kernel clock frequency via the RCC	Assuming that the RCC is programmed to provide a kernel clock (adf_ker_ck) of 6.144 MHz
Select the proper ADF kernel clock source via the RCC	Refer to the RCC of the product.
Enable the ADF clocks via the RCC	Refer to the RCC of the product.
Reset the ADF via the RCC	Refer to the RCC of the product.
AFMUX programming	Program the AFMUX to select ADF_SD0 and ADF_CCK0 functions.
Enable ADF processing clock: ADF_CKGCR = 0x0201 0023	PROCDIV = 2 (division by 3): adf_proc_ck frequency is 6.144 MHz. CCKDIV = 1 (division by 2): ADF_CCK0 clock frequency is 1.024 MHz. The ADF_CCK0 pin is set in output and generates a clock so that the microphone can exit from low-power mode.

Table 312. Programming sequence (CIC5) (continued)

Operations	Comments
Serial interfaces configuration: ADF_SITF0CR = 0x0000 1F01	SCKSRC = 0 to select ADF_CCK0 as serial clock. SIFTMOD = 0 to select LF_MASTER SIPI mode. Clock absence feature is not working in this mode. The serial interface is enabled.
Bitstream matrix configuration: ADF_BSMX0CR = 0x0000 0000	DFLT0 filter takes the bitstream of SITF0, sampled on rising edge
Digital filter control: ADF_DFLT0CR = 0x0000 0006	NBDIS = 0: no samples discarded ACQMOD = 0: Asynchronous continuous acquisition mode DMAEN = 1: DMA interface not used DFLTEN = 0: Filter disabled FTH = 1: RXFIFO half-full
Filters configuration (CIC): ADF_DFLT0CICR = 0x0010 0F50	SCALE = 0x01(+3.5 dB): Output limited to 22 bits due to RSFLT. MCICD = 0xF (decimation by 16) CICMOD = 5 to select a $Sinc^5$ DATSCR = 0 to select data coming from BSMX
Filters configuration (RSFLT and HPF): ADF_DFLT0RSFR = 0x0000 0200	HPFC = 2: cut-off frequency of 16 kHz * 0.0025 = 40 Hz HPFBYP = 0: HPF not bypassed RSFLTBYP = 0: RSFLT not bypassed RSFLTD = 0: Decimation by 4
Micro delay adjust: ADF_DLY0CR = 0x0000 0000	Not used in this example
Enable interrupt events: ADF_DFLT0IER = x	Enable the interrupt events the application wants to handle.
Clear status flags: ADF_DFLT0ISR = 0x0000 0FFF	Clear all the status flags before running the ADF
Program the DMA Enable the DMA	The DMA must be programmed in order to read the data inside the ADF_DLT0DR register every time a DMA request is generated.
Start acquisition: ADF_DFLT0CR = 0x0000 0007	The ADF starts to filter data.

36.7.3 Connection examples

Figure 379 shows simple connection examples of the ADF to external sensors.

- Picture on the left: two digital microphones connected to the ADF

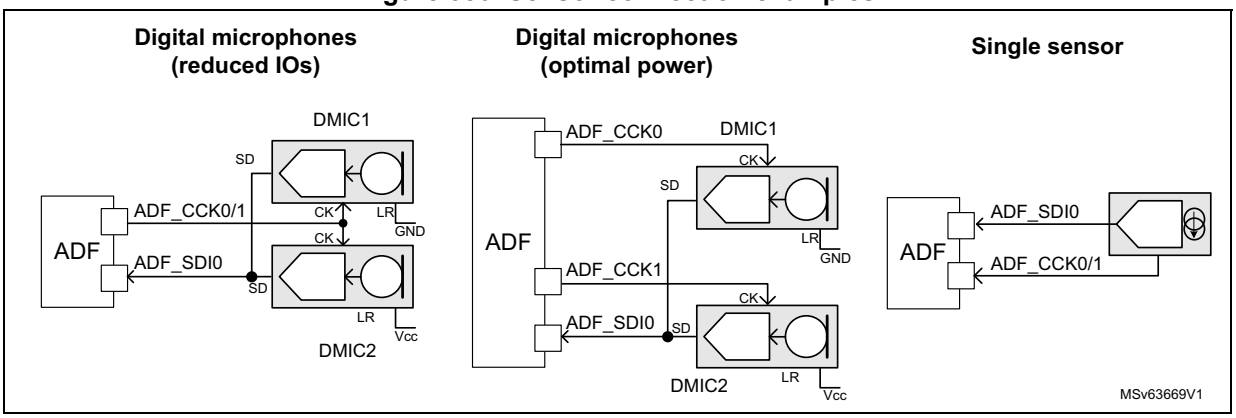
  In this connection, the amount of connections is optimized; DMIC1 and DMIC2 are sharing the same data line, and the same clock line. BSMX allows the application to connect the digital filter either to DMIC1 or to DMIC2. In this configuration when one of the microphone is used, the other is activated as well, as they share the same clock.
- Picture in the center: two digital microphones connected to the ADF

  In this connection, DMIC1 and DMIC2 are sharing the same data line, but have a dedicated clock line. BSMX allows the application to connect the filter either to DMIC1 or to DMIC2. When the application wants to use a microphone, it is possible to keep the other in low-power mode by forcing its clock line to 0 (CCKyEN = 0, CCKyDIR = 1).

- Picture on the right: single sensor connected to the ADF

  It is also possible to configure the CCK0 and CCK1 pins to input in order to connect sensors providing the clock.

Figure 380. Sensor connection examples



36.7.4 Global frequency response

Figure 380 shows the global frequency response for a 16 kHz audio signal with a digital microphone working at 1.024 MHz. The filter configuration is the following:

- CIC order 4 or 5, with a decimation ratio of 16
- RSFLT enabled, with a decimation ratio of 4
- HPF enabled with a cut-off frequency of 40 Hz

Figure 381 shows the theoretical frequency response using a CIC4 and a CIC5.

Figure 381. Global frequency response

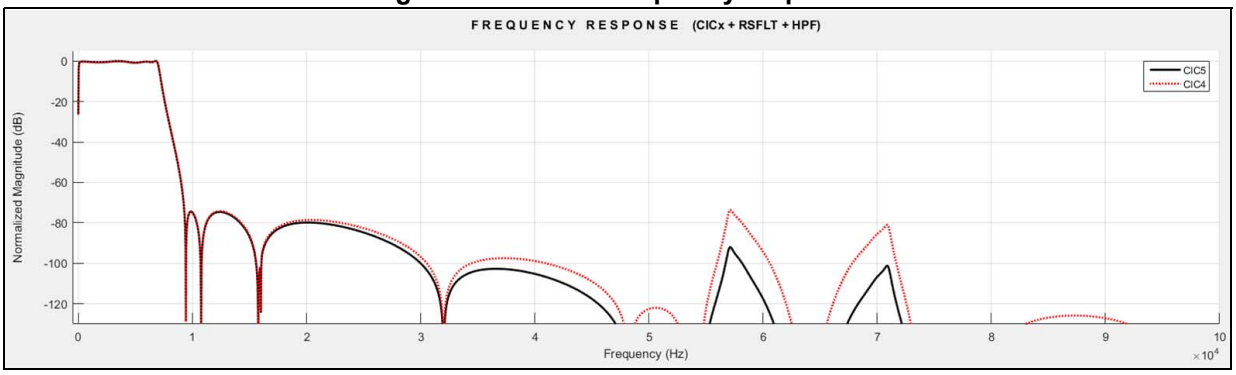


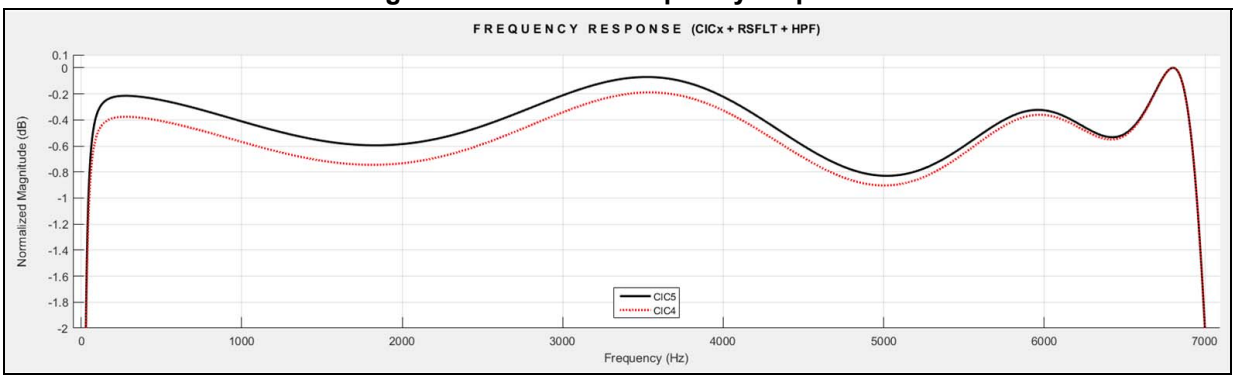
Figure 382 shows the in-band ripple for a 16 kHz audio signal with a digital microphone working at 1.024 MHz. The filter configuration is the following:

- CIC order 4 or 5, with a decimation ratio of 16
- RSFLT enabled, with a decimation ratio of 4
- HPF enabled with a cut-off frequency of 20 Hz

The resulting in-band ripple is ± 0.41 dB for CIC5, and ± 0.45 dB for CIC4.

The - 3 dB cut-off frequency is 7061 Hz.

Figure 382. Detailed frequency response



36.7.5 Total ADF gain

This section details how to compute the signal level provided by the ADF according to the filter settings.

A signal level may be expressed in dBFS (decibel full scale). A 0 dBFS level is assigned to the maximum possible digital level. For example, a signal that reaches 50 % of the maximum level, has a – 6 dBFS level (6 dB below full scale).

For example, for the ADF offering a final data width of 24 bits, a signal having an amplitude of 2 * 10^6 LSB has a level of:

$$20 \times \log_{10}\left(\frac{2 \times 10^6}{2^{(24-1)}}\right) = -12.45 \text{ dBFS}$$

In addition, the data size of a signal having an amplitude (Amp) expressed in LSB is given by:

$$DS = \left(\frac{\ln(Amp)}{\ln(2)} + 1\right) \text{ bits}$$

One bit need to be added for negative values.

So a signal having an amplitude of 2 * 10^6 LSB, has a data size of 21.9 bits.

CIC gain

The CIC gain (G_{CIC} and GdB_{CIC}) can be deduced from the following formula giving data size in bits (DS_{CIC}).

$$DS_{CIC} = (N \times \log_2(D1)) + DSin$$

where N represents the CIC order (selected by CICMOD[2:0]), and D1 is the decimation ratio (given by MCICD[8:0]).

DSin represents the data size (in bits) of the signal at CIC input.

Warning: DS_{CIC} is very important for CIC filter. In order to work fine, DS_{CIC} must not exceed 26 bits.

The CIC gain G_{CIC} is given by:

$$G_{CIC} = (D1)^N$$

which gives in decibels:

$$GdB_{CIC} = 20 \times \log_{10}((D1)^N)$$

Data size at SCALE output

The data size at SCALE output (including the CIC gain), is a key information as the RSFLT starts to have some saturations, if the peak-to-peak signal amplitude at SCALE output is higher than 22 bits.

If the RSFLT is bypassed, then a peak-to-peak signal amplitude of 24 bits is accepted.

The signal amplitude at SCALE output is:

$$Asout_{SCALE} = D1^N \times 10^{\frac{GdB_{SCALE}}{20}} \times Asin_{DFLT}$$

GdB_{SCALE} represents the gain selected by SCALE[5:0], in dB.

$Asout_{SCALE}$ is the signal amplitude at SCALE output (in LSB), and $Asin_{DFLT}$ is the signal amplitude at CIC input (LSB).

$$DS_{SCALE} = \frac{\ln(Asout_{SCALE})}{\ln(2)} + 1$$

The data size at SCALE output (DS_{SCALE}) is expressed in bits.

RSFLT gain

The RSFLT gain in the useful bandwidth is typically 9.5 dB, but due to ripple a margin of about ± 0.41 dB must be considered.

$$G_{RSFLT} = 10^{\frac{9.5\ dB}{20}} = 2.98 \text{ typical}$$

Note: The HPF filter has a gain of 0 dB.

SAD gain

The SAD is using only the 16 MSB on the signal, as a consequence, from the SAD point of view, the truncation from 24 to 16 bits can be seen as an attenuation.

$$GdB_{SAD} = -48.1dB$$

and

$$G_{SAD} = 0.003906$$

The *Figure 383* shows a simplified view of the filter path and gives, for each significant component, the expression of the bit growth and the gain.

Figure 383. Simplified DFLT view with gain information

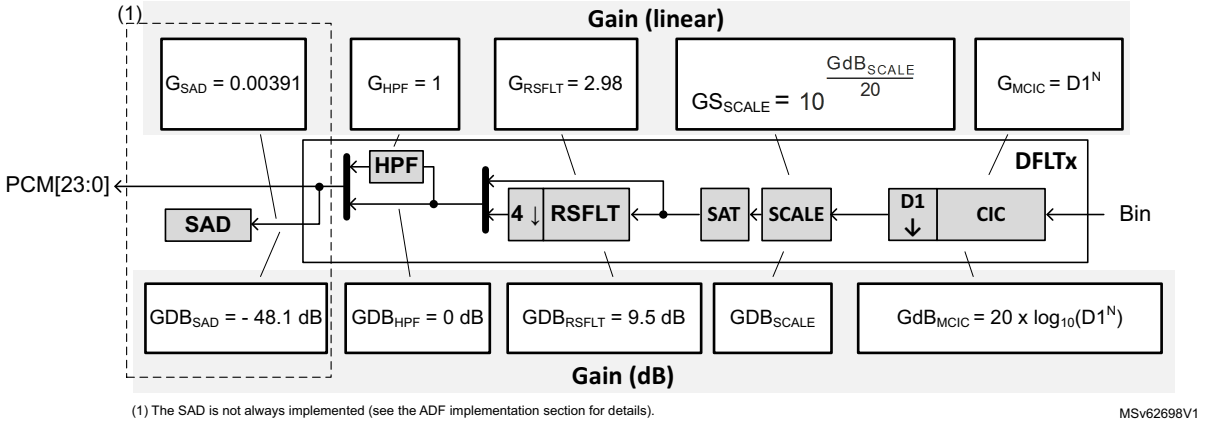


Table 313 summarizes of the final data size for different filter configurations.

Table 313. Output signal levels

Filter configurations	Final signal amplitude (LSB)
CIC + RSFLT + HPF + SAD	Samples provided to the RXFIFO: $Asout_{RXFIFO} = D1^N \times 10^{\frac{GdB_{SCALE}}{20}} \times 10^{\frac{9.5}{20}} \times Asin_{DFLT}$ Samples provided to the SAD: $Asout_{SAD} = D1^N \times 10^{\frac{GdB_{SCALE}}{20}} \times 10^{\frac{9.5}{20}} \times 0.003906 \times Asin_{DFLT}$ DS_{SCALE} must be lower than 22 bits.
CIC + RSFLT (+ HPF)	$ASout_{HPF} = D1^N \times 10^{\frac{GdB_{SCALE}}{20}} \times 10^{\frac{9.5}{20}} \times Asin_{DFLT}$ DS_{SCALE} must be lower than 22 bits.
CIC (+ HPF)	$ASout_{HPF} = D1^N \times 10^{\frac{GdB_{SCALE}}{20}} \times Asin_{DFLT}$ DS_{SCALE} must be lower than 24 bits.

Example using the main filter chain

If the ADF filter is programmed as follows:

- The input signal is coming from a serial interface ($Dsin_{RSFLT}$ = 1 bit).
- CIC order = 5 (N), with a decimation value of 24 (D1).
- SCALE[5:0] is set to - 12 dB.

- RSFLT enabled, and the decimation by four is enabled.
- HPF is enabled.

Check first the data size at CIC output:

$$DS_{CIC} = (5 \times \log_2(24)) + 1 \text{ bit} = 23.92 \text{ bits}$$

The size is lower than 26 bits, so the CIC works in good conditions.

The data size at CIC output is very close to 24 bits, so the SCALE must be adjusted in order to provide a 22-bit max signal to the RSFLT. An attenuation of 12 dB is needed.

Then the signal level provided to the RSFLT is:

$$Asout_{SCALE} = 24^5 \times 10^{\frac{-12}{20}} \times 1 = 2.10^6$$

$$DS_{SCALE} = \frac{\ln(2.10^6)}{\ln(2)} + 1 = 21.93 \text{ bits}$$

If a higher gain is used, the RSFLT may saturate the output signal for strong input signals.

At the end, the final signal amplitude is:

$$Asout_{HPF} = 24^5 \times 10^{\frac{-12}{20}} \times 10^{\frac{9.5}{20}} \times 1 = 5.9711 \times 10^6$$

$$Dsout_{HPF} = \left(\frac{\ln(5.9711 \times 10^6)}{\ln(2)} + 1\right) = 23.51 \text{ bits}$$

or:

$$SDB_{OUT} = 20 \times \log_{10}\left(\frac{2^{23.51}}{2^{24}}\right) = -2.84 \text{ dBFS}$$

36.7.6 How to compute SAD thresholds

The SAD does not compute the RMS value of the converted signal, but the average of the absolute values. As a consequence, the estimated level differs from the RMS value of the signal:

- For a sine signal having an RMS value of 1, the SAD computes a level of 0.9.
- For a white or pink noise signal having an RMS value of 1, the SAD computes a level of about 0.8.

Note: FRSIZE[2:0] has a big influence on the accuracy of the level estimation: big FRSIZE[2:0] values give better results.

Threshold programming with SADMOD = 01

Consider the case of a sound capture where the application wants to wake up the system when the captured sound is bigger than 63 dBSPL.

The sound capture can be performed with a digital microphone such as the MP45DT02.

The sensitivity of this microphone is typically -26 dBFS for an input signal of 94 dBSPL.

An acoustic signal at 63 dBSPL produces a digital signal of about:
- 26 dBFS - (94 - 63) = - 57 dBFS.

- SCALE value adjustment

  For this example, the filter configuration is the following:

  – CIC5 with a decimation by 16
  – RSFLT enabled with a decimation by 4
  – HPF enabled

  A SCALE value of 3.5 dB is recommended for this configuration. As DFTL0 provides samples only used for a sound detection (samples not provided to the application), a bigger gain value can be applied: it increases the SAD accuracy and a saturation does not affect the SAD behavior (for example, a SCALE value of 15.6 dB).

- Input signal amplitude

  The input signal is - 57 dBFS, corresponding to an amplitude

  Asin = $10^{(-57/20)}$ = 0.00141 LSB.

- Signal level at SAD input

  The total filter gain for the SAD is:

  $$G_{SAD} = 16^5 \times 10^{\frac{15.6}{20}} \times 10^{\frac{9.5}{20}} \times \frac{1}{256} = 73.68 \times 10^3 \text{ or } 97.3 \text{ dB}$$

  The signal amplitude received by the SAD is:

  $$Ain_{SAD} = 0.00141 \times G_{SAD} \sim 104 \text{ LSB}$$

  The gain can be increased if the expected amplitude is too small. For the targeted application, 104 LSB is fine.

  If the input signal is expected to be a sine, the sound level for a signal amplitude of 104 LSB is:

  $$SDLVL = \frac{Ain_{SAD} \times \sqrt{2}}{2} \times 0.9 \sim 66 \text{ LSB}$$

  where 0.9 is the correction factor to apply with respect to the RMS value.

- Program the trigger value

  ANMIN and SNTHR must be programmed to trigger the SAD when the input signal level reaches 66 LSB.

  For SADMOD[1:0] = 01, the threshold value is given by:

  $$THRH = ANMIN \times 10^{\frac{GdB_{SNTHR}}{20}}$$

  where GdB_{SNTHR} represents the decibel value selected by SNTHR[3:0].

  When SNTHR[3:0] = 6 dB for example, this formula becomes:

  $$THRH = 2 \times ANMIN$$

  So ANMIN = THRH/2 = 66 /2 = 33 LSB.

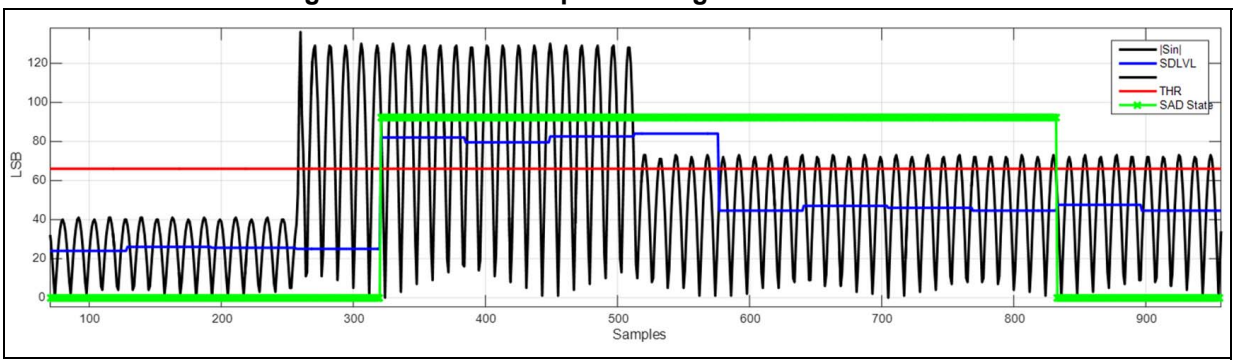
  In *Figure 383*, the trigger value (THRH in red) is fixed to 66 LSB. The input signal is at - 65dBFS during 256 samples, then its value goes to - 55 dB for 256 samples, and finally it is reduced to - 60 dBFS.

The blue curve is showing the sound level estimation (SDLVL) versus time. Fluctuation on the estimated value can be observed due to windowing effect of FRSIZE samples.

The SAD DETECT state (when green signal is high) is maintained during four additional frames due to hangover function value.

In this example ANSLP = FRSIZE = 3 (64 samples), LFRNB = 0 (2 frames), HGOVR = 0 (4 frames), SNTHR = 1 (6 dB) and ANMIN = 33.

Figure 384. SAD example working with SADMOD = 01



Threshold programming with SADMOD = 1x

Consider the case of a sound capture where the application wants to wake up the system when the captured sound is bigger than 57 dBSPL.

The sound capture can be performed with a digital microphone such as the MP45DT02. The sensitivity of this microphone is typically - 26 dBFS for an input signal of 94 dBSPL.

An acoustic signal at 57 dBSPL produces a digital signal of about:
- 26 dBFS - (94-63) = - 63 dBFS.

- Adjust SCALE value

  For this example, the filter configuration is the following:
  - CIC4 with a decimation by 48
  - RSFLT bypassed
  - HPF enabled

  A SCALE value of 3.5 dB is recommended for this configuration. The samples provided by DFTL0 are only used for a sound detection, without providing the samples to the application, a bigger gain value can be provided: it increases the SAD accuracy and a saturation does not affect the SAD behavior (for example, a SCALE value of 24 dB).

- Input signal amplitude

  The input signal is - 63 dBFS, corresponding to an amplitude:

  Asin = $10^{(-63/20)}$ = 0.000708 LSB.

- Signal level at SAD input

  The total filter gain for the SAD is:

$$G_{SAD} = 48^4 \times 10^{\frac{24}{20}} \times 0.003906 = 328.6 \times 10^3 \text{ or } 110.3 \text{ dB}$$

  The signal amplitude received by the SAD is:

$$Ain_{SAD} = 0.000708 \times G_{SAD} \sim 232 \text{ LSB}$$

The gain can be increased if the expected amplitude is too small.

If the input signal is expected to be a sine, the sound level for a signal amplitude of 232 LSB is:

$$SDLVL = \frac{Ain_{SAD} \times \sqrt{2}}{2} \times 0.9 \sim 148 \text{ LSB}$$

where 0.9 is the correction factor to apply with respect to the RMS value.

Note: ANLVL converges to average of SDLVL values, with a long constant time.
So SDLVL ~ ANLVL = 148 LSB for a constant input signal at 57 dBSPL.

- Programming trigger value

  For SADMOD = 1', the SAD compares the estimated ambient noise multiplied by the gain selected by SNTHR[3:0] to ANMIN[12:0] * 4.

  For simplification, SNTHR[3:0] is set to 1 (6 dB), meaning that ANLVL is multiplied by two.

  The SAD triggers if 2 * ANLVL > THRH.

  In this mode:

  $$THRH = 4 \times ANMIN$$

  So the SAD triggers if:

  $$ANLVL > 2 \times ANMIN$$

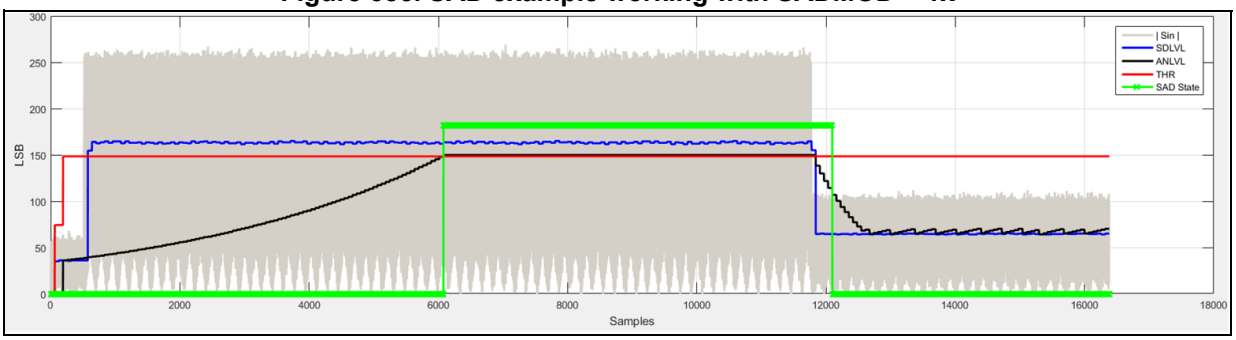
  So ANMIN = 148 / 2 = 74 LSB

  In *Figure 384*, the trigger value (THRH in red) is fixed to 148 LSB. The input signal is at - 75 dBFS during 512 samples, then its value goes to - 62 dB for 11000 samples, and finally it is reduced to - 70 dBFS.

  The blue curve shows the sound level estimation (SDLVL) versus time. The black curve shows the ambient noise estimation versus time, increasing or decreasing logarithmically. During the learning phase, it reaches the SDLVL value.

  In this example ANSLP = 6, FRSIZE = 3 (64 samples), LFRNB = 0 (2 frames), HGOVR = 0 (4 frames), SNTHR = 1 (6 dB) and ANMIN = 74.

Figure 385. SAD example working with SADMOD = 1x



36.8 ADF registers

All the ADF registers must be accessed either in word (32-bit) or half-word (16-bit) formats.

36.8.1 ADF global control register (ADF_GCR)

Address offset: 0x000

Reset value: 0x0000 0000

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

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

Bit 0 **TRGO**: Trigger output control

This bit is set by software and reset by hardware. It is used to start the acquisition of several filters synchronously. It is also used to synchronize several ADF together by controlling the adf_trgo signal.

0: Write 0 has no effect. Read 0 means that the trigger can be set again to 1.

1: Write 1 generates a positive pulse on the adf_trgo signal and triggers the acquisition on enabled filter having their ACQMOD[2:0] = 01x and selecting TRGO as trigger. Read 1 means that the trigger pulse is still active.

36.8.2 ADF clock generator control register (ADF_CKGCR)

Address offset: 0x004

Reset value: 0x0000 0000

This register is used to control the clock generator. The clock adf_proc_ck must be enabled before enabling other ADF parts.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CKGACTIVE	PROCDIV[6:0]							Res.	Res.	Res.	Res.	CCKDIV[3:0]			
r	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRGSRC[3:0]				Res.	Res.	Res.	TRGSENS	Res.	CCK1DIR	CCK0DIR	CKGMOD	Res.	CCK1EN	CCK0EN	CKGDEN
rw	rw	rw	rw				rw		rw	rw	rw		rw	rw	rw

Bit 31 **CKGACTIVE**: Clock generator active flag

This bit is set and cleared by hardware. Ii is used by the application to check if the clock generator is effectively enabled (active) or not. The protected fields of this function can only be updated when CKGACTIVE = 0 (see *Section 36.4.13: Register protection* for details).

The delay between a transition on CKGDEN and a transition on CKGACTIVE is two periods of AHB clock and two 2 periods of adf_proc_ck.

0: The clock generator is not active and can be configured if needed.

1: The clock generator is active and protected fields cannot be configured.

Bits 30:24 **PROCDIV[6:0]**: Divider to control the serial interface clock

This bitfield is set and reset by software. It is used to adjust the frequency of the clock provided to the SITF.

$$F_{adf_itf_ck} = \frac{F_{adf_ker_ck}}{(PROCDIV + 1)}$$

This bitfield must not be changed if the filter is enabled (DFTEN = 1).

0: adf_ker_ck provided to the SITF

1: adf_ker_ck / 2 provided to the SITF

2: adf_ker_ck / 3 provided to the SITF

...

127: adf_ker_ck / 128 provided to the SITF

Note: This bitfield can be write-protected (see Section 36.4.13: Register protection for details).

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **CCKDIV[3:0]**: Divider to control the ADF_CCK clock

This bitfield is set and reset by software. It is used to adjust the frequency of the ADF_CCK clock. The input clock of this divider is the clock provided to the SITF. More globally, the frequency of the ADF_CCK is given by the following formula:

$$F_{ADF_CCK} = \frac{F_{adf_ker_ck}}{(PROCDIV + 1) \times (CCKDIV + 1)}$$

This bitfield must not be changed if the filter is enabled (DFTEN = 1).

0000: The ADF_CCK clock is adf_proc_ck.

0001: The ADF_CCK clock is adf_proc_ck / 2.

0010: The ADF_CCK clock is adf_proc_ck / 3.

...

1111: The ADF_CCK clock is adf_proc_ck / 16.

Note: This bitfield can be write-protected (see Section 36.4.13: Register protection for details).

Bits 15:12 **TRGSRC[3:0]**: Digital filter trigger signal selection

This bitfield is set and cleared by software. It is used to select which external signals trigger the corresponding filter. This bitfield is not significant if the CKGMOD = 0.

000x: TRGO selected

0010: adf_trg1 selected

Others: Reserved

Note: This bitfield can be write-protected (see Section 36.4.13: Register protection for details).

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **TRGSENS**: CKGEN trigger sensitivity selection

This bit is set and cleared by software. It is used to select the trigger sensitivity of the trigger signals. This bit is not significant if the CKGMOD = 0.

0: A rising edge event triggers the activation of CKGEN dividers.

1: A falling edge even triggers the activation of CKGEN dividers.

Note: When the trigger source is TRGO, the sensitivity is forced to falling edge, thus TRGSENS value is not taken into account. This bit can be write-protected (see Section 36.4.13: Register protection for details).

Bit 7 Reserved, must be kept at reset value.

Bit 6 **CCK1DIR**: ADF_CCK1 direction

This bit is set and reset by software. It is used to control the direction of the ADF_CCK1 pin.

0: The ADF_CCK1 pin direction is in input.

1: The ADF_CCK1 pin direction is in output.

Note: This bit can be write-protected (see Section 36.4.13: Register protection for details).

Bit 5 **CCK0DIR**: ADF_CCK0 direction

This bit is set and reset by software. It is used to control the direction of the ADF_CCK0 pin.

0: The ADF_CCK0 pin direction is in input.

1: The ADF_CCK0 pin direction is in output.

Note: This bit can be write-protected (see Section 36.4.13: Register protection for details).

Bit 4 **CKGMOD**: Clock generator mode

This bit is set and reset by software. It is used to define the way the clock generator is enabled. This bit must not be changed if the filter is enabled (DFTEN = 1).

0: The kernel clock is provided to the dividers as soon as CKGDEN is set to 1.

1: The kernel clock is provided to the dividers when CKGDEN is set to 1 and the trigger condition met.

Note: This bit can be write-protected (see Section 36.4.13: Register protection for details).

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CCK1EN**: ADF_CCK1 clock enable

This bit is set and reset by software. It is used to control the generation of the bitstream clock on the ADF_CCK1 pin.

0: Bitstream clock not generated

1: Bitstream clock generated on the ADF_CCK1 pin.

Bit 1 **CCK0EN**: ADF_CCK0 clock enable

This bit is set and reset by software. It is used to control the generation of the bitstream clock on the ADF_CCK0 pin.

0: Bitstream clock not generated

1: Bitstream clock generated on the ADF_CCK0 pin

Bit 0 **CKGDEN**: CKGEN dividers enable

This bit is set and reset by software. It is used to enable/disable the clock dividers of the CKGEN: PROCDIV and CCKDIV.

0: CKGEN dividers disabled

1: CKGEN dividers enabled

36.8.3 ADF serial interface control register 0 (ADF_SITF0CR)

Address offset: 0x080

Reset value: 0x0000 1F00

This register is used to control the serial interface SITF0.

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

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

Bit 31 **SITFACTIVE**: Serial interface active flag

This bit is set and cleared by hardware. It is used by the application to check if the serial interface is effectively enabled (active) or not. The protected fields of this function can only be updated when SITFACTIVE is set to 0 (see *Section 36.4.13: Register protection* for details).
The delay between a transition on SITFEN and a transition on SITFACTIVE is two periods of AHB clock and two periods of adf_proc_ck.
0: The serial interface is not active, and can be configured if needed.
1: The serial interface is active and protected fields cannot be configured.

Bits 30:13 Reserved, must be kept at reset value.

Bits 12:8 **STH[4:0]**: Manchester symbol threshold/SPI threshold

This bitfield is set and cleared by software. It is used for Manchester mode to define the expected symbol threshold levels (see *Manchester mode* for details on computation).
In addition this bitfield is used to define the timeout value for the clock absence detection in Normal SPI mode. STH[4:0] values lower than four are invalid.

Note: This bitfield can be write-protected (see Section 36.4.13: Register protection for details).

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **SITFMOD[1:0]**: Serial interface type

This bitfield is set and cleared by software. It is used to define the serial interface type.
00: LF_MASTER SPI mode
01: Normal SPI mode
10: Manchester mode: rising edge = logic 0, falling edge = logic 1
11: Manchester mode: rising edge = logic 1, falling edge = logic 0

Note: This bitfield can be write-protected (see Section 36.4.13: Register protection for details).

Bit 3 Reserved, must be kept at reset value.

Bits 2:1 **SCKSRC[1:0]**: Serial clock source

This bitfield is set and cleared by software. It is used to select the clock source of the serial interface.
00: Serial clock source is ADF_CCK0.
01: Serial clock source is ADF_CCK1.
others: reserved

Note: This bitfield can be write-protected (see Section 36.4.13: Register protection for details).

Bit 0 **SITFEN**: Serial interface enable

This bit is set and cleared by software. It is used to enable/disable the serial interface.
0: Serial interface disabled
1: Serial interface enabled

36.8.4 ADF bitstream matrix control register 0 (ADF_BSMX0CR)

Address offset: 0x084

Reset value: 0x0000 0000

This register is used to select the bitstream to be provided to DFLT0.

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

Bit 31 **BSMXACTIVE**: BSMX active flag

This bit is set and cleared by hardware. It is used by the application to check if the BSMX is effectively enabled (active) or not. BSSEL[4:0] can only be updated when BSMXACTIVE is set to 0. This BSMXACTIVE flag cannot go to 0 if DFLT0 is enabled.

0: BSMX is not active and can be configured if needed.

1: BSMX is active and protected fields cannot be configured.

Bits 30:5 Reserved, must be kept at reset value.

Bits 4:0 **BSSEL[4:0]**: Bitstream selection

This bitfield is set and cleared by software. It is used to select the bitstream to be processed for DFLT0.

00000: bs0_r provided to DFLT0

00001: bs0_f provided to DFLT0

others: reserved

Note: This bitfield can be write-protected (see Section 36.4.13: Register protection for details).

36.8.5 ADF digital filter control register 0 (ADF_DFLT0CR)

Address offset: 0x088

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DFLTACTIVE	DFLTRUN	Res.	Res.	NBDIS[7:0]								Res.	Res.	Res.	Res.
r	r			rw	rw	rw	rw	rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRGSRC[3:0]				Res.	Res.	Res.	TRGSENS	Res.	ACQMOD[2:0]			Res.	FTH	DMAEN	DFLTEN
rw	rw	rw	rw				rw		rw	rw	rw		rw	rw	w

Bit 31 **DFLTACTIVE**: DFLT0 active flag

This bit is set and cleared by hardware. It indicates if DFLT0 is active: can be running or waiting for events.

0: DFLT0 not active (can be re-enabled again, via DFLTEN bit, if needed)

1: DFLT0 active

Bit 30 **DFLTRUN**: DFLT0 run status flag

This bit is set and cleared by hardware. It indicates if DFLT0 is running or not.

0: DFLT0 not running and ready to accept a new trigger event

1: DFLT0running

Bits 29:28 Reserved, must be kept at reset value.

Bits 27:20 **NBDIS[7:0]**: Number of samples to be discarded

This bitfield is set and cleared by software. It is used to define the number of samples to be discarded every time DFLT0 is re-started.

0: No sample discarded

1: 1 sample discarded

2: 2 samples discarded

...

255: 255 samples discarded

Note: This bitfield can be write-protected (see Section 36.4.13: Register protection for details).

Bits 19:16 Reserved, must be kept at reset value.

Bits 15:12 **TRGSRC[3:0]**: DFLT0 trigger signal selection

This bitfield is set and cleared by software. It is used to select which external signals trigger DFLT0.

0000: TRGO selected

0010: adf_trgi selected

Others: Reserved

Note: This bitfield can be write-protected (see Section 36.4.13: Register protection for details).

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **TRGSENS**: DFLT0 trigger sensitivity selection

This bitfield is set and cleared by software. It is used to select the trigger sensitivity of the external signals

0: A rising edge event triggers the acquisition.

1: A falling edge even triggers the acquisition.

Note: When the trigger source is TRGO, TRGSENS value is not taken into account. When TRGO is selected, the sensitivity is forced to falling edge.
This bit can be write-protected (see Section 36.4.13: Register protection for details).

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ACQMOD[2:0]**: DFLT0 trigger mode

This bitfield is set and cleared by software. It is used to select the filter trigger mode.

000: Asynchronous continuous acquisition mode

001: Asynchronous single-shot acquisition mode

010: Synchronous continuous acquisition mode

011: Synchronous single-shot acquisition mode

100: Window continuous acquisition mode

Others: Same as 000

Note: This bitfield can be write-protected (see Section 36.4.13: Register protection for details).

Bit 3 Reserved, must be kept at reset value.

Bit 2 **FTH**: RXFIFO threshold selection

This bit is set and cleared by software. It is used to select the RXFIFO threshold.

0: RXFIFO threshold event generated when the RXFIFO is not empty

1: RXFIFO threshold event generated when the RXFIFO is half-full

Note: This bit can be write-protected (see Section 36.4.13: Register protection for details).

Bit 1 **DMAEN**: DMA requests enable

This bit is set and cleared by software. It is used to control the generation of DMA request to transfer the processed samples into the memory.

0: DMA interface for the corresponding digital filter disabled

1: DMA interface for the corresponding digital filter enabled

Note: This bit can be write-protected (see Section 36.4.13: Register protection for details).

Bit 0 **DFLTEN**: DFLT0 enable

This bit is set and cleared by software. It is used to control the start of acquisition of the DFLT0 path. This bit behavior depends on ACQMOD[2:0] and external events. The serial or parallel interface delivering the samples must be enabled as well.

0: Acquisition immediately stopped

1: Acquisition immediately started if ACQMOD[2:0] = 00x or 101, or acquisition started when the proper trigger event occurs if ACQMOD[2:0] = 01x.

36.8.6 ADF digital filer configuration register 0 (ADF_DFLT0CICR)

Address offset: 0x08C

Reset value: 0x0000 0000

This register is used to control the main CIC filter.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	SCALE[5:0]						Res.	Res.	Res.	MCICD 8
						rw	rw	rw	rw	rw	rw				rw

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

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

Bits 25:20 **SCALE[5:0]**: Scaling factor selection

This bitfield is set and cleared by software. It is used to select the gain to be applied at CIC output (see *Table 301* for details). If the application attempts to write a new gain value while the previous one is not yet applied, this new gain value is ignored. Reading back this bitfield informs the application on the current gain value.

000000: 0 dB

000001: + 3.5 dB

000010: + 6 dB or shift left by 1 bit

...

011000: + 72 dB or shift left by 12 bits

100000: - 48.2 dB or shift right by 8 bits (default value)

100001: - 44.6 dB

100010: - 42.1 dB or shift right by 7 bits

100011: - 38.6 dB

...

101110: -6 dB or shift right by 1 bit

101111: -2.5 dB

Others: Reserved

Bits 19:17 Reserved, must be kept at reset value.

Bits 16:8 **MCICD[8:0]**: CIC decimation ratio selection
This bitfield is set and cleared by software.It is used to select the CIC decimation ratio. A decimation ratio smaller than two is not allowed. The decimation ratio is given by (CICDEC+1).
0: Decimation ratio is 2.
1: Decimation ratio is 2.
2: Decimation ratio is 3.
3: Decimation ratio is 4.
...
511: Decimation ratio is 512.
Note: This bitfield can be write-protected (see Section 36.4.13: Register protection for details).

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **CICMOD[2:0]**: Select the CIC order
This bitfield is set and cleared by software. It is used to select the order of the MCIC.
100: MCIC configured in single Sinc^4 filter
101: MCIC configured in single Sinc^5 filter
Others: Reserved
Note: This bitfield can be write-protected (see Section 36.4.13: Register protection for details).

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **DATSRC[1:0]**: Source data for the digital filter
This bitfield is set and cleared by software.
0x: Stream coming from the BSMX selected
10: Stream coming from the ADCITF1 selected
11: Stream coming from the ADCITF2 selected
Note: This bitfield can be write-protected (see Section 36.4.13: Register protection for details).

36.8.7 ADF reshape filter configuration register 0 (ADF_DFLT0RSFR)

Address offset: 0x090

Reset value: 0x0000 0000

This register is used to control the reshape and HPF filter.

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

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

Bits 9:8 **HPFC[1:0]**: High-pass filter cut-off frequency
This bitfield is set and cleared by software. it is used to select the cut-off frequency of the high-pass filter. F_{PCM} represents the sampling frequency at HPF input.
00: Cut-off frequency = 0.000625 x FPCM
01: Cut-off frequency = 0.00125 x FPCM
10: Cut-off frequency = 0.00250 x FPCM
11: Cut-off frequency = 0.00950 x FPCM
Note: This bitfield can be write-protected (see Section 36.4.13: Register protection for details).

Bit 7 **HPFBYP**: High-pass filter bypass
This bit is set and cleared by software. It is used to bypass the high-pass filter.
0: HPF not bypassed (default value)
1: HPF bypassed
Note: This bit can be write-protected (see Section 36.4.13: Register protection for details).

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **RSFLTD**: Reshaper filter decimation ratio
This bit is set and cleared by software. It is used to select the decimation ratio for the reshape filter
0: Decimation ratio is 4 (default value).
1: Decimation ratio is 1.
Note: This bit can be write-protected (see Section 36.4.13: Register protection for details).

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **RSFLTBYP**: Reshaper filter bypass
This bit is set and cleared by software. It is used to bypass the reshape filter and its decimation block.
0: Reshape filter not bypassed (default value)
1: Reshape filter bypassed
Note: This bit can be write-protected (see Section 36.4.13: Register protection for details).

36.8.8 ADF delay control register 0 (ADF_DLY0CR)

Address offset: 0x0A4

Reset value: 0x0000 0000

This register is used for the adjustment stream delays.

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

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

Bit 31 **SKPBF**: Skip busy flag
This bit is set and cleared by hardware. It is used to control if the delay sequence is completed.
0: ADF ready to accept a new value into SKPDLY[6:0]
1: Last valid SKPDLY[6:0] still under precessing

Bits 30:7 Reserved, must be kept at reset value.

Bits 6:0 **SKPDLY[6:0]**: Delay to apply to a bitstream

This bitfield is set and cleared by software. It defines the number of input samples that are skipped. Skipping is applied immediately after writing to this bitfield, if SKPBF = 0 and DFLTEN = 1. If SKPBF = 1, the value written into the register is ignored by the delay state machine.

0: No input sample skipped

1: 1 input sample skipped

...

127: 127 input samples skipped

36.8.9 ADF DFLT0 interrupt enable register (ADF_DFLT0IER)

Address offset: 0x0AC

Reset value: 0x0000 0000

This register is used for allowing or not the events to generate an interrupt.

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

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **SDLVLIE**: SAD sound-level value ready enable

This bit is set and cleared by software.

0: Sound-level-ready interrupt disabled

1: Sound-level-ready interrupt enabled

Bit 12 **SDDETIE**: Sound activity detection interrupt enable

This bit is set and cleared by software.

0: Sound-trigger interrupt disabled

1: Sound-trigger interrupt enabled

Bit 11 **RFOVRIE**: Reshape filter overrun interrupt enable

This bit is set and cleared by software.

0: Reshape filter overrun interrupt disabled

1: Reshape filter overrun interrupt enabled

Bit 10 **CKABIE**: Clock absence detection interrupt enable

This bit is set and cleared by software.

0: Clock absence interrupt disabled

1: Clock absence interrupt enabled

Bit 9 **SATIE**: Saturation detection interrupt enable

This bit is set and cleared by software.

0: Saturation interrupt disabled

1: Saturation interrupt enabled

Bits 8:2 Reserved, must be kept at reset value.

Bit 1 **DOVRIE**: Data overflow interrupt enable
This bit is set and cleared by software.
0: Data overflow interrupt disabled
1: Data overflow interrupt enabled

Bit 0 **FTHIE**: RXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: RXFIFO threshold interrupt disabled
1: RXFIFO threshold interrupt enabled

36.8.10 ADF DFLT0 interrupt status register 0 (ADF_DFLT0ISR)

Address offset: 0x0B0

Reset value: 0x0000 0000

This register contains the status flags for the digital filter path.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	SDLVLF	SDDETF	RFOVRF	CKABF	SATF	Res.	Res.	Res.	Res.	Res.	RXNEF	Res.	DOVRF	FTHF
		rc_w1	rc_w1	rc_w1	rc_w1	rc_w1						r		rc_w1	r

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **SDLVLF**: Sound level value ready flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that new sound level value is not ready. Write 0 has no effect.
1: Read 1 means that new sound level value is ready. Write 1 clears this flag.

Bit 12 **SDDETF**: Sound activity detection flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no sound activity is detected. Write 0 has no effect.
1: Read 1 means that sound activity is detected. Write 1 clears this flag.

Bit 11 **RFOVRF**: Reshape filter overrun detection flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no reshape filter overrun is detected. Write 0 has no effect.
1: Read 1 means that reshape filter overrun is detected. Write 1 clears this flag.

Bit 10 **CKABF**: Clock absence detection flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no clock absence is detected. Write 0 has no effect.
1: Read 1 means that a clock absence is detected. Write 1 clears this flag.

Bit 9 **SATF**: Saturation detection flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no saturation is detected. Write 0 has no effect.
1: Read 1 means that a saturation is detected. Write 1 clears this flag.

Bits 8:4 Reserved, must be kept at reset value.

Bit 3 **RXNEF**: RXFIFO not empty flag
This bit is set and cleared by hardware according to the RXFIFO level.
0: RXFIFO empty
1: RXFIFO not empty

Bit 2 Reserved, must be kept at reset value.

Bit 1 **DOVRF**: Data overflow flag
This bit is set by hardware and cleared by software by writing this bit to 1.
0: Read 0 means that no overflow is detected. Write 0 has no effect.
1: Read 1 means that an overflow is detected; Write 1 clears this flag.

Bit 0 **FTHF**: RXFIFO threshold flag
This bit is set by hardware, and cleared by the hardware when the RXFIFO level is lower than the threshold.
0: RXFIFO threshold not reached
1: RXFIFO threshold reached

36.8.11 ADF SAD control register (ADF_SADCR)

Address offset: 0x0B8

Reset value: 0x0000 0000

This register is used for the configuration and the control of the sound activity detection.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SADACTIVE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	SADMOD[1:0]		Res.	FRSIZE[2:0]			HYSTEN	Res.	SADST[1:0]		DETCFG	DATCAP[1:0]		SADEN
		rw	rw		rw	rw	rw	rw		r	r	rw	rw	rw	rw

Bit 31 **SADACTIVE**: SAD Active flag
This bit is set and cleared by hardware. It is used to check if the SAD is effectively enabled (active) or not. The protected fields and registers of this function can only be updated when the SADACTIVE is set to 0 (see *Section 36.4.13: Register protection* for details).
The delay between a transition on SADEN and a transition on SADACTIVE is two periods of AHB clock and two periods of adf_proc_ck.
0: SAD not active and can be configured if needed
1: SAD active and protected fields cannot be configured.

Bits 30:14 Reserved, must be kept at reset value.

Bits 13:12 **SADMOD[1:0]**: SAD working mode

This bitfield is set and cleared by software. It is used to define the way the SAD works.

00: Threshold value computed according to the estimated ambient noise

The SAD triggers when the sound level (SDLVL) is bigger than the defined threshold. In this mode, the SAD works like a voice activity detector.

01: Threshold value equal to ANMIN[12:0], multiplied by the gain selected by SNTHR[3:0]

The SAD triggers when the sound level (SDLVL) is bigger than the defined threshold. In this mode, the SAD works like a sound detector.

1x: Threshold value given by 4 x ANMIN[12:0]

The SAD triggers when the estimated ambient noise (ANLVL), multiplied by the gain selected by SNTHR[3:0] is bigger than the defined threshold. In this mode, the SAD is working like an ambient noise estimator. Hysteresis function cannot be used in this mode.

Note: This bitfield can be write-protected (see Section 36.4.13: Register protection for details).

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **FRSIZE[2:0]**: Frame size

This bitfield is set and cleared by software. it is used to define the size of one frame and also to define how many samples are taken into account to compute the short-term signal level.

000: 8 PCM samples used to compute the short-term signal level

001: 16 PCM samples used to compute the short-term signal level

010: 32 PCM samples used to compute the short-term signal level

011: 64 PCM samples used to compute the short-term signal level

100: 128 PCM samples used to compute the short-term signal level

101: 256 PCM samples used to compute the short-term signal level

11x: 512 PCM samples used to compute the short-term signal level

Note: This bitfield can be write-protected (see Section 36.4.13: Register protection for details).

Bit 7 **HYSTEN**: Hysteresis enable

This bit is set and cleared by software. It is used to enable/disable the hysteresis function (see *Table 301* for details). This bit must be kept to 0 when SADMOD[1:0] = 1x.

0: Hysteresis function disabled. THR_H is always used.

1: Hysteresis function enabled. THR_H is used for MONITOR to DETECT transition and THR_L is used for DETECT to MONITOR transition.

Note: This bit can be write-protected (see Section 36.4.13: Register protection for details).

Bit 6 Reserved, must be kept at reset value.

Bits 5:4 **SADST[1:0]**: SAD state

This bitfield is set and cleared by hardware. It indicates the SAD state and is meaningful only when SADEN = 1. The SAD state can be:

- LEARN when the SAD is in learning phase or in SDLVL computation mode
- MONITOR when the SAD is in monitoring phase
- DETECT when the SAD detects a sound

00: SAD in LEARN state

01: SAD in MONITOR state

11: SAD in DETECT state

Bit 3 **DETCFG**: Sound trigger event configuration

This bit is set and cleared by software. It is used to define if the sddet_evt event is generated only when the SAD enters to MONITOR state or when the SAD enters or exits the DETECT state.

0: sddet_evt generated when SAD enters the MONITOR state

1: sddet_evt generated when SAD enters or exits the DETECT state

Note: This bit can be write-protected (see Section 36.4.13: Register protection for details).

Bits 2:1 **DATCAP[1:0]**: Data capture mode
This bitfield is set and cleared by software. It is used to define in which conditions, the samples provided by DLFT0 are stored into the memory.
00: Samples from DFLT0 not transfered into the memory
01: Samples from DFLT0 transfered into the memory when SAD is in DETECT state
1x: Samples from DFLT0 transfered into memory when SAD and DFLT0 are enabled
Note: This bitfield can be write-protected (see Section 36.4.13: Register protection for details).

Bit 0 **SADEN**: Sound activity detector enable
This bit is set and cleared by software. It is used to enable/disable the SAD.
0: SAD disabled and SAD state reset
1: SAD enabled

36.8.12 ADF SAD configuration register (ADF_SADCFGR)

Address offset: 0x0BC

Reset value: 0x0000 0000

This register is used for the configuration of the sound activity detection.

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

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:16 **ANMIN[12:0]**: Minimum noise level
This bitfield is set and cleared by software. It is used to define the minimum noise level and then the sensitivity. It represents a positive number.
Note: This bitfield can be write-protected (see Section 36.4.13: Register protection for details).

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **HGOVR[2:0]**: Hangover time window
This bitfield is set and cleared by software. Once the SAD state is DETECT, this parameter is used to define the amount of time the sound is allowed to remain below the threshold, before switching the SAD to MONITOR state (see FRSIZE bitfield for the description of a frame).
000: SAD back to MONITOR state if sound is below threshold for 4 frames
001: SAD back to MONITOR state if sound is below threshold for 8 frames
010: SAD back to MONITOR state if sound is below threshold for 16 frames
011: SAD back to MONITOR state if sound is below threshold for 32 frames
100: SAD back to MONITOR state if sound is below threshold for 64 frames
101: SAD back to MONITOR state if sound is below threshold for 128 frames
110: SAD back to MONITOR state if sound is below threshold for 256 frames
111: SAD back to MONITOR state if sound is below threshold for 512 frames
Note: This bitfield can be write-protected (see Section 36.4.13: Register protection for details).

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **LFRNB[2:0]**: Number of learning frames

This bitfield is set and cleared by software. It is used to define the number of learning frames to perform the first estimate of the noise level.

000: 2 frames used to compute the initial noise level
001: 4 frames used to compute the initial noise level
010: 8 frames used to compute the initial noise level
011: 16 frames used to compute the initial noise level
1xx: 32 frames used to compute the initial noise level

Note: This bitfield can be write-protected (see Section 36.4.13: Register protection for details).

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ANSLP[2:0]**: Ambient noise slope control

This bitfield is set and cleared by software. It is used to define the positive and negative slope of the noise estimator, in charge of updating the ANLVL (see *Ambient noise estimation (ANLVL)* for information about programming this bitfield).

Note: This bitfield can be write-protected (see Section 36.4.13: Register protection for details).

Bits 3:0 **SNTHR[3:0]**: Signal to noise threshold

This bitfield is set and cleared by software. It is used to define THR_H (and THR_L if hysteresis is enabled). See *Table 301* for details.

0000: THR_H is 3.5 dB higher than ANLVL
0001: THR_H is 6.0 dB higher than ANLVL
0010: THR_H is 9.5 dB higher than ANLVL
0011: THR_H is 12 dB higher than ANLVL
0100: THR_H is 15.6 dB higher than ANLVL
0101: THR_H is 18 dB higher than ANLVL
0110: THR_H is 21.6 dB higher than ANLVL
0111: THR_H is 24.1 dB higher than ANLVL
1000: THR_H is 27.6 dB higher than ANLVL
1001: THR_H is 30.1dB higher than ANLVL
others: Reserved

Note: This bitfield can be write-protected (see Section 36.4.13: Register protection for details).

36.8.13 ADF SAD sound level register (ADF_SADSDLVR)

Address offset: 0x0C0

Reset value: 0x0000 0000

This register contains the short-term sound-level computed by the SAD.

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

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

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:0 **SDLVL[14:0]**: Short term sound level

This bitfield is set by hardware. It contains the latest sound level computed by the SAD. To refresh this value, SDLVLF must be cleared.

36.8.14 ADF SAD ambient noise level register (ADF_SADANLVR)

Address offset: 0x0C4

Reset value: 0x0000 0000

This register contains the ambient noise level computed by the SAD.

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

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:0 **ANLVL[14:0]**: Ambient noise level estimation

This bitfield is set by hardware. It contains the latest ambient noise level computed by the SAD. To refresh this bitfield, the SDLVLF flag must be cleared.

36.8.15 ADF digital filter data register 0 (ADF_DFLT0DR)

Address offset: 0x0F0

Reset value: 0x0000 0000

This register is used to read the data processed by the digital filter.

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

Bits 31:8 **DR[23:0]**: Data processed by DFT0

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

36.8.16 ADF register map

Table 314. ADF register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	ADF_GCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRGO
	Reset value																																0
0x004	ADF_CKGCR	CKGACTIVE	PROCDIV[6:0]							Res.	Res.	Res.	Res.	CCKDIV [3:0]				TRGSRC [3:0]				Res.	Res.	Res.	TRGSENS	Res.	CCK1DIR	CCK0DIR	CKGMOD	Res.	CCK1EN	CCK0EN	CKGDEN
	Reset value	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0				0		0	0	0		0	0	0

Table 314. ADF register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x008 - 0x07C	Reserved																																
0x080	ADF_SITF0CR	SITFACTIVE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STH[4:0]					Res.	Res.	SITFMOD [1:0]		Res.	SCKSRC [1:0]		SITFEN
	Reset value	0																			1	1	1	1	1			0	0		0	0	0
0x084	ADF_BSMX0CR	BSMXACTIVE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BSSEL[4:0]				
	Reset value	0																											0	0	0	0	0
0x088	ADF_DFLT0CR	DFLTACTIVE	DFLTRUN	Res.	Res.	NBDIS[7:0]								Res.	Res.	Res.	Res.	TRGSRC [3:0]				Res.	Res.	Res.	TRGSENS	Res.	ACQMOD [2:0]			Res.	FTH	DMAEN	DFLTEN
	Reset value	0	0			0	0	0	0	0	0	0	0					0	0	0	0				0		0	0	0		0	0	0
0x08C	ADF_DFLT0CICR	Res.	Res.	Res.	Res.	Res.	Res.	SCALE[5:0]						Res.	Res.	Res.	MCICD[8:0]									Res.	CICMOD[2:0]			Res.	Res.	DATSRC[1:0]	
	Reset value							0	0	0	0	0	0				0	0	0	0	0	0	0	0	0		0	0	0			0	0
0x090	ADF_ DFLT0RSFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HPFC[1:0]		HPFBYP	Res.	Res.	RSFLTD	Res.	Res.	Res.	RSFLTBYP
	Reset value																							0	0	0			0				0
0x094 - 0x0A0	Reserved																																
0x0A4	ADF_DLY0CR	SKPBF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SKPDLY[6:0]						
	Reset value	0																									0	0	0	0	0	0	0
0x0A8	Reserved																																
0x0AC	ADF_DFLT0IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDLVLIE	SDDETIE	RFOVRIE	CKABIE	SATIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DOVRIE	FTHIE
	Reset value																			0	0	0	0	0	0	0			0		0	0	0
0x0B0	ADF_DFLT0ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDLVLF	SDDETF	RFOVRF	CKABF	SATF	Res.	Res.	Res.	Res.	Res.	RXNEF	Res.	DOVRF	FTHF
	Reset value																			0	0	0	0	0						0		0	0
0x0B4	Reserved																																

Table 314. ADF register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0B8	ADF_SADCR	SADACTIVE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SADMOD[1:0]		Res.	FRSIZE[2:0]			HYSTEN	Res.	SADST[1:0]		DETCFG	DATCAP[1:0]		SADEN
	Reset value	0																		0	0		0	0	0	0		0	0	0	0	0	0
0x0BC	ADF_SADCFGR	Res.	Res.	Res.	ANMIN[12:0]													Res.	HGOVR [2:0]			Res.	LFRNB [2:0]			Res.	ANSLP[2:0]			Res.	SNTHR [2:0]		
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0		0	0	0		0	0	0		0	0	0
0x0C0	ADF_SADSDLVR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SDLVL[14:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C4	ADF_SADANLVR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ANLVL[14:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C8 - 0x0EC	Reserved																																
0x0F0	ADF_DFLT0DR	DR[23:0]																								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

37 Camera subsystem

37.1 Introduction

The camera subsystem is built around a double path:

- a low-resolution path, with the DCMI and a low-frequency parallel interface. It supports sensors up to 24 Mpixel/s.
- a high-resolution path, with the DCMIPP and a high-frequency parallel interface or serial CSI-2 interface, either RGB or RawBayer. It targets sensors up to 5 Mpixels at 30 fps.

When plugging a camera sensor, it is recommended to use the DCMIPP, a higher performance pipe.

The previous DCMI is recommended only in two noticeable cases:

- to get a backward compatibility with former platforms that embed also the DCMI
- to input the pixels from a second camera sensor

Note: The DCMIPP inputs pixels from one sensor via the CSI-2 interface, while the DCMI gets pixels from the second sensor via the parallel interface.

The DCMI path offers the following summarized maximum features:

- Target sensor: 24 Mpixel/s 16 bpp (limited by the DMA)
- Parallel input: 80 MHz on 14-bit input capability
- Central DMA to extract and dump its pixels
- Software extraction of its pixel data

The DCMIPP path offers the following summarized maximum features:

- Target sensors: 5 Mpixel at 30 fps max
- Parallel input: 120 MHz on 16-bit input capability
- Serial input: CSI-2 at 2.5 Gbps/lane on 2 data lanes
- ISP (image signal processor): demosaicing, exposure, white-balance, contrast, bad-pixel
- Application pipes: 2 pipes with crop, downsize, gamma, YUV conversion, YUV420

Some over-target use cases are possible, but with specific constraints:

- sensors with resolution above 5 Mpixels
- sensors with pixel rate above 150 Mpixel/s
- double sensors in parallel

37.2 Camera path main features

37.2.1 DCMI path main features

The DCMI path shows the following features:

- 8-, 10-, 12- or 14-bit parallel interface at 80 MHz (no I/O ReTime)
- Embedded/external line and frame synchronization
- Continuous or snapshot mode

- Crop feature
- Following data formats supported (dumped in memory, extracted by software):
  - 8-, 10-, 12-, 14-bit progressive video, either monochrome or RawBayer
  - YCbCr 4:2:2 progressive video
  - RGB 565 progressive video
  - Compressed data: JPEG (in fact, any default/byte data)

See *Section 38: Digital camera interface (DCMI)* for more details.

37.2.2 DCMIPP path main features

The DCMIPP path groups the DCMIPP pixel pipeline block, its included parallel interface, and the abutted CSI-2-Host and PHY.

The parallel interface shows the following features:

- Input rate: 120 MHz up to 16-bit capability (with I/O ReTime)
- Pixel format: RGB565, 888, YUV422, RawBayer/Mono 8/10/12/14
- Synchronization: embedded versus external line and frame synchronization

See *Section 39: Digital camera interface pixel pipeline (DCMIPP)* for more details.

The CSI-2 serial interface shows the following features:

- Standard: MIPI CSI2 v1.3
- Input rate: 2 data lanes @ 2.5 Gbps/lane
- Pixel rate: if RGB888: up to 200 Mpixel/s, if RawBayer 10: up to 333 Mpixel/s
- Pixel format: any MIPI CSI2 v1.3: RGB565, 888, YUV422, RawBayer
- Miscellaneous: interlaced video, interleaved packets, virtual channels

See *Section 40: CSI-2 Host (CSI)* for more details.

DCMIPP sizings and limitations

- Camera target: 5 Mpixels (typ. 2560x2048 or 2688x1944) before decimation
- Pipe resolution: 4094x4094 pixels max
  - Demosaicing: 2688 pixels wide max line-buffers (decimation available ahead)
  - Downsize: 2688 pixels wide max line-buffers
  - YUV420: 2688 pixels wide max line-buffers.
- Pipe pixel rate: 333 Mpixel/s peak.
- AXI FIFO: 5 Kbytes available, with a flexible allocation between all outputs.
- AXI master: AXI-64 bits @ 400 MHz, with FIFO for latencies up to 2 µs, 16 outstanding.

The DCMIPP provides three different pipes to process data:

- Pipe0: to dump data without any processing, branched before any ISP

  It is aimed at dumping metadata or statistics as-is from the sensor, or at dumping the full picture as RawBayer data, for a potential long but optimal software processing.
- Pipe1: main full-featured pipe (such as ISP, then downsize, YUV-multi-planar)

  It provides a picture ready for any application (such as display, video encode, or neural processing).

- Pipe2: ancillary semi-featured pipe

  Same ISP as pipe1, then downsize, but no YUV-multiplanar. It provides an RGB output for a display use, or neural processing if using RGB as input.

DCMIPP ISP capabilities common to pipe1 and 2, are the following:

- flow selection: parallel-vs-CSI2, CSI virtual packets
- capture mode: continuous versus snapshot mode
- pixel extraction: RGB565, 888, YUV422, RawBayer/Mono 8/10/12/14.
- statistic removal: to remove unused metadata (like statistics) from received pictures
- bad-pixel removal: automatic detection and correction of bad pixels from sensor array
- decimation: one pixel every 1/2/4/8, used to fit the max 2688 pixels
- exposure control: to amplify the exposure from 0.0 to 255.99x
- statistic extraction: average R,G,B,L values, dynamic bins
- demosaicing: to convert RawBayer to RGB, supports NIR patterns (2x2, 4x4)
- white-balance: to adapt to sensor white and global Illumination
- contrast: to enhance the contrast, via luminance

The DCMIPP application capabilities, replicated for pipe1 and 2, are listed below:

- cropping: to select a region of interest or stabilize the picture
- downsize: box-filtering, with any decimal ratio, up to 8x8 (64x64 with decimation)
- gamma conversion: predistortion with gamma = 2.2 for display purposes
- YUV color conversion: to convert to YUV to later store a video-friendly buffer (pipe1 only).
- YUV422/420: chroma subsampling to YUV422 or YUV420
- output streaming capability of three pixel pipes to a slave peripheral (VENC, NPU, or LP/HPDMA1), using a small multiline internal buffer located in the system memory
- regions of interest (ROI): draw capability of a colored rectangle around each ROI

Output pixel format:

- Pipe1 and 2: 888, 565, 422-1, Y8, ARGB and RGBA (coplanar only)
- Pipe1: to adds422-2, 420-2, 420-3 (multi-planar possibility)

The full features are detailed in *Section 38: Digital camera interface (DCMI)*.

37.3 Camera subsystem implementation

37.3.1 Hardware settings

The camera subsystem is implemented with the following specific settings:

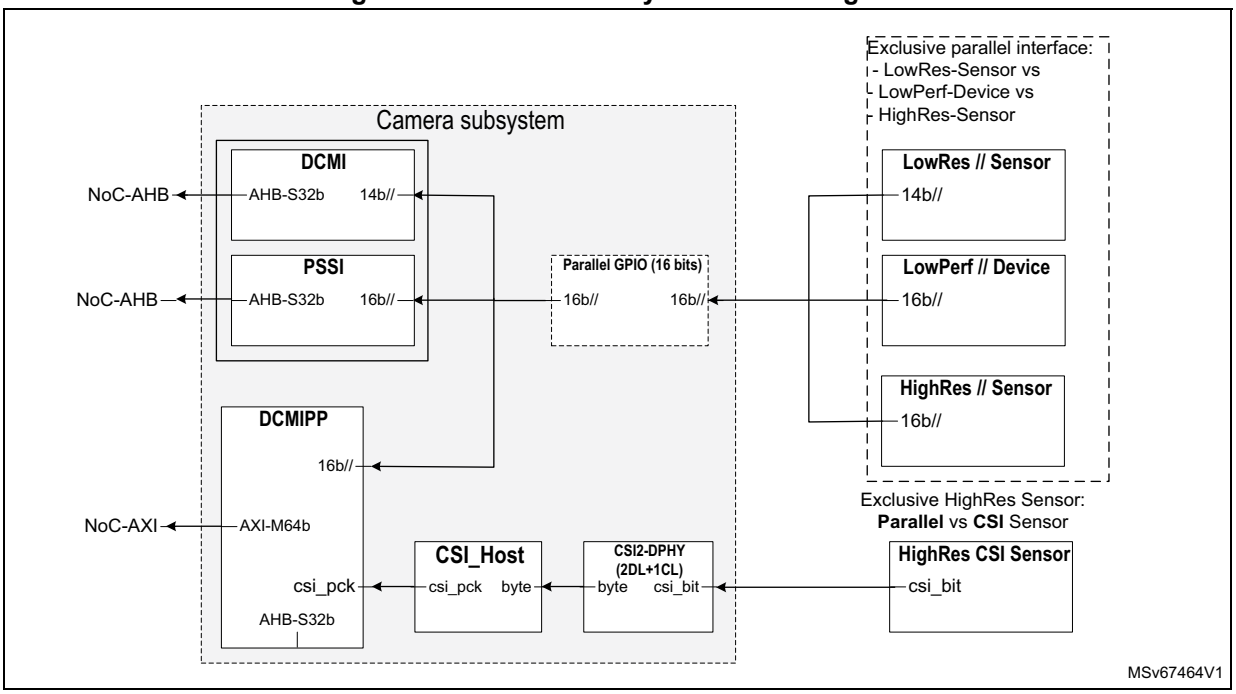
- CSI2: two data lanes (+ one clock lane) up to 2.5 Gbps/lane.
- DCMIPP FIFO: 5 Kbytes for AXI
- DCMIPP with the following subfeatures:
  - line-buffers: 2688 pixels (impacts the demosaicing, downsize, YUV420 conversion)
  - one dump pipe + two pixel pipes (pipe1 with YUV420/multi-planar only)

37.4 Camera subsystem functional description

37.4.1 Camera subsystem block diagram

Figure 386 shows the blocks in the camera subsystem.

Figure 386. Camera subsystem block diagram



Note: The PSSI is not used for camera purposes, but rather for general-purpose parallel input (not described in this section).

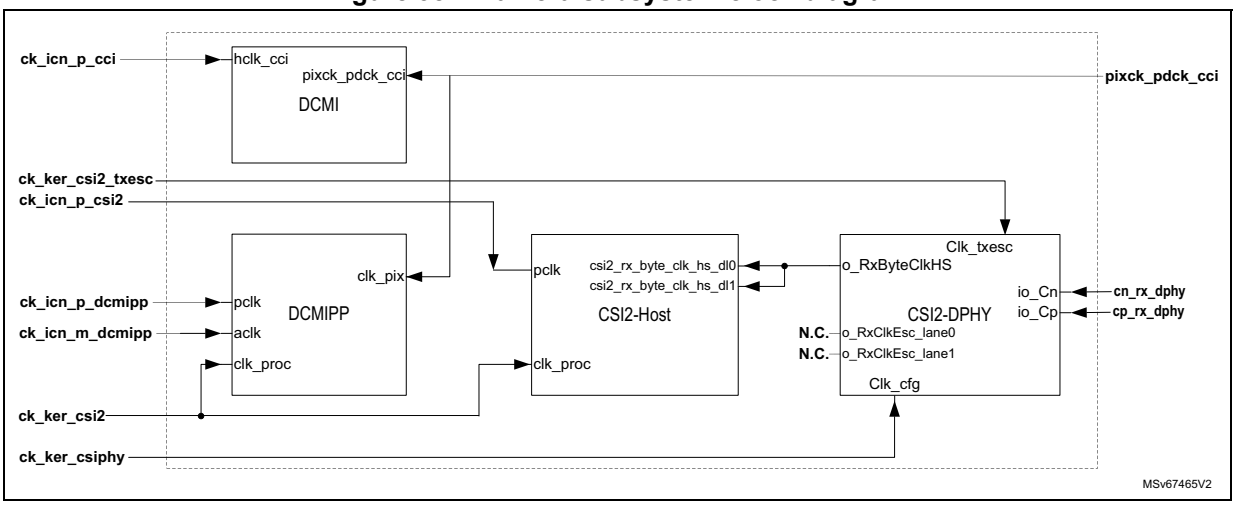
37.4.2 Camera subsystem pins and external signal interface

Details about the pins can be found in the respective block sections.

37.4.3 Camera subsystem reset and clocks

The clock diagram of the camera subsystem is shown in *Figure 387*.

Figure 387. Camera subsystem clock diagram



The camera subsystem has been implemented with the following clock max frequencies:

- DCMI // I/F (pixck_pdck_cci): 80 MHz (no I/O ReTime: async path from pads)
- DCMIPP // I/F (pixck_pdck_cci): 120 MHz (I/O ReTime mandatory)
- CSI2 bit clock (cn/p_rx_dphy): 1250 MHz (2.5 Gbps/lane)
- CSI2 byte clock (rxbyte_clkHS): 312.5 MHz
- CSI2 TxEsc clock (ck_ker_csi2_txesc): 20 MHz
- CSI2 Cfg clock (ck_ker_csiphy): 20 MHz
- Pixel clock (ck_ker_csi2): 333 MHz
- Register clocks (ck_icn_p_*): 200 MHz (all APB/AHB register files)
- AXI clock (ck_icn_m_dcmipp): 400 MHz

37.4.4 Streaming from the camera subsystem to slave peripherals

The DCMIPP in the camera subsystem can stream pixels to a set of slave peripherals NPU, HP/GPDMA, and VENC. This avoids to write the full frame into a frame buffer (and allocate this frame buffer).

The streaming is done via a small intermediate buffer in the system memory, organized in double ping-pong buffer. The double buffer must be a power-of-two amount of lines, with a recommended size of 16 lines high each.

The processing is as follows:

1. The DCMIPP starts writing into the first buffer.
2. After step 1 completion, the DCMIPP emits an hardware trigger signal to the slave peripheral, and continues to write to the second buffer.
3. At hardware trigger reception, the slave peripheral reads from the first buffer. It is expected to read and process the buffer faster than the DCMIPP fills the other.

4. After completing writing the second buffer, the DCMIPP emits again a trigger, and wraps back its write address counter to restart writing into the first buffer.
5. After finishing reading the second buffer, the slave peripheral wraps back reading into the first buffer.

All DCMIPP pipes can be streamed, but not to all slave peripherals:

- Pipe0 (dump pipe) can be streamed to HP/GPDMA.
- Pipe1 (main pixel pipe) can be streamed to HP/GPDMA, NPU, or VENC.
- Pipe2 (aux pixel pipe) can be streamed to HP/GPDMA or NPU.

The VENC can only get streamed from Pipe1 (only pipe that can output a YUV420 pixel format, needed for its H264 or JPG encode).

A same pipe can be streamed simultaneously to several peripherals if the resolution and pixel format match: after receiving the same broadcasted trigger, the various slave peripherals simply read in parallel from the same shared intermediate buffer.

More information about the streaming can be found in the respective peripheral sections:

- *Section 39: Digital camera interface pixel pipeline (DCMIPP)* (see pixel packing and streaming across a ping-pong buffer)
- *Section 20: Neural-ART accelerator™ (NPU)*
- *Section 18: High-performance direct memory access controller (HPDMA)*
- *Section 19: General purpose direct memory access controller (GPDMA)*
- *Section 45: Video encoder (VENC)*

37.4.5 Camera subsystem security

The security of the camera subsystem is based on the protection of its register map, and of its master access to the system interconnect, based on default RISUP and RIMU respectively.

More information can be found in *Section 3.5.3: RIF infrastructure*.

The RISUP differentiates the access right of accesses performed toward the following RIF protected peripheral ID (indexes refer to *Table 20: RISUP indexes*):

- DCMIPP: tag = DCMIPP for the configuration of pixel pipelines (index 93)
- CSI2-Host: tag = CSI2H for Host and PHY configurations (index 926)
- DCMI and PSSI: tag = DCMI for its configuration, and to retrieve pixels by DMA (index 94 and 95 respectively)

The RIMU (RIMU_AXI_DCMIPP) gives the access right to the DCMIPP AXI master: CID, S and P are set using the RIF RIMC_ATTR9 register (no inheritance from RISPU to RIMU).

- CID is set either by the RISA (RISUP86 register) of directly by software.

By default, the camera subsystem is expected to work in a nonsecure mode:

- DCMIPP, CSI2, DCMI, and PSSI are left unprotected, so that any OS software can configure and handle the camera subsystem.
- The AXI master can be configured with a DCMIPP CID = DISP or CID = VID to mention that the AXI write access of the camera subsystem contains pixels (like the display system).

For secure applications, the user can put the camera subsystem in S, P, or S-and-P mode, with the following impacts:

- As a benefit, the user is sure that the content of the dumped buffer does truly come from the camera sensor:

  The output of the DCMIPP is differentiated from other masters: only the DCMIPP can write into its buffer. This buffer content is known to come from the camera sensor, which makes impossible to fill-in (by software) a fake face to be recognized.
- The camera subsystem needs to be transfered to the secure software world (with the verification implications), or the system must make sure that the access rights of the DCMIPP cannot harm the system.

The above recommended non-protected settings are shown in *Table 315: Camera subsystem RIF peripheral ID*.

The table below lists vertically all camera subsystem blocks, and horizontally the associated RIF peripheral ID.

Table color legend for the recommended settings: gray for a permanently non-protected setting, blue for a temporarily protected versus non-protected, and pink for a permanently protected state: by default, all the settings are non-protected.

Table 315. Camera subsystem RIF peripheral ID

Subsystem	Block	Address range	Start address	RISUP - slave		
				94 and 95	93	92
				DCMI and PSSI	DCMIPP	CSI2H
Camera low resolution	DCMI	Configuration	0x0000	Y	-	-
		Pixel Rx	0x0A00	Y	-	-
		Configuration	0x0B00	Y	-	-
		Data in/out	0x0C00	Y	-	-
Camera high resolution	DCMIPP	Common configuration	0x0	-	Y	-
		P1 configuration	0x0	-	Y	-
		P2 configuration	0x0	-	Y	-
		P3 configuration	0x0	-	Y	-
	CSI	Configuration control	0x0000	-	-	Y
		Configuration wrap	0x1000	-	-	Y
		CSI DPHY-Rx (no register)	-	-	-	-

37.5 Camera subsystem programmable parameters

This section describes the configuration of the various main use cases supported by the camera subsystem. Its purpose is to provide the settings that are global to the camera subsystem, and not provided by the local relevant blocks.

The use cases supported by default are the following:

- low-resolution parallel camera, using the DCMI path (see *Section 37.5.1*)

- high-resolution parallel camera, using the DCMIPP path (see *Section 37.5.2*)
- high-resolution CSI2 camera, using the DCMIPP path (see *Section 37.5.3*)

Some over-target use cases are possible, but with specific constraints:

- sensors over target-resolution (above 5 Mpixels)

  The limiting factors are
  - the clock (related to max pixel rate)
  - AXI bandwidth (related to the pixel rate and pixel format of the active pipes)
  - line buffers of the demosaicing, downsize, and YUV420 chroma decimation (when active)

  To cope with these limitations, the decimation can be used, ahead of the pipes, to reduce the width and/or height of the frame, and/or deactivate the limiting operators, the demosaicing, downsize, YUV420.
- sensors over target pixel rate (above 150 Mpixel/s)
- double sensors like 3D camera (with their left and right sensors)

37.5.1 Low-resolution parallel camera and DCMI

This use case is recommended only for backward compatibility, or to grab pictures simultaneously from two sensors (a low and a high resolution).

The steps to setup this path are the following:

- Clock and reset
  - pixck_pdck_cci is provided by the sensor (via the PCB on the input I/O, 80 MHz max).
  - The RCC provides ck_icn_p_cci, that must be ≥ 2.5x pixck_pdck_cci due to the DCMI. The default clock is 200 MHz, which allows the input clock to be at 80 MHz.
  - *Section 20: Reset and clock control (RCC)* provides the needed details.
- DMA
  - The GPDMA and HPDMA retrieve pixels from the DCMI, that signals to the DMA that its FIFO is full and must be read-out.
  - *Section 18* and *Section 19* provide the needed details.

37.5.2 High-resolution parallel camera and DCMIPP

This is the default use case to grab a picture from a parallel camera (up to the target 5 Mpixels @30fps). Any RGB or RawBayer picture can be grabbed.

The steps to setup this path are the following:

- Clock and reset:
  - pixck_pdck_cci is provided by the sensor (via the PCB on the input I/O, 120 MHz max). Each pixel needs 1 to 3 cycles to get in.
  - ck_icn_p_dcmipp is provided by the RCC for the DCMIPP configuration (default 200 MHz).
  - ck_ker_csi2 is provided by the RCC for the pixel pipelines of the DCMIPP (default 200 MHz).
  - ck_icn_m_dcmipp is provided by the RCC for the DCMIPP AXI (default 400 MHz).

  - *Section 20: Reset and clock control (RCC)* provides the needed details.
- DCMIPP configuration:
  - Select the parallel interface as input interface.
  - Setup the DCMIPP (parallel interface, ISP, application path, AXI-plug) as described in *Section 39: Digital camera interface pixel pipeline (DCMIPP)*.

37.5.3 High-resolution CSI2 camera and DCMIPP

This is the default use-case to grab a picture from a CSI2 camera (up to the target 5 Mpixels @30 fps). Any RGB or RawBayer picture can be grabbed.

The steps to setup this path are the following:

- Clock and reset:
  - cn/p_rx_dphy are provided by the sensor (via the PCB on the input I/O, 1.25 GHz/2.5 Gbit/s max).
  - RxByteClkHS is automatically divided (1/8) from cn/p_rx_dphy (312.5 MHz max).
  - ck_icn_p_csi2 is provided by the RCC for the CSI2-Host configuration (default 200 MHz).
  - ck_icn_p_dcmipp is provided by the RCC for the DCMIPP configuration (default 200 MHz).
  - ck_ker_csiphy is provided by the RCC for CSI2-Host in low-power (default 20 MHz).
  - ck_ker_csi2_txesc is provided by the RCC for the TxEscape mode of CSI2 (default 20 MHz).
  - ck_ker_csi2 is provided by the RCC for the pixel pipelines of the DCMIPP (default 200 MHz).
  - ck_icn_m_dcmipp is provided by the RCC for the DCMIPP AXI (default 400 MHz).
  - *Section 20: Reset and clock control (RCC)* provides the needed details.
- CSI2-Host configuration:
  - Setup the CSI2-Host (such as packet type).
- DCMIPP configuration:
  - Select the CSI2 interface as input interface.
  - Setup the DCMIPP (CSI2, ISP, application path, AXI-plug) as described in *Section 39: Digital camera interface pixel pipeline (DCMIPP)*.

37.5.4 Sensors over target resolution

The sensors over the target resolution (5 Mpixels) are supported with the following restrictions:

- CSI2 interface must be used (parallel interface quickly limited with its 120 MHz performances).
- The sensor must be able to output its bit rate onto the available two data lanes of the CSI2 link. It is recommended to use RawBayer (less bit/pixel than RGB).
- Width and height must both remain below 4094 (up to 16 Mpixels can be reached).
- Decimation is located ahead of most processing, and can be used to decrease the grabbed resolution.
- Demosaicing is limited with a max width of 2688: if demosaicing needs to be activated (RawBayer input, RGB output), then the decimation must be used to fall under that 2688 pixel limit. There is no height limit.
- Downsize and YUV420 are limited (similarly as demosaicing) to 2688, and require a decimation ahead if they need to be activated. The horizontal downsize occurs before the vertical one: the first can be used to fall below the 2688 pixels limit, before being processed by the second.
- If all demosaicing, downsize, and YUV420 are disabled. Pictures up to 4094x4094 can be received and dumped by the AXI.

After taking into account these restrictions, the configuration itself is the same as for the default high-resolution CSI2 camera, described in previous section.

37.5.5 Sensors over target pixel rate

The sensors over the target pixel rate (150 Mpixel/s average, ~200 Mpixel/s peak) are supported with the following restrictions:

- The sensor must be able to output its bit rate onto the available two data lanes of the CSI2 link.
- The peak pixel rate must remain below (with some margin) the max 333 MHz of the DCMIPP pixel clock (with a 10% margin recommended).
- DCMIPP bandwidth and FIFO allocation must be able to dump this load: it is recommended to reduce the amount of parallel dumps (best not all Pipe0, Pipe1, and Pipe2 active simultaneously). In the DCMIPP plug, more FIFO and outstanding allocation can be made in favor of the working pipe.

After taking into account these restrictions, the configuration itself is the same as for the default high-resolution CSI2 camera, described in *Section 37.5.3*.

37.5.6 Double sensors

Two sensors are simultaneously and independently supported with the following restriction:

- One sensor must be low-resolution, so that it can use the DCMI (the other sensor uses the DCMIPP).
- Two input interfaces must be available: one sensor is CSI2, and the other parallel.

The configuration of the double sensors is the same as for the default low-resolution and default high-resolution sensors, described in previous sections.

37.6 Camera subsystem interrupts

Sections for each peripheral describe in details the provided interrupts.

37.7 Camera subsystem registers

All the registers are described in sections for each peripheral.

38 Digital camera interface (DCMI)

38.1 DCMI introduction

The digital camera is a synchronous parallel interface able to receive a high-speed data flow from an external 8-, 10-, 12- or 14-bit CMOS camera module. It supports different data formats: YCbCr4:2:2/RGB565 progressive video and compressed data (JPEG).

38.2 DCMI main features

- 8-, 10-, 12- or 14-bit parallel interface
- Embedded/external line and frame synchronization
- Continuous or snapshot mode
- Crop feature
- Supports the following data formats:
  - 8/10/12/14-bit progressive video: either monochrome or raw Bayer
  - YCbCr 4:2:2 progressive video
  - RGB 565 progressive video
  - Compressed data: JPEG

38.3 DCMI functional description

The digital camera interface is a synchronous parallel interface that can receive high-speed data flows. It consists of up to 14 data lines (DCMI_D[13:0]) and a pixel clock line (DCMI_PIXCLK). The pixel clock has a programmable polarity, so that data can be captured on either the rising or the falling edge of the pixel clock.

The data are packed into a 32-bit data register (DCMI_DR) and then transferred through a general-purpose DMA channel. The image buffer is managed by the DMA, not by the camera interface.

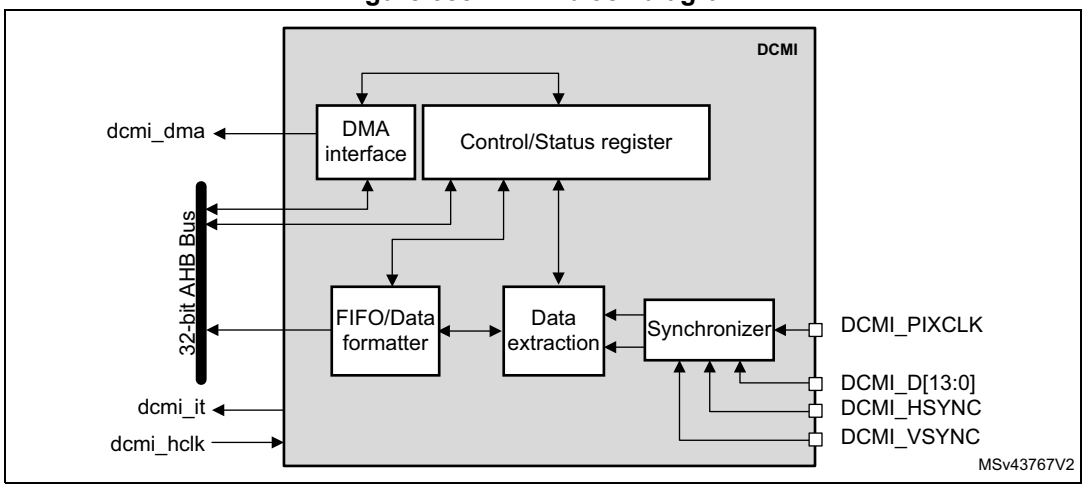
The data received from the camera can be organized in lines/frames (raw YUB/RGB/Bayer modes) or can be a sequence of JPEG images. To enable JPEG image reception, the JPEG bit (bit 3 of DCMI_CR register) must be set.

The data flow is synchronized either by hardware using the optional DCMI_HSYNC (horizontal synchronization) and DCMI_VSYNC (vertical synchronization) signals or by synchronization codes embedded in the data flow.

38.3.1 DCMI block diagram

Figure 388 shows the DCMI block diagram.

Figure 388. DCMI block diagram



38.3.2 DCMI pins and internal signals

The following table shows DCMI pins.

Table 316. DCMI input/output pins

Mode	Pin name	Signal type	Description
8 bits 10 bits 12 bits 14 bits	DCMI_D[7:0] DCMI_D[9:0] DCMI_D[11:0] DCMI_D[13:0]	Inputs	DCMI data
DCMI_PIXCLK		Input	Pixel clock
DCMI_HSYNC		Input	Horizontal synchronization / Data valid
DCMI_VSYNC		Input	Vertical synchronization

The following table shows DCMI internal signals.

Table 317. DCMI internal input/output signals

Internal signal name	Signal type	Description
dcmi_dma	Output	DCMI DMA request
dcmi_it	Output	DCMI interrupt request
dcmi_hclk	Input	DCMI interface clock

38.3.3 DCMI clocks

The digital camera interface uses two clock domains, DCMI_PIXCLK and HCLK. The signals generated with DCMI_PIXCLK are sampled on the rising edge of HCLK once they are stable. An enable signal is generated in the HCLK domain, to indicate that data coming from the camera are stable and can be sampled. The maximum DCMI_PIXCLK period must be higher than 2.5 HCLK periods.

38.3.4 DCMI DMA interface

The DMA interface is active when the CAPTURE bit of the DCMI_CR register is set. A DMA request is generated each time the camera interface receives a complete 32-bit data block in its register.

38.3.5 DCMI physical interface

The interface is composed of 11/13/15/17 inputs. Only the Slave mode is supported.

The camera interface can capture 8-bit, 10-bit, 12-bit or 14-bit data depending on the EDM[1:0] bits of the DCMI_CR register. If less than 14 bits are used, the unused input pins must be connected to ground.

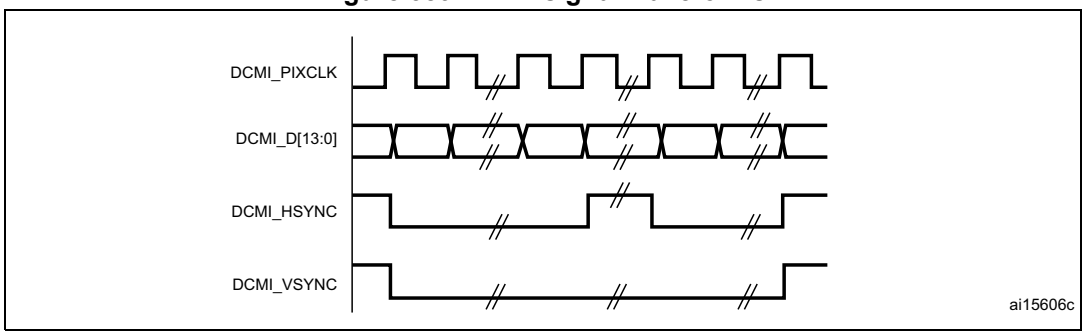
DCMI pins are shown in *Table 316*.

The data are synchronous with DCMI_PIXCLK and change on the rising/falling edge of the pixel clock depending on the polarity.

The DCMI_HSYNC signal indicates the start/end of a line.

The DCMI_VSYNC signal indicates the start/end of a frame

Figure 389. DCMI signal waveforms



1. The capture edge of DCMI_PIXCLK is the falling edge, the active state of DCMI_HSYNC and DCMI_VSYNC is 1.
2. DCMI_HSYNC and DCMI_VSYNC can change states at the same time.

8-bit data

When EDM[1:0] = 00 in DCMI_CR the interface captures 8 LSBs at its input (DCMI_D[7:0]) and stores them as 8-bit data. The DCMI_D[13:8] inputs are ignored. In this case, to capture a 32-bit word, the camera interface takes four pixel clock cycles.

The first captured data byte is placed in the LSB position in the 32-bit word and the 4th captured data byte is placed in the MSB position in the 32-bit word. The table below gives an example of the positioning of captured data bytes in two 32-bit words.

Table 318. Positioning of captured data bytes in 32-bit words (8-bit width)

Byte address	31:24	23:16	15:8	7:0
0	D_{n+3}[7:0]	D_{n+2}[7:0]	D_{n+1}[7:0]	D_n[7:0]
4	D_{n+7}[7:0]	D_{n+6}[7:0]	D_{n+5}[7:0]	D_{n+4}[7:0]

10-bit data

When EDM[1:0] = 01 in DCMI_CR, the camera interface captures 10-bit data at its input DCMI_D[9:0] and stores them as the 10 least significant bits of a 16-bit word. The remaining most significant bits of the DCMI_DR register (bits 11 to 15) are cleared to zero. So, in this case, a 32-bit data word is made up every two pixel clock cycles.

The first captured data are placed in the LSB position in the 32-bit word and the 2nd captured data are placed in the MSB position in the 32-bit word as shown in the table below.

Table 319. Positioning of captured data bytes in 32-bit words (10-bit width)

Byte address	31:26	25:16	15:10	9:0
0	0	D_{n+1}[9:0]	0	D_n[9:0]
4	0	D_{n+3}[9:0]	0	D_{n+2}[9:0]

12-bit data

When EDM[1:0] = 10 in DCMI_CR, the camera interface captures the 12-bit data at its input DCMI_D[11:0] and stores them as the 12 least significant bits of a 16-bit word. The remaining most significant bits are cleared to zero. So, in this case a 32-bit data word is made up every two pixel clock cycles.

The first captured data are placed in the LSB position in the 32-bit word and the 2nd captured data are placed in the MSB position in the 32-bit word as shown in the table below.

Table 320. Positioning of captured data bytes in 32-bit words (12-bit width)

Byte address	31:28	27:16	15:12	11:0
0	0	D_{n+1}[11:0]	0	D_n[11:0]
4	0	D_{n+3}[11:0]	0	D_{n+2}[11:0]

14-bit data

When EDM[1:0] = 11 in DCMI_CR, the camera interface captures the 14-bit data at its input DCMI_D[13:0] and stores them as the 14 least significant bits of a 16-bit word. The remaining most significant bits are cleared to zero. So, in this case a 32-bit data word is made up every two pixel clock cycles.

The first captured data are placed in the LSB position in the 32-bit word and the 2nd captured data are placed in the MSB position in the 32-bit word as shown in the table below.

Table 321. Positioning of captured data bytes in 32-bit words (14-bit width)

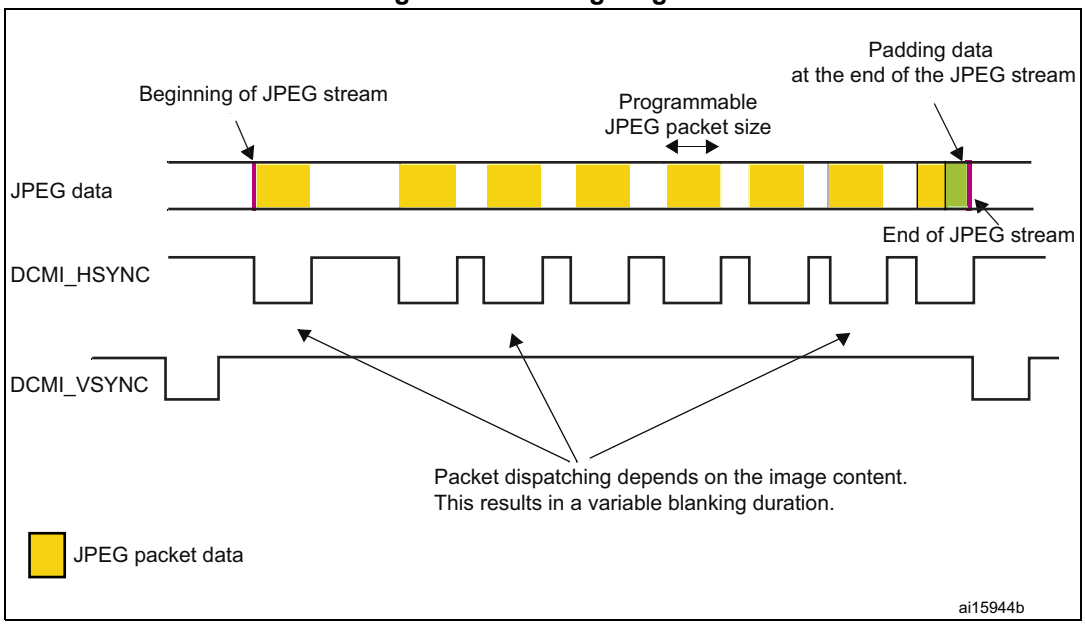
Byte address	31:30	29:16	15:14	13:0
0	0	D_{n+1}[13:0]	0	D_n[13:0]
4	0	D_{n+3}[13:0]	0	D_{n+2}[13:0]

38.3.6 DCMI synchronization

The digital camera interface supports embedded or hardware (DCMI_HSYNC and DCMI_VSYNC) synchronization. When embedded synchronization is used, it is up to the digital camera module to make sure that the 0x00 and 0xFF values are used ONLY for synchronization (not in data). Embedded synchronization codes are supported only for the 8-bit parallel data interface width (that is, in the DCMI_CR register, the EDM[1:0] bits must be cleared).

For compressed data, the DCMI supports only the hardware synchronization mode. In this case, DCMI_VSYNC is used as a start/end of the image, and DCMI_HSYNC is used as a Data Valid signal. *Figure 390* shows the corresponding timing diagram.

Figure 390. Timing diagram



Hardware synchronization mode

In hardware synchronization mode, the two synchronization signals (DCMI_HSYNC/DCMI_VSYNC) are used.

Depending on the camera module/mode, data may be transmitted during horizontal/vertical synchronization periods. The DCMI_HSYNC/DCMI_VSYNC signals act like blanking signals since all the data received during DCMI_HSYNC/DCMI_VSYNC active periods are ignored.

In order to correctly transfer images into the DMA/RAM buffer, data transfer is synchronized with the DCMI_VSYNC signal. When the hardware synchronization mode is selected, and

capture is enabled (CAPTURE bit set in DCMI_CR), data transfer is synchronized with the deactivation of the DCMI_VSYNC signal (next start of frame).

Transfer can then be continuous, with successive frames transferred by DMA to successive buffers or the same/circular buffer. To allow the DMA management of successive frames, a VSIF (Vertical synchronization interrupt flag) is activated at the end of each frame.

Embedded data synchronization mode

In this synchronization mode, the data flow is synchronized using 32-bit codes embedded in the data flow. These codes use the 0x00/0xFF values that are *not* used in data anymore. There are 4 types of codes, all with a 0xFF0000XY format. The embedded synchronization codes are supported only in 8-bit parallel data width capture (in the DCMI_CR register, the EDM[1:0] bits must be cleared). For other data widths, this mode generates unpredictable results and must not be used.

Note: Camera modules can have 8 such codes (in interleaved mode). For this reason, the interleaved mode is not supported by the camera interface (otherwise, every other half-frame would be discarded).

- Mode 2

  Four embedded codes signal the following events
  - Frame start (FS)
  - Frame end (FE)
  - Line start (LS)
  - Line end (LE)

  The XY values in the 0xFF0000XY format of the four codes are programmable (see *Section 38.5.7: DCMI embedded synchronization code register (DCMI_ESCR)*).

  A 0xFF value programmed as a “frame end” means that all the unused codes are interpreted as valid frame end codes.

  In this mode, once the camera interface has been enabled, the frame capture starts after the first occurrence of the frame end (FE) code followed by a frame start (FS) code.
- Mode 1

  An alternative coding is the camera mode 1. This mode is ITU656 compatible.

  The codes signal another set of events:
  - SAV (active line) - line start
  - EAV (active line) - line end
  - SAV (blanking) - end of line during interframe blanking period
  - EAV (blanking) - end of line during interframe blanking period

This mode can be supported by programming the following codes:

- FS $\leq$ 0xFF
- FE ≤ 0xFF
- LS ≤ SAV (active)
- LE ≤ EAV (active)

An embedded unmask code is also implemented for frame/line start and frame/line end codes. Using it, it is possible to compare only the selected unmasked bits with the programmed code. A bit can therefore be selected to compare in the embedded code and

detect a frame/line start or frame/line end. This means that there can be different codes for the frame/line start and frame/line end with the unmasked bit position remaining the same.

Example

FS = 0xA5

Unmask code for FS = 0x10

In this case the frame start code is embedded in the bit 4 of the frame start code.

38.3.7 DCMI capture modes

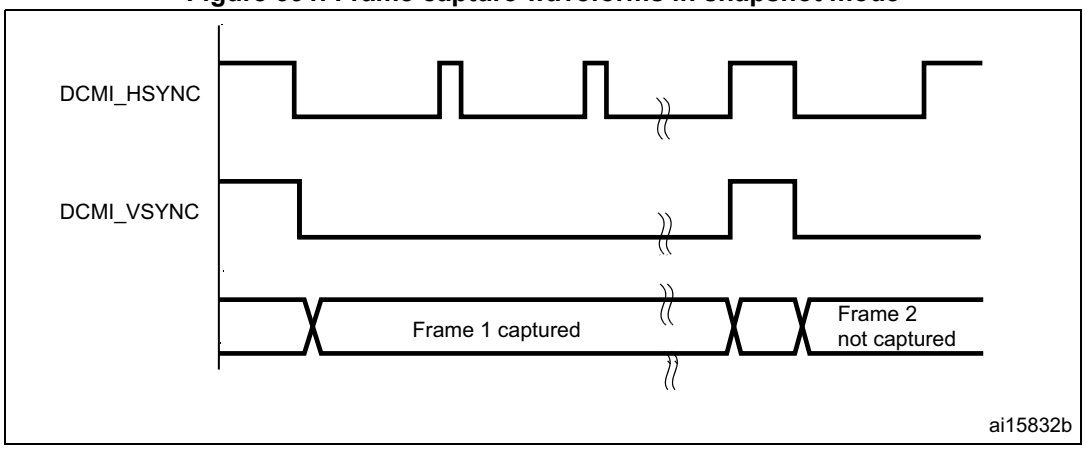
This interface supports two types of capture: snapshot (single frame) and continuous grab.

Snapshot mode (single frame)

In this mode, a single frame is captured (CM = 1 of the DCMI_CR register). After the CAPTURE bit is set in DCMI_CR, the interface waits for the detection of a start of frame before sampling the data. The camera interface is automatically disabled (CAPTURE bit cleared in DCMI_CR) after receiving the first complete frame. An interrupt is generated (IT_FRAME) if it is enabled.

In case of an overrun, the frame is lost and the CAPTURE bit is cleared.

Figure 391. Frame capture waveforms in snapshot mode

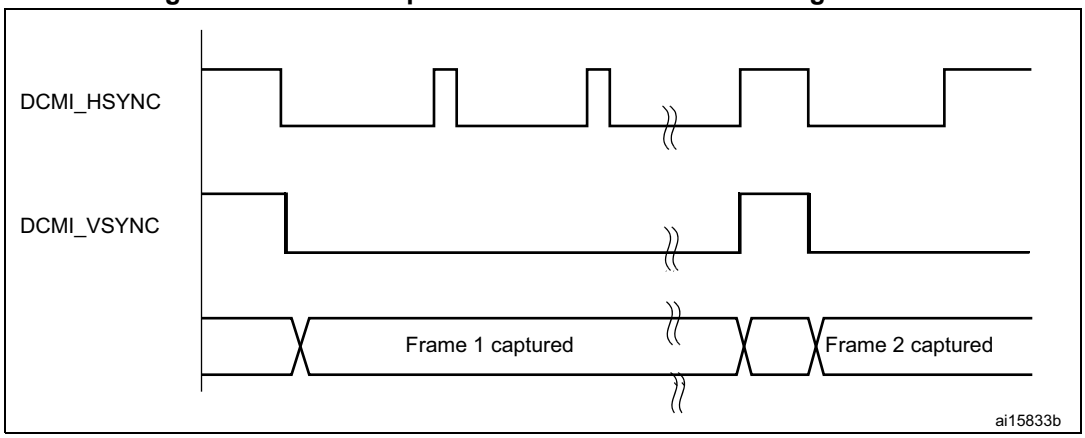


1. Here, the active state of DCMI_HSYNC and DCMI_VSYNC is 1.
2. DCMI_HSYNC and DCMI_VSYNC can change states at the same time.

Continuous grab mode

In this mode (CM bit = 0 in DCMI_CR), once the CAPTURE bit has been set in DCMI_CR, the grabbing process starts on the next DCMI_VSYNC or embedded frame start depending on the mode. The process continues until the CAPTURE bit is cleared in DCMI_CR. Once the CAPTURE bit has been cleared, the grabbing process continues until the end of the current frame.

Figure 392. Frame capture waveforms in continuous grab mode



1. Here, the active state of DCMI_HSYNC and DCMI_VSYNC is 1.
2. DCMI_HSYNC and DCMI_VSYNC can change states at the same time.

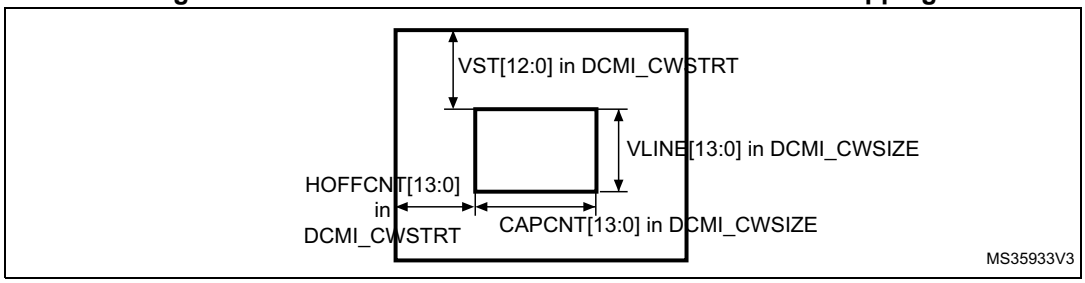
In continuous grab mode, the FCRC[1:0] bits in DCMI_CR can be configured to grab all pictures, every second picture or one out of four pictures to decrease the frame capture rate.

Note: In the hardware synchronization mode (ESS = 0 in DCMI_CR), the IT_VSYNC interrupt is generated (if enabled) even when CAPTURE = 0 in DCMI_CR so, to reduce the frame capture rate even further, the IT_VSYNC interrupt can be used to count the number of frames between 2 captures in conjunction with the Snapshot mode. This is not allowed by embedded data synchronization mode.

38.3.8 DCMI crop feature

With the crop feature, the camera interface can select a rectangular window from the received image. The start (upper left corner) coordinates and size (horizontal dimension in number of pixel clocks and vertical dimension in number of lines) are specified using two 32-bit registers (DCMI_CWSTRT and DCMI_CWSIZE). The size of the window is specified in number of pixel clocks (horizontal dimension) and in number of lines (vertical dimension).

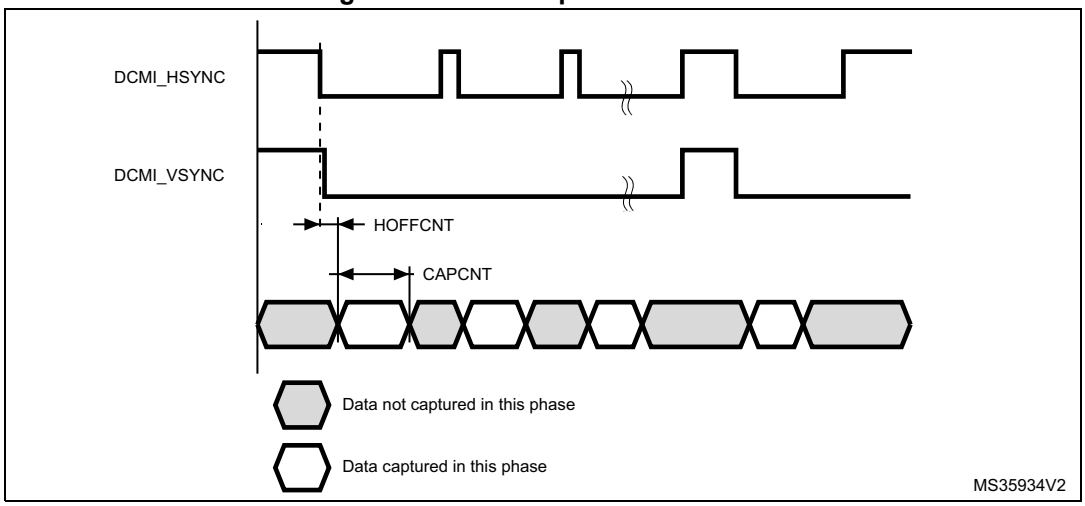
Figure 393. Coordinates and size of the window after cropping



These registers specify the coordinates of the starting point of the capture window as a line number (in the frame, starting from 0) and a number of pixel clocks (on the line, starting from 0), and the size of the window as a line number and a number of pixel clocks. The CAPCNT value can only be a multiple of 4 (two least significant bits are forced to 0) to allow the correct transfer of data through the DMA.

If the DCMI_VSYNC signal goes active before the number of lines is specified in the DCMI_CWSIZE register, then the capture stops and an IT_FRAME interrupt is generated when enabled.

Figure 394. Data capture waveforms



1. Here, the active state of DCMI_HSYNC and DCMI_VSYNC is 1.
2. DCMI_HSYNC and DCMI_VSYNC can change states at the same time.

38.3.9 DCMI JPEG format

To allow JPEG image reception, it is necessary to set the JPEG bit of the DCMI_CR register. JPEG images are not stored as lines and frames, so the DCMI_VSYNC signal is used to start the capture while DCMI_HSYNC serves as a data enable signal. The number of bytes in a line may not be a multiple of 4. This case must be carefully handled since a DMA request is generated each time a complete 32-bit word has been constructed from the captured data. When an end of frame is detected and the 32-bit word to be transferred has not been completely received, the remaining data are padded with zeros and a DMA request is generated.

The crop feature and embedded synchronization codes cannot be used in JPEG format.

38.3.10 DCMI FIFO

A 8-word FIFO is implemented to manage data rate transfers on the AHB. The DCMI features a simple FIFO controller with a read pointer incremented each time the camera interface reads from the AHB, and a write pointer incremented each time the camera interface writes to the FIFO. There is no overrun protection to prevent the data from being overwritten if the AHB interface does not sustain the data transfer rate.

In case of overrun or errors in the synchronization signals, the FIFO is reset and the DCMI interface waits for a new start of frame.

38.3.11 DCMI data format description

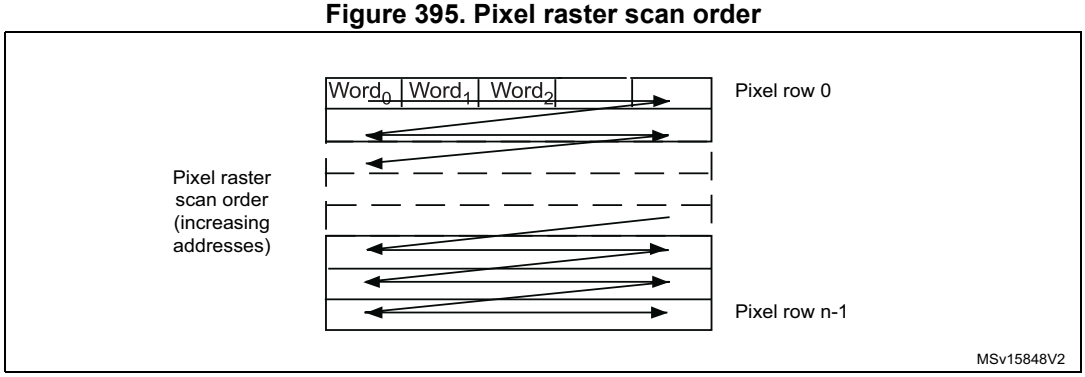
Data formats

The following types of data are supported:

- 8/10/12/14-bit progressive video: either monochrome or raw Bayer format
- YCbCr 4:2:2 progressive video
- RGB565 progressive video. A pixel coded in 16 bits (5 bits for blue, 5 bits for red, 6 bits for green) takes two clock cycles to be transferred
- compressed data: JPEG

For monochrome, RGB and YCbCr, the frame buffer is stored in raster mode. 32-bit words are used. Only the little-endian format is supported.

Figure 395. Pixel raster scan order



Monochrome format

Characteristics:

- Raster format
- 8 bits per pixel

The table below shows how the data are stored.

Table 322. Data storage in monochrome progressive video format

Byte address	31:24	23:16	15:8	7:0
0	n + 3	n + 2	n + 1	n
4	n + 7	n + 6	n + 5	n + 4

RGB format

Characteristics:

- Raster format
- RGB
- Interleaved: one buffer: R, G and B interleaved (such as BRGBRGBRG)
- Optimized for display output

The RGB planar format is compatible with standard OS frame buffer display formats.

Only 16 BPP (bits per pixel): RGB565 (2 pixels per 32-bit word) is supported.

The 24 BPP (palletized format) and gray-scale formats are not supported. Pixels are stored in a raster scan order, that is from top to bottom for pixel rows, and from left to right within a pixel row. Pixel components are R (red), G (green) and B (blue). All components have the same spatial resolution (4:4:4 format). A frame is stored in a single part, with the components interleaved on a pixel basis.

The table below shows how the data are stored.

Table 323. Data storage in RGB progressive video format

Byte address	31:27	26:21	20:16	15:11	10:5	4:0
0	Red n + 1	Green n + 1	Blue n + 1	Red n	Green n	Blue n
4	Red n + 4	Green n + 3	Blue n + 3	Red n + 2	Green n + 2	Blue n + 2

YCbCr format

Characteristics:

- Raster format
- YCbCr 4:2:2
- Interleaved: one buffer: Y, Cb and Cr interleaved (such as CbYCrYCbYCr)

Pixel components are Y (luminance or “luma”), Cb and Cr (chrominance or “chroma” blue and red). Each component is encoded in 8 bits. Luma and chroma are stored together (interleaved) as shown in the table below.

Table 324. Data storage in YCbCr progressive video format

Byte address	31:24	23:16	15:8	7:0
0	Y n + 1	Cr n	Y n	Cb n
4	Y n + 3	Cr n + 2	Y n + 2	Cb n + 2

YCbCr format - Y only

Characteristics:

- Raster format
- YCbCr 4:2:2
- The buffer only contains Y information - monochrome image

Pixel components are Y (luminance or “luma”), Cb and Cr (chrominance or “chroma” blue and red). In this mode, the chroma information is dropped. Only the luma component of each pixel, encoded in 8 bits, is stored as shown in *Table 325*.

The result is a monochrome image having the same resolution as the original YCbCr data.

Table 325. Data storage in YCbCr progressive video format - Y extraction mode

Byte address	31:24	23:16	15:8	7:0
0	Y n + 3	Y n + 2	Y n + 1	Y n
4	Y n + 7	Y n + 6	Y n + 5	Y n + 4

Half resolution image extraction

This is a modification of the previous reception modes, being applicable to monochrome, RGB or Y extraction modes.

This mode is used to only store a half resolution image. It is selected through OELS and LSM control bits.

38.4 DCMI interrupts

Five interrupts are generated. All interrupts are maskable by software. The global interrupt (dcmi_it) is the OR of all the individual interrupts. The table below gives the list of all interrupts.

Table 326. DCMI interrupts

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exits Sleep mode	Exists Stop and Standby modes
dcmi_it	End of line	LINE_RIS	LINE_IE	Set LINE_ISC	Yes	No
	End of frame capture	FRAME_RIS	FRAME_IE	Set FRAME_ISC	Yes	No
	Overrun of data reception	OVR_RIS	OVR_IE	Set OVR_ISC	Yes	No
	Synchronization frame	VSYNC_RIS	VSYNC_IE	Set VSYNC_ISC	Yes	No
	Detection of an error in the embedded synchronization frame detection	ERR_RIS	ERR_IE	Set ERR_ISC	Yes	No

38.5 DCMI registers

Refer to *Section 1.2 on page 156* for list of abbreviations used in register descriptions. All DCMI registers must be accessed as 32-bit words, otherwise a bus error occurs.

38.5.1 DCMI control register (DCMI_CR)

Address offset: 0x00

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	ENABLE	Res.	Res.	EDM[1:0]		FCRC[1:0]		VSPOL	HSPOL	PCKPOL	ESS	JPEG	CROP	CM	CAPTURE
	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **OELS:** Odd/Even Line Select (Line Select Start)
This bit works in conjunction with the LSM field (LSM = 1).
0: Interface captures first line after the frame start, second one being dropped.
1: Interface captures second line from the frame start, first one being dropped.

Bit 19 **LSM:** Line Select mode
0: Interface captures all received lines.
1: Interface captures one line out of two.

Bit 18 **OEBS:** Odd/Even Byte Select (Byte Select Start)
This bit works in conjunction with BSM field (BSM ≠ 00).
0: Interface captures first data (byte or double byte) from the frame/line start, second one being dropped.
1: Interface captures second data (byte or double byte) from the frame/line start, first one being dropped.

Bits 17:16 **BSM[1:0]:** Byte Select mode
00: Interface captures all received data.
01: Interface captures every other byte from the received data.
10: Interface captures one byte out of four.
11: Interface captures two bytes out of four.
Note: This mode only works for EDM[1:0] = 00. For all other EDM values, this field must be programmed to the reset value.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **ENABLE:** DCMI enable
0: DCMI disabled
1: DCMI enabled
Note: The DCMI configuration registers must be programmed correctly before enabling this bit.

Bits 13:12 Reserved, must be kept at reset value.

Bits 11:10 **EDM[1:0]:** Extended data mode
00: Interface captures 8-bit data on every pixel clock.
01: Interface captures 10-bit data on every pixel clock.
10: Interface captures 12-bit data on every pixel clock.
11: Interface captures 14-bit data on every pixel clock.

Bits 9:8 **FCRC[1:0]:** Frame capture rate control
These bits define the frequency of frame capture. They are meaningful only in Continuous grab mode. They are ignored in snapshot mode.
00: All frames are captured.
01: Every alternate frame captured (50% bandwidth reduction)
10: One frame out of four captured (75% bandwidth reduction)
11: reserved

Bit 7 **VSPOL:** Vertical synchronization polarity
This bit indicates the level on the DCMI_VSYNC pin when the data are not valid on the parallel interface.
0: DCMI_VSYNC active low
1: DCMI_VSYNC active high

Bit 6 **HSPOL:** Horizontal synchronization polarity

This bit indicates the level on the DCMI_HSYNC pin when the data are not valid on the parallel interface.

0: DCMI_HSYNC active low

1: DCMI_HSYNC active high

Bit 5 **PCKPOL:** Pixel clock polarity

This bit configures the capture edge of the pixel clock.

0: Falling edge active

1: Rising edge active

Bit 4 **ESS:** Embedded synchronization select

0: Hardware synchronization data capture (frame/line start/stop) is synchronized with the DCMI_HSYNC/DCMI_VSYNC signals.

1: Embedded synchronization data capture is synchronized with synchronization codes embedded in the data flow.

Note: Valid only for 8-bit parallel data. HSPOL/VSPOL are ignored when the ESS bit is set.

This bit is disabled in JPEG mode.

Bit 3 **JPEG:** JPEG format

0: Uncompressed video format

1: This bit is used for JPEG data transfers. The DCMI_HSYNC signal is used as data enable. The crop and embedded synchronization features (ESS bit) cannot be used in this mode.

Bit 2 **CROP:** Crop feature

0: The full image is captured. In this case the total number of bytes in an image frame must be a multiple of four.

1: Only the data inside the window specified by the crop register is captured. If the size of the crop window exceeds the picture size, then only the picture size is captured.

Bit 1 **CM:** Capture mode

0: Continuous grab mode - The received data are transferred into the destination memory through the DMA. The buffer location and mode (linear or circular buffer) is controlled through the system DMA.

1: Snapshot mode (single frame) - Once activated, the interface waits for the start of frame and then transfers a single frame through the DMA. At the end of the frame, the CAPTURE bit is automatically reset.

Bit 0 **CAPTURE:** Capture enable

0: Capture disabled

1: Capture enabled

The camera interface waits for the first start of frame, then a DMA request is generated to transfer the received data into the destination memory.

In snapshot mode, the CAPTURE bit is automatically cleared at the end of the first frame received.

In continuous grab mode, if the software clears this bit while a capture is ongoing, the bit is effectively cleared after the frame end.

Note: The DMA controller and all DCMI configuration registers must be programmed correctly before enabling this bit.

38.5.2 DCMI status register (DCMI_SR)

Address offset: 0x04

Reset value: 0x0000 0000

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

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

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

Bit 2 **FNE**: FIFO not empty

This bit gives the status of the FIFO.
1: FIFO contains valid data.
0: FIFO empty

Bit 1 **VSYNC**: Vertical synchronization

This bit gives the state of the DCMI_VSYNC pin with the correct programmed polarity. When embedded synchronization codes are used, the meaning of this bit is the following:
0: active frame
1: synchronization between frames
In case of embedded synchronization, this bit is meaningful only if the CAPTURE bit in DCMI_CR is set.

Bit 0 **HSYNC**: Horizontal synchronization

This bit gives the state of the DCMI_HSYNC pin with the correct programmed polarity. When embedded synchronization codes are used, the meaning of this bit is the following:
0: active line
1: synchronization between lines
In case of embedded synchronization, this bit is meaningful only if the CAPTURE bit in DCMI_CR is set.

38.5.3 DCMI raw interrupt status register (DCMI_RIS)

DCMI_RIS gives the raw interrupt status and is accessible in read only. When read, this register returns the status of the corresponding interrupt before masking with the DCMI_IER register value.

Address offset: 0x08

Reset value: 0x0000 0000

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

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LINE_RIS	VSYNC_RIS	ERR_RIS	OVR_RIS	FRAME_RIS
											r	r	r	r	r

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

Bit 4 **LINE_RIS**: Line raw interrupt status

This bit gets set when the DCMI_HSYNC signal changes from the inactive state to the active state. It goes high even if the line is not valid.

In the case of embedded synchronization, this bit is set only if the CAPTURE bit in DCMI_CR is set.

It is cleared by setting the LINE_ISC bit of the DCMI_ICR register.

Bit 3 **VSYNC_RIS**: DCMI_VSYNC raw interrupt status

This bit is set when the DCMI_VSYNC signal changes from the inactive state to the active state.

In the case of embedded synchronization, this bit is set only if the CAPTURE bit is set in DCMI_CR.

It is cleared by setting the VSYNC_ISC bit of the DCMI_ICR register.

Bit 2 **ERR_RIS**: Synchronization error raw interrupt status

0: No synchronization error detected

1: Embedded synchronization characters are not received in the correct order.

This bit is valid only in the embedded synchronization mode. It is cleared by setting the ERR_ISC bit of the DCMI_ICR register.

Note: This bit is available only in embedded synchronization mode.

Bit 1 **OVR_RIS**: Overrun raw interrupt status

0: No data buffer overrun occurred

1: A data buffer overrun occurred and the data FIFO is corrupted.

The bit is cleared by setting the OVR_ISC bit of the DCMI_ICR register.

Bit 0 **FRAME_RIS**: Capture complete raw interrupt status

0: No new capture

1: A frame has been captured.

This bit is set when a frame or window has been captured.

In case of a cropped window, this bit is set at the end of line of the last line in the crop. It is set even if the captured frame is empty (for example window cropped outside the frame).

The bit is cleared by setting the FRAME_ISC bit of the DCMI_ICR register.

38.5.4 DCMI interrupt enable register (DCMI_IER)

The DCMI_IER register is used to enable interrupts. When one of the DCMI_IER bits is set, the corresponding interrupt is enabled. This register is accessible in both read and write.

Address offset: 0x0C

Reset value: 0x0000 0000

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

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

Bit 4 **LINE_IE**: Line interrupt enable
0: No interrupt generation when the line is received
1: An Interrupt is generated when a line has been completely received.

Bit 3 **VSYNC_IE**: DCMI_VSYNC interrupt enable
0: No interrupt generation
1: An interrupt is generated on each DCMI_VSYNC transition from the inactive to the active state.
The active state of the DCMI_VSYNC signal is defined by the VSPOL bit.

Bit 2 **ERR_IE**: Synchronization error interrupt enable
0: No interrupt generation
1: An interrupt is generated if the embedded synchronization codes are not received in the correct order.
Note: This bit is available only in embedded synchronization mode.

Bit 1 **OVR_IE**: Overrun interrupt enable
0: No interrupt generation
1: An interrupt is generated if the DMA was not able to transfer the last data before new data (32-bit) are received.

Bit 0 **FRAME_IE**: Capture complete interrupt enable
0: No interrupt generation
1: An interrupt is generated at the end of each received frame/crop window (in crop mode).

38.5.5 DCMI masked interrupt status register (DCMI_MIS)

This DCMI_MIS register is a read-only register. When read, it returns the current masked status value (depending on the value in DCMI_IER) of the corresponding interrupt. A bit in this register is set if the corresponding enable bit in DCMI_IER is set and the corresponding bit in DCMI_RIS is set.

Address offset: 0x10

Reset value: 0x0000 0000

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

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

Bit 4 **LINE_MIS**: Line masked interrupt status
This bit gives the status of the masked line interrupt.
0: No interrupt generation when the line is received
1: An Interrupt is generated when a line has been completely received and the LINE_IE bit is set in DCMI_IER.

Bit 3 **VSYNC_MIS**: VSYNC masked interrupt status

This bit gives the status of the masked VSYNC interrupt.

0: No interrupt is generated on DCMI_VSYNC transitions.

1: An interrupt is generated on each DCMI_VSYNC transition from the inactive to the active state and the VSYNC_IE bit is set in DCMI_IER.

The active state of the DCMI_VSYNC signal is defined by the VSPOL bit.

Bit 2 **ERR_MIS**: Synchronization error masked interrupt status

This bit gives the status of the masked synchronization error interrupt.

0: No interrupt is generated on a synchronization error.

1: An interrupt is generated if the embedded synchronization codes are not received in the correct order and the ERR_IE bit in DCMI_IER is set.

Note: This bit is available only in embedded synchronization mode.

Bit 1 **OVR_MIS**: Overrun masked interrupt status

This bit gives the status of the masked overflow interrupt.

0: No interrupt is generated on overrun.

1: An interrupt is generated if the DMA was not able to transfer the last data before new data (32-bit) are received and the OVR_IE bit is set in DCMI_IER.

Bit 0 **FRAME_MIS**: Capture complete masked interrupt status

This bit gives the status of the masked capture complete interrupt

0: No interrupt is generated after a complete capture.

1: An interrupt is generated at the end of each received frame/crop window (in crop mode) and the FRAME_IE bit is set in DCMI_IER.

38.5.6 DCMI interrupt clear register (DCMI_ICR)

The DCMI_ICR register is write-only. Setting a bit of this register clears the corresponding flag in the DCMI_RIS and DCMI_MIS registers. Writing 0 has no effect.

Address offset: 0x14

Reset value: 0x0000 0000

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

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

Bit 4 **LINE_ISC**: line interrupt status clear

Setting this bit clears the LINE_RIS flag in the DCMI_RIS register.

Bit 3 **VSYNC_ISC**: Vertical Synchronization interrupt status clear

Setting this bit clears the VSYNC_RIS flag in the DCMI_RIS register.

Bit 2 **ERR_ISC**: Synchronization error interrupt status clear

Setting this bit clears the ERR_RIS flag in the DCMI_RIS register.

Note: This bit is available only in embedded synchronization mode.

Bit 1 **OVR_ISC**: Overrun interrupt status clear
Setting this bit clears the OVR_RIS flag in the DCMI_RIS register.

Bit 0 **FRAME_ISC**: Capture complete interrupt status clear
Setting this bit clears the FRAME_RIS flag in the DCMI_RIS register.

38.5.7 DCMI embedded synchronization code register (DCMI_ESCR)

Address offset: 0x18

Reset value: 0x0000 0000

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

Bits 31:24 **FEC[7:0]**: Frame end delimiter code
This byte specifies the code of the frame end delimiter. The code consists of 4 bytes in the form of 0xFF, 0x00, 0x00, FEC.
If FEC is programmed to 0xFF, all the unused codes (0xFF0000XY) are interpreted as frame end delimiters.

Bits 23:16 **LEC[7:0]**: Line end delimiter code
This byte specifies the code of the line end delimiter. The code consists of 4 bytes in the form of 0xFF, 0x00, 0x00, LEC.

Bits 15:8 **LSC[7:0]**: Line start delimiter code
This byte specifies the code of the line start delimiter. The code consists of 4 bytes in the form of 0xFF, 0x00, 0x00, LSC.

Bits 7:0 **FSC[7:0]**: Frame start delimiter code
This byte specifies the code of the frame start delimiter. The code consists of 4 bytes in the form of 0xFF, 0x00, 0x00, FSC.
If FSC is programmed to 0xFF, no frame start delimiter is detected. But, the first occurrence of LSC after an FEC code is interpreted as a start of frame delimiter.

38.5.8 DCMI embedded synchronization unmask register (DCMI_ESUR)

Address offset: 0x1C

Reset value: 0x0000 0000

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

Bits 31:24 **FEU[7:0]**: Frame end delimiter unmask

This byte specifies the mask to be applied to the code of the frame end delimiter.
0: The corresponding bit in the FEC byte in DCMI_ESCR is masked while comparing the frame end delimiter with the received data.
1: The corresponding bit in the FEC byte in DCMI_ESCR is compared while comparing the frame end delimiter with the received data.

Bits 23:16 **LEU[7:0]**: Line end delimiter unmask

This byte specifies the mask to be applied to the code of the line end delimiter.
0: The corresponding bit in the LEC byte in DCMI_ESCR is masked while comparing the line end delimiter with the received data.
1: The corresponding bit in the LEC byte in DCMI_ESCR is compared while comparing the line end delimiter with the received data.

Bits 15:8 **LSU[7:0]**: Line start delimiter unmask

This byte specifies the mask to be applied to the code of the line start delimiter.
0: The corresponding bit in the LSC byte in DCMI_ESCR is masked while comparing the line start delimiter with the received data.
1: The corresponding bit in the LSC byte in DCMI_ESCR is compared while comparing the line start delimiter with the received data.

Bits 7:0 **FSU[7:0]**: Frame start delimiter unmask

This byte specifies the mask to be applied to the code of the frame start delimiter.
0: The corresponding bit in the FSC byte in DCMI_ESCR is masked while comparing the frame start delimiter with the received data.
1: The corresponding bit in the FSC byte in DCMI_ESCR is compared while comparing the frame start delimiter with the received data.

38.5.9 DCMI crop window start (DCMI_CWSTRT)

Address offset: 0x20

Reset value: 0x0000 0000

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

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:16 **VST[12:0]**: Vertical start line count

The image capture starts with this line number. Previous line data are ignored.
0x0000: line 1
0x0001: line 2
0x0002: line 3
....

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

Bits 13:0 **HOFFCNT[13:0]**: Horizontal offset count

This value gives the number of pixel clocks to count before starting a capture.

38.5.10 DCMI crop window size (DCMI_CWSIZE)

Address offset: 0x24

Reset value: 0x0000 0000

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

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:16 **VLINE[13:0]**: Vertical line count

This value gives the number of lines to be captured from the starting point.
0x0000: 1 line
0x0001: 2 lines
0x0002: 3 lines
....

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

Bits 13:0 **CAPCNT[13:0]**: Capture count

This value gives the number of pixel clocks to be captured from the starting point on the same line. It value must corresponds to word-aligned data for different widths of parallel interfaces.
0x0000: 1 pixel
0x0001: 2 pixels
0x0002: 3 pixels
....

38.5.11 DCMI data register (DCMI_DR)

Address offset: 0x28

Reset value: 0x0000 0000

The digital camera Interface packages all the received data in 32-bit format before requesting a DMA transfer. A 8-word deep FIFO is available to leave enough time for DMA transfers and avoid DMA overrun conditions.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BYTE3[7:0]								BYTE2[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BYTE1[7:0]								BYTE0[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **BYTE3[7:0]**: Data byte 3

Bits 23:16 **BYTE2[7:0]**: Data byte 2

Bits 15:8 **BYTE1[7:0]**: Data byte 1

Bits 7:0 **BYTE0[7:0]**: Data byte 0

38.5.12 DCMI register map

Table 327. DCMI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	DCMI_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OELS	LSM	OEBS	BSM[1:0]		Res.	ENABLE	Res.	Res.	EDM[1:0]		FCRC[1:0]		VSPOL	HSPOL	PCKPOL	ESS	JPEG	CROP	CM	CAPTURE
	Reset value												0	0	0	0	0		0			0	0	0	0	0	0	0	0	0	0	0	0
0x04	DCMI_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FNE	VSYNC	HSYNC
	Reset value																														0	0	0
0x08	DCMI_RIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LINE_RIS	VSYNC_RIS	ERR_RIS	OVR_RIS	FRAME_RIS
	Reset value																												0	0	0	0	0
0x0C	DCMI_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LINE_IE	VSYNC_IE	ERR_IE	OVR_IE	FRAME_IE
	Reset value																												0	0	0	0	0
0x10	DCMI_MIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LINE_MIS	VSYNC_MIS	ERR_MIS	OVR_MIS	FRAME_MIS
	Reset value																												0	0	0	0	0
0x14	DCMI_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LINE_ISC	VSYNC_ISC	ERR_ISC	OVR_ISC	FRAME_ISC
	Reset value																												0	0	0	0	0
0x18	DCMI_ESCR	FEC[7:0]								LEC[7:0]								LSC[7:0]								FSC[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	DCMI_ESUR	FEU[7:0]								LEU[7:0]								LSU[7:0]								FSU[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	DCMI_CWSTRT	Res.	Res.	Res.	VST[12:0]													Res.	Res.	HOFFCNT[13:0]													
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	DCMI_CWSIZE	Res.	Res.	VLINE[13:0]														Res.	Res.	CAPCNT[13:0]													
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	DCMI_DR	BYTE3[7:0]								BYTE2[7:0]								BYTE1[7:0]								BYTE0[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

39 Digital camera interface pixel pipeline (DCMIPP)

39.1 DCMIPP introduction

The DCMIPP is the pixel pipeline section of a high-resolution camera subsystem: it gets pixels from a parallel or a CSI interface, and after some processing (such as decimation, cropping, downsize, color conversion, gamma correction, auto-exposure) dumps them to the memory.

DCMIPP supports multiple types of external sensors, among others:

- dumb sensors (without internal ISP), which output raw Bayer pixels
- smart sensors (with an internal ISP), which usually output RGB or YUV pixels
- smart sensors with internal compression, which output a bit-stream (such as JPEG, or H264)
- sensors with interlaced video, which output their odd and even fields sequentially

The DCMIPP input interface integrates a parallel interface (up to 16 bits at 120 MHz, with internal/external synchronization), and makes it possible to abut an external CSI-2 host (one to two data lanes, up to 2.5 Gbps/lane).

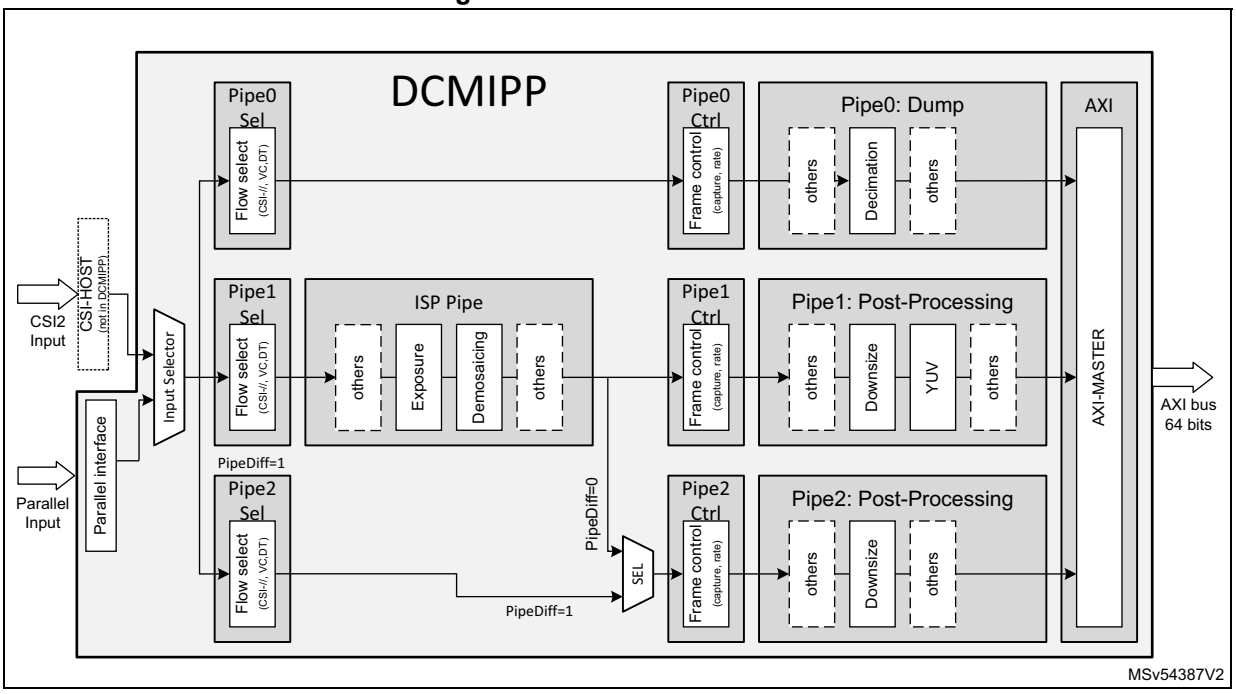
A first common part of the DCMIPP selects the input exclusively from the parallel or the CSI interfaces. Data go to dedicated pipeline(s) before they are sent to memory for further processing or display purposes.

Table 328. Available pipelines

ID	Function	Description
Pipe0	Dump pipe	Used to dump the received data as-is, without any processing (for example. without pixel extraction, no formatting), with an exception: it formats the CSI 10/12/14-bit raw Bayer components mapped unpacked into 16-bit memory words, to ease the job of a downstream application. 2D crop operation and basic decimation can be performed within this pipe.
Pipe1	Main pipe	Typically used for the main needs (such as video encode, software analysis, neural network), it allows to downsize the input, convert its color and dump to multiple pixel formats, including multi-planar ones. It can adapt the luminosity (gamma conversion) as well as auto-exposure and black level calibration based on its own embedded image processing functions, handling camera sensor without internal ISP. It can convert the raw Bayer to RGB (demosaicing). Pipe1 shares all the image processing functions with Pipe2.
Pipe2	Ancillary pipe	Typically used for ancillary needs (such as a permanent display), also allows to downsize the input and dump to several pixel formats, however restricted to coplanar ones (as specialized for display-like usage).

Figure 396 shows the DCMIPP main functions, namely the integrated parallel interface, the abutted CSI-2 host, the ISP functions in Pipe1 and shareable with Pipe2, the differentiated applicative post-processing pipelines with their crop, downsize, and pixel formatting.

Figure 396. DCMIPP overview



In the ISP processing, and therefore before the post-processing pipes (Pipe1 and Pipe2), the component dynamic is reduced to 8 bits per component. If the software intends to benefit of the wider dynamic, the input data must be dumped via Pipe0, which keeps the input pixel format (10/12/14 bpp) when storing them in memory.

Table 329. Glossary

Item	Definition
A	Alpha component, used to define transparency, opaque = 0xFF, for example ARGB.
3A (or AAA)	In ISP domain: acronym for AE-AF-AWB, algorithms automatically correcting pictures: – AE: automatic exposure control (adapts exposure or sensibility of sensor) – AF: automatic focus control (adapts the focus) – AWB: automatic white balance (adapts the color tones to have a white balance).
BPC	Bits per component (for example, RGB565 has five bits for the red component, hence 5 bpc)
BPP	Bits per pixel (for example, RGB565 has 16 bits per pixel, hence 16 bpp)
CSI	Camera serial interface, and its latest standard by MIPI®, the CSI-2 v1.3 (and v2.0)
DCMIPP_	Digital camera interface - Pixel pipeline
DT	Data type, used in the CSI-2 standard. The data type is a concept similar to the pixel format, but adds support for other data like the byte format.
Interlaced	Interlaced video: the field with odd lines is transmitted first, followed, after the last odd line, by the field with even lines. The two fields are thus consecutive but exclusive.
Interleaved	Interleaved packets, applies to CSI-2: two groups of packets transmitted by the sensor can be emitted mixed, as a packet granularity.

Table 329. Glossary (continued)

Item	Definition
ISP	Image signal processing: algorithms that retune the sensor output (through correction of pixels, demosaicing, lens-correction, light/contrast correction) to build a good quality picture.
NPU	Neural network processing engine (also known as NN, neural network).
Planar	Defines in how many sub-buffers a pixel buffer is split into: 1 (coplanar), 2 (semi-planar), 3 (full planar).
ROI	Region of interest (pixel box surrounding the region of interest)
VC	Virtual channel, used in the CSI-2 standard, makes it possible to simultaneously transmit up to four parallel flows (in CSI-2 v1.3) or more (in CSI-2 v2.0).
YUV444	Pixel format, YUV color reference, all three components (Y, U, V) given per pixel.
YUV422	Pixel format, YUV color reference, chroma component (U, V) sub-sampled 1/2 in X

39.2 DCMIPP main features

- Parallel input interface:
  - Pixel rate: up to 16 bits in parallel, 100 Mpixel/s (typically 1080p30, maximum 2048x2048 on processing pipelines after decimation).
  - Pixel format: RGB565, RGB888, YUV422, raw Bayer/Mono 8/10/12/14, and ByteStream (JPEG)
- CSI-2 host input interface:
  - Pixel rate: 32 bits in parallel at 200 Mpixel/s (typically 5 MPixel x30), supports MIPI® CSI-2 v1.3
  - Pixel format: all MIPI CSI-2 v1.3, among others RGB565, RGB888, YUV422, raw Bayer (see *Table 335*)
  - Abutted CSI-2 host IP: 1-to-2 DL at up to 2.5 Gbps/lane
  - Features: Interleaved packets, four virtual channels
- Pipeline maximum resolution:
  - Dimensioned for 5 Mpixel (typically 2560 x 2048 or 2688 x 1944) before decimation
  - Maximum pipe resolution: 4094 x 4094
  - Maximum demosaicing or downsize or YUV420 width: 2688 pixels
- Pipe pixel rate: 200 Mpixel/s peak (150 Mpixel/s average + blanking)
- Maximum frame per second (fps): typical examples are defined with 30 fps, it is possible to reach higher rates (for example 120 fps) by decreasing image resolution to guarantee the maximum pixel rate constraints.
- Flow selection and frame control:
  - CSI-2 interleaved packets supported (with some restrictions)
  - Capture mode in Continuous vs. Snapshot mode
- Byte-to-pixel conversion:
  - Enables the above flexible input pixel formats to be extracted and fed to pipes
  - Non-extracted pixel formats can still be dumped via Pipe0, to allow an ad-hoc software extraction
- Statistic removal:
  - To remove unused metadata (like statistics) from received pictures
- Bad pixel removal (automatic detection and correction of bad pixels from sensor array)
- Decimation of one pixel every 1/2/4/8:
  - Decimation to fit the demosaicing (maximum 2688 pixel width)
  - Raw Bayer format support, via a decimation of components (2 by 2)
- Integrated image processing block, which makes possible to connect very low cost camera modules without any embedded ISP
  - Black level calibration
  - Statistics extraction
    - > To extract average R, G, B, L values (for auto exposure and white balance)
    - > To sort any R, G, B, L value into 12 dynamic bins (for contrast enhancement)
  - Exposure control (amplifies the exposure up to 256x)
    - > To smoothly adapt to dark exposures

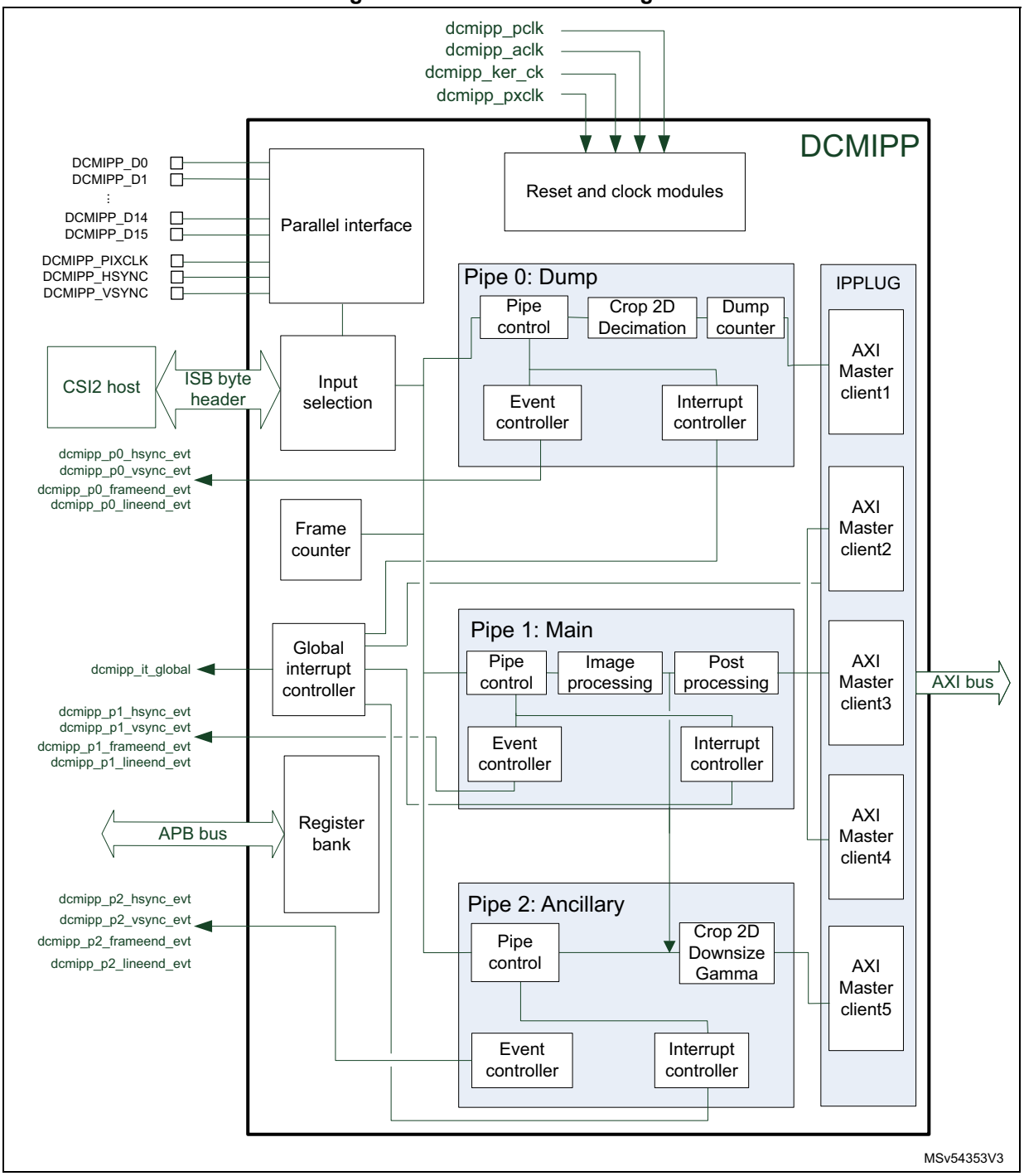
  - Raw Bayer to RGB conversion (demosaicing)
    - > Mid quality conversion for display and software analysis purposes (high quality conversion can be done in software upon trigger, for still-image only)
    - > Algorithm with a 4 x 3 heuristic analysis matrix
  - Color conversion to adapt to the sensor and tune the illumination
    - > Configurable coefficients, to map any linear tuning
  - Contrast enhancement
    - > Non-linear luminance tuning, linearized across eight luminance segments.
- Decimation on Pipe0 (dump pipe)
  - Pure sub-sampling, ratio 1 vs. 1, 2, 4 in X and ratio 1 vs. 1, 2 in Y (no X if CSI source)
- Parallel pipelines for parallel applications:
  - Pipe0 (for data dump) for a direct dump without processing
  - Pipe1 (for main usage) with downsize, ColorConv, decimation, multi-planar
  - Pipe2 (for ancillary usage) with downsize, decimation, coplanar
- Downsize
  - Decimation, optional pre-processing, sub-sampling ratio 1 vs. 1, 2, 4, 8 in X and Y
  - Box-filtering, with any decimal ratio up to 8x8, on Pipe1 and Pipe2
- Up to eight regions of interest (ROI) can be defined for displaying purpose (only supporting RGB data format)
  - Up to 64 line colors can be selected by the software, per ROI
  - Up to four line widths can be configurable (1, 2, 4, 8 pixels, common to all ROIs)
  - Pipe0 does not support ROI
- Gamma conversion for display:
  - Non-linear transform, with seven interpolated segments, with static coefficient for G = 2.2
- Color conversion 2 for RGB to YUV conversion:
  - to/from YUV: 12 coefficients (BT.601, 709, any), with optional range ext/clamp
- Output pixel format:
  - Pipe0: any data as-is, Y/Rb: 8/10/12/14 statistics, bit streams
  - Pipe2: 888, 565, 422-1, Y8, ARGB and RGBA (coplanar only)
  - Pipe1: adds 422-2, 420-2, 420-3 (multi-planar possible) to Pipe2 formats
- AXI master:
  - Output FIFOs to drain pixel/data to the memory (linked to the memory latency) from the pipe(s)
  - The unique AXI-Master interface implements one client per pipe, with semi- or multi-planar pipes using, respectively, two or three clients
  - Double buffer mode capab/ility per pipe
  - Address wrapping to write into a ping-pong buffer
- Overrun detection:
  - Notification in case of frame impacted by pixel congestion (no impact on the following frame if the congestion cause is solved)

39.3 DCMIPP functional description

39.3.1 DCMIPP block diagram

The block diagram of the DCMIPP is shown in *Figure 397*.

Figure 397. DCMIPP block diagram



39.3.2 DCMIPP pads and internal signals

Table 330. DCMIPP input/output pads

Pin name	Signal type	Description
DCMIPP_Dn (n = 0 to 15)	Input	Bit n of the parallel data bus coming from the camera sensor
DCMIPP_PIXCLK	Input	Pixel clock sent by the master (parallel camera sensor module)
DCMIPP_VSYNC	Input	VSYNC signal (vertical synchronization) coming from the camera sensor
DCMIPP_HSYNC	Input	HSYNC signal (horizontal synchronization) coming from the camera sensor

Table 331. DCMIPP input/output pins

Internal signal name	Signal type	Description
dcmipp_pclk	Input	DCMIPP APB clock (APB bus)
dcmipp_aclk	Input	DCMIPP AXI clock (AXI bus)
dcmipp_ker_ck	Input	Interface from the CSI2 host
dcmipp_it_global	Output	DCMIPP interrupts (refer to *Table 364* for the full list of interrupt sources)
dcmipp_p0_hsync_evt	Output	Pipe0 Hsync event
dcmipp_p0_vsync_evt		Pipe0 Vsync event
dcmipp_p0_frameend_evt		Pipe0 frame end event
dcmipp_p0_lineend_evt		Pipe0 line event
dcmipp_p1_hsync_evt	Output	Pipe1 Hsync event
dcmipp_p1_vsync_evt		Pipe1 Vsync event
dcmipp_p1_frameend_evt		Pipe1 frame end event
dcmipp_p1_lineend_evt		Pipe1 line event
dcmipp_p2_hsync_evt	Output	Pipe2 Hsync event
dcmipp_p2_vsync_evt		Pipe2 Vsync event
dcmipp_p2_frameend_evt		Pipe2 frame end event
dcmipp_p2_lineend_evt		Pipe2 line event

39.3.3 DCMIPP reset and clocks

Table 332. DCMIPP clocks

Domain	Clock	Maximum frequency	Comments
Parallel	dcmipp_pxclk	120 MHz	Up to 2x 120 MB/s in 16-bit mode
Pipeline	dcmipp_ker_ck	333 MHz	Up to 333 Mpixels/s (minus the blanking)
AXI bus	dcmipp_aclk	400 MHz	SoC AXI frequency
APB bus	dcmipp_pclk	200 MHz	SoC APB frequency

Table 333. DCMIPP resets

Domain	Reset type	Comments
Parallel + Pipeline	Asynchronous	Resynchronized internally on dcmipp_pxclk/dcmipp_ker_ck
AXI bus	Synchronous	Resets directly the dcmipp_aclk domain
APB bus	Synchronous	Resets directly the dcmipp_pclk domain

Clocks and pixel rate limitations

This paragraph lists the DCMIPP limitations and bottlenecks:

- The target is to sustain an average pixel rate (PixRateAvg):
  - A 5 Mpixel sensor at 30 fps requires 150 Mpixel/s (CSI-2 interface)
  - Higher resolution sensors can be supported, but with lower FPS (for example 10 Mpixels at 15 fps)
- Vertical and horizontal blanking:
  - The full system (sensor, interface, pipe, AXI) embeds limited capacity FIFOs. The idle times that the sensor inserts during its blanking periods impact all the downstream elements, including interface, pipe, and AXI. A pixel rate (PixRatePeak) higher than average must be used. The idle time due to the blanking can reach 33% of the frame duration, sometimes more (check sensor specification).
  - PixRatePeak = 1.33 * PixRateAvg = 200 Mpixel/s in our example.
- Parallel interface clock (dcmipp_pxclk):
  - On the parallel interface, pixels are received sequentially, in one, two or three cycles. The maximum implemented clock (120 MHz) constrains the peak pixel rate, depending on the ratio of cycles per pixel.
  - PixRatePeak ≤ dcmipp_pxclk / (1, 2 or 3) = 120 / (1, 2 or 3)
  - Clock frequency:
    - > dcmipp_pxclk ≤ dcmipp_ker_ck (due to resync interface between them)
    - > dcmipp_pxclk ≤ dcmipp_aclk
  - Consequences: see *Table 334*.

Table 334. Parallel interface maximum resolution (80 MHz)

Pixel		Transmission on the interface			Resolution (30 fps), including blanking	
Format	Width (bpp)	Width	Cycles/pixel	Pixel rate	Mpixel/frame	Example
Mono/raw	8	8	1	80	2.15	1080 p
	14	14	1	80	2.15	1080 p
RGB565	16	8	2	40	1.08	720 p
		16	1	80	2.15	1080 p
RGB888	24	8	3	27	0.72	qHD
		12	2	40	1.08	720 p
YUV422	16	16	1	80	2.15	1080 p
		16	2	40	1.08	720 p
YUV444	24	12	2	40	1.08	720 p
		16	2	40	1.08	720 p

  - Clock setting: dcmipp_pxclk ≥ PixRatePeak * Ratio(1,2,3)

- Pipe clock dcmipp_ker_ck (when using the CSI-2 host interface)
  - CSI-2 PHY provides data bit at the clk_bit rate, on DL parallel data lanes.
  - Rationale: internally, the CSI-2 host interface packs the serial stream on 32-bit words, resynchronized on the pipe clock (dcmipp_ker_ck).
  - Clock setting: dcmipp_ker_ck ≥ clk_bit * DL / 32 bpp = 156.25 MHz.
  - Rationale 2: The pixel pipe processes 1 pixel/clk, so that dcmipp_ker_ck must be higher than the wanted PixRatePeak.
  - Clock setting: dcmipp_ker_ck ≥ PixRatePeak.
  - For small footprint pixels (like Raw8), the DCMIPP is limited by its maximum dcmipp_ker_clk, and can be unable to handle the maximum CSI pixel rate (clk_bit * DL / 8 bpp for Raw8).
- Downsize:
  - The line buffer of the vertical downsize operator is limited to a width of 2688 pixels after the horizontal downsize operation, limiting the resolution, when the downsizing is active, to 2688 x 1944 (1944 is the usual maximum height with a width of 2688), thus 5.1 Mpixel.
  - If a camera resolution higher than 5 Mpixel is targeted, a basic decimation has to be performed to decrease the resolution to a maximum of 5 Mpixel before processing the downsizing operation.
- AXI clock:
  - The pixels must be reclocked from the processing to the AXI domain, resulting in
    > dcmipp_ker_ck ≤ dcmipp_aclk
  - The AXI bus must drain the running pipelines. The maximum footprint of pixel (or data) pipes is 32 bpp (Pipe1 and Pipe2) and 16 bpp (Pipe0). AXI is on 64 bits, with an efficiency of 8 / 9 for the typical 64-byte burst.
  - dcmipp_aclk ≥ dcmipp_ker_ck * (32 bpp + 32 bpp + 16 bpp) / (64bit / bus) * 9 / 8 (efficiency) = dcmipp_ker_ck * 1.4.

39.3.4 DCMIPP maximum resolution

The limitations related to the maximum resolution supported by the DCMIPP are:

- Maximum width and height across pipes: 4094 x 4094 pixels
  - Registers and counters are limited to 12 bits
- Maximum performance: 333 Mpixel/s on each pipe
  - Larger pixel rate can be used, but with less pipe active or lower memory latency
  - Maximum pixel rate on each pipe must remain below the pixel clock
- Maximum width on demosaicing: 2688 components wide
  - limitation related to the Line-buffer size
- Maximum width after downsize: 2688 pixels
  - limitation related to the Line-buffer size
- Maximum width for conversion to YUV420: 2688 pixels
  - limitation related to the Line-buffer size
- Max line stride: 32767 bytes
  - Registers and counters, to handle 4094 pixels x 32 bits x 2 (provision for interleaving)

39.3.5 DCMIPP minimum requirements for frame structure

DCMIPP imposes the following minimum requirements for the frame architecture:

Blanking phase

- Dump pipe
  - The minimal duration of the blanking between two frames must be an equivalent of 666 datawords of 32 bits
- Pixel pipe
  - Vertical blanking: minimum is 5 lines. If the line is less than 12 pixels, a minimum of 24 pixels equivalent for the vertical blanking parameter is requested
  - Horizontal blanking: a minimum duration of 12 pixels is required between two lines

Data image area

- Vertical image area: minimum one line (two lines if raw Bayer conversion is active)
- Horizontal image area: minimum two pixels
- The minimum total frame size is fixed to 36 pixels

39.3.6 Description of DCMIPP pixel format support

Table 335 lists the supported pixel formats in the input interfaces (parallel, CSI), as input to the pixel Pipe1 and Pipe2, and in the three output pipes.

Table 335. Supported pixel formats

Index (1)	Pixel format	BPP	Parallel input (clk/pix)(2)	CSI input	Pipe0 output (dump)	Pipe1 and Pipe2 input	Pipe1 output (main)	Pipe2 output (ancillary)
-	Bytes (CSI: long packets and user)	8	-	Y	Y	-	-	-
13(3)	Byte stream (JPEG, compressed video)	8	Y, 1	-	Y	-	-	-
14(4)	Other data	8 to 16(4)	Y, 1	-	Y	-	-	-
5	Raw6	6	(5)	Y(6)	Y(6)	-	-	-
5	Raw7	7	(5)	Y(6)	Y(6)	-	-	-
5	Raw8	8	Y, 1	Y	Y	Y	Y	Y
6	Raw10	10	Y, 1	Y	To 16 bpc	To 8 bpc	-	-
7	Raw12	12	Y, 1	Y	To 16 bpc	To 8 bpc	-	-
8	Raw14	14	Y, 1	Y	To 16 bpc	To 8 bpc	-	-
9	Mono8	8	Y, 1	-	Y	Y	Y	Y
10	Mono10	10	Y, 1	-	To 16 bpc	To 8 bpc	-	-
11	Mono12	12	Y, 1	-	To 16 bpc	To 8 bpc	-	-
12	Mono14	14	Y, 1	-	To 16 bpc	To 8 bpc	-	-
2	RGB444	12	(5)	Y	(5)	To 8 bpc	-	-
2	RGB555	15	(5)	Y	(5)	To 8 bpc	-	-

Table 335. Supported pixel formats (continued)

Index (1)	Pixel format	BPP	Parallel input (clk/pix)(2)	CSI input	Pipe0 output (dump)	Pipe1 and Pipe2 input	Pipe1 output (main)	Pipe2 output (ancillary)
2	RGB565	16	Y, 1, 2	Y	Y	To 8 bpc	Y	Y
3	RGB666	18	(5)	Y	(5)	To 8 bpc	-	-
4	RGB888 / YUV444(6)	24	Y, 2, 3	Y	Y	Y	Y	Y
-	xRGB888 / xYUV444(6)	32	-	-	Y	-	Y	Y
-	RGBx888 / YUVx444(6)	32	-	-	Y	-	Y	Y
1	YUV422-1 (YUYV)	16	Y, 1, 2	Y	Y	Y	Y	Y
-	YUV422-2	16	-	Y(6)	Y(6)	-	Y	-
-	YUV420-2 (NV21/NV12)	12	-	Y(6)	Y(6)	-	Y	-
-	YUV420-3 (YV12)	12	-	Y(6)	Y(6)	-	Y	-
-	YUV422 10 bpc	20	-	Y(6)	Y(6)	-	-	-
-	YUV420 10 bpc	15	-	Y(6)	Y(6)	-	-	-
-	Any other CSI-2	Any	-	Y(6)	Y(6)	-	-	-

1. Index is used to link the input pixel format to the DCMIPP_PRCR register. The index has a meaning only for camera connected with the parallel interface
2. Pixel formats on 16 bpp (like RGB565 or YUV422-1) can be received either on an 8-bit interface (using 2 cycles), or on a 16-bit input interface (using one cycle). Pixel formats on 24 bpp (like RGB888) can be received either on an 8-bit interface (using three cycles), or on a 12-bit input interface (using two cycles).
3. In Byte stream mode, the bytes are dumped consecutively with only a 32-bit padding (if needed) at the end of the frame (or JPEG stream). There is no 32-bit padding operation inside the stream until the VSYNC deassertion (end of frame, or end of JPEG stream).
4. If the data format is different from those in this table, user can choose “others” configuration into bit field FORMAT in the DCMIPP_PRCR register, using bit-field EDM to select how many data bits must be captured in one pixel clock. It is possible to capture two 8-bit data mapped on 16-bit parallel input data in a single cycle (for instance FORMAT = Others, EDM = 100, i.e. 16-bit capture on each pixel clock). Data are 32-bit padded at the end of a line to have a complete line aligned on 32-bit width.
5. On the parallel interface input, some sensor pixel formats are supported by the DCMIPP, by selecting a wider pixel format in DCMIPP, by mapping the sensor wires onto the MSB of the DCMIPP interface. On the input missing pins, it is recommended to replicate the sensor MSB pins onto the missing input LSB pins. This helps achieving full dynamic on the extended pixel format. The work-around can be applied on the following formats (sensor output on left, mapping on DCMIPP format on right):
   - Raw6: Raw8 with 1-cycle input, processed as a 8 bpp
   - Raw7: Raw8 with 1-cycle input, processed as a 8 bpp
   - RGB444: RGB565 with 1-cycle input, processed as a 16 bpp
   - RGB555: RGB565 with 1-cycle input, processed as a 16 bpp
   - RGB666: RGB888 with 3-cycle input, processed as a 24 bpp
6. The same bit mapping is used for the input RGB888 and YUV444, the input xRGB and xYUV, and similarly for the outputs. The difference between RGB and YUV is the color reference, set in the color conversion that occurs after the input or before the output.

39.4 DCMIPP input and flow control

This section describes the functional elements of the DCMIPP. For each, it details the features, the software configuration, and provides a software configuration example (when needed).

39.4.1 DCMIPP common configuration

The DCMIPP common configuration register (DCMIPP_CMCR) enables the selection of the input interface—either parallel DCMI or serial CSI-2—from which to fetch the input pixels. It also allows swapping the R/B color components of the incoming pixels and selecting the pipe used by the frame counter.

Table 336. DCMIPP_CMCR bit function

Bit ID	Function	Comments
INSEL	Input selection	Select the input interface to fetch pixels from. The selection is common to all pipes and must not be changed while they are active. – 0: DCMI – 1: CSI-2
PSFC	Pipe selection for the frame counter	Select which pipe the frame counter maps to.
CFC	Clear frame counter	Clear the frame counter when set.
SWAPRB	Swaps R-versus-B components (and U-versus-V) of the input pixels	Operate only on pixel pipes, not on dump pipe.

39.4.2 Parallel input interface

The parallel interface is a flexible pixel interface that clocks in a bus of 8- to 16-bit in parallel, with an externally provided clock, and adds vertical and horizontal synchronization, provided either with specific pins (external sync) or flagged via specific data (as in CCIR601).

Interface (all inputs)

- Clk: clock up to 120 MHz, internally named dcmipp_pxclk
- Data: 16 IO pins, up to 120 Mbps/pin, 8- to 16-bit, pixels potentially serialized
- HSync, VSync, but no enable (EN)

Features

- Horizontal and vertical synchronization:
  - External, via the HSync and VSync pins.
  - Internal, via the CCIR601 flagging.

  Data provided during the horizontal or vertical blanking cannot be captured.
- Input pixel format (see *Section 39.3.6*)
  - Binary stream like JPG
  - Single cycle data:
    - Raw Bayer and monochrome: 8/10/12/14 bits in parallel
    - YUV422, RGB565:16 bpp on 16 bits in parallel
  - Double-cycle data:
    - YUV422, RGB565:16 bpp on 8 bits in parallel
    - RGB888/YUV444: 24 bpp on 12 bits in parallel
  - Triple-cycle data:
    - RGB888/YUV444: 24 bpp on 8 bits in parallel
- Flexible input:
  - MSB-vs-LSB bits can be swapped to ease PCB placement
  - Cycle0-vs-Cycle1 can be swapped (for 2-cycle long pixels).
  - Red-vs-blue components can be swapped (for RGB, or U-vs-V for YUV), this last swap operates only on the pixel pipes and not the dump pipe.

Hardware synchronization mode

In this mode two synchronization signals (DCMIPP_HSYNC and DCMIPP_VSYNC) are used.

Depending on the camera module/mode, data can be transmitted during horizontal/vertical synchronization periods. DCMIPP_HSYNC/DCMIPP_VSYNC act as blanking signals, as all data received during DCMIPP_HSYNC/DCMIPP_VSYNC active periods are ignored.

To correctly transfer images in the RAM buffer, data transfer is synchronized with the DCMIPP_HSYNC/DCMIPP_VSYNC signals. When the hardware synchronization mode is selected, and capture is enabled (CPTREQ bit set in DCMIPP_PxFCTCR), data transfer is synchronized with the deassertion of the DCMIPP_VSYNC signal (next start of frame).

Transfer can then be continuous, with successive frames transferred by the IP-Plug to successive buffers or the same/circular buffer.

Embedded data synchronization mode

In this synchronization mode, the data flow is synchronized using embedded 32-bit codes, using the 0x00/0xFF values not used in data anymore.There are four types of codes, all with 0xFF00 00XY format. The embedded synchronization codes are supported only in 8-bit parallel data capture (in the DCMIPP_PRCR register, the EDM[2:0] bits must be programmed to “000”). For other data widths, this mode generates unpredictable results.

Note: Camera modules generate up to eight synchronization codes when in interleaved mode, while the DCMIPP reacts to one single code. As a consequence, an interleaved flow with an embedded synchronization has one every other frame discarded as not detected.

Mode 2

Four embedded codes signal the following events:

- Frame start (FS)
- Frame end (FE)
- Line start (LS)
- Line end (LE)

The XY values in the 0xFF00 00XY format of these codes are programmable (see *Section 39.14.9*).

A 0xFF value programmed as a frame end means that all the unused codes are interpreted as valid frame end codes.

In this mode, once the camera interface has been enabled, the frame capture starts after the first occurrence of the frame end (FE) code followed by a frame start (FS) code.

Mode 1

An alternative coding is the camera mode 1. This mode is ITU656 compatible.

The codes signal another set of events:

- SAV (active line) - Line start
- EAV (active line) - Line end
- SAV (blanking) - Start of line during interframe blanking period
- EAV (blanking) - End of line during interframe blanking period

This mode can be supported by programming the following codes:

- FS ≤ 0xFF
- FE ≤ 0xFF
- LS ≤ SAV (active)
- LE ≤ EAV (active)

An embedded unmask code is also implemented for frame/line start and frame/line end codes. Using it, it is possible to compare only the selected unmasked bits with the programmed code. User can therefore select a bit to compare in the embedded code and detect a frame/line start or frame/line end. This means that there can be different codes for the frame/line start and frame/line end with the unmasked bit position remaining the same.

Example:

- FS = 0xA5
- Unmask code for FS = 0x10

In this case the frame start code is embedded in bit 4 of the frame start code.

Note: FEC sequence must be sent before the transfer of the first frame, otherwise other codes are not decoded and the first frame can be lost after the DCMIPP has been enabled. After the first FE sequence, the following frame is captured based on FS sequence.

Error conditions

Error conditions can be detected when using the embedding synchronization data modes. Flags PRERRF in DCMIPP_CMSR2 and ERRF in DCMIPP_PRSR are used for this function. An interruption can be triggered based on error detection if bit PRERRIE in

DCMIPP_CMIER is set for the global interrupt (or bit ERRIE in DCMIPP_PRIER for a local interrupt line).

For instance, such kind of wrong sequences or values generate an error:

- FF 00 00 XY sequence detected with XY value different from the one set for FS/LS/FE/LE
- LS = 0xFF or LE = 0xFF
- If an LS or FS sequence follows an LS sequence

Software configuration

Table 337. DCMIPP_PRCR bit function

Bit ID	Function	Comments
ENABLE	Enables the parallel interface	Low-power mode when disabled.
PCKPOL	Defines clock polarity	– 0: Falling edge – 1: Rising edge
ESS	Defines if VSync and HSync synchronizations are provided by sideband (specific hardware IO) or by in-band (specific embedded codes) signals	– 0: Hardware (external) synchronization: using DCMIPP_HSYNC and DCMIPP_VSYNC pins. – 1: Embedded (internal) synchronization: using codes FSC, FEC, LSC, LEC and masks in registers DCMIPP_PRESCR and DCMIPP_PRESURy.
VSPOL, HSPOL	Defines the polarity of the IO DCMIPP_VSYNC and DCMIPP_HSYNC used for, respectively, the vertical and horizontal synchronization	– 0: Active low – 1: Active high
FORMAT	Defines the pixel format used on the IO DCMIPP_Dx	From 0 to 15
EDM	Defines the amount of valid bits received per cycle	8/10/12/14/16 bits
	The duration of a pixel is extracted by the bit-per-pixel (given by FORMAT) and bit-per-cycle (given by EDM) ratios	1, 2 or 3 cycles
SWAPBITS	Swaps the 16 bits of the parallel interface	15:0 becomes 0:15
SWAPCYCLES	Swaps, when a pixel is received in two cycles, the data from the first cycle with the data from the second cycle	Software must not activate it when a pixel is received in one or three cycles.

The embedded synchronization codes are configured by means of DCMIPP_PRESCR and DCMIPP_PRESUR registers (see *Table 338*).

Table 338. DCMIPP_PRESCR and DCMIPP_PRESUR bit function

Bit ID	Function
FSC, FEC	Frame-start and frame-end synchronization, when embedded synchronization ESS = 1
FSU, FEU	Masks for FSC and FEC codes

Table 338. DCMIPP_PRESCR and DCMIPP_PRESUR bit function (continued)

Bit ID	Function
LSC, LEC	Line-start and line-end synchronization, when embedded synchronization ESS = 1
LSU, LEU	Masks for LSC and LEC codes

39.4.3 Interface from CSI-2 host

The CSI-2 host and DPHY_RX (both external to the DCMIPP) provide the data from the CSI-2 camera sensor, forward them into the DCMIPP using the ISB byte header interface.

The ISB byte header features are:

- CSI-2 clock interface aligned with the DCMIPP processing clock (333 MHz)
- Data width: 32 bit (data and CSI headers are interleaved)
  - Frame synchronization short packet
  - Frame number, Data ID, and packet type are provided (see *Section 39.7.9*).
- Line Sync short packets are not received.
- Headers of data long packet:
  - Packet length, Data ID, and packet type are provided (see *Section 39.7.9*).
- Payloads of data long packet:
  - Data as 32-bit words, as collected by the CSI-2 host.
- Packet CRC is not received.

39.4.4 Input selection

Input selection block is ahead of any pipe to select the camera sensor module interface connected to the DCMIPP. Two interfaces are supported:

- Parallel interface: up to 16-bit parallel interface with physical synchronization signals (DCMIPP_VSYNC/DCMIPP_HSYNC) or software embedded ones (as in CCIR601).
- MIPI CSI-2 interface: Host ISB byte header internal connection from the CSI2 host controller placed in front of the DCMIPP.

Both interfaces are exclusive. The selected one is applied to all the input pipes.

The camera sensor interface is selected by means of bit INSEL of the DCMIPP_CMCR register.

39.4.5 Flow selection

The module, present in each pipe, handles the flow control around the CSI2 camera sensor interface. For each pipe, the software needs to configure which virtual channel (VC) and which Datatypes (DT) are processed by the pipe.

For Pipe0 and Pipe1, multiple Datatypes can be selected whereas only one virtual channel can be defined for a frame. Configuration can be changed frame by frame if the application requires it, offering some flexibility.

For Pipe2, when the pipe is independent with PIPEDIFF = 1, only one virtual channel and only one data type by frame can be selected.

The software has to reconfigure the flow selection module to handle interleaved frames from a sensor if it uses virtual channels as a mean to make the differentiation, accepting to lose frames since there can be only one virtual channel active for a given frame (it decreases the FPS).

Multiple cameras data flow can be received simultaneously through the CSI2 ISB header bus. For instance, for a system having two different camera data flows, it is possible to redirect one virtual channel on a pipe and the other one on the other pipe. It can be interesting to dump them on Pipe0, but only one frame over two since the flow selection block has to be reconfigured to switch from one VC to another one.

The flow selection module is used also to share some image processing features (demosaicing, color conversion, statistics, contrast enhancement, auto exposure, statistics removal and basic decimation) between Pipe1 and Pipe2, whatever the camera sensor interface solution (parallel interface or CSI2 interface): when bit PIPEDIFF in the DCMIPP_P1FSCR register is 0, or the input is on the parallel interface that provides unique flow, the Pipe2 samples pixels from the output of the ISP of Pipe1 (see *Figure 396*). PIPEDIFF must be reconfigured when Pipe1 and Pipe2 are disabled.

Software configuration

The following configurations must be set first, because static in the DCMIPP_PxFSCR registers:

Pipe0

- DTMODE[1:0]: Mode for CSI flow selection into that pipe
  - 0: only flow VC/DTIDA is forwarded in the pipe
  - 1: flows VC/DTIDA and/or VC/DTIDB are forwarded in the pipe
  - 2: all Datatypes of the selected virtual channel, except the VC/DTIDA or VC/DTIDB, are forwarded in the pipe
  - 3: all Datatypes of the selected virtual channel are forwarded in the pipe
- VC[1:0]: Virtual channel ID of the CSI flow to select for further processing on the pipe
- DTIDA[5:0]: Data type ID A to select in the CSI flow for further processing in the pipe
- DTIDB[5:0]: Data type ID B to select in the CSI flow for further processing in the pipe
- PIPEN: activation of the Pipes, to be set after the above configurations.

Pipe1

- DTMODE[1:0]: mode for CSI flow selection into that pipe
  - 0: only flow VC/DTIDA is forwarded in the pipe
  - 1: flows VC/DTIDA and VC/DTIDB are forwarded in the pipe
- VC[1:0]: virtual channel ID of the CSI flow to select for further processing on the pipe
- DTIDA[5:0]: data type ID A to select in the CSI flow for further processing in the pipe
- DTIDB[5:0]: data type ID B to select in the CSI flow for further processing in the pipe
- FDTFEN: when the DTIDx is detected depending on DTMODE, the software forces the data type format to the value set into FDTF[5:0] for the pixel pipe processing. It can be used, for instance, to address sensor 3D camera through specific bridge using CSI2

"user data type" to make distinction between left and right camera sensor (each camera has its own user data type value, but sharing the same pixel format).

- FDTF[5:0]: data type value to be forced
- PIPEDIFF: Pipe2 to take same input as Pipe1 or differentiated:
  - 0: Pipe1 and Pipe2 take the flow selection configuration of Pipe1. Pipe1 shares the image processing functions with Pipe2.
  - 1: Pipe1 and Pipe2 are considered as two independent pipes, without any shared configuration.
- PIPEN: activation of the pipes, to be set after the above configurations.

Pipe2

- VC[1:0]: Virtual channel ID of the CSI flow to select for further processing on the pipe.
- DTIDA[5:0]: Data type ID A to select in the CSI flow for further processing in the pipe
- FDTFEN: When the DTIDx is detected depending on DTMODE, the software forces the data type format to the value set into FDTF[5:0] for the pixel pipe processing. It can be used to address sensor 3D camera through specific bridge using CSI2 "user data type" to make distinction between left and right camera sensor for instance (each camera having its own user data type value, but sharing the same pixel format).
- FDTF[5:0]: Data type value to be forced
- PIPEN: activation of the pipes, to be set after the above configurations.

If PIPEDIFF = 0 in DCMIPP_P1FSCR, the configuration bits are directly those set in the DCMIPP_P1FSCR register.

The above bit fields are meaningful when PIPEDIFF = 1.

39.4.6 Frame counter

A frame counter is available for a tag purpose and counts all the frame received on the selected pipe.

The counter is 32-bit, read-only. It is active at least when bit PIPEN of the pipe to whom it is connected is enabled (PIPEN = 1).

It provides an (almost) unique frame number, with a loop time of 4.5 years (if 30 fps).

The counter can be incremented, depending on the PSFC in DCMIPP_CMCR, at the FrameStart event of the selected virtual channel, when a CSI-2 camera sensor is connected, or at the FrameStart event of the pipe when the camera sensor module is connected with parallel interface.

The frame counter is cleared by setting CFC in the DCMIPP_CMCR register.

Software configuration

The frame counter must be configured before it is used for the pipe. Software action(s) must be performed in the DCMIPP_CMCR register:

- Select on which pipe the frame counter is connected by configuring bit PSFC[1:0]
- Frame counter runs as soon as the selected pipe is active (PIPEN = 1)

Note: It is possible to have the Frame counter run on a pipe, while having the pipe not flowing any pixel: this is achieved with pipe enable active (PIPEN = 1) to have the frame counter run, but with CPTREQ = 0 to avoid any pixel flow.

39.4.7 Frame control

The module is replicated for each dump and pixel pipe. It handles the frame control and the capture of each pipe, grouping data into frames, and capturing them when requested.

Definitions

There are two notions of frame events

- Event used inside the hardware (provided by the input interfaces):
  - FrameStart: when a frame starts at the input of the pipes (at frame control)
    - > Parallel interface with VSync fall
    - > CSI-2 interface: SoF (Start-of-Frame) packet
  - FrameEnd: when a frame finishes at the input of pipes (at frame control)
    - > Parallel interface: VSync rise (End-of-Frame)
    - > CSI-2 interface: EoF (End-of-Frame) packet
- Events exposed to software via DCMIPP events and interrupts:
  - VSYNC
    - > When a frame starts, observed at the input of the pipes (at frame control)
    - > It indicates that the current frame has started and has sampled its shadow registers, and that the software can start to configure the next frame
    - > It is recommended to use that interrupt to trigger the software that reconfigures the DCMIPP.
  - FRAME END
    - > When a frame finishes, observed at the output of pipes (when the last pixel has been written on the AXI)
    - > It indicates that the current frame has finished writing all its data to memory, and that the application can immediately start using them
    - > It is recommended to use that interrupt to trigger the software that forwards the memory buffer to other applications

Features

- Frame delineation: allows to dump data with the required frame granularity.
  - Extracts, per pipe, the FrameStart and FrameEnd event, based on the input Start-of-Frame and End-of-Frame.
    - > The FrameStart immediately generates a VSYNC event.
    - > The FrameEnd ripples till the end of the pipe, at the pixel packer level, where it generates the FRAME event.
  - Based on the algorithm detailed in *Extraction algorithm for FrameStart and FrameEnd*.
- Capture mode: can capture a frame snapshot, or capture continuously.
- Capture rate: can capture all frames, or one out of 2, 4, 8. It is only active when the Capture mode is in Continuous mode.
- Frame status: stores infos about the last captured frame (PxLSTFRM, PxLSTLINE), re-sampled at the FrameEnd event.

Extraction algorithm for FrameStart and FrameEnd

The extraction of the FrameStart and FrameEnd is straightforward.

In CSI-2 mode, each pipe is handling a unique virtual channel ID by frame. In case of interleaved flow in input of the DCMIPP, the software can decide to:

- Dedicate each virtual channel on a single pipe in case of multiple pipes, each of them having its own virtual channel number selected
- Decide to decrease the FPS ratio by changing the pipe configuration swapping with the other virtual channel ID (mainly in case of single pipe)

The interlaced mode can be handled by the software in CSI-2 mode by changing the pipe configuration frame by frame.

The following usecases are supported and detailed in the figures below:

- Default single VC: straightforward behavior, support in both pixel and dump pipes (*Figure 398*)
- Overlapping VC (supported on a single pipe via the loss of one frame when switching)
  - It can be supported by affecting each VC on a dedicated pipe (*Figure 400*)
- Exclusive VC (supported in both pixel and dump pipes), see *Figure 399*:

The use cases with two exclusive VCs can be handled either using one single pipe, by reconfiguring it from frame to frame, or by splitting the flow on more than one pipe, each of them handling its own VC:

- each flow arrives exclusively in one pipe and is processed as an independent frame, with its Start-of-Frame and End-of-Frame
- FrameStart and FrameEnd are generated per VC flow.

Figure 398. VC flow processed by one pipe

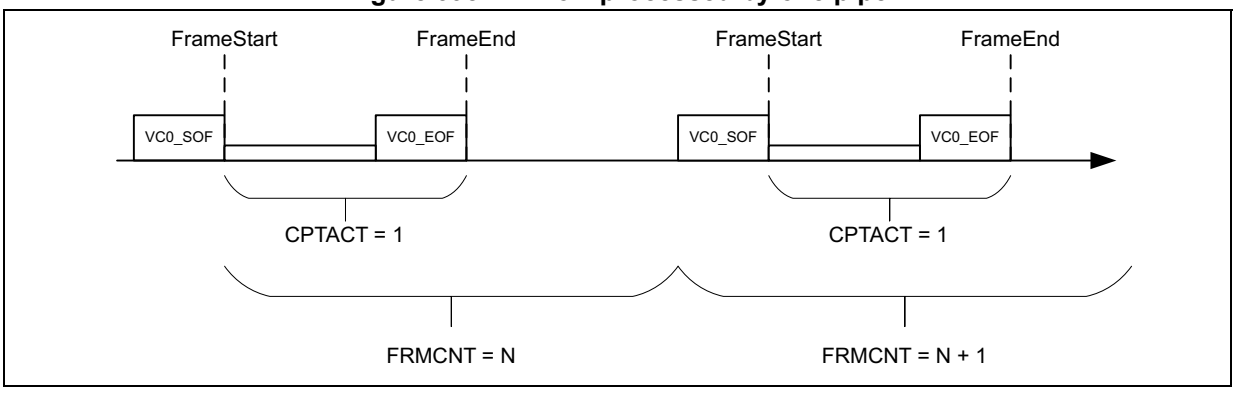
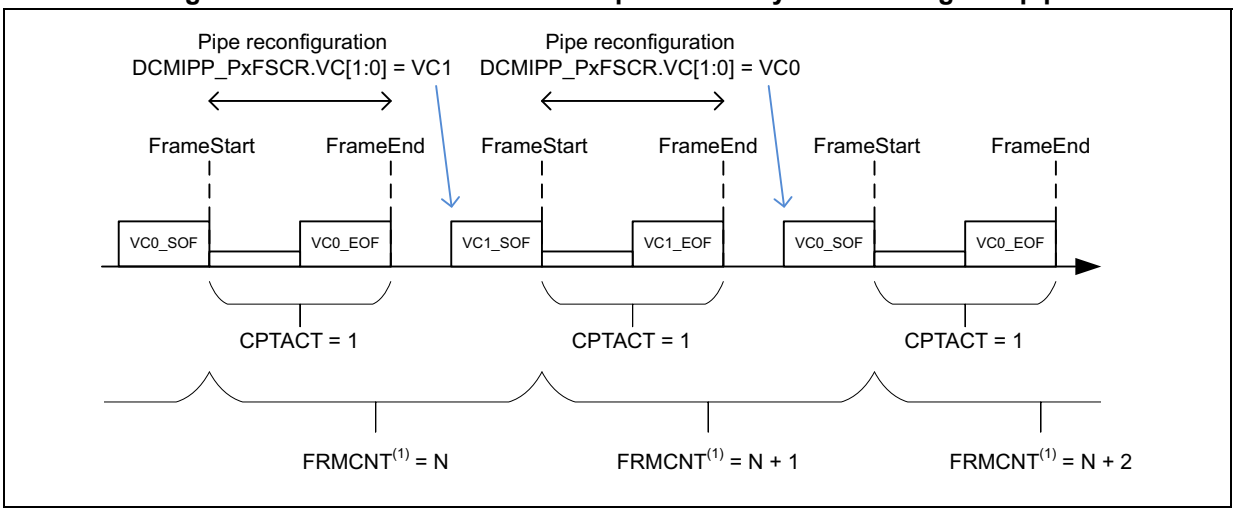
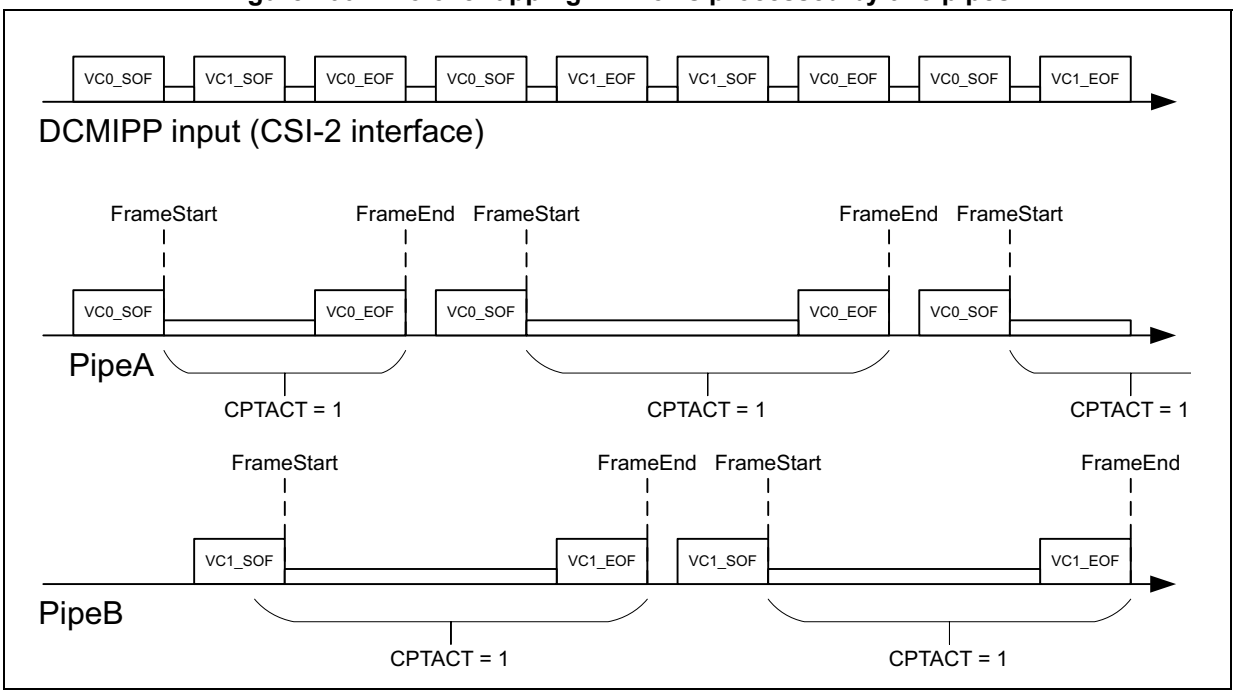


Figure 399. Two exclusive VC flows processed by one reconfigured pipe



1. If frame counter is attached to the pipe and if reset is not requested when changing VC.

Figure 400. Two overlapping VC flows processed by two pipes



Use cases

The Continuous mode is typically used to send to display or to software-analysis a continuous stream of frames.

The Snapshot mode is typically used to dump a high-resolution frame.

The Frame rate mode is typically used for a low-power continuous analysis, where only one frame every eight is dumped and analyzed.

The frame informations are typically used for interlaced video use cases, where the sensors transmit alternatively odd and even fields (top vs. bottom), and where software must be able to retrieve the type of field. Depending on the sensor, it can be retrieved from:

- the VC-ID, if the sensor uses one VC for one field and another for the other
- the LSB of the counted number of lines in that frame (indicates if the number is odd or even)
- the LSB of the frame counter extracted from the CSI information (if counting this pipe), which can be odd or even.

Software configuration

The frame control supports two different capture modes:

- Snapshot mode: the software captures single frames
- Continuous mode: the software captures a continuous sequence.

The timing diagram of these two modes is shown in *Figure 401*.

Figure 401. Snapshot (CPTMODE = 1) and Continuous (CPTMODE = 0) capture modes

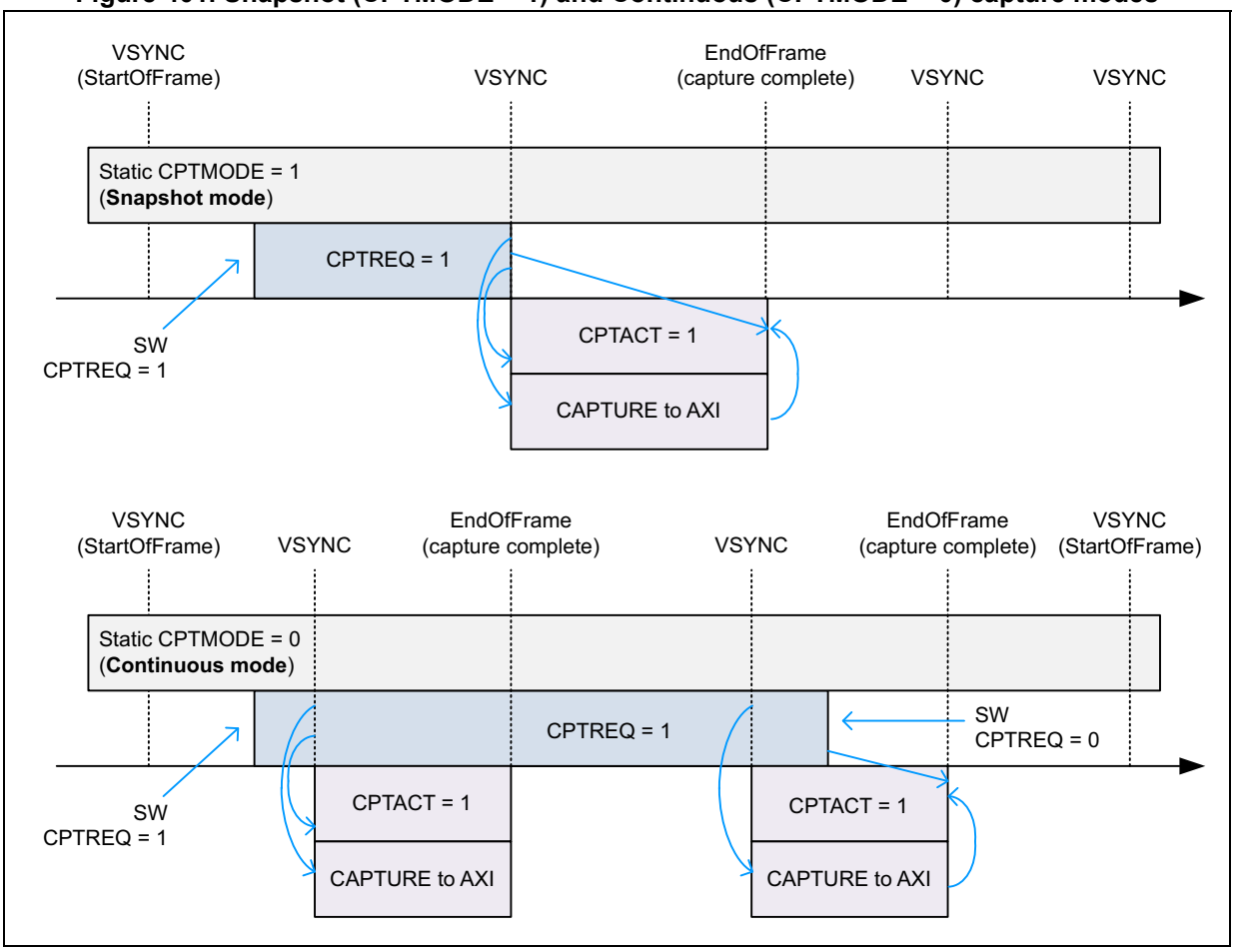


Table 339. DCMIPP_PxFCTCR bit function

Bit ID	Function	Values
CPTMODE	Capture mode	– 0: Continuous – 1: Snapshot
CPTREQ	Capture request	– 1: Capture requested
FRATE	Frame rate	– 0: Full rate capture – 1: 1/2-rate capture – 2: 1/4-rate capture – 3: 1/8-rate capture
CPTACT(1)	Capture status	– 1: Capture currently active

1. Bit read from the DCMIPP_PxSR register.

Configuration example

The software operations when using the Snapshot mode are the following:

1. Software sets CPTMODE = 1, to use the Snapshot mode.
2. Software sets PIPEN = 1, to let the flow selection send data to PipeN.
3. Software sets CPTREQ = 1, to request the capture of one frame.
4. At the first following VSync, the HW samples CPTREQ at 1, with the following impact:
   a) The capture effectively starts: pixels flow into the pipe and are dumped in memory.
   b) CPTACT is set to 1, to mention that a capture is currently ongoing, and that it is best to not modify the configuration of the pipe operators, unless shadowed.
   c) CPTREQ is reset to 0, so that only a single frame is dumped.
5. At the following capture complete interrupt
   a) CPTACT is reset to 0, to signal that the capture is over.
   b) The capture is complete, no more pixels are flowing, hence software can restart to update the configuration of non-shadowed registers without any issue, and use the captured pixels in memory. Depending upon the sensor blanking, this period can be short.
6. At the next VSync, CPTREQ usually is sampled at 0, so that no more frames are captured
   – As soon as CPTREQ is reset to 0 the software can set it again to 1, to request a capture at the following frame. It means that a continuous sequence of frame can be captured by setting again CPTREQ to 1, continuously frame after frame.

The software operations when using the Continuous mode are the following:

1. Software sets CPTMODE = 0, to use the Continuous mode.
2. Software sets PIPEN = 1, to let the flow selection send data to PipeN.
3. Software sets CPTREQ = 1, to request the continuous capture of frames.
4. At the first following VSync, CPTREQ is sampled at 1, with the following impact:
   – As in Snapshot mode, the capture starts, and CPTACT is set at 1.
   – Unlike the Snapshot mode, CPTREQ is not modified and remains at 1.
5. At the following capture complete interrupt:
   – As in Snapshot mode, CPTACT is reset to 0, capture stops, software can shortly reconfigure the pipes, and software can use the captured pixels.
6. At the next VSync, and as long as CPTREQ remains at 1, the capture restarts, similarly as described above.
7. To stop capturing later frames, software resets CPTREQ to 0.
   – An ongoing capture continues until completion.
8. At the next VSync, CPTREQ is sampled at 0, and the next capture does not restart.

39.4.8 Pipe deactivation

It is possible to abort a pipe to stop any frame acquisition and to potentially offer a mean to reprogram it completely (including non-shadowed registers), or to let it fully disabled for some time. The following operating mode has to be considered to correctly stop the pipe:

1. Disable the CPTREQ bit of the DCMIPP_PxFCTCR register of the corresponding pipe
2. Poll CPTACT = 0 status bit in the DCMIPP_CMSR1 register to check if the pipe is no longer active (idle state)
3. Disable PIPEN in the DCMIPP_PxFCTCR register.

When disabling the pipe, the last frame is processed completely by respecting the above operating mode.

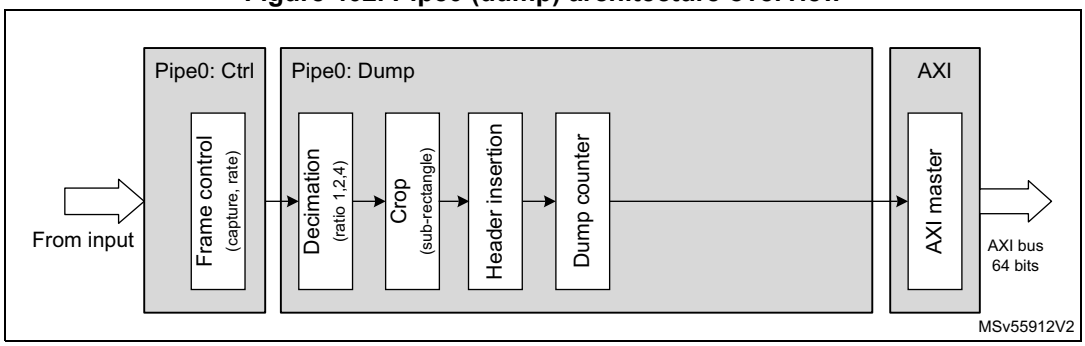
Note: To disable the parallel interface (PREN = 0 in the DCMIPP_PxFSCR), it is recommended to first disable the pipes considering the above operation mode, before switching off the parallel interface.

39.5 Pipe0 (dump pipe)

39.5.1 Overview

Pipe0 works as a dump pipe: it extracts data from the camera sensor module and dumps them (as-is) to the targeted memory, with some basic decimation and cropping 2D operations in between.

Figure 402. Pipe0 (dump) architecture overview



Pipe0 retrieves data from the camera sensor module connected to the DCMIPP through the 16-bit parallel interface or the CSI-2 interface, by means of the CSI-2 host controller peripheral. The selection between interfaces is managed by means of bit INSEL in the DCMIPP_CMCR register.

The flow selection chooses the virtual channels and data types to be processed by the pipe from the CSI2 host controller, see *Section 39.4.5* for more details.

The frame controller handles mainly the camera acquisition mode (continuous or snapshot, frame rate), see *Section 39.4.7* for more details.

When the input data from the camera is valid and supposed to be processed by Pipe0, 2D-cropping and decimation operations can be configured by the software. A dump counter combining with a limit amount of data to be set is offered to handle unknown length of data or to avoid the amount of data to be too wide within a frame.

Pipe0 can be reconfigured from time to time. Some events can be used to trigger the software reconfiguration routines (refer to *Section 39.13.2: Interrupts*).

It is also possible to write into the shadow registers within the processing of a frame. The values are loaded into the physical register based on the start of frame event.

39.5.2 Decimation

The decimation allows to cheaply downsize a frame.

Based on parallel interface camera module:

- Factor 1x, 2x, 4x at data/byte level depending on the configuration of bit field BSM[1:0] of the DCMIPP_P0PPCR register. The decimation is not working directly at pixel level, but it is based on either byte or 16-bit word granularity (for padded raw 10, for instance) to make equivalent decimation to one pixel out of two. Bit OEBS is used to start from the odd or even byte (or data) to capture first.
- The decimation can also be based on entire line, by capturing all the lines, or one line out of two thanks to bit LSM. Here, again, choice is given to the software to reject odd or even lines within the frame by configuring bit OELS.

When the camera sensor module is designed on CSI-2 interface, the decimation can be based filter out potentially, one line out of two. Indeed, each packet is by protocol composed for an entire line.

Note: Care must be taken when the CSI-2 header is dumped, as it is 32-bit wide, and it increases by one unit the data counter, taking part of the decimation. It is advised not to use the CSI header option when decimation takes place.

Table 340. DCMIPP_P0PPCR bit function

Bit ID	Function
BSM[1:0]	To select how much byte/data have to be captured within a line. This feature is available only for a parallel interface camera module.
OEBS	Allows the user to chose if the filtering starts from the odd or the even byte.
LSM	Possibility to filter out one line out of two.
OELS	The software can select if the filtering rejects the odd or even lines.

39.5.3 Crop/statistics selection/suppression

The dump pipe (Pipe0) has the capability to handle 2-D crop processing. The vertical cropping is based on line, thanks to HSYNC event (from the physical IOs or the embedded code detection if selected). It can be extracted from the CSI-2 packet information (each packet is one line wide by protocol).

The horizontal area on which the crop is applied is based on data (32-bit wide), and not on pixels since this pipe is not directly considering the pixels. It handles the data flow with 32-bit granularity. The CROP functionality is not supported when the pipe is convoying JPEG format. The ENABLE bit must be kept cleared into DCMIPP_P0SCSZR to avoid any unpredictable behavior.

The area to be captured within the frame has to be specified configuring the registers DCMIPP_P0SCSTR and DCMIPP_P0SCSZR. The starting point on the two axis are set accordingly as well as the width in both directions. It is possible to take the data inside or outside this area by means of bit POSNEG in the DCMIPP_P0SCSZR register. Some sensors can send sensor configuration data and statistics (histogram) on the very first and last lines of a frame. By specifying active the area outside of this window selection (POSNEG = 1), only statistic data are extracted by the way, and the software can decide to apply some processing on the data to correct, such as contrast or exposure.

Features

- 2D-Crop operation with granularity:
  - Lines in the vertical axis
  - 32-bit data in the horizontal axis
- Possibility to capture data inside or outside the defined window (valid image data or statistic data)
- Cropping with 0 value for HSIZE[11:0] or VSIZE[11:0] has no crop effect to the dimension for which the 0 value has been applied (vertical or horizontal axis).

Software configuration

Registers DCMIPP_P0SCSTR and DCMIPP_P0SCSZR have to be configured to set the crop feature into the dump pipe:

1. Configure the cropped horizontal starting point with HSTART[11:0] and the width with HSIZE[11:0] (both with a 32-bit data granularity).
2. Configure the cropped lines, starting at line VSTART[11:0] and with a height of VSIZE[11:0] (both with a line granularity)
3. If any value HSIZE or VSIZE is set to 0, the hardware does not consider crop operation in the vertical or horizontal dimension for which the 0 value has been applied, so that all the pixels in that dimension are sampled.
4. Select the inner or the outer part of the window for the data capture by configuring POSNEG bit.
5. Enable the crop feature setting ENABLE bit. This bit must be cleared when JPEG is selected as the input format for Pipe0.

Note: Cropping out the picture size (with too large HSTART or VSTART) leads to a not guaranteed processed frame.

39.5.4 Header insertion

The module works on the dump pipe(s) and it optionally inserts an header before the data of a former CSI packet.

Features

- Optionally inserts non-data information to help software interpreting the following data payload. The non-data info is Frame sync short packets and Header of data long packets (both non-data info into a 32-bit word), mapped as follows:
  - Frame sync short packet: defined by Frame type = SOF or EOF
    - > bits 31:16: Frame number
    - > bits 15:8: Reserved (forced to 0)
    - > bits 7:4: Virtual channel ID in CSI-2_v1.3, bits 7:6 = 00
    - > bits 3:0: Frame type (SOF = 0x1 or EOF = 0x2)
  - Header of data long packet (defined by below Frame type = DATA)
    - > bits 31:16: Amount of bytes in the payload
    - > bits 15:8: Data type ID in CSI-2_v1.3, bits 15:14 = 00
    - > bits 7:4: Virtual channel ID in CSI-2_v1.3, bits 7:6 = 00
    - > bits 3:0: Frame type (DATA = 0x0)

To keep consistency between the amount of bytes provided by the header versus those effectively dumped, it is recommended to not activate the header insertion simultaneously with the crop or decimation, as these functions modify the amount of dumped data.

When header insertion is active, the FrameEnd packet is dumped as a header, and is considered as the unique part of an additional line, inducing an event.

Software configuration

It is configured with the DCMIPP_P0PPCR register:

- HEADEREN:
  - 0: No header insertion
  - 1: Insertion of the headers, aligned on 32 bits.

39.5.5 Dump counter

The dump counter is present on dump pipe. It is used to count the amount of data that are dumped in that frame, and to potentially limit the amount dumped if too large. It allows the software to know the size of dumped buffer, and to make sure that no dump is made out of a preallocated buffer.

It is specifically useful when dumping a content whose length is unknown prior to reception, like an encoded JPG stream. When dumping a pixel frame, the size is known thanks to the configured width and height of the frame.

The counter is counting 32-bit data for almost all the input formats, even if the value is expressed in number of bytes. The counter increment is 4-bytes granularity for all formats except the JPEG byte stream input mode, for which the counter granularity is 8-bit because the application does not know the amount of dumped data.

Features

- Dump counter, counting the amount of bytes dumped. It counts the amount of dumped 32-bit words (the increment is 4), except when connected to a parallel interface camera and dumping a 8-bit data in byte stream mode (see FORMAT bit field in the DCMIPP_PRCR register), where it counts per byte (the increment is 1).
  - The counter saturates at 0x03FF_FFFF.
  - At FrameEnd, the counter value is recopied in a readable shadow register and the counter itself is reset to start the count of the next frame.
- Dump limit, that clamps the dumped words below a maximum limit:
  - If the counter reaches the limit, all following bytes are deleted until the next VSync.
  - The counter continues to count over the limit, to indicate how many bytes have been received and must have been dumped if not deleted.
- Interrupt limit is triggered when some bytes are deleted due to dump limit crossed.
- CSI headers, when dumped (HEADEREN = 1), participate to the counted and limited data volumes.
- Monochrome or raw Bayer padding (alignment of 10/12/14 bpp to 16 bpp) is also taken into account in data volumes.

Software configuration

Table 341. DCMIPP_P0DCCNTR and DCMIPP_P0DCLMTR bit function

Bit ID	Function
ENABLE	1: Enables the limit check.
CNT	Read-only, counts the amount of data with a 4-byte granularity for most of the input formats or with an 8-bit data granularity in JPEG byte stream mode selected into DCMIPP_PRCR.FORMAT). The counter increment is based on dumped data at the pipe output standpoint. The value is expressed in bytes, whatever the input format.
LIMIT	Gives the maximum amount of 32-bit data that can be dumped. Limit value 0 is inconsistent and the processed frame is not guaranteed if ENABLE bit is set.
LIMITIE	DCMIPP_PxIER.LIMITIE bit enables the interrupt.

Note: The dump counter sees the CSI-2 packet headers as normal 32-bit words. Raw Bayer and monochrome pixels on 10/12/14 bpc are padded onto 16 bits, and are thus slightly larger than their 10/12/14 bit size.

39.5.6 Double buffer mode

The dump pipe uses an AXI master interface to dump the data from the internal FIFO to the external memory. In the application, it is possible to handle frame data swapping memory area frame by frame. The double buffer mode fills up this function. There are two memory address registers set to initialize the base addresses of these memories areas. Each start of captured frame event swaps the memory base address to handle double buffering mode.

The double buffer mode can be used to allow post-processing on a buffer (frame buffer) while the other buffer is read to be displayed (display buffer).

Software configuration

Double buffering mode requires an activation, as well as addresses configuration, to define the two memory areas in which data are consecutively stored frame by frame at the output of the pipeline:

- DCMIPP_PxPPM0AR1 and DCMIPP_PxPPM0AR2 must be filled with the memory addresses associated to the double buffer memory locations.
- Bit DBM in the DCMIPP_PxPPCR register(s) is set to enable the double buffer mode.

39.6 Pipe1 (ISP part)

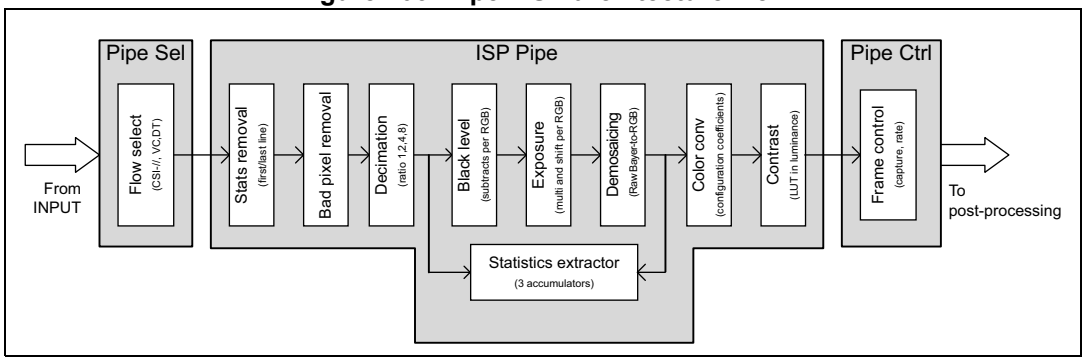
39.6.1 Overview

Pipe1 is a pixel pipeline in which some image processing (ISP) and post-processing functions are integrated, to relieve the software from very specific (time and memory consuming) operations. The ISP part implements functions like:

- Statistics removal within the dataflow
- Bad pixel detection to automatically remove noisy pixels
- Basic decimation to reduce high resolution camera sensor to fit with the image processing maximum resolution tolerance
- Auto exposure and statistics extraction based on pixel analysis
- Demosaicing algorithm to convert raw Bayer input pixels to RGB ones
- Color conversion and contrast enhancement functions

Pipe1 ISP block diagram is shown in *Figure 403*.

Figure 403. Pipe1 ISP architecture view



Pipe1 is the most featured pixel pipeline, even if the image processing functions can be shared with Pipe2 (refer to *Section 39.8.2*).

Pipe1 retrieves data from the camera sensor module connected to the DCMIPP with the 16-bit parallel interface or with the CSI-2 interface, thanks to the external CSI-2 host controller peripheral. The selection between interfaces is managed by means of bit INSEL in the DCMIPP_CMCR register.

The flow selection chooses the virtual channels and data types to be processed by the pipe from the CSI2 host controller. Refer to *Section 39.4.5: Flow selection* for more details.

The frame controller handles mainly the camera acquisition mode (continuous or snapshot, the frame rate). Refer to *Section 39.4.7: Frame control*.

Pipe1 can be reconfigured from time to time. Some events can be used to trigger the software reconfiguration routines (refer to *Section 39.13.2: Interrupts*).

It is also possible to write into the shadow registers during the processing of a frame. The values are loaded into the physical register based on the following start of frame event.

39.6.2 Byte-to-pixel conversion

The module works on the pixel pipes: it gets 32-bit words from the input (parallel or CSI) interfaces, extracts the pixels depending on the provided pixel format, and maps them onto

the triple-component pixel pipeline, that is 14 bpc at this stage (and 8 bpc later downstream).

Functionality

- Extracts various input pixel formats to a triple R, G, B lane
  - All CSI-2 formats of versions CSI v1.3 and v2.0 are supported, except Raw 6/7 bpc, YUV422-10 bpc, YUV420 (see *Table 335*)
- YUV420-to-YUV444 conversion is not supported
  - Pipe1 and Pipe2 cannot extract and process YUV420 pixels received from the CSI interface
- YUV422-to-YUV444 conversion:
  - Done by replication of the U and V components. It thus assumes that the location of the common chroma is defined at mid-distance between the two pixels.
- Extension of short triple component (RGB/YUV) to the output triple 8 bpc:
  - The extension is done by replicating the input MSB onto the missing output LSB. The replication onto the LSB allows to map any input dynamic on the full downstream dynamic.
  - Examples of 5 and 6 bpc extended to the output 8 bpc:
    - > Input Di[4:0]: Output Do = Di[4:0] and Di[4:2]
    - > Input Di[5:0]: Output Do = Di[5:0] and Di[5:4]
- Keeps as-is the single-components, usually wide:
  - The monochrome and raw Bayer 10/12/14 bpc are fed into the pipes as-is (the monochrome assumes a Y, with U = V = 128).

Software configuration

The used pixel format depends upon the source of the pixels:

- CSI-2 interface: the pixel format is embedded in the received CSI-2 packets, and defined in the data type field of the packet header.
  - When FDTFEN = 0, the byte-to-pixel conversion uses the data type provided by the CSI-2 packet, and works automatically, without need for any configuration.
  - When FDTFEN = 1, the data type is configured by software in the FDTF field.
- Parallel interface: the pixel format is determined by the source of the pixels, i.e. the camera sensor. The byte-to-pixel extracts the pixels based on the pixel format provided by the software in the parallel interface configuration, in the field of the DCMIPP_PRCR register.

39.6.3 Statistics removal

The module is used to removing potential statistic data inserted by a sensor, prior to starting the demosaicing conversion that is multi-pixel and would be impacted by remaining statistics. It is typically used by sensors with a parallel interface, which does not have any way to distinguish between statistics and pixel data. CSI-2 sensors typically split these flows using different virtual channel IDs.

Features

- Suppression of some of the first lines of the frame.
- Suppression of the last lines of the frame, after a configured line N.

Software configuration

Table 342. DCMIPP_P1SRCR bit function

Bit ID	Values	Comments
CROPEN	1: Active	If CROPEN = 0 all pixels are fed through.
FIRSTLINEDEL	0 to 7	Amount of first lines to delete when CROPEN = 1
LASTLINE	0 to 4094	Amount of lines to keep after the removed first lines. If LASTLINE is null, all remaining lines after FIRSTLINEDEL are fed through.

Configuration example

If the received frame contains four lines of statistics, followed by 100 lines of pixels and eight lines of ancillary data, the configuration to extract the 100 lines of pixels is the following:

- CROPEN = 1, to have the removal active
- FIRSTLINEDEL = 4, to remove the first four lines
- LASTLINE = 100, to keep the next 100 lines

39.6.4 Bad pixel removal

This module aims at detecting and correcting artifacts generated by a bad pixel on the sensor array. It is based on a correlation extraction, and not on fixed locations.

Functionality

- Works for raw Bayer flows only, as RGB flows are assumed as already processed.
- Scans all the arriving components, and extracts the horizontal continuity correlation versus the two left and two right neighbors, on both the local component and on the adjacent raw Bayer component:
  - if a component is seen as too light, it is declared as bad
  - too dark components are not extracted, to avoid false positives
- A component assumed bad is replaced by the horizontal average of its two neighbors (with the same component).

Software configuration

Table 343. DCMIPP_P1BPRCR and DCMIPP_P1BPRSR bit function

Bit ID	Function
ENABLE	– 1: Active – 0: No component checked nor replaced. To be enabled only for raw Bayer flows, as it corrupts RGB flows.
STRENGTH	3 bits, 0 (high tolerance) to 7 (aggressive detection)
BADCNT	12-bit counter, reporting the amount of bad components corrected. The counter saturates at its maximum value (4095). At FrameEnd, the counter is re-clocked (the read value remains stable one full frame), even if the frame is skipped (not captured), and reset to restart accumulation at the next frame.

Configuration example

A too aggressive bad pixel detection might detect false bad pixels and erroneously correct them (it can occur for very thin light vertical lines that cannot be distinguished from bad pixels in the implemented horizontal filter). To avoid such artifacts, it is advised to setup a software loop-back that tunes the filter strength so that the detected bad pixels remain within a predefined valid range.

Start configuration (or without software feedback loop):

- ENABLE = 1 (enabled)
- STRENGTH = 0 (high tolerance)

Software feedback loop:

Assumes a maximum amount of 100 bad pixels on the array (any other value can be used).

- BADCNT read: gets the amount of corrected bad pixels of the last frame
- Low-pass filter on the BADCNT, and/or hysteresis
  - BADCNT > 100: too aggressive, decrease STRENGTH
  - BADCNT < 30: too tolerant, increase STRENGTH (assuming there are bad pixel on the array)

Note: The expected amount of bad pixels (100 in the above example) can be set to other values, or made dependent upon the equivalent gain of the sensor (analog gain, exposure duration). A high-gain picture has more noisy pixels and can accept a more aggressive filter to smooth the noise, while a low-gain picture has mostly correct pixels, only very uncorrelated pixels need to be corrected.

39.6.5 Input decimation

The decimation makes it possible to cheaply downsize a frame, by a factor 2, 4, or 8, in both horizontal and vertical directions. It supports a special mode adapted to a raw Bayer format in input.

The decimation is redundant with the downsize, that brings a higher quality (the decimation discards the data of the dropped pixels, while the downsize averages all pixels and does not lose information).

The decimation has two main usages:

- Reduces the width of very high resolution sensors to one accepted by the raw Bayer-to-RGB conversion. Indeed, the raw Bayer-to-RGB embeds a line buffer that limits its processed images to a maximum width of 2688 pixels. The decimation makes possible, for instance, to plug a 12 Mpixel sensor (4000 x 3000), decimate its width by a factor 2x to 2000 pixels wide, and have it converted to RGB.
- Complements the downsize (limited to ratio of 8 x 8), to have a combined downsizing ratio of 64 x 64. It is recommended to have the largest ratio implemented in the downsize, and the decimation complement to that. For instance, for a ratio 30x, is it recommended to have a ratio 4x in decimation, and a ratio 30 / 4 = 7.5 in the downsize.

Features

- Forwards one item every 1, 2, 4, 8 pixels, independently, vertical and horizontal.
- Supports raw Bayer formats, where RG-GB components must remain grouped. In that mode the above forwarded items are not pixels, but a double component (for example R and G or G and B for a RGGB format), in both vertical and horizontal directions. For

instance, with a decimation of 2x, the input sequence R0, G0, R1, G1, R2, G2, R3, G3 gets forwarded as R0, G0, R2, G2 and not R0, R1, R2. The same applies vertically.

Software configuration

Table 344. DCMIPP_P1DECR bit function

Bit ID	Function	Value	Comments
HDEC[1:0]	Ratio (2 bits)	0	No decimation
		1	Ratio 1/2
		2	Ratio 1/4
		3	Ratio 1/8
VDEC[1:0]		0	No decimation
		1	Ratio 1/2
		2	Ratio 1/4
		3	Ratio 1/8

The raw Bayer mode in decimation is indirectly activated when the Datatype is defined as raw Bayer.

39.6.6 Black level calibration

This module suppresses a potential positive black level offset, which happens when the sensor sends non-null pixels when capturing a black picture.

Functionality

- Subtracts an offset from the R, G, B components.
- A different constant is used for R vs. G vs. B, including within raw Bayer pixels.

Software configuration

Table 345. DCMIPP_P1BLCCR bit function

Bit ID	Value	Comments
ENABLE	– 1: Active – 0: Bypasses R, G, B components	-
BLCR	-	8-bit offset, for red
BLCG	-	8-bit offset, for green
BLCB	-	8-bit offset, for blue

The offset subtracted is aligned components MSB: for instance, if components are defined on 12 bits, the 8-bit offset subtracted for red is BLCR « 4.

Configuration example

It is possible to use the statistics extraction, to extract the long-term most-dark R, G, B components, and assign their darkness as black. Their RGB values become the black level offset for each RGB.

For a raw Bayer sensor with 12-bit components, if the measured black level offset for red is 64 (out of a maximum of 4095), configure BLCR as 16.

39.6.7 Exposure compensation and white-balance calibration

This module aims at compensating bad exposure (too short/dark or too long/light), and at compensating a bad color balance to recenter the colors to more white ones.

The exposure and white balance operations are purely linear, and do not distort colors if performed correctly.

Functionality

- Multiplies R, G, B by a configured floating-point constant.
- A different constant is used for R vs. G vs. B, including within raw Bayer pixels.

Software configuration

Table 346. DCMIPP_P1EXCR1 and DCMIPP_P1EXCR2 bit function

Bit ID	Value	Comments
ENABLE	– 1: Active – 0: Bypasses R, G, B components	-
MULTR, SHFR	-	8-bit multiplier (mantissa), and up-shift value (0, 1, 2, 3, 4, 5, 6, 7) for red components.
MULTG, SHTG	-	8-bit multiplier and exponent, for green components.
MULTB, SHTB	-	8-bit multiplier and exponent, for blue components.

The idle setting (that sends out a similar value as input) is:

- MULTR = MULTG = MULTB = 128
- SHFR = SHFG = SHFB = 0

Similar idle couples are MULTx = 64 and SHFx = 1, or 32 and 2, or 16 and 3.

Configuration example for exposure alone

It is recommended to use the exposure control in conjunction with the contrast enhancement below in the pipe: the idea is to deliberately keep an exposure lower than ideal, to avoid burning the whites: the contrast enhancement, thanks to non-linear operator on luminance, lightens the dark pixels, and reduces the dynamic of the light pixels without burning them.

It is recommended to use the statistics to extract the average luminance across the whole frame (done best by extracting the R, G, B averages, and computing the luminance from the RGB).

The idea is to push the average output luminance to 128, or if working with the contrast enhancement too, to push it to a lower value, for instance 64.

When measuring an average luminance of 26 (target is 64), the exposure factor to apply is 64 / 26 = 2.5x = 1.25x * 2, hence the following configuration must be set:

- MULTR = MULTG = MULTB = 128 * 1.25 = 160
- SHFR = SHFG = SHFB = 1

Configuration example for an additional white balance

An example of easy way to recenter the colors is using the Gray-world algorithm.

This algorithm assumes that a default scene has an inherent balance of its red, green, and blue pixels, so that the average values of the red, green, and blue components across the ideal picture are the same. If the measured averages are not the same, it means that there has been a distortion of the white balance, that must be corrected by amplifying or reducing the red, green, and blue strengths in the exposure module, thanks to the differentiated red vs. green vs. blue multipliers and exponents.

In practice this is the same approach used for the exposure compensation, but, instead of forcing the output luminance to a default target (64 in the example), user must force the red, green, and blue averages towards the same exposure target.

As an example:

- Exposure target is defined as 64.
- Measured red is 26: red factor is 64 / 26 = 2.5x, hence MULTR = 160, SHFR = 1.
- Measured green is 32: green factor is 64 / 32 = 2.0x, hence MULTG = 128, SHFG = 1.
- Measured blue is 10: blue factor is 64 / 10 = 6.4x, hence MULTB = 205, SHFB = 2.

39.6.8 Demosaicing

The module works on the pixel pipes, and implements a mid-quality conversion from raw Bayer 8 bpp to an RGB 24 bpp, with no implicit half-sizing: the output RGB pixel rate is the same as the input component rate.

The conversion targets real-time but medium quality purposes, like display or low-resolution software analysis. For higher quality conversions, it is recommended to convert the raw Bayer in software, with higher quality algorithms but limited to non-real-time, thus still-picture applications.

The algorithm takes the surrounding 3 x 3 components in input, applies linear heuristics to determine the most likely missing components, and generates an output RGB888 pixel.

To provide the 3 x 3 input matrix to the algorithm, the demosaicing module embeds a double line-buffer, 2 x 2688 components wide. As such, it limits the maximum width of a converted demosaicing buffer to 2688 components/pixels wide. Wider camera sensor resolution can be connected and goes through the first decimation stage in front of the demosaicing operation.

Note: This demosaicing conversion algorithm can create some artifacts, like Zipper. Such artifacts are much reduced by a later downsizing or even suppressed for a 2 x 2 downsizing.

Applications like display, video encode, or software analysis usually require such a downsize, so the artifacts become invisible.

The underlying heuristic filters try to extract and adapt to edges, horizontal and vertical lines, and lone pixels.

Functionality

- Input pixel format: from 1 (raw Bayer) to three (RGB/YUV) components on 8 bits
- Max input width of 2688 components
- Demosaicing conversion:
  - 2-line buffer
  - provides a 3 x 3 matrix to the algorithm + one trailing component (2 left)
  - Extraction and optimization for specific shapes:
    - > Edges (on left, right, top, bottom)
    - > Lines (horizontal and vertical)
    - > Peaking (lone pixels detection)
- Output pixel format: three RGB components on 8 bits.

Software configuration

Table 347. DCMIPP_P1DMCR bit function

Bit ID	Values	Comments
ENABLE	– 1: Active – 0: Bypasses R, G, B components.	If the camera sensor transmits an RGB or YUV frame and not raw Bayer, the demosaicing conversion must be disabled, by setting Enable = 0. All other parameters are ignored.
TYPE	– 0:RGGB – 1:GRBG – 2:GBRG – 3:BGGR	Default raw Bayer with three components (R, G, B)
EDGE	– 0: No edge detection, pure linear interpolation – 1 to 7: Relative algorithm strength (respectively 3, 4, 6, 8, 12, 16, 24)	-
LINEH	– 0: No horizontal line detection, pure linear interpolation – 1 to 7: Relative algorithm strength (respectively 3, 4, 6, 8, 12, 16, 24)	-
LINEV	– 0: No vertical line detection, pure linear interpolation – 1 to 7: Relative algorithm strength (respectively 3, 4, 6, 8, 12, 16, 24)	-
PEAK	– 0: No peak detection, pure linear interpolation – 1 to 7: Relative algorithm strength (respectively 3, 4, 6, 8, 12, 16, 24)	-

The four filters, EDGE, LINEH, LINEV, PEAK, improve the zoomed accuracy of the filtered picture (thus mostly luminance correctness, like no Zipper artifact), however at the expense of some chrominance blurring or distortion. Some scenes can require specific settings:

- Scenes where chrominance distortion is very visible (like a mostly black and white scene viewed by an RGB sensor) can require lower strength of the filters.
- On the opposite, scenes heavily loaded with geometric shapes (and thus source of artifacts) can require strengthening the filters.

Globally, 6/4/4/2 is a good compromise for the filter strength.

Configuration example

If the camera sensor is raw Bayer, with a grid with a red top-left pixel, the next configuration can be used:

- ENABLE = 1 to have the demosaicing active
- TYPE = 0 to have a top-red pixel, thus a RGGB grid
- EDGE, LINEH, LINEV, PEAK = (4, 4, 4, 4) as typical setting (so an internal strength of 8 for each parameter).

39.6.9 Color conversion

This module implements a color conversion with flexible (fully programmable coefficients), which can be used to map the following features:

- YUV-to-RGB: when the pixels of a YUV-sensor are used as RGB (display, GPU)
- RGB-to-YUV: when the pixels of a RGB-sensor are used by software (for example video encode)
- Tone change: to map special color references (sepia), emphasize some tone/colors
- Monochrome: to compute a monochrome from RGB/YUV with a specific linear mix
- Expansion: to handle a YUV (or RGB) expansion or compression
- Gamma: to emulate a gamma correction via a linear scale

Features

- Linear rotation of color space: 3 x 3 matrix + 3 x 1 column.
- Clamping optional on either [16 to 235] for Y, R, G, B and [16 to 240] for U, V.

Software configuration

Table 348. DCMIPP_PxCCyy (yy = R0, R1, G0, G1, B0, B1) bit function

Bit ID	Value	Comments
ENABLE	– 1: Activates the color conversion	-
CLAMP	– 1: Clamps the output pixel, when using a reduced dynamic	-
TYPE	– 1: Clamps as RGB – 0: Clamps as YUV	-

Table 348. DCMIPP_PxCCyy (yy = R0, R1, G0, G1, B0, B1) bit function (continued)

Bit ID	Value	Comments
Matrix3x3_Coeff	-	11 bits (sign, 2 int, 8 dec, with sign as complement to 2)
Added3x1_Coeff	-	10 bits (sign, 9 int, with sign as complement to 2)

Configuration example

Table 349 provides examples of the most used conversions:

- To / from RGB from/to YUV (as SDTV ITU-R BT.601 or HDTV BT.709).
- With a full (0 to 255) or a reduced (16 to 235) dynamic for Y and RGB, 16 to 240 for CbCr.
- Matrix fields are defined as XY bit fields in the color conversion coefficient registers, with X being the input R or G or B, and Y being the output R or G or B. For conversions to/from YUV, the R must be assumed as Cr (= V), G must be assumed as Y, and B must be assumed as Cb (= U).
- Coefficients are defined signed, and multiplied by 256 when there are decimals (the matrix coefficients). Negative values are on 11 bits: N11(val) = XOR (val, 0x7FF) + 1, on 10 bits for Added_Coeff: N10(val) = XOR (val, 0x3FF) + 1.

Table 349. Color conversion: examples of coefficients

Conversion	Output components	Input components				
RGB-255 to YUV601-255	-	R	G	B	Added	Clamping
	Cr	131	-110	-21	128	[0 to 255]
	Y	77	150	29	0	[0 to 255]
	Cb	-44	-87	131	128	[0 to 255]
YUV601-255 to RGB255	-	Cr	Y	Cb	Added	Clamping
	R	351	256	0	-175	[0 to 255]
	G	-179	256	-86	132	[0 to 255]
	B	0	256	443	-222	[0 to 255]
RGB-255 to YUV709-255	-	R	G	B	Added	Clamping
	Cr	131	-119	-12	128	[0 to 255]
	Y	55	183	18	0	[0 to 255]
	Cb	-30	-101	131	128	[0 to 255]
YUV709-255 to RGB255	-	Cr	Y	Cb	Added	Clamping
	R	394	256	0	-197	[0 to 255]
	G	-118	256	-47	82	[0 to 255]
	B	0	256	456	-232	[0 to 255]

Table 349. Color conversion: examples of coefficients (continued)

Conversion	Output components	Input components				
RGB-255 to YUV601-240	-	R	G	B	Added	Clamping
	Cr	112	-94	-18	128	[16 to 240]
	Y	66	129	25	16	[16 to 235]
	Cb	-38	-74	112	128	[16 to 240]
YUV601-240 to RGB255	-	Cr	Y	Cb	Added	Clamping
	R	409	298	0	-223	[0 to 255]
	G	-208	298	-100	135	[0 to 255]
	B	0	298	517	-277	[0 to 255]
RGB-255 to YUV709-240	-	R	G	B	Added	Clamping
	Cr	112	-102	-10	128	[16 to 240]
	Y	47	157	16	16	[16 to 235]
	Cb	-26	-87	112	128	[16 to 240]
YUV709-240 to RGB255	-	Cr	Y	Cb	Added	Clamping
	R	459	298	0	-248	[0 to 255]
	G	-137	298	-55	77	[0 to 255]
	B	0	298	541	-289	[0 to 255]
RGB-240 to YUV601-240	-	R	G	B	Added	Clamping
	Cr	131	-110	-21	128	[16 to 240]
	Y	77	150	29	0	[16 to 235]
	Cb	-44	-87	131	128	[16 to 240]
YUV601-240 to RGB240	-	Cr	Y	Cb	Added	Clamping
	R	351	256	0	-175	[16 to 235]
	G	-179	256	-86	132	[16 to 235]
	B	0	256	443	-222	[16 to 235]
RGB-240 to YUV709-240	-	R	G	B	Added	Clamping
	Cr	131	-119	-12	128	[16 to 240]
	Y	55	183	18	0	[16 to 235]
	Cb	-30	-101	131	128	[16 to 240]
YUV709-240 to RGB240	-	Cr	Y	Cb	Added	Clamping
	R	394	256	0	-197	[16 to 235]
	G	-118	256	-47	82	[16 to 235]
	B	0	256	465	-232	[16 to 235]

39.6.10 Contrast enhancement

This module ends the image processing pipe. It makes possible to enhance the contrast of the frame, by non-linear emphasis of the luminance of the pixels, and to provide more luminance quantification steps at highly used luminances (usually central grays), by reducing the amount of quantification steps less used by the picture (usually the few very dark and very light pixels).

Functionality

- Inputs RGB from the pipe
- Computes the luminance of the local pixel
- Extracts the desired luminance increase/decrease factor, in function of the input luminance, non-linearly (using a look-up table)
- Multiplies the input RGB with the amplification factor
- Outputs the clamped RGB components on 8 bits

Software configuration

Table 350. DCMIPP_P1CTCR1,2,3 bit function

Bit ID	Values	Comments
ENABLE	– 1: Active – 0: Bypasses R, G, B components	-
LUMx (x = 0 to 8, respectively inputs at 0, 32, 64, 96, 128, 160, 192, 224, 256)	0 to 63	Amplification factor: – LUMx = 8 halves the pixel luminance – LUMx = 16 does not modify the pixel luminance – LUMx = 32 doubles the pixel luminance

Configuration example

There are multiple approaches to handle contrasts.

It is possible to flatten the spectrum of luminance density (that is, to try have a similar amount of pixel per each segment of luminance), which is to fill each of the eight segments of luminance with 12.5% of the pixels of the frame.

An intermediate approach is to proceed with only the upper and lower half pixels: half of pixels must have a luminance below and another half above the mid-dynamic (128 for 8-bit components).

It is advised to use the statistics extraction module, sample the luminance after the exposure control (to avoid an impact on the exposure compensation), and sort it in the 12 available bins (refer to *Section 39.6.11*). From the extracted bins the user must linearly extrapolate the luminance of the median of the amount of pixels. The luminance of median pixel must be increased/decreased to be moved to the targeted output median luminance, which is recommended to be the median dynamic (128).

For instance, a 400 kpixels VGA picture can have 200 kpixels below a luminance of 64, and the other 200 kpixels above that value. The ideal luminance amplification factor can be set to 128 / 64 = 2x:

- pixels darker than 64 are lightened by a factor 2x
- pixels lighter than 64 are lightened a bit less, decreasingly
- pixel of luminance of 255 get an idle, 1x, factor.

The above assertions can be used to compute the LUMx values to configure:

- Ratio 2x from lum = 0 to 64:
  - LUM0 = LUM1 = LUM2 = 32.
- Decreasing ratio from 32 to 256, best exponentially decreasing:
  - LUM3 = 27
  - LUM4 = 23
  - LUM5 = 20
  - LUM6 = 18
  - LUM7 = 17
  - LUM8 = 16 (the maximum luminance, 256, must remain idle).

39.6.11 Statistics extraction

This module extracts several statistics across the frame, to provide inputs to the software image compensation algorithms, that configure feedbacks to the following implemented hardware modules: black level calibration, exposure compensation, white balance, and contrast enhancement.

The statistics are accumulated during a full frame, and sampled at the end of the frame, thus allows the software to have the full next frame to read the accumulated statistics, process them, define the adapted configuration for the hardware feedback modules, and to configure these modules.

Thanks the shadowing mechanism, the newly configuration parameters are active at the following frame, thus active at an average two frames after the statistics are accumulated. The statistics extraction block is considering all frames selected by means of the DCMIPP_P1FSCR configuration registers. Statistics block is running even when the frame is not supposed to be captured, respect to the DCMIPP_FCTCR configuration register.

The latency is not an issue, as the captured scene is expected to be almost static, and typical software compensation algorithms filter the statistic variations to avoid oscillations and visual artifacts (the low-pass filter typically has a one-second period).

Functionality

- Three statistics accumulators available, allowing three statistics extractions per frame.
  - Different types of statistics can be extracted frame over frame
- 32-bit accumulators, allowing to accumulate 8-bit samples over 16 Mpixels.
- Average mode, that extracts averages (accumulates component values)
  - Capability to discard extreme values (if value is <16 or >240, same for 32 or 64).
- Binning mode, which extracts presences (accumulates 256 per fit)
  - 12 available bins (values: <4, <8, <16, <32, <64, <128, and reverse to 256).
- Capability to define a sub-rectangle within the frame where statistics are extracted
  - Allows to focus the statistics on a specific region of interest.
- Raw Bayer supported, with the statistics using the specific locally available R, G, B components of the raw Bayer
- Statistics can be performed on either the input data (before any pixel modification) or after demosaicing, to benefit of a full RGB and thus luminance presence per pixel.

Software configuration

Table 351. DCMIPP_P1STyCR (y = 1, 2, 3), DCMIPP_P1STySR (y = 1, 2, 3), DCMIPP_P1STSTR and DCMIPP_P1STSZR bit function

Bit ID	Value	Comments
ENABLE	– 1: Active – 0: Bypasses R, G, B components	-
MODE	– 0: Average mode	Values of pixels are accumulated as-is.
	– 1: Bin mode	Values of pixels are used to see if the pixel fits one of the 12 bins
SRC (3 bits)	– (R, G, B, L)_up	– Up: refers to sampling after decimation. – L: luma (Y') computed as R / 4 + G / 2 + B / 4 when receiving RGB pixels, (R or G or B) when in raw Bayer mode.
	– (R, G, B, L)_down	– Down: refers to sampling after demosaicing (raw Bayer to RGB conversion), thus after both the exposure and white balance compensation, and when true RGB components (and thus luminance components) are available.

Table 351. DCMIPP_P1STyCR (y = 1, 2, 3), DCMIPP_P1STySR (y = 1, 2, 3), DCMIPP_P1STSTR and DCMIPP_P1STSZR bit function (continued)

Bit ID		Value	Comments
BINS (2 bits)	MODE = Average	– 0: AllPixels: no components are rejected – 1: NoExt16: components are rejected if < 16 or ≥ 240 – 2: NoExt32: components are rejected if < 32 or ≥ 224 – 3: NoExt64: components are rejected if < 64 or ≥ 192	Enables to reject extreme data, not accumulated for statistics. Additional statistics (with MODE = Bin) are needed to count the amount of selected pixels. BINS = 0 is likely to be the most used setting.
	MODE = Bin	– 0: Low: Accu0 counts if V < 4, Accu1 if V < 8, Accu2 if V < 16 – 1: LowMid: Accu0 counts if V < 32, Accu1 if V < 64, Accu2 if V < 128 – 2: UpMid: Accu0 counts if V ≥ 128, Accu1 if V ≥ 192, Accu2 if V ≥ 224 – 3: Up: Accu0 counts if V ≥ 240, Accu1 if V ≥ 248, Accu2 if V ≥ 252	Counts the amount of components V.
CROPEN (1 bit)		– 1: Statistics extracted from sub-rectangle – 0: Statistics extracted from whole frame	-
VSTART (12 bits)		-	Upper position of rectangle, from which statistics are extracted.
HSTART (12 bits)		-	Left position of the rectangle, from which statistics are extracted.
VSIZE (12 bits)		-	Height of the sub-rectangle, from which statistics are extracted.
HSIZE (12 bits)		-	Width of the sub-rectangle, from which statistics are extracted.
ACCU (24 bits)		-	Result of the 32-bit accumulation, re-sampled at frame boundary, and divided by 256 to fit the ACCU width of 24 bits.

Note: If HSIZE or VSIZE = 0 when the CROPEN bit is set, the block internally forces the “0” value at the maximum for the corresponding bit field (i.e. 0xFFE).

Table 352. Statistics extraction: collected data vs. modes

Mode	Average		Bin	
SRC	**RGB**	**Raw**	**RGB**	**Raw**
R, G, B up	012 = R / G / B	012 = RGB / 0 / 0	012 = R / G / B	012 = RGB / 0 / 0
L up	012 = R / 4 + G / 2 + B / 4	012 = RGB + 0 + 0	012 = R / 4 + G / 2 + B / 4	012 = RGB + 0 + 0
R, G, B down	012 = R / G / B	-	012 = R / G / B	-
L down	012 = R / 4 + G / 2 + B / 4	-	012 = R / 4 + G / 2 + B / 4	-

Configuration example for exposure control

For exposure control and white balance calibration, user would typically extract the average values of the luminance, and of the R, G, B. It can be done in a single frame capture campaign, with the following settings:

- ENABLE = 1 (enabled)
- MODE = 0 (average mode)
- SRC = 0 for red, 1 for G, 2 for B (upper values, before processing)
- BINS = 0 (all pixel values)
- CROPEN = 1 (1 below: to select a 300 x 220 close to top-left corner: 10,10)
- VSTART = 10
- HSTART = 10
- VSIZE= 220
- HSIZE= 300

After one frame (after a VSync), the Accu0, Accu1, Accu2 counters contain, respectively, the average red, green, blue accumulated across the 300 x 220 rectangle.

The value to expect from the Accu0, Accu1, Accu2 can be computed according to the following hypotheses:

- Amount of pixels: 300 x 220 (or full frame if statistics has CROPEN = 0)
- Amount of components:
  - Raw Bayer-red: 1/4, as one pixel every four in raw Bayer has a red component.
  - Raw Bayer-green: 1/2, as one pixel every two in raw Bayer has a green component.
  - Raw Bayer-blue: 1/4, as one pixel every four in raw Bayer has a blue component.
  - RGB: 1, as all pixels contain R, G, B components.
- Average value on 8 bits: 128 typically for a mid-gray picture (RGB: 128, 128, 128).
- Re-sampling: 1/256 (re-sampling accumulator to ACCU at end of frame).

The expected values in the accumulators are:

- Accu0 (red): 300 x 220 / 4 x 128 / 256 = 8250
- Accu1 (green): 300 x 220 / 2 x 128 / 256 = 16500
- Accu2 (blue): 300 x 220 / 4 x 128 / 256 = 8250

Configuration example for contrast enhancement

For contrast enhancement, the amount of pixels with a luminance inside a bin is typically extracted. As statistics are extracted 3 by 3, and as 12 bins can be accumulated, data are extracted during four consecutive frames, with three different bins for each frame.

The typical configuration is the following:

- ENABLE = 1 (enabled)
- MODE = 1 (bin mode)
- SRC = 3 (upper value, luminance selected)
- BINS = 0 (then, at next frame 1, then 2, then 3)
- CROPEN = 0 (no stat sub-rectangle)

After one frame (after a VSync), Accu0, Accu1, Accu2 counters contain the amount of pixels fitting bins <4, <8, <16, respectively.

The value to expect from the Accu0, Accu1, Accu2 can be computed by multiplying the following:

- Amount of pixels: 640 x 480 (assuming a full frame resolution of VGA).
- Amount of components: 1 (all pixels contribute to the luminance, even in raw Bayer).
- All pixels contribute to the statistics, both in raw Bayer and in RGB. In raw Bayer, a fake statistics is extracted, assuming a RGB-balanced pixel (Gray-world algorithm).
- Average value for bins: 256 (as in Bin mode, 256 is added for each fitting presence).
- Re-sampling: 1/256 (at end of frame)

The resulting value in accumulators are:

- Accu0 (bin < 4): 640 x 480 x 1 x 256 / 256 x 1% = 3072 (assuming 1% of pixels have lum < 4)
- Accu1 (bin < 8): 640 x 480 x 1 x 256 / 256 x 3% = 9000 (assuming 3% of pixels have lum < 8)
- Accu2 (bin < 16): 640 x 480 x 1 x 256 / 256 x 8% = 24000 (assuming 8% of pixels have lum < 16)

At the following frame, with BINS = 1, the statistics for luminance <32, <64, <128 are extracted, providing figures like 72k (25%), 144k (50%), and 210k (75%).

Such values indicate a median luminance ~64, as 50% of the pixels have a luminance <64.

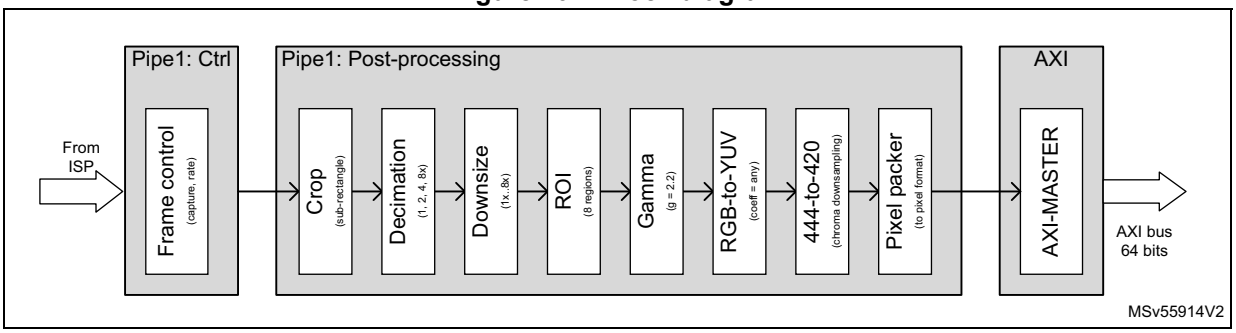
39.7 Pipe1 (post-processing part)

39.7.1 Overview

The post-processing Pipe1 (whose diagram is shown in *Figure 404*) is a pixel pipeline in which post-processing functions are integrated to relieve the software from very specific operations (time and memory consuming), like:

- Crop, to keep only the part of the picture required by the targeted application
- Decimation, as a cheap preprocessing before the downsize
- Downsize, to reduce the picture resolution (size reduction)
- ROI signaling
- Gamma correction before sending the data to the memory for further displaying operations
- YUV conversion with possible down-sampling option, to optimize the memory size needed to store the corresponding image or to match a YUV420 video encoder.

Figure 404. Block diagram



The applicative Pipe1 is the most featured pixel pipeline, even if the image processing functions can be shared with Pipe2 (refer to *Section 39.8.2*).

Pipe1 can be reconfigured from time to time. Some events can be used to trigger the software reconfiguration routines (refer to *Section 39.13.2*).

It is also possible to write into the shadow registers during the processing of a frame. The values are loaded into the physical register based on the following start of frame event.

39.7.2 Pixel 2D cropping

The pixel 2D cropping aims at extracting and forwarding a rectangle of pixels.

The cropping assigns an X and Y location to its input pixels, and selects a configured rectangle among the pixels, and forward the pixels of this rectangle farther in the pixel pipe. The pixels around the selected rectangle are discarded.

Functionality

- Counts lines thanks to HSync (if from parallel interface) or packets (if from CSI-2)
- Counts pixels inside a line
- Positive cropping (forwards a selected rectangle of pixels in the pipe)
- Cropping with 0 value for HSIZE[11:0] or VSIZE[11:0] forces the internal crop to get the maximum value on the parameter where the software sets to the 0 value

Software configuration

Table 353. DCMIPP_PxCRSTR and DCMIPP_PxCRSZR bit function

Bit ID	Values	Comments
ENABLE	– 1: Cropping active – 0: All data transmitted	-
HSTART, VSTART	0 to 4094	Top-left corner of the cropped rectangle.
HSIZE, VSIZE	0 to 4094	Width and height of the cropped rectangle. Can be larger than the received picture to grab everything.

If the cropped window is larger than the received window, only the received pixels are forwarded (no pixels are inserted to match the window size).

If the software sets 0 in one or both horizontal/vertical size bit fields (HSIZE, VSIZE), the crop forces internally the value to be at the maximum on the corresponding bit field.

Configuration example

To select a small rectangle (size 100 x 50), whose upper pixel is located at position (30, 10), provide the following setting:

- ENABLE = 1 to crop a sub-rectangle
- HSTART, VSTART = (30, 10) to start cropping at x = 30, y = 10 of the received picture
- HSIZE, VSIZE = (100, 50)

The most bottom-right selected pixels have a position (129, 59) from the received picture.

The frame forwarded is handled downstream in the pipe as a 100 x 50 frame.

39.7.3 Decimation pre-downsize

The decimation (prior to downsize) makes it possible to cheaply downsize a frame by a factor 2, 4, or 8x, in both horizontal and vertical directions.

The decimation is partially redundant with the downsize, that brings a higher quality (the decimation discards the data of the dropped pixels, while the downsize averages all pixels and does not lose information).

The decimation complements the downsize (limited to ratio of 8 x 8), to have a combined downsizing ratio of 64 x 64. It is recommended to have the largest ratio implemented in the higher quality downsize, and have needed the decimation complement to that. For instance, for a ratio 30x, is it recommended to have a ratio 4x in decimation, and a ratio 30 / 4 = 7.5 in the downsize.

Features

- Forwards one item (typically pixel) every 1, 2, 4, 8 pixels, independently vertical and horizontal.

Software configuration

Table 354. DCMIPP_PxDCCR bit function

Bit ID	Function	Value	Comments
HDEC[1:0]	Ratio (2 bits)	0	No decimation
		1	Ratio 1/2
		2	Ratio 1/4
		3	Ratio 1/8
VDEC[1:0]		0	No decimation
		1	Ratio 1/2
		2	Ratio 1/4
		3	Ratio 1/8

39.7.4 Downsize

This module implements a downsize based on box filtering, which allows ratios from 1x to 8x, independently, in both X and Y directions.

The box filter has properties close to a bilinear filter for small ratios (from 1x to 2x), but provides the advantage of avoiding some Moiré effect for higher ratios (from 2x to 8x) thanks to using all the samples.

Functionality

- Features:
  - Integer + decimal ratio, different vertically and horizontally.
  - Ratio: 1.0x to 8x
  - Limited to a maximum output buffer width of 2688 pixels due to the needed linebuffer.
- Operations:
  - Increments defined with 3.13 precision (<1 pixel error after maximum width of 4094)
  - Internal computations for filter defined with 3.3 precision
  - Accumulates skipped pixels to allow ratio up to 8x with saturation on 11 bits
  - Weighting of shared pixels, to allow good quality at ratio close to 1
  - Independent H filter, then V filter
  - Init phase: left of left-most pixel is aligned on input pixel (see below Pi0, Po0)

Software configuration

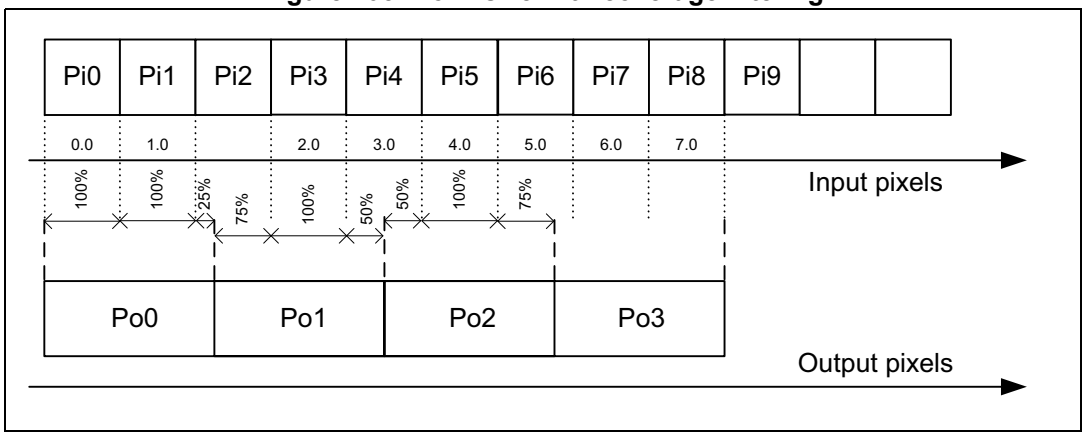
Table 355. DCMIPP_PxDSCR, DCMIPP_PxDSRTIOR, DCMIPP_PxDSSZR

Bit ID	Function	Values	Comments
ENABLE	-	– 1: Downsize active – 0: Transmission without downsize	-
HRATIO, VRATIO	Downsize ratio (respectively horizontally and vertically)	16-bit fields, used as unsigned 3 integer 13 decimal. Valid values are 8192 (= 1x) to 65535 (= 7.999x).	Key parameters that create output pixels. The optimal is computed as xRATIO = Floor (8192 * xRatioFloatingPoint), maximum value 65535 (x = H or V).
HDIV, VDIV	Inverted ratio, must be consistent with Ratio	10-bit fields, used as unsigned 10 decimal. Valid values are 128 (= inv8x) to 1023 (= inv1x).	A side parameter, impacts only the luminance of the output pixels. The optimal in function of xRATIO is xDIV = Floor((1024 * 8192 -1) / xRATIO) (x = H or V).
HSIZE, VSIZE	Final size after the downsize	-	The downsize filter clips pixels if the post-downsize size is not correct. The HSIZE and VSIZE are partially redundant vs. the cropping size divided by the downsize ratio. If VSIZE = 0 or HSIZE = 0, the filter is disabled in that dimension.

Figure 405 shows the principle of the computations:

- Po0 = (1.00 x Pi0 + 1.00 x Pi1 + 0.25 Pi2) / 2.25
- Po1 = (0.75 x Pi2 + 1.00 x Pi3 + 0.50 Pi4) / 2.25
- Po2 = (0.50 x Pi4 + 1.00 x Pi5 + 0.75 Pi6) / 2.25

Figure 405. Downsize with coverage filtering



Configuration example

- ENABLE = 1 to activate the downsize
- Downsize ratio 1x: HRATIO = VRATIO = 8192, HDIV = VDIV = 1023
- Downsize ratio 2x: HRATIO = VRATIO = 16384, HDIV = VDIV = 512
- Downsize ratio 3x: HRATIO = VRATIO = 24576, HDIV = VDIV = 341
- Downsize ratio 8x: HRATIO = VRATIO = 65535, HDIV = VDIV = 128

39.7.5 Regions of interest (ROIs)

Pixel pipes can defined up to eight different ROIs to highlight areas on a picture. The purpose of this feature is to draw one rectangle with user size defined for each enabled ROI.

There is no hardware processing within the defined area applied on the image. As the aim is to draw rectangles on a display, the ROI is supposed to be used when the data flow is RGB at the output of the image processing blocks.

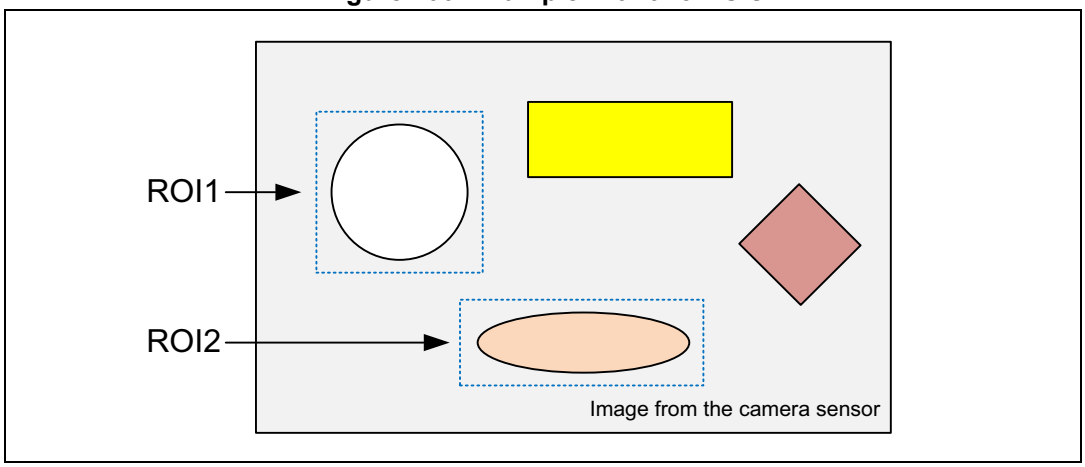
The line width is configurable commonly for each ROI thanks to bit-field ROILSZ[1:0] of the DCMIPP_PxCMRICR register. The color line is configurable per ROI to make easy visual distinction between each region of interest.

Each configuration is valid for a frame and it may be changed frame by frame by reprogramming the registers by software.

Each ROI can be enabled independently for displaying purpose.

Figure 406 shows a basic example of two ROIs displayed within a frame to isolate two flights (it is only visual information). The application can decide to crop the area to work on these two ROIs, by reprogramming the crop area for the next frames to make appropriate analysis linked to the application target.

Figure 406. Example with two ROIs



Software configuration

The first action is the configuration of DCMIPP_PxCMRICR, the common register to activate the ROI and to select the line width applied to all the active ROIs within a frame:

- ROIyEN (y = 1 to 8): each ROI can be enabled during the frame.
- ROILSZ: select the line width in pixel (1, 2, 4 or 8 pixels wide).

The second step is the configuration of each ROI, independently, through registers DCMIPP_PxRIyCR1 and DCMIPP_PxRIyCR2. The color line, among others, is set during this step.

- VSTART and HSTART bit fields allow to set the origin point of the rectangle for a given ROI. The origin is located on the top-left pixel, within the colored lines.
- VSIZE and HSIZE bit fields are used to set up the vertical and horizontal size in pixel for the area to draw. The size of the rectangle is measured the colored lines.
- CLR, CLG, CLB are used to select the color among 64. Each component has its two bits duplicated to 8-bit, to get an RGB888 at the output of the ROI block

39.7.6 Gamma conversion

This feature implements a gamma compression on each R, G, B component, using a static gamma exponent 2.2, to store gamma-compressed pictures, as defined in the sRGB standard.

Features

- The gamma is implemented as a 7-segment linear curve.
- It crosses the following way-points, in logarithm spacing:
  - (0,0)
  - (4,39)
  - (8,53)
  - (16,72)
  - (32,99)
  - (64,136)
  - (128,186)
  - (256,256)

Software configuration

ENABLE = 1 in the DCMIPP_PxGMCR registers activate the color conversion.

39.7.7 YUV conversion

This block is a replication of the one described in *Section 39.6.9*, used to convert RGB in YUV color format (basically, since the conversion YUV to RGB is assumed to be performed by the color conversion block into the image processing main function). Bit fields are the same of the color conversion module, only the register name and the memory mapping are different. The register functions are the same, the register names are DCMIPP_P1YUVxxx instead of DCMIPP_P1CCxx.

39.7.8 Chroma down-sampling

The Chroma down-sampling module reduces the chroma resolution from its arriving YUV444 format to the potential YUV420 output, and sends them to a different channel.

Features

- Downsizes in H first, if YUV420, with (0.5 + 0.5) filter
- Downsizes in V then, if YUV420, with (0.5 + 0.5) filter
- Memory: embeds one half-line buffer for each U and V

Software configuration

Configuration is indirectly taken from the pixel packing output format, located in the DCMIPP_PxPPCR register.

- The Chroma down-sampling is active when the output pixel format is YUV422-1, YUV422-2, YUV420-2, YUV420-3.

Output

- One to three channels: RGB/Y, UV (or U) and V.

39.7.9 Pixel packing

This module works on the dump pipe (Pipe0) and pixel pipes (Pipe1 .. N), setting the arriving pixels in memory words.

Features

- DBM addresses: double buffer mode with their associated address locations if the application need to swap memory address for one frame to the next one.
- Flexible pitch (line stride): width up to 4094 pixels, 32 bpp, potentially interleaved.
- Pixel format: all usual formats, as indicated in *Section 39.3.6* (Pipe1 allows semi-planar and full planar buffer, Pipe2 coplanar buffer only).
- Swap R-vs-B: swap the red vs. the blue components of the output pixel.
- Padding: MSB vs. LSB padding of 10/12/14 bit onto 16 bit. (such formats are supported only on the dump pipe).
- Header enable: allows to dump (or not) the CSI packet headers, to allow software an easier extraction of data in case of Interleaved CSI packets in the captured buffer.
- Multi-line event: event generated (for DMA and INT) at each multiple of sent lines.
- Multi-line address wrapping: allows to write the full frame across a ping-pong buffer, thanks to wrapping back to the BaseAddress after a power-of-2 amount of lines.

Note: Some features (for example padding) are available only on the dump pipe, while others (like swap R-vs-B) are available only on the pixel pipes.

Table 356. DCMIPP_PxPPCR bit function

Bit ID	Function	Value	Comments
M0A, M1A, M2A	Base address of pixel buffer in memory, 32-bit wide, aligned on 16 bytes	-	The line alignment of the pixel buffers in memory is assumed on 16 bytes. Therefore the buffer base address (i.e. M0A, M1A, M2A) must be a multiple of 16 bytes, and similarly for line PITCH.
PITCH	Address jump from line to line, up to 32 Kbyte - 1.	-	Aligned on 16 bytes.

Table 356. DCMIPP_PxPPCR bit function (continued)

Bit ID	Function	Value	Comments
SWAPRB	Swaps R-vs-B components (and U-vs-V, same operation if Y mapped on G) available on pixel pipe and not on dump pipe.	-	-
HEADEREN	-	– 1: Dumps headers	Feature present only in dump pipe.
PAD	MSB vs. LSB padding of the raw Bayer and monochrome to 16 bpp.	– 0 (align on LSB): backward compatibility with the former DCMI IP – 1 (align on MSB): easier use by software or GPU	Applies only on the dump pipe and for the 10/12/14 (not 8)-bit pixels (the pixel pipe convert pixels into 8 bpc, and padding is therefore not relevant).
FORMAT	Output pixel format	-	– Pipe0: not defined as default input format used for dump. It allows unique formats ByteData, ByteHeader and Mono/raw Bayer 8/10/12/14. – Pipe1: adds semi- and multi-planar formats: YUV 422-2, 420-2, 420-3 if Alpha or X is dumped, they are forced at 255). – Pipe2: supports coplanar pixel formats xRGB8888 (=xYUV444), RGB888 (=YUV444), RGB565, YUV422-1 and Y8.
LINEMULT	Periodicity of the line (HSync) interrupt and event, as power of 2 of LINEMULT configuration (i.e. every 1, 2, ..,128 lines).	-	-
LMAWE	Enable line multi address wrapping (see *LMAWM*)	-	-
LMAWM	Periodicity of the address wrapping to BaseAddress, period given as power of 2 of LMAWM configuration (i.e. every 1, 2, ..,128 lines).	-	-

Double buffer mode

The dump pipe uses an AXI master interface to dump the data from the internal FIFO to the external memory. In the application, it is possible to handle frame data swapping memory area frame by frame. The double buffer mode fills up this function. There are two memory address registers set to initialize the base addresses of these memories areas. Each start of captured frame event swaps the memory base address to handle double buffering mode. The double buffer mode can be used to allow post-processing on a buffer (frame buffer), while the other buffer is read to be displayed (display buffer).

Software configuration

Double buffering mode requires an activation and addresses configuration to define the two memory areas in which data are consecutively stored frame by frame at the output of the pipeline:

- DCMIPP_PxPPM0AR1 and DCMIPP_PxPPM0AR2 must be filled with the memory addresses associated to the double buffer memory locations.
- Bit DBM in the DCMIPP_PxPPCR registers is set to enable the double buffer mode.

Streaming across a ping-pong buffer

The DCMIPP has the capability (pixel pipes only) to stream pixels to a slave IP (like VENC or any) without storing the full frame in the main memory, by writing to a short intermediate multi-line buffer. It can do so by configuring the LINEMULT, LMAWE and LMAWM fields.

As an example, the slave IP is a JPG encoder that expects to be provided lines of macroblocks (16x16 pixels). The DCMIPP writes into a double (ping-pong) buffer, with each 16 lines high.

The streaming flow is as follows:

1. At a start of frame, the DCMIPP writes its pixels at an incremented address starting at the buffer BaseAddress (M1A/M2A).
2. After having written 16 lines, it emits a hardware Line event (dcmipp1_pXline_evt). At the reception of this hardware trigger, the slaved IP (JPG encoder here) starts to read the first 16-line buffer at the BaseAddress, and to encode its pixels.
3. The DCMIPP continues to write at the following addresses, thus after line 16.
4. After having written another 16 lines, the DCMIPP emits another Line event, to that the slaved IP can read and process the following 16 lines (at BaseAddress + Line16).
5. The DCMIPP wraps back to the BaseAddress, writes the following lines (32 to 47) starting at this BaseAddress, and then sends a trigger to the slave IP to process these new 16 lines.

The needed configuration is the following one:

- LINEMULT = 0x4, to send an event after having written each 16 lines
- LMAWM = 0x5, to wrap the address after having written each 32 lines
- LMAWE = 1, to enable the address wrapping of the address.

39.7.10 Overrun detection

This logic handles flow-control hazards: the DCMIPP is provided a continuous flow of data (via parallel or CSI2 interfaces) without any wait/hold capability, while downstream of DCMIPP, the access to memory can be temporarily stuck.

In some cases the internal FIFOs of the DCMIPP get full and pixels may be deleted. The overrun detection logic handles such hazard at best.

Features

- Software gets warned in case of an hazard:
  - Interrupt OVR is triggered at the first deletion of a pixel, repeated at every frame, as long as the overrun hazard is present.
  - It is recommended to skip / trash the impacted frame:
    > Overflowing pixels are deleted, so that when the access to memory resumes, the remaining of the line is discarded (not written out). The normal processing resumes at the start of the next line. Potentially, several full lines can get discarded.
- No Frame-to-Frame impact in case of an hazard:
  - VSync resets what is needed in the DCMIPP, so that any hazard on a previous frame has no impact at all on the pixels dumped in next frames.

Software configuration

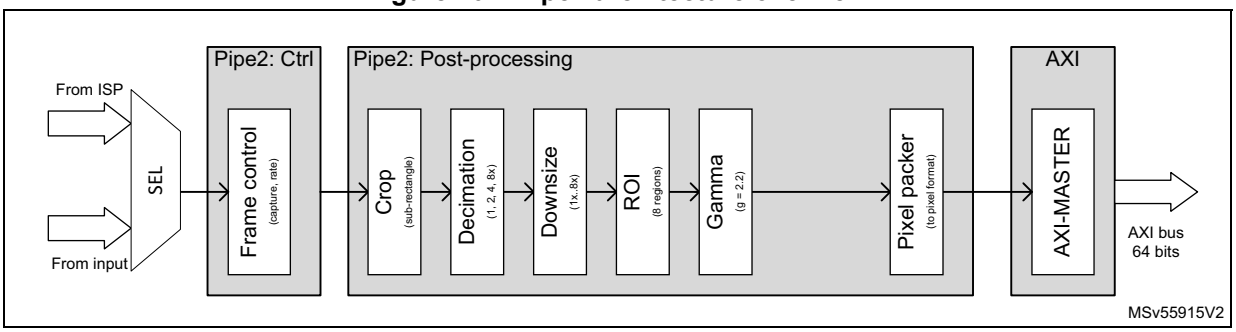
- Interrupt DCMIPP_PxSR.OVRF is warned in case of hazard.

39.8 Pipe2 (post-processing)

39.8.1 Overview

Pipe2, compared to Pipe1, is a lighter pixel pipeline offering no embedded image processing blocks. This pipe can work in standalone mode, or share the image processing functions with Pipe1 (refer to *Section 39.8.2*).

Figure 407. Pipe2 architecture overview



The main blocks (controller and post-processing general functions) of Pipe2 are almost the same as those of Pipe1. The flow selection and frame controller allow to configure Pipe2 behavior when it is used in standalone, meaning without image processing heritage from Pipe1. Refer to *Section 39.4.5* and *Section 39.4.6* for more details about the use cases and the configuration of the pipe.

The set of post-processing blocks is reduced compared to Pipe1, as neither the YUV conversion nor the down-sampling YUV444 to YUV420 are integrated in Pipe2. This pipe can have its own image crop area, its dedicated downsizing, and take benefit of the gamma correction, like Pipe1. Refer to *Section 39.7* and to its subsections for more details on how to use and configure by software these post-processing functions.

Software configuration

To use Pipe2 as a standalone pixel pipe, without any image processing functions shared with Pipe1, bit PIPEDIFF must be set in the DCMIPP_P1FSCR register. This bit has a meaning only when the flows can be differentiated, that is, when the camera connected and processed by the DCMIPP is a CSI2 camera interface.

In this non-shared mode, it is required that different flows are sent to Pipe1 and Pipe2, or a non-supported behavior can occur.

The flow to Pipe1 and the flow to Pipe2 may potentially be interleaved at packet level.

Pipe2 may be reconfigured time to time. Some events may be used to trigger the software reconfiguration routines (refer to *Section 39.13.2: Interrupts*).

It is possible also to write into the shadow registers within the processing of a frame. The values are loaded into the physical register based on the start of frame event.

39.8.2 Pipe1 and Pipe2 sharing image processing functions

Following the application, it may be interesting to share the image processing functions belonging to Pipe1 with Pipe2 when the dataflow is shared for both. One pipe may be used for display operation, and the other one used for statistics or software analysis. In this case PIPEDIFF bit of DCMIPP_P1FSCR register must be cleared.

If the CSI-2 module is connected to the DCMIPP, the VC DT and DTMODE of Pipe2 are the same as those configured into DCMIPP_P1FSCR register, making non relevant the values for these fields into the DCMIPP_P2FSCR register. However, bits CPTREQ and CPTMODE remain active into the DCMIPP_P2FCTCR register to control the flow inside Pipe2.

39.9 Application use cases

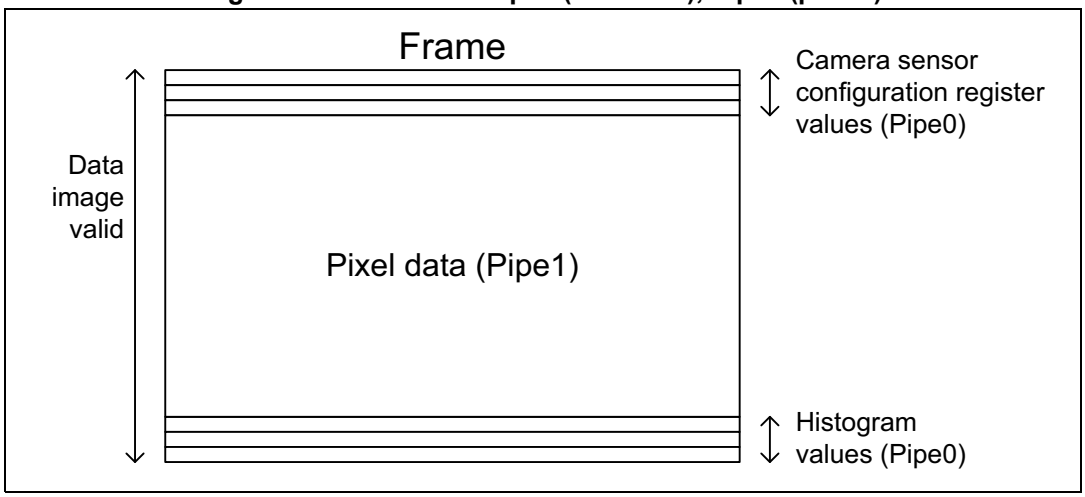
There are several application use cases that can fit the DCMIPP architecture, a few of them are described in this section.

39.9.1 Parallel interface camera sensor module

A parallel interface camera sensor can transfer images as well as statistics. It is easier to manage the software processing by splitting the statistics and the image data on different memory storages. Two pipes are used in such case, with Pipe0 extracting the statistics and Pipe1 dealing with the image data (whatever the data format supported by the DCMIPP). Both work in parallel on the same frame, frame by frame. Each output FIFO attached to the pipe conveys the data on the targeted memory for further processing.

The camera sensor module may use some lines of the frame to transfer statistics or sensor configuration data. These lines can be placed at the beginning and at the end of the frame (see *Figure 408*), even if there is no real standard.

Figure 408. Use case: Pipe0 (statistics), Pipe1 (pixels)



Software configuration

- Pipe0: select the crop area to extract statistics only
  - Configure HSTART[11:0] = 0 and VSTART[11:0] bits in the DCMIPP_P0SCSTR register from the line number corresponding to the first pixel data line (after first statistics line).
  - Configure HSIZE [11:0] (horizontal screen resolution in number of data 32-bit equivalent to the horizontal resolution of the screen in pixels) and VSIZE [11:0] to the value corresponding to the offset line number from the first pixel data area. These bits are in the DCMIPP_P0SCSZR register.
  - Set bit POSNEG of the DCMIPP_P0SCSZR register to reject data belonging to the rectangle defined, to keep the outside area of the rectangle, where the statistics are taken in this use case.
- Pipe1: Select the area to extract the statistics into DCMIPP_P1SRCR register, by configuring bit FIRSTLINEDEL[2:0] and LASTLINE[11:0] (it avoids to start the image processing on statistics data). Only the pixel data are forwarded, if needed, to the image processing blocks (for example demoisaicing, auto exposure, contrast enhancement) for further processing.

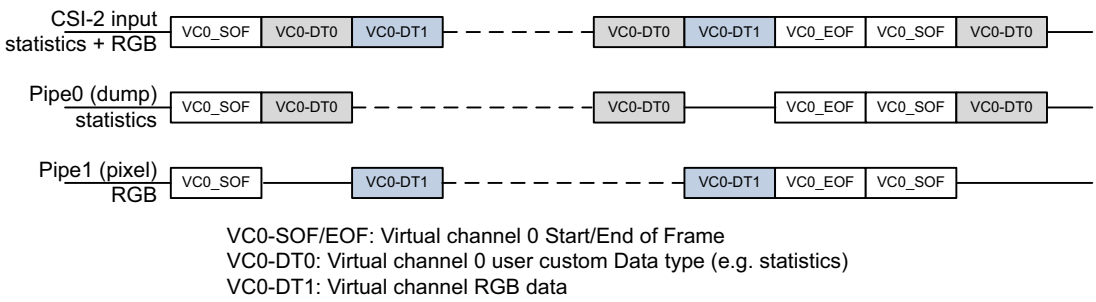
39.9.2 CSI2 camera sensor module

The CSI2 protocol allows, thanks to Datatypes, to make distinction between different data categories. The application may request to handle interlaced video as well as interleaved data flow like described below.

Single virtual channel involving two pipes (for example Pipe0 and Pipe1)

- Use case: the CSI provides two data flows in parallel, typically RGB (or raw Bayer) with its sideband statistics.
- Handling: the statistics are dumped by the dump pipe, while the pixels are sent to the pixel pipes.

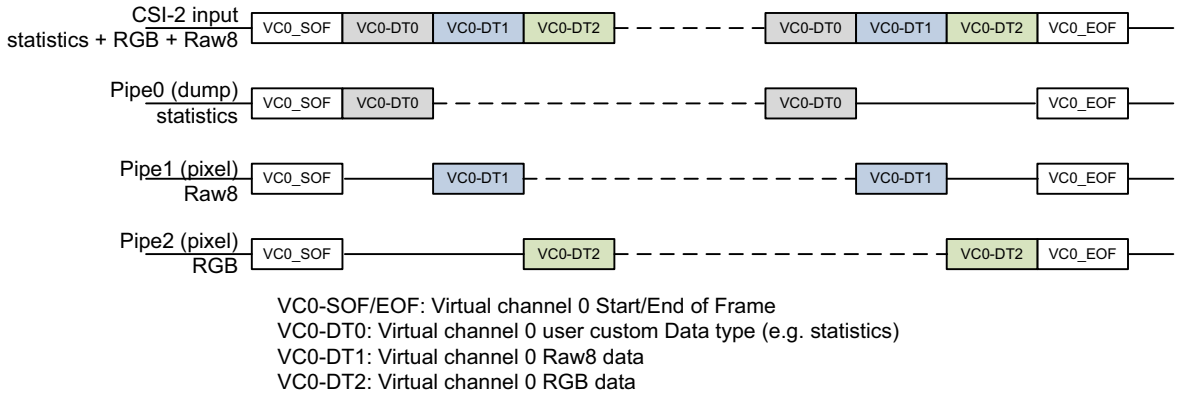
Figure 409. Statistics and RGB data for single virtual channel data flow



Single virtual channel involving three pipes (for example Pipe0, Pipe1 and Pipe2)

Another use case (*Figure 410*) employs different Datatypes in addition to the statistics within the CSI-2 flow based on a single virtual channel.

Figure 410. Three pipes within a single virtual channel in CSI2 mode



Interleaved packets (based on multiple virtual channels)

- Use case: the CSI provides two pixel flows in parallel, typically a JPG-compressed flow and an RGB888 or raw Bayer flow.
- Handling: the JPG is dumped by the dump pipe, while the pixels are sent to the pixel pipes.

Both Pipe1 and Pipe2 can process these pixels. The shared demosaicing mode can be used, hence both Pipe1 and Pipe2 benefit of the potential demosaicing conversion.

Figure 411. Handling interleaved packet

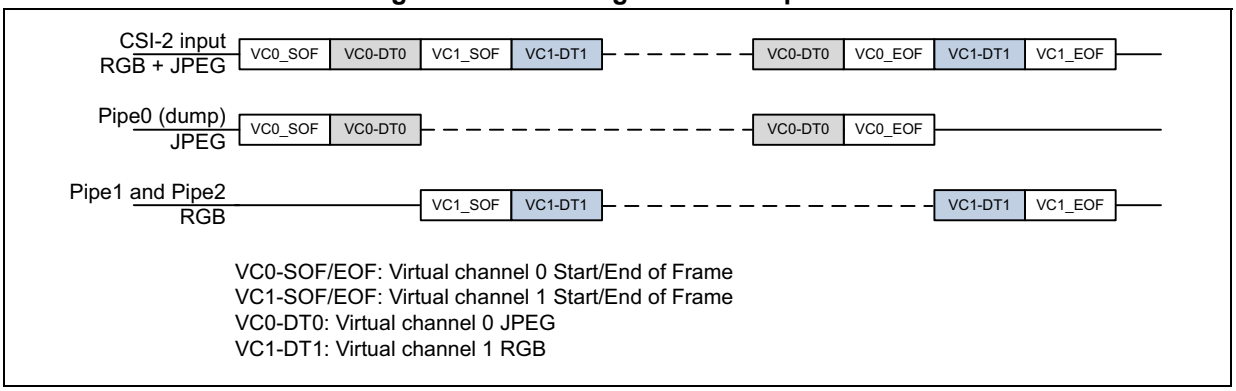
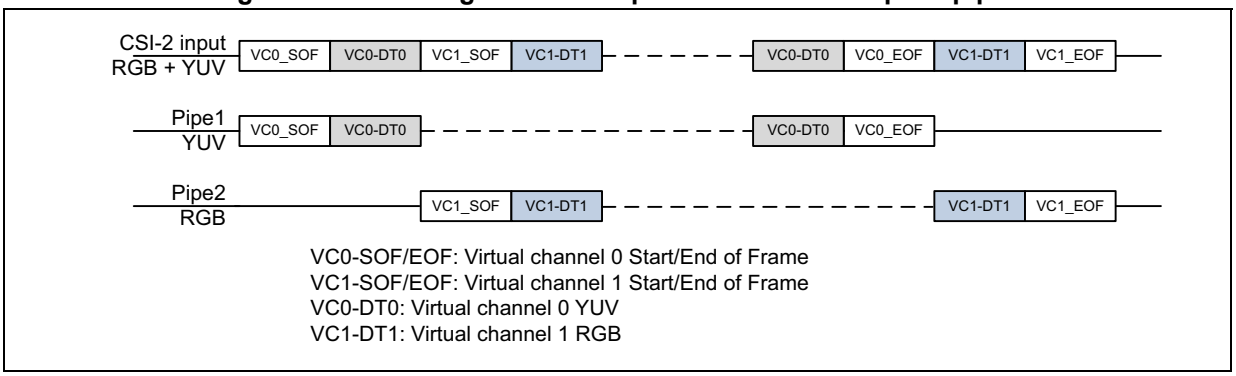


Figure 412. Handling interleaved packets across two pixel pipes



Interlaced video

- Use case: the CSI provides the odd and even frames of an interlaced video onto two different virtual channels (usually with a same data type).
- Handling: the flows of both the virtual channel of the odd field and the virtual channel of the even field are sent to the pixel pipes.
- Single pixel pipe may be configured if the image processing function is required by the application to handle both interlaced frame (refer to *Figure 413*). In such case, a pipe reconfiguration is needed frame by frame to change the virtual channel number and the address memory for the output buffer if the software distinctly keeps the odd and even frame buffers in their respective memory locations.
- Pipe1 and Pipe2 may be used if the application software does not need image processing functions (refer to *Figure 414*). Each pipe is configured with its own virtual channel and its own output buffer address in memory. The split of the odd and even frames is explicitly defined in a static way, avoiding any pipe reconfiguration between two frames.
- Interlaced video can be managed with a single virtual channel identifier, using a different data type to identify odd and even frames. In such case, the CSI2 bridge uses user Datatypes to distinguish between them.
- Limitation: the frames of the two virtual channels must be time-exclusive, by default the case for usual interlaced video sensors.

Figure 413. Interlaced video (based on single pixel pipe)

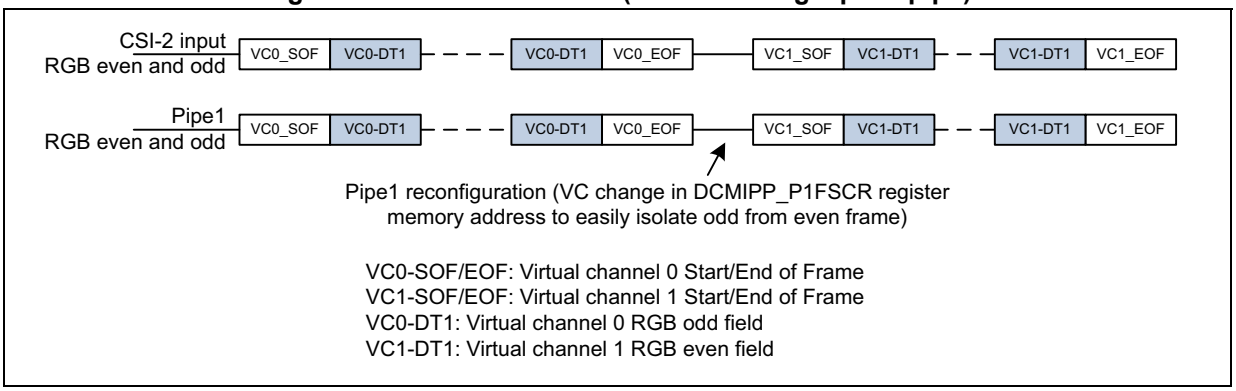
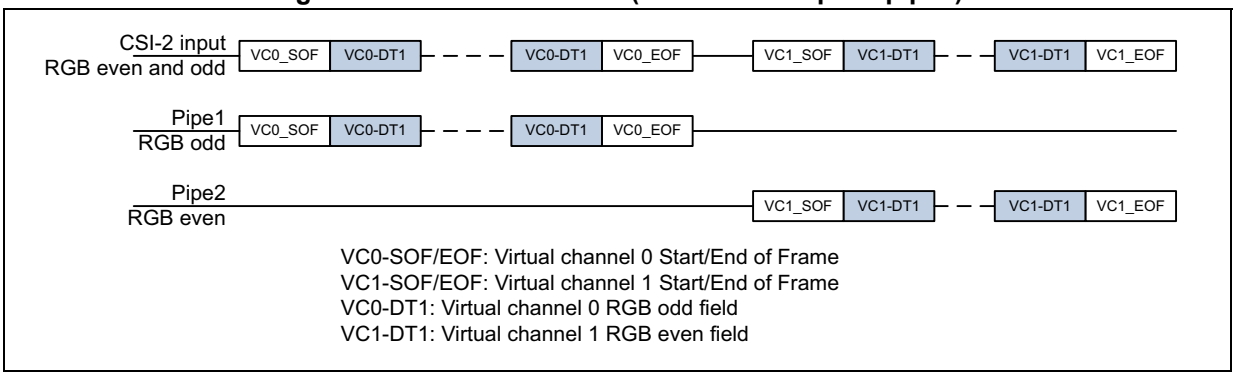


Figure 414. Interlaced video (based on two pixel pipes)



Software configuration

The following configuration must be set first into the static DCMIPP_PxFSCR and DCMIPP_CMCR registers:

- INSEL: Selects the input flow (parallel vs. CSI) in the DCMIPP_CMCR register
- DTMODE[1:0]: mode for CSI flow selection into that pipe:
  - 0: only flow VC/DTIDA is forwarded in the corresponding pipe.
  - 1: flows VC/DTIDA and/or VC/DTIDB are forwarded for Pipe0, and VC/DTIDA and VC/DTIDB for Pipe1.
  - 2: all flows, except the VCDTIDA or VCDTIDB, are forwarded in the pipe (this last setting, 3, is available only for Pipe0).
  - 3: all Datatypes from the selected virtual channel are forwarded in Pipe0. It is valid only for Pipe0 (the pixel formats that are not supported by Pixel pipes are suppressed).

  This bit field is not valid for Pipe2.
- DTIDA/DTIDB: Data type selection ID-A and ID-B for Pipe0 and Pipe1.
- DTIDA: Data type selection for the pixel Pipe2 when PIPEDIFF = 1.
- FDTFEN: Forces the value of the data type (value defined in FDTF[5:0]) to be processed by the pixel Pipe1 and Pipe2, based on the DTIDA/DTIDB detection in the flow selection block of the pipe. Not available for Pipe0.
- FDTF[5:0]: Data type value overwritten the one received by the CSI-2 physical node when equal to DTIDA/DTIDB. Not available for Pipe0.
- PIPEDIFF: Pipe2 to take same input as Pipe1 or differentiated:
  - 0: Pipe2 takes the same VC/DTIDx as well as the same FDTFEN and FDTF[5:0] configured into the DCMIPP_P1FSCR register.
  - 1: Pipe1 and Pipe2 take their own VC/DTID, FDTFEN and FDTF[5:0] values in the data flow.

After all the pipes have been configured, the processing can be activated:

- PIPEN: Set to 1 to start the frame control logic of the pipe. When reset at 0, it immediately resets the frame control logic, which results in immediately stopping an ongoing dump (CPTREQ = 0, CPTACT = 0), see *Figure 398*, *Figure 399*, *Figure 400*.
- CPTREQ: Set to 1 to activate the capture at the next start-of-frame in the DCMIPP_PxFCTCR register.

Note: To be cleanly sampled for the next frame, the VCIDA, VCIDB, DTIDA, and DTIDB fields must be updated after a VSync and after the first HSync.

Configuration example

The following list details five possible use cases, among others interlaced video and interleaved frames, and a 3D sensor (left + right eye):

- Default, with a parallel dump of statistics and processing of an image on both Pipe1 and Pipe2:
  - Image forwarded to the pixel Pipe1 and Pipe2, and processed, with the following settings:
    - Pipe1 / DTMODE = 0, to extract a single flow.
    - Pipe1 / VC / DTIDA = vv_dddddd, with VC and DTIDA matching the image flow
    - Pipe1 / PIPEDIFF = 0, to send the flow to both Pipe1 and Pipe2
  - Statistics are extracted and dumped on Pipe0 with the following config:
    - Pipe0 / DTMODE = 0, to extract a single flow.
    - Pipe0 / VC / DTIDA = vv_dddddd, with VC and DTIDA matching the statistics flow.
- Interlaced video, with the consecutive but exclusive odd and even (requiring the use of the image processing functions):
  - Interlaced video, if having differentiated VC and DT, is forwarded to the pixel Pipe1 and Pipe2, and processed, with the following settings:
    - Pipe1 / DTMODE = 0, to extract the VC /DTIDA flow
    - Pipe1 / VC/DTIDA = vv_dddddd, with VC and DTIDA matching the odd or the even field by reprogramming between two frames
    - Pipe1 / PIPEDIFF = 0, to send the flow to both Pipe1 and Pipe2
    - Pipe1 / Pipe2 / CPTMODE = 1, Snapshot mode
    - Pipe1 / Pipe2 / CPTREQ = 1 alternatively for Pipe1 and Pipe2, frame by frame
  - Any other (different) data can be potentially dumped with the following settings:
    - Pipe0 / DTMODE = 2, to extract all but flows A and B
    - Pipe0 / VCDTIDA = Pipe1 / VC / DTIDA corresponding to the odd field
    - Pipe0 / VCDTIDB = Pipe1 / VC / DTIDB corresponding to the even field

The software needs to reprogram frame to frame the DTIDx fields of the DCMIPP_P1FSCR register to select the data to process into Pipe1 or Pipe2. The software sets CPTMODE bit to work in Snapshot mode for Pipe1 and Pipe2 in the DCMIPP_P1FCTCR and DCMIPP_P2FCTCR registers. Alternatively, bit CPTREQ is set frame by frame in one pipe then in the other one to allow the acquisition of data on the allocated pipe. The data type identifying the odd or even fields needs to by updated into the DCMIPP_P1FSCR register only, as Pipe2 shares the image processing block of Pipe1.

- Interlaced video, with consecutive but exclusive odd and even (bypassing the image processing functions):
  - Interlaced video, if having differentiated VC and/or DT, is forwarded to pixel Pipe1 and Pipe2, and processed, with the following settings:
    - Pipe1 / DTMODE = 0, to extract the VC /DTIDA flow
    - Pipe1 / VC / DTIDA = vv_dddddd, with VC and DTIDA matching (for instance) the odd field data type
    - Pipe2 / DTMODE = 0, to extract the VC /DTIDA flow

> Pipe2 / VC / DTIDA = vv_dddddd, with VC and DTIDA matching (for instance) the even field data type
> Pipe1 / PIPEDIFF = 1, to consider Pipe1 and Pipe2 as totally independent
> Pipe1 / Pipe2 / CPTMODE = 0, Continuous grab mode
> Pipe1 / Pipe2 / CPTREQ = 1 to capture data corresponding to the VCDTIDA attached to the corresponding pipe

– Any other different data can be potentially dumped with the following settings:
> Pipe0 / DTMODE = 2, to extract all but flows A and B
> Pipe0 / VCDTIDA = Pipe0 / VC / DTIDA corresponding to the odd field
> Pipe0 / VCDTIDB = Pipe0 / VC / DTIDB corresponding to the even field

In this case the interlaced video must be differentiated with the data type and not with the virtual channel (since there is a single VC by pipe), if the application wants to dump data other than the odd and even frames on Pipe0.

- Interleaved flows, with a parallel dump of an RGB picture and a JPEG-encoded flow:
  – The RGB picture is extracted and forwarded to pixel Pipe1 to be processed, with the following settings:
  > Pipe1 / DTMODE = 0, to extract one flow, the RGB picture
  > Pipe1 / VC /DTIDA = vv_dddddd, with VC and DTIDA matching the RGB picture data type
  > Pipe1 / PIPEDIFF = 1, to send the RGB on Pipe1 and use Pipe2 for other purposes

  – The JPEG-encoded stream is transmitted under the CSI user-defined format, is extracted and sent to the pipe to be dumped, thanks to the following:
  > Pipe0 / DTMODE = 0, to extract one flow, the JPEG-encoded flow
  > Pipe0 / VCDTIDA = vv_dddddd, with VC and DTIDA matching the JPEG stream data type
  > Pipe0 / VCDTIDB = not used

  – Another pixel picture, like YUV, can be dumped using Pipe2, by overwriting fields:
  > Pipe2 / DTMODE = 0, to extract a second flow for the YUV picture
  > Pipe2 / VCDTIDA = vv_dddddd, with VC and DTIDA matching the YUV picture data type

- 3D sensor, with a parallel transmission of two RGB or YUV pictures: one with data type DT1 for the left eye, and one with data type DT2 for the right eye: the DCMIPP can process each eye on a different pipe (for example left eye on Pipe1 and right eye on Pipe2). Note that, as Pipe1 and Pipe2 are differentiated, the demosaicing is assigned to Pipe1 only and is not available for Pipe2. The left and right eye flows are similar (raw Bayer or RGB), hence they both need to be RGB and not raw Bayer:
  – The left eye RGB picture is extracted and forwarded to the pixel Pipe1:
  > Pipe1 / DTMODE = 0, to extract one flow, the RGB picture
  > Pipe1 / VC / DTIDA = vv_dddddd, with VC and DTIDA matching the left eye data type (DT1)
  > Pipe1 / PIPEDIFF = 1, to split and differentiate the flow between Pipe1 and Pipe2

  – The right eye RGB picture is extracted and forwarded to the pixel Pipe2:
  > Pipe2 / VC / DTIDA = vv_dddddd, with VC and DTIDA matching the right eye

data type (DT2)

  - Another flow can be dumped onto the dump pipe:
    - > Pipe0 / DTMODE = 0, to extract one flow
    - > Pipe0 / VCDTIDA = vv_dddddd, with VC and DTIDA matching the flow to dump for instance
- Sensors in 3D, with left and right image inside a unique global image:
  - If the left and right images are differentiated with a left and right part of a single image, that double image can be extracted and duplicated to both the pixel pipes, and there, cropped differently at input of Pipe1 and Pipe2 respectively
- Sensors with multiple DT:
  - The sensor is supported by sending all (or most) of the flows to the dump pipe, and dumped consecutively, inserting ahead of each dumped packet an header containing the VC, DT, and packet length. See *Section 39.7.9* for more details on the header format. One or two pixel flows can still be extracted and processed by Pipe1 and Pipe2, and potentially suppressed from the extraction of the data dump, thanks to DCMIPP_PxFSCR.DTMODE.

39.9.3 Force data type format from CSI-2 data flow (pixel pipes only)

Application may require to use a user data type in the CSI-2 data flow without specifying the format used for the pixel transfer from a camera bridge to the DCMIPP, thanks to CSI-2 interface physical link.

However, the pixel pipes need to know the data format to process them in a proper way along the pipeline. It is possible to force a data type by software thanks to bits FDTF[5:0] and FDTFEN[5:0]. Only the pixel pipes (Pipe1 and Pipe2) can benefit of this capability. Indeed, Pipe0 manages the data and not the pixel itself.

If FDTFEN bit is set in the DCMIPP_PxFSCR register, the data type set into DTIDA/DTIDB (coupled with DTMODE) is used to detect the data type to be overwritten by FDTF[5:0] value. The value used to force the data type must follow CSI-2 data type format specification to identify the CSI-2 normalized data format.

Note: The forced data type value is unique. This means that, if two data types have been selected to be propagated into Pipe1 (DTMODE = 1), both DTIDA and DTIDB detected in input of the pipe are considered by Pipe1 with the forced value FDTF[5:0] if bit FDTFEN is set.

If the virtual channel and data type values are identical between Pipe1 and Pipe2, the force value FDTF[5:0] needs to be aligned within these two pipes if the respective FDTFEN bits are set, otherwise the behavior is not guaranteed.

If PIPEDIFF = 1, any data type format linked to a raw format can be forced on Pipe2 since it is not sharing the demoisaicing block of Pipe1.

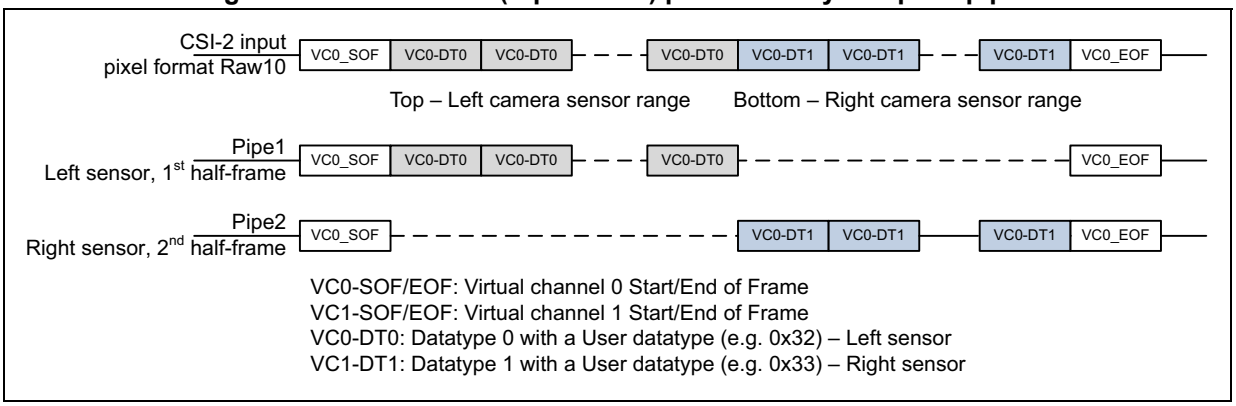
Typical application example: 3D sensors camera (top-bottom)

Some CSI-2 bridges concatenate data from left and right camera in a top-bottom approach. It means that within a frame, first half of the lines are the ones generated by the left camera, whereas the half last lines of a frame are attached to the right camera sensor. The bridge may typically make a distinction between lines coming from left camera and right camera by identifying the data type with a user data type value linked to a camera.

The user data type does not identify a pixel format, but rather a data linked to a camera. To help the pipeline process in a correct way the pixel, the software must inform the pipeline about the pixel format in input attached to the user Datatypes.

Let us imagine that the application has to separate pixels from the left camera from those of the right camera. Two pixel pipes can be configured for that, with, for instance, Pipe1 for left camera and Pipe2 dedicated to extract pixels from the right camera, as shown in *Figure 415*.

Figure 415. 3D sensors (top-bottom) processed by two pixel pipes



DCMIPP_P1FSCR configuration

- PIPEDIFF = 0, Pipe1 shares the demosaicing block with Pipe2 to transform, for instance, Raw10 in RGB888 for the output pixel format.
- VC = Virtual channel number
- DTMODE = 1, both Datatypes (0x32 and 0x33) selected to be propagated into Pipe1 through the demoisaicing, shared with Pipe2.
- DTIDA = 0x32, to let pixel data from the left camera sensor (identified by the user data type 0x32) to be processed by the demosaicing block, and further Pipe1 post processing stages (crop, downsize).
- DTIDB = 0x33, to let pixel data from the right camera sensor (identified by the user data type 0x33) to be processed by the demosaicing block, and further Pipe2 post processing stages (crop, downsize).
- FDTF[5:0] = 0x2B, to inform about the Raw10 data type normalized value in CSI-2 specification as input pixel format of the pipes.
- FDTFEN = 1, to force the user Datatypes DTIDA and DTIDB to be decoded as FDTF[5:0] input pixel format value.
- PIPEN = 1, to activate Pipe1.

Pipe1 crop registers (DCMIPP_P1CRSTR and DCMIPP_P1CRSZR) must be configured to crop only the first half lines of the frame corresponding to the pixels coming from the left camera sensor.

DCMIPP_P2FSCR configuration:

- VC / DTIDA / DTIDB / FDTF[5:0] / FDTFEN = don’t care since it is the same as the one defined in DCMIPP_P1FSCR (as PIPEDIFF = 0).
- PIPEN = 1 to activate Pipe2.

Pipe2 crop registers (DCMIPP_P2CRSTR and DCMIPP_P2CRSZR registers) must be configured to crop only the second half lines of the frame corresponding to the pixels coming from the right camera sensor.

39.10 Pixel format description

This section describes the pixel formats used in the parallel interface and in the pixel pipes and dump pipe. The CSI-2 pixel format is described in the CSI-2 v1.3 Standard.

The support for the pixel formats, per interface (parallel, CSI-2, pixel output, dump output) are summarized in *Table 357*, *Table 335*.

Note: In the following tables the components are represented with their symbolic color: red and Cr (= V) in red, blue and Cb (= U) in blue, green in green, and Y, monochrome, and all raw Bayer components in gray.

39.10.1 Parallel interface formats

This paragraph describes the input pixel format as supported by the parallel interface of the DCMIPP and the possible swap combinations.

Note: The parallel interface does not input specifically RGB444/RGB555 (and RGB666). However, a sensor with these output can connect them onto the DCMIPP, by selecting RGB565 (and RGB888), and by either connecting the missing bits with the MSB of the sensor output or by strapping them.

Table 357. Parallel interface input pixel formats

Index	IO pin		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
13	Byte	Cycle 1/1	-	-	-	-	-	-	-	-	7	6	5	4	3	2	1	0
5	Raw Bayer 8	Cycle 1/1	-	-	-	-	-	-	-	-	7	6	5	4	3	2	1	0
6	Raw Bayer 10	Cycle 1/1	-	-	-	-	-	-	9	8	7	6	5	4	3	2	1	0
7	Raw Bayer 12	Cycle 1/1	-	-	-	-	11	10	9	8	7	6	5	4	3	2	1	0
8	Raw Bayer 14	Cycle 1/1	-	-	13	12	11	10	9	8	7	6	5	4	3	2	1	0
9	Monochrome 8	Cycle 1/1	-	-	-	-	-	-	-	-	7	6	5	4	3	2	1	0
10	Monochrome 10	Cycle 1/1	-	-	-	-	-	-	9	8	7	6	5	4	3	2	1	0
11	Monochrome 12	Cycle 1/1	-	-	-	-	11	10	9	8	7	6	5	4	3	2	1	0
12	Monochrome 14	Cycle 1/1	-	-	13	12	11	10	9	8	7	6	5	4	3	2	1	0
2-1	RGB565 - 8 bits	Cycle 1/2	-	-	-	-	-	-	-	-	R4	R3	R2	R1	R0	G5	G4	G3
		Cycle 2/2	-	-	-	-	-	-	-	-	G2	G1	G0	B4	B3	B2	B1	B0
2-2	RGB565 - 16 bits	Cycle 1/1	R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0
2-2	PCB: Sensor444 to DCMIPP-RGB565	Cycle 1/1	R3	R2	R1	R0	R3	G3	G2	G1	G0	G3	G2	B3	B2	B1	B0	B3
2-2	PCB: Sensor555 to DCMIPP-RGB565	Cycle 1/1	R4	R3	R2	R1	R0	G4	G3	G2	G1	G0	G4	B4	B3	B2	B1	B0
4-1	RGB888 - 12 bits	Cycle 1/2	-	-	-	-	R7	R6	R5	R4	R3	R2	R1	R0	G7	G6	G5	G4
		Cycle 2/2	-	-	-	-	G3	G2	G1	G0	B7	B6	B5	B4	B3	B2	B1	B0
	YUV444 -12 bits	Cycle 1/2	-	-	-	-	V7	V6	V5	V4	V3	V2	V1	V0	Y7	Y6	Y5	Y4
		Cycle 2/2	-	-	-	-	Y3	Y2	Y1	Y0	U7	U6	U5	U4	U3	U2	U1	U0

Table 357. Parallel interface input pixel formats (continued)

Index	IO pin		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
4-2	RGB888 - 8 bits	Cycle 1/3	-	-	-	-	-	-	-	-	R7	R6	R5	R4	R3	R2	R1	R0
		Cycle 2/3	-	-	-	-	-	-	-	-	G7	G6	G5	G4	G3	G2	G1	G0
		Cycle 3/3	-	-	-	-	-	-	-	-	B7	B6	B5	B4	B3	B2	B1	B0
	YUV444 - 8 bits	Cycle 1/3	-	-	-	-	-	-	-	-	V7	V6	V5	V4	V3	V2	V1	V0
		Cycle 2/3	-	-	-	-	-	-	-	-	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
		Cycle 3/3	-	-	-	-	-	-	-	-	U7	U6	U5	U4	U3	U2	U1	U0
3	PCB: Sensor 666 to RGB888	Cycle 1/3	-	-	-	-	-	-	-	-	R5	R4	R3	R2	R1	R0	R5	R4
		Cycle 2/3	-	-	-	-	-	-	-	-	G5	G4	G3	G2	G1	G0	G5	G4
		Cycle 3/3	-	-	-	-	-	-	-	-	B5	B4	B3	B2	B1	B0	B5	B4
1-1	YUV422 - 8 bits (YUY'V)	Cycle 1/4	-	-	-	-	-	-	-	-	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
		Cycle 2/4	-	-	-	-	-	-	-	-	U7	U6	U5	U4	U3	U2	U1	U0
		Cycle 3/4	-	-	-	-	-	-	-	-	Y7'	Y6'	Y5'	Y4'	Y3'	Y2'	Y1'	Y0'
		Cycle 4/4	-	-	-	-	-	-	-	-	V7	V6	V5	V4	V3	V2	V1	V0
1-2	YUV422 - 16 bits (YUY'V)	Cycle 1/2	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	U7	U6	U5	U4	U3	U2	U1	U0
		Cycle 2/2	Y7'	Y6'	Y5'	Y4'	Y3'	Y2'	Y1'	Y0'	V7	V6	V5	V4	V3	V2	V1	V0

Table 358. Correspondence between index and DCMIPP_PRCR register values

Index	FORMAT[7:0]	EDM[2:0]
1-1	0x1E	0x00
1-2	0x1E	0x04
2	0x22	0x04
3	0x23	0x00
4-1	0x24	0x02
4-2	0x24	0x00
5	0x2A	0x00
6	0x2B	0x01
7	0x2C	0x02
8	0x2D	0x03
9	0x4A	0x00
10	0x4B	0x01
11	0x4C	0x02
12	0x4D	0x03
13	Other values	0x00

To adapt to non-standard sensors, the parallel interface is flexible and allows to swap bits, cycles, components, namely:

- Bits: LSB-vs.-MSB bits are swapped, across the whole 16-bit IO pins.
- Cycles: cycle 1/2 and cycle 2/2 are swapped. No impact for 1 or 3 cycles.
- Components: R-vs-B components are swapped (or V-vs-U if YUV).

Table 359 shows these permutations, based on the RGB565 format. It lists the sampled bits of cycles 1 and 2, on pins 0 to 7, and for each of them returns the assignation of the RGB output components.

Table 359. Parallel interface input pixel formats

IO pin		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NATIVE RGB565 8 bits	Cycle 1/2	-	-	-	-	-	-	-	-	R4	R3	R2	R1	R0	G5	G4	G3
	Cycle 2/2	-	-	-	-	-	-	-	-	G2	G1	G0	B4	B3	B2	B1	B0
SWAP LSB-MSB RGB565 - 8 bits	Cycle 1/2	G3	G4	G5	R0	R1	R2	R3	R4	-	-	-	-	-	-	-	-
	Cycle 2/2	B0	B1	B2	B3	B4	G0	G1	G2	-	-	-	-	-	-	-	-
SWAP cycles RGB565 8 bits	Cycle 1/2	-	-	-	-	-	-	-	-	G2	G1	G0	B4	B3	B2	B1	B0
	Cycle 2/2	-	-	-	-	-	-	-	-	R4	R3	R2	R1	R0	G5	G4	G3
SWAP R-vs-B RGB565 8 bits (not available on dump pipe)	Cycle 1/2	-	-	-	-	-	-	-	-	B4	B3	B2	B1	B0	G5	G4	G3
	Cycle 2/2	-	-	-	-	-	-	-	-	G2	G1	G0	R4	R3	R2	R1	R0

39.10.2 Pixel pipe formats

Table 360. Pixel pipe output pixel format(1)

Format	Memory bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	RGB888 (packed)	B7	B6	B5	B4	B3	B2	B1	B0	R7	R6	R5	R4	R3	R2	R1	R0	G7	G6	G5	G4	G3	G2	G1	G0	B7	B6	B5	B4	B3	B2	B1	B0
		Pixel N + 1								Pixel N + 0																							
		G7	G6	G5	G4	G3	G2	G1	G0	B7	B6	B5	B4	B3	B2	B1	B0	R7	R6	R5	R4	R3	R2	R1	R0	G7	G6	G5	G4	G3	G2	G1	G0
		Pixel N + 2																Pixel N + 1															
		R7	R6	R5	R4	R3	R2	R1	R0	G7	G6	G5	G4	G3	G2	G1	G0	B7	B6	B5	B4	B3	B2	B1	B0	R7	R6	R5	R4	R3	R2	R1	R0
		Pixel N + 3																								Pixel N + 2							
1	RGB565	R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0	R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0
		Pixel N + 1																Pixel N + 0															
2	xRGB888 (32 bpp)	x (Filling = 0xFF)								R7	R6	R5	R4	R3	R2	R1	R0	G7	G6	G5	G4	G3	G2	G1	G0	B7	B6	B5	B4	B3	B2	B1	B0
		Pixel N																															
3	RGBx888 (32 bpp)	R7	R6	R5	R4	R3	R2	R1	R0	G7	G6	G5	G4	G3	G2	G1	G0	B7	B6	B5	B4	B3	B2	B1	B0	x (filling = 0xFF)							
		Pixel N																															

Table 360. Pixel pipe output pixel format(1) (continued)

Format	Memory bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
4	Mono or raw Bayer (8 bpp)	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
		Pixel N + 3								Pixel N + 2								Pixel N + 1								Pixel N + 0							
5	AYUV888 (32 bpp, AYUV)	A (Filling = 0xFF)								Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	U7	U6	U5	U4	U3	U2	U1	U0	V7	V6	V5	V4	V3	V2	V1	V0
		Pixel N																															
6	YUV422-1 buffer (YUYV)	V7	V6	V5	V4	V3	V2	V1	V0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	U7	U6	U5	U4	U3	U2	U1	U0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
		Pixel N + 1																Pixel N + 0															
7	YUV422-2 buffers (-)	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
		Buffer0: Pixel N + 3								Pixel N + 2								Pixel N + 1								Pixel N + 0							
		V7	V6	V5	V4	V3	V2	V1	V0	U7	U6	U5	U4	U3	U2	U1	U0	V7	V6	V5	V4	V3	V2	V1	V0	U7	U6	U5	U4	U3	U2	U1	U0
		Buffer1: Pixel N + 2 and 3																Pixel N + 0 and 1															
8	YUV420-2 buffers (NV21, NV12 if SwapRB)	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
		Buffer0: Pixel N + 3								Pixel N + 2								Pixel N + 1								Pixel N + 0							
		V7	V6	V5	V4	V3	V2	V1	V0	U7	U6	U5	U4	U3	U2	U1	U0	V7	V6	V5	V4	V3	V2	V1	V0	U7	U6	U5	U4	U3	U2	U1	U0
		Buffer1: Pixel N + 2 and 3																Pixel N + 0 and 1															
9	YUV420-3 buffers (YV12)	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0
		Buffer0: Pixel N + 3								Pixel N + 2								Pixel N + 1								Pixel N + 0							
		U7	U6	U5	U4	U3	U2	U1	U0	U7	U6	U5	U4	U3	U2	U1	U0	U7	U6	U5	U4	U3	U2	U1	U0	U7	U6	U5	U4	U3	U2	U1	U0
		Buffer1: Pixel N + 6 and 7								Pixel N + 4 and 5								Pixel N + 2 and 3								Pixel N + 0 and 1							
		V7	V6	V5	V4	V3	V2	V1	V0	V7	V6	V5	V4	V3	V2	V1	V0	V7	V6	V5	V4	V3	V2	V1	V0	V7	V6	V5	V4	V3	V2	V1	V0
		Buffer2: Pixel N + 6 and 7								Pixel N + 4 and 5								Pixel N + 2 and 3								Pixel N + 0 and 1							
10	YUV422-1 buffer (UYVY)	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	V7	V6	V5	V4	V3	V2	V1	V0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	U7	U6	U5	U4	U3	U2	U1	U0
		Pixel N + 1																Pixel N + 0															

1. Format identifies the value set in bitfield FORMAT[3:0] of DCMIPP_PxPPCR registers (x = 1 to 2).

39.10.3 Dump pipe formats

The dump pipe, thanks to its capability to dump, supports more formats than the pixel pipe, namely:

- CSI-2 packet headers and data
- CSI-2 raw Bayer pixels on 6/7bits (that can be dumped as-is but are not supported by the pixel pipe and hence cannot be processed as pixels).
- Monochrome and raw Bayer pixels on 10/12/14 bits, padded on 16 bpp.

Table 361 details the support for these additional pixel formats. Other formats are assumed to be pixels with the same width. The other pixel formats used by the dump pipe are similar to the pixel formats used by the pixel pipe.

Table 361. Dump pipe OUTPUT pixel formats(1)

Bits	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4	3 2 1 0
CSI-2 header	Packet data (FrameNumber or ByteCount)	DataType-ID	VC-ID	FrameType
	Header 32 bits			
CSI-2 data	31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4	3 2 1 0
	Data 32 bits			
CSI-2 raw Bayer 6 bpp (words 0 to 2)	1 0 5 4 3 2 1 0 5 4 3 2 1 0	5 4 3 2 1 0 5 4 3 2 1 0	5 4 3 2 1 0	5 4 3 2 1 0
	PN + 5 / PN + 4	PN + 3	PN + 2	PN + 1 / PN + 0
	3 2 1 0 5 4 3 2 1 0 5 4 3 2	1 0 5 4 3 2 1 0 5 4 3 2 1 0	5 4 3 2 1 0 5 4 3 2 1 0	5 4 3 2
	PN + 11	PN + 10	PN + 9	PN + 8 / PN + 7 / PN + 6
	5 4 3 2 1 0 5 4 3 2 1 0 5 4	3 2 1 0 5 4 3 2 1 0 5 4 3 2	1 0 5 4 3 2 1 0 5 4 3 2 1 0	5 4
	PN + 16	PN + 15	PN + 14	PN + 13 / PN + 12 / PN +11
CSI-2 raw Bayer 7 bpp (words 0 to 7)	3 2 1 0 6 5 4 3 2 1 0 6 5 4	3 2 1 0 6 5 4 3 2 1 0 6 5 4	3 2 1 0 6 5 4 3 2 1 0 6 5 4	3 2 1 0 6 5 4 3 2 1 0 6 5 4
	PN + 4	PN + 3	PN + 2	PN + 1 / PN + 0
	31 6 5 4 3 2 1 0 6 5 4 3 2 1 0	6 5 4 3 2 1 0 6 5 4 3 2 1 0	6 5 4 3 2 1 0 6 5 4 3 2 1 0	6 5 4
	PN + 9 / PN + 8	PN + 7	PN + 6	PN + 5 / PN + 4
	4 3 2 1 0 6 5 4 3 2 1 0 6 5 4	3 2 1 0 6 5 4 3 2 1 0 6 5 4	3 2 1 0 6 5 4 3 2 1 0 6 5 4	3 2 1
	PN + 13	PN + 12	PN + 11	PN + 10 / PN + 9
	1 0 6 5 4 3 2 1 0 6 5 4 3 2 1 0	6 5 4 3 2 1 0 6 5 4 3 2 1 0	6 5 4 3 2 1 0 6 5 4 3 2 1 0	6 5
	PN + 18	PN + 17	PN + 16	PN + 15 / PN + 14 / PN +13
	5 4 3 2 1 0 6 5 4 3 2 1 0 6 5 4	3 2 1 0 6 5 4 3 2 1 0 6 5 4	3 2 1 0 6 5 4 3 2 1 0 6 5 4	3 2
	PN + 22	PN + 21	PN + 20	PN + 19 / PN + 18
	2 1 0 6 5 4 3 2 1 0 6 5 4 3 2 1 0 6 5 4 3 2 1 0 6 5 4 3 2 1 0 0			
	PN + 27	PN + 26	PN + 25	PN + 24 / PN + 23
	6 5 4 3 2 1 0 6 5 4 3 2 1 0 6 5 4 3 2 1 0 6 5 4 3 2 1 0 6 5 4 3			
	PN + 31	PN + 30	PN + 29	PN + 28 / PN + 27
Byte/ mono/raw 8 bpp	7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0	7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0	
	PN + 3	PN + 2	PN + 1	PN + 0

Table 361. Dump pipe OUTPUT pixel formats(1) (continued)

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PAD = 0 Mono/raw 10 bpp	0						9	8	7	6	5	4	3	2	1	0	0						9	8	7	6	5	4	3	2	1	0
	PN + 1																PN + 0															
PAD = 0 Mono/raw 12 bpp	0				11	10	9	8	7	6	5	4	3	2	1	0	0				11	10	9	8	7	6	5	4	3	2	1	0
	PN + 1																PN + 0															
PAD = 0 Mono/raw 14 bpp	0	13	12	11	10	9	8	7	6	5	4	3	2	1	0		0	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	PN + 1																PN + 0															
PAD = 1 Mono/raw 10 bpp	9	8	7	6	5	4	3	2	1	0	0	0	0	0	0	0	9	8	7	6	5	4	3	2	1	0	0	0	0	0	0	0
	PN + 1																PN + 0															
PAD = 1 Mono/raw 12 bpp	11	10	9	8	7	6	5	4	3	2	1	0	0	0	0	0	11	10	9	8	7	6	5	4	3	2	1	0	0	0	0	0
	PN + 1																PN + 0															
PAD = 1 Mono/raw 14 bpp	13	12	11	10	9	8	7	6	5	4	3	2	1	0	0	0	13	12	11	10	9	8	7	6	5	4	3	2	1	0	0	0
	PN + 1																PN + 0															
RGB 16 bpp	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PN + 1																PN + 0															
	R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0	R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0
RGB 24 bpp (words 0 to 2)	7	6	5	4	3	2	1	0	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PN + 1								PN + 0																							
	B7	B6	B5	B4	B3	B2	B1	B0	R7	R6	R5	R4	R3	R2	R1	R0	G7	G6	G5	G4	G3	G2	G1	G0	B7	B6	B5	B4	B3	B2	B1	B0
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
	PN + 2																PN + 1															
	G7	G6	G5	G4	G3	G2	G1	G0	B7	B6	B5	B4	B3	B2	B1	B0	R7	R6	R5	R4	R3	R2	R1	R0	G7	G6	G5	G4	G3	G2	G1	G0
	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	23	22	21	20	19	18	17	16
	PN + 3																						PN + 2									
	R7	R6	R5	R4	R3	R2	R1	R0	G7	G6	G5	G4	G3	G2	G1	G0	B7	B6	B5	B4	B3	B2	B1	B0	R7	R6	R5	R4	R3	R2	R1	R0
YUV422-1 (1 word)	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	PN + 1																PN + 0															
	Y'7	Y'6	Y'5	Y'4	Y'3	Y'2	Y'1	Y'0	V7	V6	V5	V4	V3	V2	V1	V0	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	U7	U6	U5	U4	U3	U2	U1	U0

1. PN indicates Pixel N.

39.10.4 AXI IP-Plug

The AXI IP-Plug dumps the data of the pipes (via the AXI) to the memory. It is unique, shared by all the pipes.

Features

- Pipe ID: AXI IP-Plug differentiates, via its ID, the source of the pipe. It allows to differentiate the QoS depending upon the clients.
- FIFOs: sized to handle up to 2 µs latencies (1 µs average round-trip + 1 µs memory slowdown), with, by default, at reset:
  a) Pipe0: 5 Gbps, thus up to 1.25 Kbytes
  b) Pipe1-Y/RGB: 200 Mpixel/s peak x 8bpp / 32bpp, thus up to 1.6 Kbytes (if ARGB)
  c) Pipe1-U: 200 Mpixel/s peak x 8 bpp, thus up to 400 bytes
  d) Pipe1-V: 200 Mpixel/s peak x 4 bpp, thus up to 200 bytes
  e) Pipe2: 200 Mpixel/s peak x 32 bpp, thus up to 1.6 Kbytes
- Burst length: 64 and 128 bytes for all the pipes.
- Outstanding capability: 16 outstandings, to cope with 2 GB/s maximum bandwidth on 128 bytes bursts.
- Bandwidth limiter (via an outstanding limitation per pipe): avoids a last-line dump that would uselessly overload the memory bandwidth.
- Unique AXI-QOS: unique QOS for the DCMIPP (without differentiation between pipes that must all work correctly with their real-time constraint).
- The total FIFO size is 5120 bytes.

Common software configuration

The AXI IP-Plug has a common configuration in the DCMIPP_IPGR1/2/3/8 registers.

The common registers are writable only when the IP-Plug is in Idle mode (see PSTART below to lock it, or below for an example).

- MEMORYPAGE: for efficient handling of DRAM: bursts do not cross this module.
- PSTART: request to freeze the plug, to reconfigure it safely.
- IDLE: status to mention the plug is frozen, to reconfigure it safely.
- IPPID[7:0], ARCHIID[4:0], REVID[4:0], DID[5:0]: version IDs of the IP-Plug.

Per-client software configuration

The AXI IP-Plug has a per-client configuration in the registers DCMIPP_IPCxR1/2/3.

The client registers are writable only when the IP-Plug is in Idle mode (see PSTART in common registers above to lock it, or below for an example).

The clients (1..n) are numbered following the sequence Pipe0..m. Inside each pipe, there are several potential clients when the planar feature is needed.

For instance, the DCMIPP with its three pipes (Pipe0, Pipe1, Pipe2) and with Pipe1 allowing a full planar capability (three buffers: Y, U, V), has the IP-Plug clients, numbered as follows:

- Client1 = Pipe0
- Client2 = Pipe1Y/RGB
- Client3 = Pipe1U
- Client4 = Pipe1V
- Client5 = Pipe2

The configurable per-client features are the following:

- OTR: outstanding transactions per client (Outstanding = OTR + 1)
- TRAFFIC[2:0]: burst size: best select 128 bytes, thus TRAFFIC = 4
- WLRU[3:0]: proportion of total bandwidth: WLRU = BWi / sum (all BW)
- SVC: system virtual channel, keep SVC = 0
- DPREGSTART: local base-address of start of FIFO for this client
- DPREGEND: local base-address of end of FIFO (included word) for this client

Configuration computations

IP-Plug common configuration:

- PSTART: set to 1, to put IP-Plug idle and configure its registers G1/3, R1/3
- IDLE: read and wait till 1, to get it idle
- MEMORYPAGE: 4, to optimize for memory pages of 256 bytes

IP-Plug per-client configuration:

- OTR (client): outstanding requests per client, among a total of 16 requests.
  - The provided value (OTR), is the outstanding amount minus 1.
- DPREG size (client): RAM shared among client, among a total of 640 x 64-bit words
  - DPREGSTART = 0 (or after another client location).
  - DPREGEND = 127 for ((127 - 0) + 1) x 8 B = 1 KB, as example if 1 KB is assigned to this client. The value to program is 127, as the word is included. DPREGEND is defined to include the last given word of memory, so that a FIFO of 128 words (starting at 0) has DPREGEND = 127.
- WLRU (client): Weighted Least Recently Used arbitration ratio, represents the bandwidth ratio for each client. The maximum ratio WLRU value is 16, meaning that the most demanding client will have 16 times the bandwidth of the least demanding client. The value to configure in DCMIPP_IPCxR2 is the WLRU ratio minus 1.

Each client is provided a proportion of the total FIFO size and of the total outstanding available transactions, in function of the local peak bandwidth. Indeed, the DCMIPP implements buffers to smooth the bandwidth, but they are not large enough to buffer full lines. The proportion of the peak bandwidth of each pipe is estimated as:

- PeakBandwidthProportion (= PBP) = OutputWidth (= OW) * PixelFormatSize (= PFS)
- OTR(client) = Total_OTR * PBP(client) / SumAll(PBP) - 1
- DPREG_Size(client) = Total_DPREG_Size * PBP(client) / SumAll(PBP)

As an example, an xRGB 720p display output, a VGA 422 video output (for software), and no statistics, results in:

- PBP(Display) = 1280 * 4 = 5120
- PBP(Video) = 640 * 2 = 1280
- OTR(Display) = 16 * 5120 / (5120 + 1280) - 1 = 12
- OTR(Video) = 16 * 1280 / (5120 + 1280) - 1 = 3
- DPREG_Size(Display) = 640 * 5120 / (5120 + 1280) = 512
- DPREG_Size(Video) = 640 * 1280 / (5120 + 1280) = 128

Video is assigned to Pipe1 (= Client2) and display to Pipe2 (Client5):

- IP-Plug common registers:
  - DCMIPP_IPGR1.MEMORYPAGE = 2 (default, 256 bytes)
  - DCMIPP_IPGR2.PSTART = 1 (locks IP-Plug for reconfiguration)
  - DCMIPP_IPGR3.IDLE to wait until = 1 (to wait before continuing)
- Video as Pipe1, statically mapped onto IP-Plug Client2:
  - DCMIPP_IPC2R1.OTR = 3
  - DCMIPP_IPC2R1.TRAFFIC = 4 (default)
  - DCMIPP_IPC2R3.DPREGSTART = 512
  - DCMIPP_IPC2R3.DPREGEND = 639
- Display as Pipe2, statically mapped onto IP-Plug Client5:
  - DCMIPP_IPC5R1.OTR = 11
  - DCMIPP_IPC5R1.TRAFFIC = 4 (default)
  - DCMIPP_IPC5R3.DPREGSTART = 0
  - DCMIPP_IPC5R3.DPREGEND = 511
- IP-Plug common registers:
  - DCMIPP_IPGR2.PSTART = 0 (unlocks IP-Plug after configuration)

Configuration example 1: Single RGB dump

It is the simplest but most usual configuration, with the DCMIPP dumping to a single xRGB buffer. It has thus a single pipe active, the dump is coplanar (no side U or V planes), and to emphasize, it does not dump separate statistics (with Pipe0), nor does use the second pixel pipe (Pipe2).

All the AXI capabilities are focused on the single active pipe, which gets all the FIFO space and outstanding capability of the DCMIPP. With this setting, a camera dump of 200 Mpixel/s peak on 32 bpp (800 MB/s bandwidth) benefits of:

- 5 Kbytes FIFO, allowing a memory peak slowdown 6 µs (a very high value).
- 16 outstandings of 128 bytes, allowing thus an average memory latency of 2.5 µs (a high value).
- IP-Plug common registers:
  - DCMIPP_IPGR1.MEMORYPAGE = 2 (default, 256 bytes)
  - DCMIPP_IPGR2.PSTART = 1 (locks IP-Plug for reconfig)
  - DCMIPP_IPGR3.IDLE to wait until = 1 (to wait before continuing)
- Single dump on Pipe1 (main pipe), statically mapped onto IP-Plug Client2:
  - DCMIPP_IPC2R1.OTR = 15 (maximum assigned here)
  - DCMIPP_IPC2R1.TRAFFIC = 4 (default, 128 bytes/burst)
  - DCMIPP_IPC2R2.WLRU = 0 (not used as single client)
  - DCMIPP_IPC2R2.SVCMAPPING = 0 (non-user)
  - DCMIPP_IPC2R3.DPREGSTART = 0 (single FIFO, start at lowest)
  - DCMIPP_IPC2R3.DPREGEND = 639 (single FIFO, end at highest)
- IP-Plug common registers:
  - DCMIPP_IPGR2.PSTART = 0 (unlocks IP-Plug after configuration)

Configuration example 2: All pipes active

This is the most flexible configuration of the DCMIPP: the five clients of the IP-Plug are configured to dump the maximum bandwidth they can receive).

Note that, as the bus capabilities are shared among the clients, that configuration is less resistant to memory latencies than the above configuration with only a single RGB dump.

The capabilities of the DCMIPP are shared among the clients with the following assumptions:

- Client1 for Pipe0: dump of raw Bayer pixels, padded to 16 bpp, BW = 2 / 12
- Client2 for Pipe1-Y: dump of ARGB pixels, on 32 bpp, BW = 4 / 12
- Client3 for Pipe1-UV: dump of UV components, on 8 bpp, BW = 1 / 12
- Client4 for Pipe1-V: dump of V component, on 4 bpp, BW = 1 / 12
- Client5 for Pipe2: dump of ARGB pixels, on 32 bpp, BW = 4 / 12

The total bandwidth proportion, when all pipes are used, is 12. As some configurations (like the outstanding capability) are divided in other granularities, Pipe1-Y/RGB has some advantages, sometimes is used alone (as in the above simpler use case).

With these settings, a camera dump of 200 Mpixel/s peak on 32 bpp (800 MB/s bandwidth) benefits, for an average pipe like Pipe2, of:

- 1.5 Kbytes FIFO, allowing a memory peak slowdown of ~2 µs
- Four outstandings of 128 bytes, allowing thus an average memory latency of 0.65 µs, (acceptable but not so large): the DCMIPP must get priority to access the memory.

It results in the following register configuration:

- IP-Plug common registers:
  - DCMIPP_IPGR1.MEMORYPAGE = 2 (default, 256 bytes)
  - DCMIPP_IPGR2.PSTART = 1 (locks IP-Plug for reconfiguration)
  - DCMIPP_IPGR3.IDLE to wait until = 1 (to wait before continuing)
- Client1 for Pipe0: (raw Bayer, 16 bpp) with 2 / 12 of the BW:
  - DCMIPP_IPC1R1.OTR = 1 (2 outstandings, among 16)
  - DCMIPP_IPC1R1.TRAFFIC = 4 (default, 128 bytes/burst)
  - DCMIPP_IPC1R2.WLRU = 1 (2 parts of BW, among 12)
  - DCMIPP_IPC1R2.SVCMAPPING = 0 (non-user)
  - DCMIPP_IPC1R3.DPREGSTART = 0 (96 parts of FIFO, among 640)
  - DCMIPP_IPC1R3.DPREGEND = 95 (96 parts of FIFO, among 640)
- Client2 for Pipe1-Y/RGB: (ARGB, 32 bpp) with 4 / 12 of the BW:
  - DCMIPP_IPC2R1.OTR = 5 (6 outstandings, among 16)
  - DCMIPP_IPC2R1.TRAFFIC = 4 (default, 128 bytes/burst)
  - DCMIPP_IPC2R2.WLRU = 3 (4 parts of BW, among 12)
  - DCMIPP_IPC2R2.SVCMAPPING = 0 (non-user)
  - DCMIPP_IPC2R3.DPREGSTART = 96 (224 parts of FIFO, among 640)
  - DCMIPP_IPC2R3.DPREGEND = 319 (224 parts of FIFO, among 640)
- Client3 for Pipe1-UV: (UV, 16 bpp) with 2 / 12 of the BW:
  - DCMIPP_IPC3R1.OTR = 1 (2 outstandings, among 16)
  - DCMIPP_IPC3R1.TRAFFIC = 4 (default, 128 bytes/burst)
  - DCMIPP_IPC3R2.WLRU = 0 (1 part of BW, among 12)
  - DCMIPP_IPC3R2.SVCMAPPING = 0 (non-user)
  - DCMIPP_IPC3R3.DPREGSTART = 320 (64 parts of FIFO, among 640)
  - DCMIPP_IPC3R3.DPREGEND = 383 (64 parts of FIFO, among 640)
- Client4 for Pipe1-V: (V, 8 bpp) with 1 / 12 of the BW:
  - DCMIPP_IPC4R1.OTR = 1 (2 outstandings, among 16)
  - DCMIPP_IPC4R1.TRAFFIC = 4 (default, 128 bytes/burst)
  - DCMIPP_IPC4R2.WLRU = 0 (1 part of BW, among 12)
  - DCMIPP_IPC4R2.SVCMAPPING = 0 (non-user)
  - DCMIPP_IPC4R3.DPREGSTART = 384 (32 parts of FIFO, among 640)
  - DCMIPP_IPC4R3.DPREGEND = 415 (32 parts of FIFO, among 640)
- Client5 for Pipe2: (ARGB, 32 bpp) with 4 / 12 of the BW:
  - DCMIPP_IPC5R1.OTR = 3 (4 outstandings, among 16)
  - DCMIPP_IPC5R1.TRAFFIC = 4 (default, 128 bytes/burst)
  - DCMIPP_IPC5R2.WLRU = 3 (4 parts of BW, among 12)

  - DCMIPP_IPC5R2.SVCMAPPING = 0 (non-user)
  - DCMIPP_IPC5R3.DPREGSTART = 416 (224 parts of FIFO, among 640)
  - DCMIPP_IPC5R3.DPREGEND = 639 (224 parts of FIFO, among 640)
- IP-Plug common registers:
  - DCMIPP_IPGR2.PSTART = 0 (unlocks IP-Plug after configuration)

39.11 Shadow registers

A dump and/or pixel pipe must have the capability to be reconfigured without stopping it. Some registers can be written in the middle of a frame without impacting the actual frame acquisition. The values are stored into some shadow registers before being written in their corresponding physical registers, based on a trigger event like described in *Table 362*.

Each physical register in the register map has its address increased by 0x200 respect to its own shadow register address.

Table 362. Shadow and physical registers

Shadow register	Physical register	Trigger event
DCMIPP_P0FSCR(1)	DCMIPP_P0CFSCR	Vsync boundaries
DCMIPP_P0FCTCR	DCMIPP_P0CFCTCR	Vsync boundaries
DCMIPP_P0SCSTR	DCMIPP_P0CSCSTR	Vsync boundaries
DCMIPP_P0SCSZR	DCMIPP_P0CSCSZR	Vsync boundaries
DCMIPP_P0PPCR	DCMIPP_P0CPPCR	Vsync boundaries
DCMIPP_P0PPM0AR1	DCMIPP_P0CPPxM0AR1	Vsync boundaries
DCMIPP_P0PPM0AR2	DCMIPP_P0CPPM0AR2	Vsync boundaries
DCMIPP_P1FSCR(1)	DCMIPP_P1CFSCR	Vsync boundaries
DCMIPP_P1FCTCR	DCMIPP_P1CFCTCR	Vsync boundaries
DCMIPP_P1BPRCR	DCMIPP_P1CBPRCR	Vsync boundaries
DCMIPP_P1BLCCR	DCMIPP_P1CBLCCR	Vsync boundaries
DCMIPP_P1EXCR1	DCMIPP_P1CEXCR1	Vsync boundaries
DCMIPP_P1EXCR2	DCMIPP_P1CEXCR2	Vsync boundaries
DCMIPP_P1STxCR (x = 1 to 3)	DCMIPP_P1CSTxCR (x = 1 to 3)	Vsync boundaries
DCMIPP_P1STSTR	DCMIPP_P1CSTSTR	Vsync boundaries
DCMIPP_P1STSZR	DCMIPP_P1CSTSZR	Vsync boundaries
DCMIPP_P1CTCRx (x = 1 to 3)	DCMIPP_P1CCTCRx (x = 1 to 3)	Vsync boundaries
DCMIPP_P1CC1CR	DCMIPP_P1CCC1CR	Vsync boundaries
DCMIPP_P1CC1RRx (x = 1 to 2)	DCMIPP_P1CCC1RRx (x = 1 to 2)	Vsync boundaries
DCMIPP_P1CC1GRx (x = 1 to 2)	DCMIPP_P1CCC1GRx (x = 1 to 2)	Vsync boundaries
DCMIPP_P1CC1BRx (x = 1 to 2)	DCMIPP_P1CCC1BRx (x = 1 to 2)	Vsync boundaries
DCMIPP_P1CRSTR	DCMIPP_P1CCRSTR	Vsync boundaries

Table 362. Shadow and physical registers (continued)

Shadow register	Physical register	Trigger event
DCMIPP_P1CRSZR	DCMIPP_P1CCRSZR	Vsync boundaries
DCMIPP_P1DCCR	DCMIPP_P1CDCCR	Vsync boundaries
DCMIPP_P1DSCR	DCMIPP_P1CDSCR	Vsync boundaries
DCMIPP_P1DSRTIOR	DCMIPP_P1CDSRTIOR	Vsync boundaries
DCMIPP_P1DSSZR	DCMIPP_P1CDSSZR	Vsync boundaries
DCMIPP_P1PPCR	DCMIPP_P1CPPCR	Vsync boundaries
DCMIPP_P1CMRICR	DCMIPP_P1CCMRICR	Vsync boundaries
DCMIPP_P1RIxCR1 (x = 1 to 8)	DCMIPP_P1CRIxCR1 (x = 1 to 8)	Vsync boundaries
DCMIPP_P1RIxCR2 (x = 1 to 8)	DCMIPP_P1CRIxCR2 (x = 1 to 8)	Vsync boundaries
DCMIPP_P1PPM0AR1	DCMIPP_P1CPPM0AR1	Vsync boundaries
DCMIPP_P1PPM0AR2	DCMIPP_P1CPPM0AR2	Vsync boundaries
DCMIPP_P1PPM0PR	DCMIPP_P1CPPM0PR	Vsync boundaries
DCMIPP_P1PPM1AR1	DCMIPP_P1CPPM1AR1	Vsync boundaries
DCMIPP_P1PPM1AR2	DCMIPP_P1CPPM1AR2	Vsync boundaries
DCMIPP_P1PPM1PR	DCMIPP_P1CPPM1PR	Vsync boundaries
DCMIPP_P1PPM2AR1	DCMIPP_P1CPPM2AR1	Vsync boundaries
DCMIPP_P1PPM2AR2	DCMIPP_P1CPPM2AR2	Vsync boundaries
DCMIPP_P2FSCR(1)	DCMIPP_P2CFSCR	Vsync boundaries
DCMIPP_P2FCTCR	DCMIPP_P2CFCTCR	Vsync boundaries
DCMIPP_P2CRSTR	DCMIPP_P2CCRSTR	Vsync boundaries
DCMIPP_P2CRSZR	DCMIPP_P2CCRSZR	Vsync boundaries
DCMIPP_P2DSCR	DCMIPP_P2CDSCR	Vsync boundaries
DCMIPP_P2DCCR	DCMIPP_P2CDCCR	Vsync boundaries
DCMIPP_P2DSRTIOR	DCMIPP_P2CDSRTIOR	Vsync boundaries
DCMIPP_P2DSSZR	DCMIPP_P2CDSSZR	Vsync boundaries
DCMIPP_P2PPCR	DCMIPP_P2CPPCR	Vsync boundaries
DCMIPP_P2CMRICR	DCMIPP_P2CCMRICR	Vsync boundaries
DCMIPP_P2RIxCR1 (x= 1 to 8)	DCMIPP_P2CRIxCR1 (x= 1 to 8)	Vsync boundaries
DCMIPP_P2RIxCR2 (x = 1 to 8)	DCMIPP_P2CRIxCR2 (x = 1 to 8)	Vsync boundaries
DCMIPP_P2PPM0AR1	DCMIPP_P2CPPM0AR1	Vsync boundaries
DCMIPP_P2PPM0AR2	DCMIPP_P2CPPM0AR2	Vsync boundaries
DCMIPP_P2PPM0PR	DCMIPP_P2CPPM0PR	Vsync boundaries

1. Bit PIPEN of this register is not shadowed.

It is mandatory to refresh the shadow registers within a frame to prepare the context to be ready for the next start of frame, to avoid any unpredictable behavior during the acquisition. The software must then react to the following interrupt sources to launch the shadow registers accesses:

- At VSYNC interrupt (FrameStart at start of pipe) an interrupt is generated, and the software can update the shadow registers loaded into the physical registers at the end of the current frame (so active for the next frame). The software has an entire frame period to complete the update.
- At LINE interrupt line selection can be used also to trigger the software to update the physical registers with the new frame context to consider. The application code must ensure that the update takes place before the end of frame event (current frame). If not, the context can be partially considered from the next frame, dealing with inconsistent settings.

In any case, the software must ensure that the shadow registers are updated before the end of the current frame, to be active from the next frame. Updates started and not completed before the end of current frame result in inconsistent frame acquisition for the next frame (mixing old and new configurations).

It is recommended to change non-shadowed registers when the corresponding pipe is disabled and in an idle state (refer to *Section 39.4.8: Pipe deactivation*).

39.12 DCMIPP low power modes

This section describes the behavior of the DCMIPP versus the modes listed in *Table 363*.

Table 363. DCMIPP low power modes

Mode	Description
Stop	Peripheral content is kept. It is recommended to follow the pipe(s) disabling procedure before entering it.
Standby	Powered-down. The peripherals must be reinitialized after exiting this mode.
Sleep	No effect. DCMIPP is still working in this mode, peripheral interrupts cause the device to exit it.

39.13 DCMIPP interrupts

This section describes the DCMIPP interrupts, and more globally, the features that are related to real-time, namely:

- Free-running: DCMIPP can run without CPU involvement
- Shadow registers: relieve timing constrains from the software
- Interrupts: source of interrupt
- Event pins: to drive external IPs, like a DMA

39.13.1 Free-running DCMIPP

After a camera flow has been configured and established, it is able to run permanently, without any software involvement. The following features are available:

- Permanent pipe activation, for Pipe0, 1, 2: when set, the respective pipe dumps frame data, for an unlimited amount of frames.
  - DCMIPP_PxFCTCR.CPTMODE = 0
  - DCMIPP_PxSR.CPTACT = 1
  - DCMIPP_PxFCTCR.FRATE = xx (depending on desired frame subsampling rate)
- Double buffered frames: the DCMIPP can be defined with two BaseAddress pointers for the output buffers:
  - Software can set up the two BaseAddress pointers to let the hardware reacting to the VSYNC event to change from one buffer BaseAddress to the other one, and this alternatively.

39.13.2 Interrupts

The DCMIPP handles five interrupts pins, the fifth being the OR of the first four.

A register set is associated to each interrupt to provide the following functionality:

- Unmasked events: reports an event that occurred and has not yet been cleared
- Masks: to mask unwanted events
- Masked events: not readable, any active unmasked bit triggers the interrupt pin
- Clear register: write to clear any active event

The interrupt pins are active when high: they go high when triggered by an unmasked event, and remain high until all the masked events of that interrupt are cleared.

The interrupts lines and registers are:

1. Parallel interface, with the next interrupt events handled:
   - ERR: bad embedded synchronization detected.
     - > Unexpected behavior in the parallel interface, usually not triggered if no issues.
     - > It is due to an unexpected sensor behavior, or to a transmission error.
2. Pipe0 (dump), with the next interrupt events handled:
   - OVR: data overflow: the memory was too slow vs. received pixels.
     - > Unexpected behavior in Pipe0, usually not triggered if no issues.
     - > Same recommendation as for above OVR: skip the impacted frame.
   - LIMIT: received volume is larger than the maximum allowed dump volume.
     - > Unexpected behavior in Pipe0, usually not triggered if no issues.
     - > The transmitted flow is too long. More space must be allocated in memory to store the transmission of a next frame.
   - VSYNC: main interrupt, where most software can sit.
     - > Trigger: at VSync (mid blanking), permanent (even if no dump active).
     - > Typically to reconfigure slowly the pipes (at least shadow registers) for next frame
     - > Typically to trigger usage of the previously captured frame.
   - FRAME: secondary interrupt, as backup for fast-software
     - > Trigger: after last data dump of this pipe, inactive if no capture has occurred.
     - > Typically to quickly reconfigure non-shadow registers (during only vertical blanking)
     - > Typically to trigger usage of the previously captured frame.
   - LINE: interruption for stripe-based operators trigger: after every 1/2/4/8/16/32/64/128 dumped lines and last line. It is measured and extracted at end of pipe, close to output to memory.
     - > Typically to trigger fast/reactive software or hardware stripe-based operators.

Note: The LINE flag (LINEF bit) is set at the end of frame, to trigger out software event when the frame height is not a multiple of the selected number of lines for which the LINEF event is triggered. The last part of the frame can, in such case, have a lower number of lines than the selection. The event/interrupt is generated even if the number of lines is incomplete. The software must consider that this last event has a lower than expected number of lines when reaching the end of frame (FRAME event).

3. Pipe1 (main pipe)
   - Same events handled as Pipe0: OVR, VSYNC, FRAME, LINE, but no LIMIT as only pixels are captured after a potential crop.
4. Pipe2 (ancillary pipe)
   - Same events handled as Pipe1: OVR, VSYNC, FRAME, LINE.
5. Common interrupt: groups the events of the above interrupts in a single register set and drives its own interrupt line.
   - The fifth interrupt is designed for systems where a single driver handles the whole DCMIPP: a single access to this interrupt status allows to retrieve the status of the whole DCMIPP.
   - The first four interrupts are designed for systems where a driver handles a pipe

and (potentially) another one drives the parallel interface. In that case, each driver handles its own individual interrupt register set, without need for semaphores to arbitrate conflicting accesses.

There is also a common interrupt that can be generated if enabled, when a transfer error is detected during an AXI transfer from the IP-Plug to memories. Error flag ATXERR in the DCMIPP_CMSR2 register is set and an interrupt can be generated to inform the software about this error status. There is no specific hardware action linked to this error, it is up to software to handle the transfer error situation.

On multiple-pipe architecture, the error informs that at least one AXI client has a transfer error, but does not identify from which client.

The DCMIPP interrupts are summarized in *Table 364*. An event that generates an interrupt does it at two locations:

– once in the common interrupt register set (when driven by a single CPU)
– once in the local interrupt register set (when driven by multiple CPUs).

Table 364 provides the local (register and bit) and global (bit only, as all bits are in the same DCMIPP_CMSR2 register) locations.

Table 364. DCMIPP interrupts

Interrupt event	Local register	Local bit	Common bit in DCMIPP _CMSR2	Event flag/interrupt clearing method	Interrupt enable control bit local/common register
Synchronization error in parallel interface	DCMIPP _PRSR	ERRF	PRERRF	Write CERRF = 1 / CPRERRF = 1	ERRIE/PRERRFIE
AXI transfer error	-	-	ATXERR	Write CATXERRF = 1	ATXERRIE
Overflow in Pipe0	DCMIPP _P0SR	OVRF	P0OVRF	Write COVRF = 1 / CP0OVRF = 1	OVRIE/P0OVRIE
Limit violation in Pipe0		LIMITF	P0LIMITF	Write CLIMITF = 1 / CP0LIMITF = 1	LIMITIE/P0LIMITIE
Frame start (VSYNC) in Pipe0		VSYNC	P0VSYNC	Write CVSYNCF = 1 / CP0VCSYNCF = 1	VSYNCIE/P0VSYNCIE
Frame end (FRAME) in Pipe0		FRAMEF	P0FRAMEF	Write CFRAMEF = 1 / CP0FRAMEF = 1	FRAMEIE/P0FRAMIE
Multi-line (LINE) in Pipe0		LINEF	P0LINEF	Write CLINEF = 1 / CP0LINEF = 1	LINEFIE/P0LINEFIE
Overflow in Pipe1	DCMIPP _P1SR	OVRF	P1OVRF	Write COVRF = 1 / CP1OVRF = 1	OVRIE/P1OVRIE
Frame start (VSYNC) in Pipe1		VSYNC	P1VSYNC	Write CLIMITF = 1 / CP1LIMITF = 1	LIMITIE/P1LIMITIE
Frame end (FRAME) in Pipe1		FRAMEF	P1FRAMEF	Write CVSYNCF = 1 / CP1VCSYNCF = 1	VSYNCIE/P1VSYNCIE
Multi-line (LINE) in Pipe1		LINEF	P1LINEF	Write CFRAMEF = 1 / CP1FRAMEF = 1	FRAMEIE/P1FRAMIE

Table 364. DCMIPP interrupts (continued)

Interrupt event	Local register	Local bit	Common bit in DCMIPP _CMSR2	Event flag/interrupt clearing method	Interrupt enable control bit local/common register
Overflow in Pipe2	DCMIPP _P2SR	OVRF	P2OVRF	Write COVRF = 1 / CP2OVRF = 1	OVRIE/P2OVRIE
Frame start (VSYNC) in Pipe2		VSYNC	P2VSYNC	Write CLIMITF = 1 / CP2LIMITF = 1	LIMITIE/P2LIMITIE
Frame end (FRAME) in Pipe2		FRAMEF	P2FRAMEF	Write CVSYNCF = 1 / CP2VCSYNCF = 1	VSYNCIE/P2VSYNCIE
Multi-line (LINE) in Pipe2		LINEF	P2LINEF	Write CFRAMEF = 1 / CP2FRAMEF = 1	FRAMEIE/P2FRAMIE

39.13.3 Event pins

Event pins are exposed to let an ancillary HW IP, like a central DMA, to synchronize with them. The event pins are the unmasked synchronization events (VSYNC, FRAME, LINE, HSYNC) of the pipes.

An internal events generates a pulse (15 APB cycles long) on its pin.

Table 365. Event connection

Pipe	Event name	Internal signal
Pipe0	HSYNC	dcmipp_p0_hsync_evt
	VSYNC	dcmipp_p0_vsync_evt
	Frame	dcmipp_p0_frameend_evt
	Line	dcmipp_p0_lineend_evt
Pipe1	HSYNC	dcmipp_p1_hsync_evt
	VSYNC	dcmipp_p1_vsync_evt
	Frame	dcmipp_p1_frameend_evt
	Line	dcmipp_p1_lineend_evt
Pipe2	HSYNC	dcmipp_p2_hsync_evt
	VSYNC	dcmipp_p2_vsync_evt
	Frame	dcmipp_p2_frameend_evt
	Line	dcmipp_p2_lineend_evt

39.14 DCMIPP registers

The registers are split into groups with a same prefix, as shown in *Table 366*.

Table 366. DCMIPP registers organization

Register name	Offset	Function
DCMIPP_IP	0x000	IP-Plug registers
DCMIPP_PR	0x100	Parallel interface
DCMIPP_CM	0x200	Common registers
DCMIPP_P0	0x500	Pipe0 (dump pipe) with shadow registers
	0x700	Pipe0 (dump pipe) with physical registers
DCMIPP_P1	0x800	Pipe1 image processing with shadow registers
	0x900	Pipe1 post-processing with shadow registers
	0xA00	Pipe1 image processing with physical registers
	0xB00	Pipe1 post-processing with physical registers
DCMIPP_P2	0xC00	Pipe2 image processing with shadow registers
	0xD00	Pipe2 post-processing with shadow registers
	0xE00	Pipe2 image processing with physical registers
	0xF00	Pipe2 post-processing with physical registers
DCMIPP_	0xFF0	IP version and configuration (without prefix)

39.14.1 DCMIPP IP-Plug global register 1 (DCMIPP_IPGR1)

Address offset: 0x000

Reset value: 0x0000 0002

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

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **QOS_MODE:** Quality of service
Set of functions enabling to build and configure an architecture meeting bandwidth and latency requirements.

Bits 23:3 Reserved, must be kept at reset value.

Bits 2:0 **MEMORYPAGE[2:0]:** Memory page size, as power of 2 of 64-byte units:
0x0: 64 bytes
0x1: 128 bytes
0x2: 256 bytes
0x3: 512 bytes
0x4: 1K bytes
0x5: 2K bytes
0x6: 4K bytes
0x7: 8K bytes

39.14.2 DCMIPP IP-Plug global register 2 (DCMIPP_IPGR2)

Address offset: 0x004

Reset value: 0x0000 0000

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

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

Bit 0 **PSTART:** Request to lock the IP-Plug, to allow reconfiguration.
0: No lock requested, IP-Plug runs on demand by background HW.
1: Lock requested: IP-Plug freezes shortly (see IDLE bit when lock is active).
PSTART must be reset to 0 after configuration is completed, to restart the IP-Plug.

39.14.3 DCMIPP IP-Plug global register 3 (DCMIPP_IPGR3)

Address offset: 0x008

Reset value: 0x0000 0001

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

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

Bit 0 **IDLE:** Status of IP-Plug
0: IP-Plug is running (on demand by background HW)
1: IP-Plug is currently locked and can be reconfigured
IDLE is set after a request by setting PSTART at 1, and reset by resetting PSTART at 0.

39.14.4 DCMIPP IP-Plug identification register (DCMIPP_IPGR8)

Address offset: 0x01C

Reset value: 0xAA04 0314

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

Bits 31:24 **IPPID[7:0]:** IP identifier (0xAA)

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:16 **ARCHIID[4:0]:** Architecture identifier (0x04)

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **REVID[4:0]:** Revision identifier (0x03)

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **DID[5:0]:** Division identifier (0x14)

39.14.5 DCMIPP IP-Plug Clientx register 1 (DCMIPP_IPCxR1)

Address offset: 0x020 + 0x10 * (x - 1), (x = 1 to 5)

Reset value: 0x0000 0004

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

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **OTR[3:0]:** Maximum outstanding transactions
0: Disabled. No outstanding transaction limitation (except via FIFO size)
1: Maximum two outstanding transactions ongoing.
...
15: Maximum 16 outstanding transactions ongoing.

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 **TRAFFIC[2:0]:** Burst size as power of 2 of 8-byte units
0x0: 8 bytes
0x1: 16 bytes
0x2: 32 bytes
0x3: 64 bytes
0x4: 128 bytes
Other values: Reserved

39.14.6 DCMIPP IP-Plug Clientx register 2 (DCMIPP_IPCxR2)

Address offset: 0x024 + 0x10 * (x - 1), (x = 1 to 5)

Reset value: 0x0001 0000

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

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **WLRU[3:0]:** Ratio for WLRU[3:0] arbitration
A client gets a portion of the total bandwidth = Ratio(client) / Sum(all ratios)
0x0: Ratio part = 1
0x1: Ratio part = 2
...
0xF: Ratio part = 16

Bits 15:0 Reserved, must be kept at reset value.

39.14.7 DCMIPP IP-Plug Clientx register 3 (DCMIPP_IPCxR3)

Address offset: 0x028 + 0x10 * (x - 1), (x = 1 to 5)

Reset value: 0x007F 0000, 0x013F 0080, 0x018F 0140, 0x01BF 0190, 0x027F 01C0

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

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

Bits 25:16 **DPREGEND[9:0]:** End word (AXI width = 64 bits) of the FIFO of Clientx.
The addressed word is included in the FIFO, so that next DPREGSTART is DPREGEND + 1.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:0 **DPREGSTART[9:0]:** Start word (AXI width = 64 bits) of the FIFO of Clientx.

39.14.8 DCMIPP parallel interface control register (DCMIPP_PRCR)

Address offset: 0x104

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	SWAP BITS	SWAP CYCLES	Res.	FORMAT[7:0]							
					rw	rw		rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	ENABLE	Res.	EDM[2:0]			Res.	Res.	VSPOL	HSPOL	PCKPOL	ESS	Res.	Res.	Res.	Res.
	rw		rw	rw	rw			rw	rw	rw	rw				

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **SWAPBITS**: Swap LSB vs. MSB within each received component
0: As received
1: Swapped MSB vs. LSB

Bit 25 **SWAPCYCLES**: Swap data (cycle 0 vs. cycle 1) for pixels received on two cycles
0: Default
1: Swap active: the data of cycle 1 is used before the data of cycle 0.
The swap must not be activated by software for pixels received in one or three cycles.

Bit 24 Reserved, must be kept at reset value.

Bits 23:16 **FORMAT[7:0]**:
0x1E: YUV422
0x22: RGB565
0x24: RGB888 (= YUV444)
0x2A: RAW8
0x2B: RAW10
0x2C: RAW12
0x2D: RAW14
0x4A: monochrome 8-bit
0x4B: monochrome 10-bit
0x4C: monochrome 12-bit
0x4D: monochrome 14-bit
0x5A: byte stream (JPEG, compressed video)
Other values: data are captured and output as-is only through the data/dump pipeline (for example JPEG or byte input format).
The monochrome Y input is inserted in the pipe as YUV pixels, with the U and V components set to neutral, to represent a gray color.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **ENABLE:** Parallel interface enable
0: Parallel interface disabled to lower power consumption
1: Parallel interface enabled
The parallel interface configuration registers must be correctly programmed before enabling this bit.

Bit 13 Reserved, must be kept at reset value.

Bits 12:10 **EDM[2:0]:** Extended data mode
0x0: Interface captures 8-bit data on every pixel clock
0x1: Interface captures 10-bit data on every pixel clock
0x2: Interface captures 12-bit data on every pixel clock
0x3: Interface captures 14-bit data on every pixel clock
0x4: Interface captures 16-bit data on every pixel clock
Other values: Reserved

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **VSPOL:** Vertical synchronization polarity
This bit indicates the level on the VSYNC pin when the data are not valid on the parallel interface.
0: VSYNC active low
1: VSYNC active high

Bit 6 **HSPOL:** Horizontal synchronization polarity
This bit indicates the level on the HSYNC pin when the data are not valid on the parallel interface.
0: HSYNC active low
1: HSYNC active high

Bit 5 **PCKPOL:** Pixel clock polarity
This bit configures the capture edge of the pixel clock
0: Falling edge active
1: Rising edge active

Bit 4 **ESS:** Embedded synchronization select
0: Hardware synchronization data capture (frame/line start/stop) is synchronized with the HSYNC/VSYNC signals.
1: Embedded synchronization data capture is synchronized with synchronization codes embedded in the data flow.
Valid only for 8-bit parallel data. HSPOL/VSPOL are ignored when this bit is set.

Bits 3:0 Reserved, must be kept at reset value.

39.14.9 DCMIPP parallel interface embedded synchronization code register (DCMIPP_PRESCR)

Address offset: 0x108

Reset value: 0x0000 0000

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

Bits 31:24 **FEC[7:0]:** Frame end delimiter code

This byte specifies the code of the frame end delimiter. The code consists of four bytes in the form of 0xFF, 0x00, 0x00, FEC.

If FEC is programmed to 0xFF, all the unused codes (0xFF00 00XY) are interpreted as frame end delimiters.

Bits 23:16 **LEC[7:0]:** Line end delimiter code

This byte specifies the code of the line end delimiter. The code consists of four bytes in the form of 0xFF, 0x00, 0x00, LEC.

Bits 15:8 **LSC[7:0]:** Line start delimiter code

This byte specifies the code of the line start delimiter. The code consists of four bytes in the form of 0xFF, 0x00, 0x00, LSC.

Bits 7:0 **FSC[7:0]:** Frame start delimiter code

This byte specifies the code of the frame start delimiter. The code consists of four bytes in the form of 0xFF, 0x00, 0x00, FSC.

If FSC is programmed to 0xFF, no frame start delimiter is detected, but the first occurrence of LSC after an FEC code is interpreted as the start of frame delimiter.

39.14.10 DCMIPP parallel interface embedded synchronization unmask register (DCMIPP_PRESUR)

Address offset: 0x10C

Reset value: 0x0000 0000

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

Bits 31:24 **FEU[7:0]:** Frame end delimiter unmask

This byte specifies the mask to be applied to the code of the frame end delimiter.

0: The corresponding bit in the FEC byte in DCMIPP_ESCR is masked while comparing the frame end delimiter with the received data.

1: The corresponding bit in the FEC byte in DCMIPP_ESCR is compared while comparing the frame end delimiter with the received data

Bits 23:16 **LEU[7:0]:** Line end delimiter unmask

This byte specifies the mask to be applied to the code of the line end delimiter.

0: The corresponding bit in the LEC byte in DCMIPP_ESCR is masked while comparing the line end delimiter with the received data

1: The corresponding bit in the LEC byte in DCMIPP_ESCR is compared while comparing the line end delimiter with the received data

Bits 15:8 **LSU[7:0]:** Line start delimiter unmask

This byte specifies the mask to be applied to the code of the line start delimiter.

0: The corresponding bit in the LSC byte in DCMIPP_ESCR is masked while comparing the line start delimiter with the received data

1: The corresponding bit in the LSC byte in DCMIPP_ESCR is compared while comparing the line start delimiter with the received data

Bits 7:0 **FSU[7:0]:** Frame start delimiter unmask

This byte specifies the mask to be applied to the code of the frame start delimiter.

0: The corresponding bit in the FSC byte in DCMIPP_ESCR is masked while comparing the frame start delimiter with the received data

1: The corresponding bit in the FSC byte in DCMIPP_ESCR is compared while comparing the frame start delimiter with the received data

39.14.11 DCMIPP parallel interface interrupt enable register (DCMIPP_PRIER)

Address offset: 0x1F4

Reset value: 0x0000 0000

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

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **ERRIE:** Synchronization error interrupt enable

0: No interrupt generation

1: An interrupt is generated if the embedded synchronization codes are not received in the correct order.

This bit is available only in embedded synchronization mode.

Bits 5:0 Reserved, must be kept at reset value.

39.14.12 DCMIPP parallel interface status register (DCMIPP_PRSR)

Address offset: 0x1F8

Reset value: 0x0003 0000

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

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

Bit 17 **VSYNC:**

This bit gives the state of the VSYNC pin with the correct programmed polarity if ENABLE bit is set into the DCMIPP_PRCR register and if the pixel clock is received. It is set during the blanking period whatever the polarity selected in VPOL bit, and cleared otherwise.

When embedded synchronization codes are used:

0: Active frame

1: Synchronization between frames

In case of embedded synchronization, this bit is meaningful only if the CAPTURE bit in DCMIPP_CR is set.

Bit 16 **HSYNC:**

This bit gives the state of the HSYNC pin with the correct programmed polarity if ENABLE bit is set into the DCMIPP_PRCR register and if the pixel clock is received. It is set during the blanking period whatever the polarity selected in HPOL bit, and cleared otherwise.

When embedded synchronization codes are used:

0: Active line

1: Synchronization between lines

In case of embedded synchronization, this bit is meaningful only if the CAPTURE bit in DCMIPP_CR is set.

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **ERRF:** Synchronization error raw interrupt status

0: No synchronization error detected

1: Embedded synchronization characters are not received in the correct order.

This bit is valid only in the embedded synchronization mode. It is cleared by writing 1 to the CERRF bit in DCMIPP_PRFCR.

This bit is available only in embedded synchronization mode.

Bits 5:0 Reserved, must be kept at reset value.

39.14.13 DCMIPP parallel interface interrupt clear register (DCMIPP_PRFCR)

Address offset: 0x1FC

Reset value: 0x0000 0000

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

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CERRF:** Synchronization error interrupt status clear

Writing a 1 into this bit clears the ERRF bit in DCMIPP_PRSR.

This bit is available only in embedded synchronization mode.

Bits 5:0 Reserved, must be kept at reset value.

39.14.14 DCMIPP common configuration register (DCMIPP_CMCR)

Address offset: 0x204

Reset value: 0x0000 0000

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

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **SWAPRB**: Swap R/U and B/V

0: RGB/VYU

1: BGR/UYV

This bit is not applicable for Pipe0.

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **CFC**: Clear frame counter

When this bit is set, the frame counter associated to a pipe is cleared. It resets DCMIPP_CMFRCR register. This bit is always read at 0.

Bit 3 Reserved, must be kept at reset value.

Bits 2:1 **PSFC[1:0]**: Pipe selection for the frame counter

These bits (set and cleared by software) select on which pipe the frame counter is mapped.

00: Frame counter mapped to Pipe0

01: Frame counter mapped to Pipe1

10: Frame counter mapped to Pipe2

Other values: Reserved

Bit 0 **INSEL**: input selection

0: DCMI

1: CSI-2

This selection is common for all the pipes, meaning that the input source (DCMI or CSI) is exclusive. This bit must not be changed if one or more pipes are enabled or still in processing state after a disabling operation (corresponding CPTACT bit set into the DCMIPP_CMSR1 register).

39.14.15 DCMIPP common frame counter register (DCMIPP_CMFRCR)

Address offset: 0x208

Reset value: 0x0000 0000

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

Bits 31:0 **FRMCNT[31:0]**: Frame counter, read-only, loops around.

Incremented following VSYNC detection mapped to the pipe configured into bits PSFC[1:0] of the DCMIPP_CMCR register. The counter is cleared using clear frame counter (CFC) bit of DCMIPP_CMCR register.

39.14.16 DCMIPP common interrupt enable register (DCMIPP_CMIER)

Address offset: 0x3F0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
P2 OVR IE	Res.	Res.	Res.	Res.	P2 VSYNC IE	P2 FRAME IE	P2 LINE IE	P1 OVR IE	Res.	Res.	Res.	Res.	P1 VSYNC IE	P1 FRAME IE	P1 LINE IE
rw					rw	rw	rw	rw					rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P0 OVR IE	P0 LIMIT IE	Res.	Res.	Res.	P0 VSYNC IE	P0 FRAME IE	P0 LINE IE	Res.	PR ERR IE	ATX ERR IE	Res.	Res.	Res.	Res.	Res.
rw	rw				rw	rw	rw		rw	rw					

Bit 31 **P2OVRIE:** Overrun interrupt status enable for Pipe2

0: No interrupt generation

1: An interrupt is generated

Bits 30:28 Reserved, must be kept at reset value.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **P2VSYNCIE:** Vertical sync interrupt enable for Pipe2

0: No interrupt generation

1: An interrupt is generated

Bit 25 **P2FRAMEIE:** Frame capture complete interrupt enable for Pipe2

0: No interrupt generation

1: An interrupt is generated

Bit 24 **P2LINEIE:** Multi-line capture complete interrupt enable for Pipe2

0: No interrupt generation

1: An interrupt is generated

Bit 23 **P1OVRIE:** Overrun interrupt enable for Pipe1

Bits 22:20 Reserved, must be kept at reset value.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **P1VSYNCIE:** Vertical sync interrupt enable for Pipe1
0: No interrupt generation
1: An interrupt is generated

Bit 17 **P1FRAMEIE:** Frame capture complete interrupt enable for Pipe1
0: No interrupt generation
1: An interrupt is generated

Bit 16 **P1LINEIE:** Multi-line capture complete interrupt status clear for Pipe1
0: No interrupt generation
1: An interrupt is generated

Bit 15 **P0OVRIE:** Overrun interrupt enable for Pipe0
0: No interrupt generation
1: An interrupt is generated

Bit 14 **P0LIMITIE:** Limit interrupt enable for Pipe0
0: No interrupt generation
1: An interrupt is generated

Bits 13:11 Reserved, must be kept at reset value.

Bit 10 **P0VSYNCIE:** Vertical sync interrupt enable for Pipe0
0: No interrupt generation
1: An interrupt is generated

Bit 9 **P0FRAMEIE:** Frame capture complete interrupt enable for Pipe0
0: No interrupt generation
1: An interrupt is generated

Bit 8 **P0LINEIE:** Multi-line capture complete interrupt enable for Pipe0
0: No interrupt generation
1: An interrupt is generated

Bit 7 Reserved, must be kept at reset value.

Bit 6 **PRERRIE:** Limit interrupt enable for the parallel Interface
0: No interrupt generation
1: An interrupt is generated

Bit 5 **ATXERRIE:** AXI transfer error interrupt enable for IP-Plug
0: No interrupt generation
1: An interrupt is generated

Bits 4:0 Reserved, must be kept at reset value.

39.14.17 DCMIPP common status register 1 (DCMIPP_CMSR1)

Address offset: 0x3F4

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
P2CPTACT	Res.	Res.	Res.	Res.	Res.	P2LSTFRM	P2LSTLINE	P1CPTACT	Res.	Res.	Res.	Res.	Res.	P1LSTFRM	P1LSTLINE
r						r	r	r						r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P0CPTACT	Res.	Res.	Res.	Res.	Res.	P0LSTFRM	P0LSTLINE	Res.	Res.	Res.	Res.	Res.	Res.	PRVSYNC	PRHSYNC
r						r	r							r	r

Bit 31 **P2CPTACT**: Active frame capture (active from start-of-frame to frame complete) for Pipe2
0: No capture currently active
1: Capture currently active

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:26 Reserved, must be kept at reset value.

Bit 25 **P2LSTFRM**: Last frame LSB bit, sampled at frame capture complete event for Pipe2
The information is extracted from the frame data number that can be delivered by the camera through the CSI2 interface.

Bit 24 **P2LSTLINE**: Last line LSB bit, sampled at frame capture complete event for Pipe2
The first line of a frame is counted as 0.

Bit 23 **P1CPTACT**: Active frame capture (active from start-of-frame to frame complete) for Pipe1
0: No capture currently active
1: Capture currently active

Bits 22:20 Reserved, must be kept at reset value.

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **P1LSTFRM**: Last frame LSB bit, sampled at frame capture complete event for Pipe1
The information is extracted from the frame data number, which can be delivered by the camera through the CSI2 interface.

Bit 16 **P1LSTLINE**: Last line LSB bit, sampled at Frame capture complete event for Pipe1

Bit 15 **P0CPTACT**: Active frame capture (active from start-of-frame to frame complete) for Pipe0
0: No capture currently active
1: Capture currently active

Bits 14:12 Reserved, must be kept at reset value.

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **P0LSTFRM**: Last frame LSB bit, sampled at Frame capture complete event for Pipe0
The information is extracted from the frame data number that can be delivered by the camera through the CSI2 interface.

Bit 8 **P0LSTLINE**: Last line LSB bit, sampled at Frame capture complete event for Pipe0

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

Bit 1 **PRVSYNC:**

This bit gives the state of the VSYNC pin with the correct programmed polarity on the parallel interface if ENABLE bit is set into the DCMIPP_PRCR register and if the pixel clock is received. It is set during the blanking period whatever the polarity selected in VPOL bit of the DCMIPP_PRCR register, and cleared otherwise.

When embedded synchronization codes are used, the meaning of this bit is the following:

0: Active frame

1: Synchronization between frames

In case of embedded synchronization, this bit is meaningful only if the CAPTURE bit in the DCMIPP_PRCR register is set.

Bit 0 **PRHSYNC:**

This bit gives the state of the HSYNC pin with the correct programmed polarity on the parallel interface if ENABLE bit is set into the DCMIPP_PRCR register and if the pixel clock is received. It is set during the blanking period whatever the polarity selected in HPOL bit of the DCMIPP_PRCR register, and cleared otherwise.

When embedded synchronization codes are used the meaning of this bit is the following:

0: Active line

1: Synchronization between lines

In case of embedded synchronization, this bit is meaningful only if the CAPTURE bit in the DCMIPP_PRCR register is set.

39.14.18 DCMIPP common status register 2 (DCMIPP_CMSR2)

Address offset: 0x3F8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
P2 OVRF	Res.	Res.	Res.	Res.	P2 VSYNCF	P2 FRAMEF	P2 LINEF	P1 OVRF	Res.	Res.	Res.	Res.	P1 VSYNCF	P1 FRAMEF	P1LINEF
r					r	r	r	r					r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P0 OVRF	P0 LIMITF	Res.	Res.	Res.	P0 VSYNCF	P0 FRAMEF	P0 LINEF	Res.	PR ERRF	ATX ERRF	Res.	Res.	Res.	Res.	Res.
r	r				r	r	r		r	r					

Bit 31 **P2OVRF:** Overrun raw interrupt status for Pipe2

0: No data buffer overrun occurred

1: A data buffer overrun occurred, and this frame data are corrupted

This bit is cleared by writing 1 to the CP2OVRF bit in DCMIPP_CMFCR.

Bits 30:28 Reserved, must be kept at reset value.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **P2VSYNCF:** VSYNC raw interrupt status for Pipe2

This bit is set when the VSYNC signal changes from the inactive state to the active state.

In the case of embedded synchronization, this bit is set only if the CAPTURE bit is set in the DCMIPP_CR register. It is cleared by writing 1 to the CP2VSYNCF bit in the DCMIPP_CMFCR register.

Bit 25 **P2FRAMEF:** Frame capture completed raw interrupt status for Pipe2

0: No capture or ongoing capture

1: All data of a frame have been captured

This bit is set when all data of a frame or window have been captured.

In case of a cropped window, this bit is set at the end of line of the last line in the crop. It is set even if the captured frame is empty (for example window cropped outside the frame).

This bit is cleared by writing 1 to the CP2FRAMEF bit in the DCMIPP_CMFCR.

Bit 24 **P2LINEF:** Multi-line capture completed raw interrupt status for Pipe2

This bit is set when at least a line has been completed, and cleared by writing 1 to the CP2LINEF bit in the DCMIPP_CMFCR register.

The periodicity of LINEF event is configured by LINEMULT bits of the DCMIPP_P2PPCR register. When reaching end of frame, this event is triggered out to allow software action even if the LINEMULT value set is not a multiple of the total lines frame. For embedded synchronization, this bit is set only if the CAPTURE bit in the DCMIPP_CR register is set.

Bit 23 **P1OVRF:** Overrun raw interrupt status for Pipe1

0: No data buffer overrun occurred

1: A data buffer overrun occurred and this frame data are corrupted

This bit is cleared by writing 1 to the CP1OVRF bit in the DCMIPP_CMFCR register.

Bits 22:20 Reserved, must be kept at reset value.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **P1VSYNCF:** VSYNC raw interrupt status for Pipe1

This bit is set when the VSYNC signal changes from inactive to active state. In the case of embedded synchronization, this bit is set only if the CAPTURE bit is set in the DCMIPP_CR register. It is cleared by writing 1 to the CP1VSYNCF bit in the DCMIPP_CMFCR register.

Bit 17 **P1FRAMEF:** Frame capture completed raw interrupt status for Pipe1

0: No capture or ongoing capture

1: All data of a frame have been captured

This bit is set when all data of a frame or window have been captured. In case of a cropped window, this bit is set at the end of line of the last line in the crop. It is set even if the captured frame is empty (for example window cropped outside the frame). This bit is cleared by writing 1 to the CP1FRAMEF bit in the DCMIPP_CMFCR register.

Bit 16 **P1LINEF:** Multi-line capture completed raw interrupt status for Pipe1

This bit is set when one/more lines have been completed. The periodicity of LINEF event is configured by LINEMULT bits into DCMIPP_P1PPCR register. When reaching end of frame, this event is triggered out to allow software action even if the LINEMULT value set is not a multiple of the total lines frame. In the case of embedded synchronization, this bit is set only if the CAPTURE bit in DCMIPP_CR is set. It is cleared by writing 1 to the CP1LINEF bit in the DCMIPP_CMFCR register.

Bit 15 **P0OVRF:** Overrun raw interrupt status for Pipe0

0: No data buffer overrun occurred

1: A data buffer overrun occurred and this frame data are corrupted

This bit is cleared by writing 1 to the CP0OVRF bit in the DCMIPP_CMFCR register.

Bit 14 **P0LIMITF:** Limit raw interrupt status for Pipe0

This bit is set when the data counter DCMIPP_P0DCCNT reaches its maximum value DCMIPP_P0DCLIMIT. It is cleared by writing 1 to the CP0LIMITF bit in the DCMIPP_CMFCR register.

Bits 13:11 Reserved, must be kept at reset value.

Bit 10 **P0VSYNCF:** VSYNC raw interrupt status for Pipe0

This bit is set when the VSYNC signal changes from the inactive state to the active state.

In the case of embedded synchronization, this bit is set only if the CAPTURE bit is set in DCMIPP_CR. It is cleared by writing 1 to the CP0VSYNCF bit in the DCMIPP_CMFCR register.

Bit 9 **P0FRAMEF:** Frame capture completed raw interrupt status for Pipe0

0: No capture or ongoing capture

1: All data of a frame have been captured

This bit is set when all data of a frame or window have been captured.

In case of a cropped window, this bit is set at the end of line of the last line in the crop, even if the captured frame is empty (for example window cropped outside the frame).

This bit is cleared by writing 1 to the CP0FRAMEF bit in the DCMIPP_CMFCR register.

Bit 8 **P0LINEF:** Multi-line capture completed raw interrupt status for Pipe0

This bit is set when one/more lines have been completed. The periodicity of LINEF event is configured by LINEMULT bits into DCMIPP_P0PPCR register. When reaching end of frame, this event is triggered out to allow software action even if the LINEMULT value set is not a multiple of the total lines frame.

In the case of embedded synchronization, this bit is set only if the CAPTURE bit in the DCMIPP_CR register is set. It is cleared by writing 1 to the CP0LINEF bit in the DCMIPP_CMFCR register.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **PRERRF:** Synchronization error raw interrupt status for the parallel interface.

0: No synchronization error detected

1: Embedded synchronization characters are not received in the correct order.

This bit is valid only in the embedded synchronization mode. It is cleared by writing 1 to the CPRERRF bit in the DCMIPP_CMFCR register.

This bit is available only in embedded synchronization mode.

Bit 5 **ATXERRF:** AXI transfer error interrupt status flag for the IP-Plug.

0: No AXI transfer error detected

1: AXI transfer error occurred on an AXI client. This bit signals an error on a client without any specific hardware action, the software must handle the situation (normally used when debugging software application code).

This bit is cleared by writing 1 to CATXERRF bit in the DCMIPP_CMFCR register.

Bits 4:0 Reserved, must be kept at reset value.

39.14.19 DCMIPP common interrupt clear register (DCMIPP_CMFCR)

Address offset: 0x3FC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CP2 OVRF	Res.	Res.	Res.	Res.	CP2 VSYNCF	CP2 FRAMEF	CP2 LINEF	CP1 OVRF	Res.	Res.	Res.	Res.	CP1 VSYNCF	CP1 FRAMEF	CP1 LINEF
w					w	w	w	w					w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CP0 OVRF	CP0 LIMITF	Res.	Res.	Res.	CP0 VSYNCF	CP0 FRAMEF	CP0 LINEF	Res.	CPR ERRF	CATX ERRF	Res.	Res.	Res.	Res.	Res.
w	w				w	w	w		w	w					

Bit 31 **CP2OVRF:** Overrun interrupt status clear
Writing 1 into this bit clears the P2OVRF bit in the DCMIPP_CMSR2 register

Bits 30:28 Reserved, must be kept at reset value.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **CP2VSYNCF:** Vertical synchronization interrupt status clear
Writing 1 into this bit clears the P2VSYNCF bit in the DCMIPP_CMSR2 register

Bit 25 **CP2FRAMEF:** Frame capture complete interrupt status clear
Writing 1 into this bit clears the P2FRAMEF bit in the DCMIPP_CMSR2 register

Bit 24 **CP2LINEF:** Multi-line capture complete interrupt status clear
Writing 1 into this bit clears P2LINEF in the DCMIPP_CMSR2 register

Bit 23 **CP1OVRF:** Overrun interrupt status clear
Writing 1 into this bit clears the P1OVRF bit in the DCMIPP_CMSR2 register

Bits 22:20 Reserved, must be kept at reset value.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **CP1VSYNCF:** Vertical synchronization interrupt status clear
Writing 1 into this bit clears the P1VSYNCF bit in the DCMIPP_CMSR2 register

Bit 17 **CP1FRAMEF:** Frame capture complete interrupt status clear
Writing 1 into this bit clears the P1FRAMEF bit in the DCMIPP_CMSR2 register

Bit 16 **CP1LINEF:** Multi-line capture complete interrupt status clear
Writing 1 into this bit clears P1LINEF in the DCMIPP_CMSR2 register

Bit 15 **CP0OVRF:** Overrun interrupt status clear
Writing 1 into this bit clears the P0OVRF bit in the DCMIPP_CMSR2 register

Bit 14 **CP0LIMITF:** limit interrupt status clear
Writing 1 into this bit clears P0LIMITF in the DCMIPP_CMSR2 register

Bits 13:11 Reserved, must be kept at reset value.

Bit 10 **CP0VSYNCF:** Vertical synchronization interrupt status clear
Writing 1 into this bit clears the P0VSYNCF bit in the DCMIPP_CMSR2 register.

Bit 9 **CP0FRAMEF:** Frame capture complete interrupt status clear
Writing 1 into this bit clears the P0FRAMEF bit in the DCMIPP_CMSR2 register.

Bit 8 **CP0LINEF:** Multi-line capture complete interrupt status clear
Writing 1 into this bit clears P0LINEF in the DCMIPP_CMSR2 register

Bit 7 Reserved, must be kept at reset value.

Bit 6 **CPRERRF:** Synchronization error interrupt status clear
Writing 1 into this bit clears the PRERRF bit in the DCMIPP_CMSR2 register.
This bit is available only in embedded synchronization mode.

Bit 5 **CATXERRF:** AXI transfer error interrupt status clear
Writing 1 into this bit clears the ATXERRF bit in the DCMIPP_CMSR2 register.

Bits 4:0 Reserved, must be kept at reset value.

39.14.20 DCMIPP Pipe0 flow selection configuration register (DCMIPP_P0FSCR)

Address offset: 0x404

Reset value: 0x0000 0000

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

Bit 31 **PIPEN**: Activation of PipeN
0: Pipe disabled
1: Pipe enabled, can start capturing with CPTMODE, CPTREQ, CPTACK

Note: This bit is not shadowed, differently from all other bits in this register.

Bits 30:21 Reserved, must be kept at reset value.

Bits 20:19 **VC[1:0]**: Flow selection mode
Virtual channel ID of the CSI flow to select for further processing on the pipe.

Bit 18 Reserved, must be kept at reset value.

Bits 17:16 **DTMODE[1:0]**: Flow selection mode
00: Only flow DTIDA from the selected virtual channel is forwarded in the pipe
01: Flows DTIDA and/or DTIDB from the selected virtual channel are forwarded in the pipe
10: All data types from the selected virtual channel (except the DTIDA or DTIDB) are forwarded in the pipe
11: All data types of the selected virtual channel VC are forwarded in the pipe

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

Bits 13:8 **DTIDB[5:0]**: Data type selection ID B
Data type ID of the CSI flow to select.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **DTIDA[5:0]**: Data type selection ID A
Data type ID of the CSI flow to select.

39.14.21 DCMIPP Pipe0 flow control configuration register (DCMIPP_P0FCTCR)

Address offset: 0x500

Reset value: 0x0000 0000

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

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

Bit 3 **CPTREQ:** Capture requested

0: Capture not requested for next frame

1: Capture requested for next frame

When PIPEN = 1 and CPTREQ = 1. the pipe waits for the first VSync, automatically starts a capture, and sets CPTACT = 1 to mention it.

In Snapshot mode the CPTREQ bit is cleared at the start of the first received frame.

In Continuous grab mode, the capture remains active and CPTREQ = 1 until the software clears CPTREQ: the capture stops and CPTACT is reset at the end of the ongoing frame.

The DCMI and pipe configuration registers must be correctly programmed before enabling this bit.

Bit 2 **CPTMODE:** Capture mode

0: Continuous grab mode - The received data are transferred into the destination memory through the AXI master.

1: Snapshot mode (single frame) - Once activated, the interface waits for the start of frame, and then transfers a single frame through the AXI master. At the end of the frame, the CPTACT bit is automatically reset.

Bits 1:0 **FRATE[1:0]:** Frame capture rate control

These bits define the frequency of frame capture. They are meaningful only in Continuous grab mode, ignored in Snapshot mode.

00: All frames are captured

01: one out of two frames captured (50% bandwidth reduction)

10: one out of four frames captured (75% bandwidth reduction)

11: one out of eight frames captured (87% bandwidth reduction)

39.14.22 DCMIPP Pipe0 statistic/crop start register (DCMIPP_P0SCSTR)

Address offset: 0x504

Reset value: 0x0000 0000

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

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **VSTART[11:0]**: Vertical start, from 0 to 4094 pixels high

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **HSTART[11:0]**: Horizontal start, from 0 to 4094 words wide

39.14.23 DCMIPP Pipe0 statistic/crop size register (DCMIPP_P0SCSZR)

Address offset: 0x508

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ENABLE	POSNEG	Res.	Res.	VSIZE[11:0]											
rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HSIZE[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **ENABLE**:

This bit is set and cleared by software.

0: Bypass. All the data are computed, if the statistic data are sent within the frame, they are sent to the processing pipe as pixels data.

1: Enable. Depending on bit POSNEG value, the rectangle defined by VSIZE, HSIZE, VSTART and HSTART can be used to extract or to remove some data (statistical extraction or removal, or basic 2D crop features).

if POSNEG = 0, the data inside the rectangle area are transmitted (it can correspond to a statistical data removal, or as a crop feature in a data valid image area).

if POSNEG = 1, the data outside of the rectangle area are transmitted (it can correspond to a statistical data extraction, rejecting all data inside the window).

This bit must be kept cleared if the input format is JPEG, to avoid unpredictable behavior of the pipe.

Bit 30 **POSNEG**:

This bit is set and cleared by software. It has a meaning only if ENABLE bit is set.

0: Positive area, the rectangle defined by VSIZE, HSIZE, and VSTART, HSTART

1: Negative area, the area excluding the rectangle defined by VSIZE, HSIZE, and VSTART, HSTART

Bits 29:28 Reserved, must be kept at reset value.

Bits 27:16 **VSIZE[11:0]**: Vertical size, from 0 to 4094 pixels high

If the value is maintained at 0 when enabling the crop by means of ENABLE bit, the crop operation is not performed on vertical direction.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **HSIZE[11:0]**: Horizontal size, from 0 to 4094 word wide (data 32-bit)

If the value is maintained at 0 when enabling the crop by means of ENABLE bit, the crop operation is not performed on horizontal direction.

39.14.24 DCMIPP Pipe0 dump counter register (DCMIPP_P0DCCNTR)

Address offset: 0x5B0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	CNT[25:16]									
						r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:0 **CNT[25:0]**: Number of data dumped during the frame.

The size of the data is expressed in bytes. It counts only the data selected by means of the CROP 2D function. The counter saturates at 0x3FFFFFF. Granularity is 32-bit for all the formats except for the byte stream formats (for example JPEG) having byte granularity.

39.14.25 DCMIPP Pipe0 dump limit register (DCMIPP_P0DCLMTR)

Address offset: 0x5B4

Reset value: 0x00FF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ENABLE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LIMIT[23:16]							
rw								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LIMIT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **ENABLE**:

0: Disabled, no check on the amount of 32-bit words transmitted

1: Enabled, check done versus limit

Bits 30:24 Reserved, must be kept at reset value.

Bits 23:0 **LIMIT[23:0]**: Maximum number of 32-bit data that can be dumped during a frame, after the crop 2D operation.

39.14.26 DCMIPP Pipe0 pixel packer configuration register (DCMIPP_P0PPCR)

Address offset: 0x5C0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBM
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LINEMULT[2:0]			Res.	OELS	LSM	OEBS	BSM[1:0]		HEADER EN	PAD	Res.	Res.	Res.	Res.	SWAP YUV
rw	rw	rw		rw	rw	rw	rw	rw	rw	rw					rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **DBM:** Double buffer mode
This bit is set and cleared by software.
0: No double buffer mode activated. Pipe0 always dumps to memory address set by DCMIPP_P0PPM0AR1.
1: Double buffer mode activated. Dump address location switches from DCMIPP_P0PPM0AR1 to DCMIPP_P0PPM0AR2 alternatively on each frame.

Bits 15:13 **LINEMULT[2:0]**: Amount of capture completed lines for LINE event and interrupt
0x0: Event after one line
0x1: Event after two lines
0x2: Event after four lines
0x3: Event after eight lines
0x4: Event after sixteen lines
0x5: Event after 32 lines
0x6: Event after 64 lines
0x7: Event after 128 lines

Bit 12 Reserved, must be kept at reset value.

Bit 11 **OELS:** Odd/even line select (line select start)
This bit works in conjunction with LSM field (LSM = 1).
0: Interface captures first line after the frame start, second one is dropped
1: Interface captures second line from the frame start, first one is dropped

Bit 10 **LSM:** Line select mode
0: Interface captures all received lines
1: Interface captures one line out of two

Bit 9 **OEBS:** Odd/even byte select (byte select start)
This bit works in conjunction with BSM field (BSM ≠ 00)
0: Interface captures the first data (byte or double byte) from the frame/line start, the second one is dropped
1: Interface captures the second data (byte or double byte) from the frame/line start, the first one is dropped
This bit has a meaning only if INSEL = 0 into the DCMIPP_CMCR register.

Bits 8:7 **BSM[1:0]:** Byte select mode
00: Interface captures all received data
01: Interface captures 1 data out of 2
10: Interface captures one byte out of four
11: Interface captures two bytes out of four
Modes 10 and 11 work only with EDM [2:0] = 000 into the DCMIPP_PRCR register.
This bit has a meaning only if INSEL = 0 into the DCMIPP_CMCR register

Bit 6 **HEADEREN:** CSI header dump enable
0: CSI-2 headers are not dumped
1: CSI-2 headers are dumped as a 32-bit word.
This setting is ignored for parallel interface mode, as there are no headers.

Bit 5 **PAD:** Pad mode for monochrome and raw Bayer 10/12/14 bpp (MSB vs. LSB alignment)
0: Aligns on LSB (and pads null bits on MSB), for backward compatibility with former DCMI.
1: Aligns on MSB (and pads null bits on LSB), for better ease of software or GPU.

Bits 4:1 Reserved, must be kept at reset value.

Bit 0 **SWAPYUV:** *S*waps, within a 32-bit word, byte 0-vs-1 and byte 2-vs-3. It corresponds, for YUV422 pixels formats, to swap between UYVY and YUYV.

0: Outputs the provided words, as described in *Section 39.10: Pixel format description*.

1: Swaps the bytes from provided words, byte 0-vs.-1 and 2-vs.-3

39.14.27 DCMIPP Pipe0 pixel packer Memory0 address register 1 (DCMIPP_P0PPM0AR1)

Address offset: 0x5C4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M0A[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M0A[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r	r

Bits 31:0 **M0A[31:0]**: Memory0 address

Base address of memory area 0, to whom data are written. It is assumed to be a multiple of 16, hence its bits 3:0 are always at 0x0.

39.14.28 DCMIPP Pipe0 pixel packer Memory0 address register 2 (DCMIPP_P0PPM0AR2)

Address offset: 0x5C8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M0A[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M0A[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r	r

Bits 31:0 **M0A[31:0]**: Memory0 address

Base address of memory area 0, to whom data are written. It is assumed to be a multiple of 16, hence its bits 3:0 are always at 0x0.

39.14.29 DCMIPP Pipe0 status Memory0 address register (DCMIPP_P0STM0AR)

Address offset: 0x5D0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M0A[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M0A[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **M0A[31:0]**: Memory0 address

Base address of memory area 0, to whom data are written. It is assumed to be a multiple of 16, hence its bits 3:0 are always at 0x0.

The register returns the status of the last captured frame.

39.14.30 DCMIPP Pipe0 interrupt enable register (DCMIPP_P0IER)

Address offset: 0x5F4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVR IE	LIMIT IE	Res.	Res.	Res.	VSYNC IE	FRAME IE	LINE IE
								rw	rw				rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **OVRIE:** Overrun interrupt enable

0: No interrupt generation

1: An interrupt is generated if the AXI master is unable to transfer the last data before new data (32-bit) are received.

Bit 6 **LIMITIE:** Limit interrupt enable

0: No interrupt generation when the limit is reached

1: An interrupt is generated when the limit is reached

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **VSYNCIE:** VSYNC interrupt enable

0: No interrupt generation

1: An interrupt is generated on each VSYNC (captured or not)

Bit 1 **FRAMEIE:** Frame capture completed interrupt enable

0: No interrupt generation

1: An interrupt is generated after the full capture of a cropped frame

Bit 0 **LINEIE:** Multi-line capture completed interrupt enable

0: No interrupt generation when the line is received

1: An interrupt is generated after the full capture of a group of lines (or last line reached)

39.14.31 DCMIPP Pipe0 status register (DCMIPP_P0SR)

Address offset: 0x5F8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPTACT	Res.	Res.	Res.	Res.	Res.	LST FRM	LST LINE
								r						r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVR F	LIMIT F	Res.	Res.	Res.	VSYNC F	FRAME F	LINE F
								r	r				r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **CPTACT:** Capture immediate status

0: Capture currently inactive

1: Capture currently active

This bit is automatically reset at the end of frame capture complete event (after all the data of that frame have been captured and the IP-Plug has started to emit the last burst on the AXI, usually before the next VSync).

Bits 22:20 Reserved, must be kept at reset value.

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **LSTFRM**: Last frame LSB bit, sampled at frame capture complete event.

The information is extracted from the frame data number that can be delivered by the camera through the CSI2 interface.

Bit 16 **LSTLINE**: Last line LSB bit, sampled at frame capture complete event.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **OVRF:** Overrun raw interrupt status

0: No data buffer overrun occurred

1: A data buffer overrun occurred and this frame data are corrupted

This bit is cleared by writing 1 to the COVRF bit in the DCMIPP_P0FCR register.

Bit 6 **LIMITF:** Limit raw interrupt status

This bit is set when the data counter DCMIPP_PxDCCNTR reaches its maximum value DCMIPP_PxDCLIMITR.

It is cleared by writing 1 to the CLIMITF bit in the DCMIPP_P0FCR register.

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **VSYNCF:** VSYNC raw interrupt status

This bit is set when the VSYNC signal changes from the inactive state to the active state. In case of embedded synchronization, this bit is set only if the CAPTURE bit is set in DCMIPP_CR.

It is cleared by writing 1 to the CVSYNCF bit in the DCMIPP_P0FCR register.

Bit 1 **FRAMEF:** Frame capture completed raw interrupt status

0: No capture or ongoing capture

1: All data of a frame have been captured

This bit is set when all data of a frame or window have been captured. In case of a cropped window, this bit is set at the end of line of the last line in the crop. It is set even if the captured frame is empty (for example window cropped outside the frame).

This bit is cleared by writing 1 to the CFRAMEF bit in DCMIPP_P0FCR.

Bit 0 **LINEF:** Multi-line capture completed raw interrupt status

This bit is set when one/more lines have been completed. For the JPEG mode, this bit is raised at the end of the frame.

The periodicity of LINEF event is configured by LINEMULT bits into DCMIPP_P0PPCR register. When reaching end of frame, this event is triggered out to allow software action even if the LINEMULT value set is not a multiple of the total lines frame.

In case of embedded synchronization, this bit is set only if the CAPTURE bit in the DCMIPP_CR register is set. It is cleared by writing 1 to the CLINEF bit in the DCMIPP_P0FCR register.

39.14.32 DCMIPP Pipe0 interrupt clear register (DCMIPP_P0FCR)

Address offset: 0x5FC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COVRF	CLIMITF	Res.	Res.	Res.	CVSYNCF	CFRAMEF	CLINEF
								w	w				w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **COVRF:** Overrun interrupt status clear

Writing 1 into this bit clears the OVRF bit in the DCMIPP_P0SR register.

Bit 6 **CLIMITF:** limit interrupt status clear

Writing 1 into this bit clears LIMITF in the DCMIPP_P0SR register.

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **CVSYNCF:** Vertical synchronization interrupt status clear

Writing 1 into this bit clears the VSYNCF bit in the DCMIPP_P0SR register.

Bit 1 **CFRAMEF:** Frame capture complete interrupt status clear

Writing 1 into this bit clears the FRAMEF bit in the DCMIPP_P0SR register.

Bit 0 **CLINEF:** Multi-line capture complete interrupt status clear

Writing 1 into this bit clears LINEF in the DCMIPP_P0SR register.

39.14.33 DCMIPP Pipe0 current flow selection configuration register (DCMIPP_P0CFSCR)

Address offset: 0x604

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PIPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VC[1:0]		Res.	DTMODE[1:0]	
r											r	r		r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	DTIDB[5:0]						Res.	Res.	DTIDA[5:0]					
		r	r	r	r	r	r			r	r	r	r	r	r

Bit 31 **PIPEN**: Current activation of PipeN
0: Pipe disabled
1: Pipe enabled, can start capturing with CPTMODE, CPTREQ, CPTACK.

Bits 30:21 Reserved, must be kept at reset value.

Bits 20:19 **VC[1:0]**: Current flow selection mode
Virtual channel ID of the CSI flow to select for further processing on the pipe.

Bit 18 Reserved, must be kept at reset value.

Bits 17:16 **DTMODE[1:0]**: Flow selection mode
00: Only flow DTIDA from the selected virtual channel is forwarded in the pipe.
01: Flows DTIDA and/or DTIDB from the selected virtual channel are forwarded in the pipe.
10: All Datatypes from the selected virtual channel, except the DTIDA or DTIDB are forwarded into the pipe.
11: All Datatypes of the selected virtual channel VC are forwarded in the pipe.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **DTIDB[5:0]**: Current data type selection ID B
Data type ID of the CSI flow to select.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **DTIDA[5:0]**: Current data type selection ID A
Data type ID of the CSI flow to select.

39.14.34 DCMIPP Pipe0 current flow control configuration register (DCMIPP_P0CFCTCR)

Address offset: 0x700

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPT REQ	CPT MODE	FRATE[1:0]	
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **CPTREQ:** Capture requested

0: Capture not requested for next frame.

1: Capture requested for next frame.

When PIPEN = 1, and when the CPTREQ is set to 1 the pipe waits for the first VSync, and automatically starts a capture and sets CPTACT = 1 to mention it.

In Snapshot mode the CPTREQ bit is automatically cleared at the start of the first frame received.

In continuous grab mode the capture remains active and CPTREQ = 1, until the software clears CPTREQ: the capture stops and CPTACT is reset at the end of the ongoing frame.

The DCMI and pipe configuration registers must be correctly programmed before enabling this bit.

Bit 2 **CPTMODE:** Capture mode

0: Continuous grab mode - The received data are transferred into the destination memory through the AXI master.

1: Snapshot mode (single frame) - Once activated, the interface waits for the start of frame, and then transfers a single frame through the AXI master. At the end of the frame the CPTACT bit is automatically reset.

Bits 1:0 **FRATE[1:0]:** Frame capture rate control

These bits define the frequency of frame capture. They are meaningful only in Continuous grab mode, ignored in Snapshot mode.

00: all frames are captured

01: one out of two frames captured (50% bandwidth reduction)

10: one out of four frames captured (75% bandwidth reduction)

11: one out of eight frames captured (87% bandwidth reduction)

39.14.35 DCMIPP Pipe0 current statistic/crop start register (DCMIPP_P0CSCSTR)

Address offset: 0x704

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	VSTART[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HSTART[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **VSTART[11:0]**: Current vertical start, from 0 to 4094 pixels high

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **HSTART[11:0]**: Current horizontal start, from 0 to 4094 words wide

39.14.36 DCMIPP Pipe0 current statistic/crop size register (DCMIPP_P0CSCSZR)

Address offset: 0x708

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ENABLE	POSNEG	Res.	Res.	VSIZE[11:0]											
r	r			r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HSIZE[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **ENABLE**: Current value of the ENABLE bit

0: Bypass. All data are computed, if the statistics data are sent within the frame, they are sent to the processing pipe as pixels data.

1: Enable: Depending on bit POSNEG value, the rectangle defined by the VSIZE, HSIZE, VSTART, HSTART can be used to extract or to remove certain amount of data (statistical extraction or removal, or basic 2D crop features)

if POSNEG = 0, the data inside the rectangle area are transmitted (can correspond to a statistical data removal, or as a crop feature in a data valid image area).

if POSNEG = 1, the data outside of the rectangle area are transmitted (can correspond to a statistical data extraction, rejecting all data inside the window)

Bit 30 **POSNEG**: Current value of the POSNEG bit

This bit has a meaning only if ENABLE bit is set.

0: Positive area. The rectangle defined by VSIZE,HSIZE, and VSTART, HSTART is the active area.

1: Negative area. The active area is the area excluding the rectangle defined by VSIZE, HSIZE, and VSTART, HSTART.

Bits 29:28 Reserved, must be kept at reset value.

Bits 27:16 **VSIZE[11:0]**: Current vertical size, from 0 to 4094 pixels high.

If the value is maintained at 0 when enabling the crop by means of the ENABLE bit, the value is forced internally at 0xFFE which is the maximum value.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **HSIZE[11:0]**: Current horizontal size, from 0 to 4094 word wide (data 32-bit).

If the value is maintained at 0 when enabling the crop by means of the ENABLE bit, the value is forced internally at 0xFFE, which is the maximum value.

39.14.37 DCMIPP Pipe0 current pixel packer configuration register (DCMIPP_P0CPPCR)

Address offset: 0x7C0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LMAWE	LMAWM[2:0]			DBM
											r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LINEMULT[2:0]			Res.	OELS	LSM	OEBS	BSM[1:0]		HEADER EN	PAD	Res.	Res.	Res.	Res.	SWAP YUV
r	r	r		r	r	r	r	r	r	r					r

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **LMAWE**: Line multi address wrapping enable bit

Enables the wrapping of the address every power-of-2 of written lines.
0: Line multi address wrapping disabled.
1: Line multi address wrapping enabled.

Bits 19:17 **LMAWM[2:0]**: Line multi address wrapping modulo

Amount of captured lines after which the write address is wrapped back to the BaseAddress. It is used to write a full frame onto a circular (ping-pong) buffer.
000: Wraps address after every line
001: Wraps address after two lines
010: Wraps address after four lines
011: Wraps address after eight lines
100: Wraps address after sixteen lines
101: Wraps address after 32 lines
110: Wraps address after 64 lines
111: Wraps address after 128 lines

Bit 16 **DBM:** Double buffer mode

0: No double buffer mode activated. Pipe0 is always dump to memory address set by DCMIPP_P0PPM0AR1 register
1: Double buffer mode activated. Dump address location switches from DCMIPP_P0PPM0AR1 register to DCMIPP_P0PPM0AR2 register alternatively on each frame.

Bits 15:13 **LINEMULT[2:0]**: Current amount of capture completed lines for LINE event and interrupt

0x0: Event after every line
0x1: Event after two lines
0x2: Event after four lines
0x3: Event after eight lines
0x4: Event after sixteen lines
0x5: Event after 32 lines
0x6: Event after 64 lines
0x7: Event after 128 lines

Bit 12 Reserved, must be kept at reset value.

Bit 11 **OELS:** Current odd/even line select (ine select start)

This bit works in conjunction with LSM field (LSM = 1)
0: Interface captures the first line after the frame start, the second one is dropped
1: Interface captures the second line from the frame start, the first one is dropped
This bit has a meaning only if INSEL = 0 in the DCMIPP_CMCR register.

Bit 10 **LSM:** Current Line select mode

0: Interface captures all received lines
1: Interface captures one line out of two
This bit has a meaning only if INSEL = 0 in the DCMIPP_CMCR register.

Bit 9 **OEBS:** Current odd/even byte select (byte select start)

This bit works in conjunction with BSM field (BSM ≠ 00)
0: Interface captures the first data (byte or double byte) from the frame/line start, the second one is dropped
1: Interface captures the second data (byte or double byte) from the frame/line start, the first one is dropped
This bit has a meaning only if INSEL = 0 in the DCMIPP_CMCR register.

Bits 8:7 **BSM[1:0]:** Current Byte select mode
00: Interface captures all received data
01: Interface captures one data out of two
10: Interface captures one byte out of four
11: Interface captures two bytes out of four
Modes 10 and 11 work only with EDM [2:0] = 000 into the DCMIPP_PRCR register.
This bit has a meaning only if INSEL = 0 into the DCMIPP_CMCR register

Bit 6 **HEADEREN:** Current CSI header dump enable
0: CSI-2 headers are not dumped.
1: CSI-2 headers are dumped as 32-bit words.
This setting is ignored for Parallel interface mode, as there are no headers.

Bit 5 **PAD:** Current Pad mode for monochrome and raw Bayer 10/12/14 bpp (MSB vs. LSB alignment)
0: Aligns on LSB (and pads null bits on MSB), for backward compatibility with former DCMI
1: Aligns on MSB (and pads null bits on LSB), for better ease of software or GPU

Bits 4:1 Reserved, must be kept at reset value.

Bit 0 **SWAPYUV:** Swaps, within a 32-bit word, byte 0 vs. 1 and byte 2 vs. 3. It corresponds, for YUV422 pixels formats, to swap between UYVY and YUYV.
0: Outputs the provided words, as described in *Section 39.10: Pixel format description*.
1: Swaps the bytes from provided words, byte 0 vs. 1 and 2 vs. 3.

39.14.38 DCMIPP Pipe0 current pixel packer Memory0 address register 1 (DCMIPP_P0CPPM0AR1)

Address offset: 0x7C4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M0A[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M0A[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **M0A[31:0]**: Memory0 address
Base address of the current memory area 0, to whom data are written. It is assumed to be a multiple of 16, hence its bits 3:0 are always at 0x0.

39.14.39 DCMIPP Pipe0 current pixel packer Memory0 address register 2 (DCMIPP_P0CPPM0AR2)

Address offset: 0x7C8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M0A[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M0A[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **M0A[31:0]**: Memory0 address

Base address of the current memory area 0, to whom data are written. It is assumed to be a multiple of 16, hence its bits 3:0 are always at 0x0.

39.14.40 DCMIPP Pipe1 flow selection configuration register (DCMIPP_P1FSCR)

Address offset: 0x804

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PIPEN	FDTFEN	FDTF[5:0]						Res.	Res.	Res.	VC[1:0]		PIPEDIFF	DTMODE[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	DTIDB[5:0]						Res.	Res.	DTIDA[5:0]					
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bit 31 **PIPEN**: Activation of PipeN

0: Pipe disabled

1: Pipe enabled, can start capturing with CPTMODE, CPTREQ, CPTACK

Bit 30 **FDTFEN**: Force Datatype format enable

0: Disable force Datatype format. The Datatype format used by the pipe is the one defined into DTIDx1: Enable force Datatype format. When the Datatype DTIDx is detected by the pipe, the real pixel format considered by the pipe is the one set into the FDTF[5:0] bit field instead of the physical one received by the CSI2 Host. It allows to handle, for instance, 3D sensors camera using user data type in bottom-up approach, and to explicitly define the pixel format to be handled by the pipe. Note that the flow is first selected using the DTMODE and DTIDA/B, and then, if selected, its data type may be forced with the to FDTC using FDTFEN = 1.

Bits 29:24 **FDTF[5:0]**: Force Datatype format

Forcing the Datatype format. This Datatype overwrites the one received by the CSI2-HOST by specifying the pixel format to consider. The value must follow the CSI2 Datatype format definition in the CSI2 standard. This bit field has no meaning if the FDTFEN bit is cleared.

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:19 **VC[1:0]**: Flow selection mode

Virtual channel ID of the CSI flow to select for further processing on the pipe.

Bit 18 **PIPEDIFF:** Differentiates Pipe2 from Pipe1

0: Pipe2 receives the same data as Pipe1 (data filtered by Pipe1 configuration for VC and DTIDA/DTIDB and DTMODE mode, as well as FDTF and FDTFEN bit).

1: Pipe1 gets pixels only from VC and DT configured in Pipe1, processed by all blocks belonging to the image processing functional block (demosaicing, auto exposure, statistics, contrast enhancement). Pipe2 gets pixels only from VC/DTID and FDTFEN/FDTF bits configured in Pipe2, processed from Crop. There are no shared functions with Pipe1, fully independent.

This bit must be kept cleared when INSEL = 0 into the DCMIPP_CMCR register.

Bits 17:16 **DTMODE[1:0]**: Flow selection mode

00: Only flow DTIDA from the selected virtual channel is forwarded in the pipe

01: Flows DTIDA and/or DTIDB from the selected virtual channel are forwarded in the pipe

Other values: Reserved.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **DTIDB[5:0]**: Data type selection ID B

Data type ID of the CSI flow to select.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **DTIDA[5:0]**: Data type selection ID A

Data type ID of the CSI flow to select.

39.14.41 DCMIPP Pipe1 stat removal configuration register (DCMIPP_P1SRCR)

Address offset: 0x820

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CROP EN	FIRSTLINEDEL[2:0]			LASTLINE[11:0]											
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **CROPEN**: Crop line enable

0: No crop (all pixels are fed through)

1: Crop

Bits 14:12 **FIRSTLINEDEL[2:0]**: Amount of first lines to delete when CROPEN = 1

Bits 11:0 **LASTLINE[11:0]**: Amount of following lines to keep when CROPEN = 1.

If LASTLINE = 0 all pixels after FIRSTLINEDEL are fed through.

39.14.42 DCMIPP Pipe1 bad pixel removal control register (DCMIPP_P1BPRCR)

Address offset: 0x824

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRENGTH[2:0]			ENABLE
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:1 **STRENGTH[2:0]**: Strength (aggressiveness) of the bad pixel detection

0x0: The filter is fairly tolerant: only a few pixels are declared bad, due to their high noise

...

0x7: The filter is very aggressive: many pixels are declared bad, even if only slightly noisy.

Bit 0 **ENABLE**:

0: Bypass: bad pixel removal is not active, all pixels are transmitted through

1: Enable: if bad pixel are detected, they are replaced by corrected pixels

Bad pixel detection must be enabled only for raw Bayer flows, as it corrupts RGB flows.

39.14.43 DCMIPP Pipe1 bad pixel removal status register (DCMIPP_P1BPRSR)

Address offset: 0x828

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	BADCNT[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **BADCNT[11:0]**: Amount of detected bad pixels

Counts the amount of bad pixels detected in each frame, and if too many, saturates at its maximum value (4095). It is reclocked at the frame end, to remain stable for reading during the next frame.

39.14.44 DCMIPP Pipe1 decimation register (DCMIPP_P1DECR)

This register controls the input decimation.

Address offset: 0x830

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VDEC[1:0]		HDEC[1:0]		ENABLE
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:3 **VDEC[1:0]**: Vertical decimation ratio

0x0: All pixels are transmitted, no vertical decimation

0x1: one line out of two transmitted (for raw Bayer data, two lines of components every four)

0x2: one line out of four transmitted (for raw Bayer data, two lines of components every eight)

0x3: one line out of eight transmitted (for raw Bayer data, two lines of components every 16)

Bits 2:1 **HDEC[1:0]**: Horizontal decimation ratio

0x0: All pixels are transmitted, no horizontal decimation

0x1: one line out of two transmitted (for raw Bayer data, two lines of components every four)

0x2: one line out of four transmitted (for raw Bayer data, two lines of components every eight)

0x3: one line out of eight transmitted (for raw Bayer data, two lines of components every 16)

Bit 0 **ENABLE**:

0: Bypass: decimation is not active, all pixels are transmitted

1: Enable: one pixel every 1, 2, 4, or 8, in H and V is transmitted, following HDEC and VDEC

39.14.45 DCMIPP Pipe1 black level calibration control register (DCMIPP_P1BLCCR)

Address offset: 0x840

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BLCR[7:0]								BLCG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BLCB[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENABLE
rw	rw	rw	rw	rw	rw	rw	rw								rw

Bits 31:24 **BLCR[7:0]**: Black level calibration - Red

Value subtracted from the red input MSB component, whether it is on 8, 10, 12, 14 bits.

Bits 23:16 **BLCG[7:0]**: Black level calibration - Green

Value subtracted from the green input MSB component, whether it is on 8, 10, 12, 14 bits.

Bits 15:8 **BLCB[7:0]**: Black level calibration - Blue
Value subtracted from the blue input MSB component, whether it is on 8, 10, 12, 14 bits.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **ENABLE**: Black level calibration
0: Bypass: black level calibration is not active, all pixels are transmitted non-modified.
1: Enable: the BLC R, G, B are subtracted.

39.14.46 DCMIPP Pipe1 exposure control register 1 (DCMIPP_P1EXCR1)

Address offset: 0x844

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	SHFR[2:0]			MULTR[7:0]								Res.	Res.	Res.	Res.
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENABLE
															rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **SHFR[2:0]**: Exposure shift - Red
0x0, 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0x7: exposure of reds is increased, respectively, by a factor 1x, 2x, 4x, 8x, 16x, 32x, 64x, 128x.

Bits 27:20 **MULTR[7:0]**: Exposure multiplier - Red
Value that is multiplied for the red input components, whether it is on 8, 10, 12, 14 bits.
Multiplying by 64 reduces the exposure (darkens the reds) by a factor 2x.
Multiplying by 128 does not modify the exposure (keeps the reds as-is).
Multiplying by 255 increase the exposure (lightens the reds) by a factor ~2x.

Bits 19:1 Reserved, must be kept at reset value.

Bit 0 **ENABLE**: Exposure control (multiplication and shift) of all red, green and blue
0: Bypass: exposure multiplier and shift are not applied
1: Enable: the exposure multiplication and shift is applied

39.14.47 DCMIPP Pipe1 exposure control register 2 (DCMIPP_P1EXCR2)

Address offset: 0x848

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	SHFG[2:0]			MULTG[7:0]								Res.	Res.	Res.	Res.
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SHFB[2:0]			MULTB[7:0]								Res.	Res.	Res.	Res.
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **SHFG[2:0]**: Exposure shift - Green

0x0, 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0x7: Exposure of greens is increased, respectively, by a factor 1x, 2x, 4x, 8x, 16x, 32x, 64x, 128x.

Bits 27:20 **MULTG[7:0]**: Exposure multiplier - Green

Value that is multiplied for the green input components, whether it is on 8, 10, 12, 14 bits.
Multiplying by 64 reduces the exposure (darkens the greens) by a factor 2x.
Multiplying by 128 does not modify the exposure (keeps the greens as-is).
Multiplying by 255 increases the exposure (lightens the greens) by a factor ~2x.

Bits 19:15 Reserved, must be kept at reset value.

Bits 14:12 **SHFB[2:0]**: Exposure shift - Blue

0x0, 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0x7: exposure of blues is increased, respectively, by a factor 1x, 2x, 4x, 8x, 16x, 32x, 64x, 128x.

Bits 11:4 **MULTB[7:0]**: Exposure multiplier - Blue

Value that is multiplied for the blue input components, whether it is on 8, 10, 12, 14 bits.
Multiplying by 64 reduces the exposure (darkens the blues) by a factor 2x.
Multiplying by 128 does not modify the exposure (keeps the blues as-is).
Multiplying by 255 increases the exposure (lightens the blues) by a factor ~2x.

Bits 3:0 Reserved, must be kept at reset value.

39.14.48 DCMIPP Pipe1 statistics1 control register (DCMIPP_P1ST1CR)

Address offset: 0x850

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE	SRC[2:0]			BINS[1:0]		Res.	ENABLE
								rw	rw	rw	rw	rw	rw		rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **MODE**: Statistics mode

0: Average: accumulates the 8-bit component value of the considered pixel components.
Typically used to extract the average value of R, G, B, or luminance across the picture, and to drive the exposure control and the white balance.
1: Bins: accumulates 256 for each considered pixel component fitting a bin dynamic.
Typically used to extract how many pixel components fall into the various dynamic bins, and to drive the contrast enhancement.

Bits 6:4 **SRC[2:0]**: Statistics source

0x0, 0x1, 0x2, 0x3: components are sampled after decimation, and sample, respectively, red, green, blue, and luminance. For raw Bayer formats, only one component is available by pixel.
0x4, 0x5, 0x6, 0x7: components are sampled after demosaicing, and sample, respectively, red, green, blue, and luminance.

Bits 3:2 **BINS[1:0]**: Current bin definition
Condition: MODE = Bins
0x0: LowerBins: Accu0 (1, 2) is incremented of 256 if Component <4 (<8, <16)
0x1: LowMid Bins: Accu0 (1, 2) is incremented of 256 if Component <32 (<64, <128)
0x2: UpMid Bins: Accu0 (1, 2) is incremented of 256 if Component >127 (>191, >224)
0x3: UpBins: Accu0 (1, 2) is incremented of 256 if Component >239 (>247, >251)
Condition: MODE = Average
0x0: All Pixels: Accu is incremented of Component, if 0 ≤ Component < 256.
0x1: NoExt16: Accu is incremented of Component, if 16 ≤ Component < 240
0x2: NoExt32: Accu is incremented of Component, if 32 ≤ Component < 224
0x3: NoExt64: Accu is incremented of Component, if 64 ≤ Component < 192

Bit 1 Reserved, must be kept at reset value.

Bit 0 **ENABLE**:
0: Disabled: statistics are not accumulated
1: Enable: statistics are accumulated

39.14.49 DCMIPP Pipe1 statistics 2 control register (DCMIPP_P1ST2CR)

Address offset: 0x854

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE	SRC[2:0]			BINS[1:0]		Res.	ENABLE
								rw	rw	rw	rw	rw	rw		rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **MODE**: Statistics mode
0: Average: accumulates the 8-bit component value of the considered pixel components.
It is typically used to extract the average value of R, G, B, or luminance across the picture, and to drive the exposure control and the white balance.
1: Bins: accumulates 256 for each considered pixel component fitting a bin dynamic.
It is typically used to extract how many pixel components fall into the various dynamic bins.
It is usually used to drive the contrast enhancement.

Bits 6:4 **SRC[2:0]**: Statistics source
0x0, 0x1, 0x2, 0x3: components are sampled after decimation, and sample, respectively, the red, green, blue components, and luminance.
0x4, 0x5, 0x6, 0x7: components are sampled after the black level calibration and exposure, and sample, respectively, red, green, blue and luminance.

Bits 3:2 **BINS[1:0]**: Bin definition
Condition: MODE = Bins
0x0: LowerBins: Accu0 (1, 2) is incremented of 256 if Component < 4 (< 8, < 16)
0x1: LowMid Bins: Accu0 (1, 2) is incremented of 256 if Component < 32 (< 64, < 128)
0x2: UpMid Bins: Accu0 (1, 2) is incremented of 256 if Component > 127 (> 191, > 224)
0x3: UpBins: Accu0 (1, 2) is incremented of 256 if Component > 239 (> 247, > 251)
Condition: MODE = Average
0x0: AllPixels: Accu is incremented of Component, if 0 ≤ Component < 256.
0x1: NoExt16: Accu is incremented of Component, if 16 ≤ Component < 240.
0x2: NoExt32: Accu is incremented of Component, if 32 ≤ Component < 224.
0x3: NoExt64: Accu is incremented of Component, if 64 ≤ Component < 192.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **ENABLE**:
0: Disabled: statistics are not accumulated
1: Enabled: statistics are accumulated

39.14.50 DCMIPP Pipe1 statistics 3 control register (DCMIPP_P1ST3CR)

Address offset: 0x858

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE	SRC[2:0]			BINS[1:0]		Res.	ENABLE
								rw	rw	rw	rw	rw	rw		rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **MODE**: Statistics mode
0: Average: accumulates the 8-bit component value of the considered pixel components.
It is typically used to extract the average value of R, G, B, or luminance across the picture, and to drive the exposure control and the white balance.
1: Bins: accumulates 256 for each considered pixel component fitting a bin dynamic.
It is typically used to extract how many pixel components fall into the various dynamic bins.
It is usually used to drive the contrast enhancement.

Bits 6:4 **SRC[2:0]**: Statistics source
0x0, 0x1, 0x2, 0x3: components are sampled after decimation, and sample, respectively, the red, green, blue components, and luminance.
0x4, 0x5, 0x6, 0x7: components are sampled after the black level calibration and exposure, and sample, respectively, the red, green, blue and luminance.

Bits 3:2 **BINS[1:0]**: Bin definition
Condition: MODE = Bins
0x0: LowerBins: Accu0 (1, 2) is incremented of 256 if Component <4 (<8, <16)
0x1: LowMid Bins: Accu0 (1, 2) is incremented of 256 if Component <32 (<64, <128)
0x2: UpMid Bins: Accu0 (1, 2) is incremented of 256 if Component >127 (>191, >224)
0x3: UpBins: Accu0 (1, 2) is incremented of 256 if Component >239 (>247, >251)
Condition: MODE = Average
0x0: AllPixels: Accu is incremented of Component, if 0 ≤ Component < 256.
0x1: NoExt16: Accu is incremented of Component, if 16 ≤ Component < 240.
0x2: NoExt32: Accu is incremented of Component, if 32 ≤ Component < 224.
0x3: NoExt64: Accu is incremented of Component, if 64 ≤ Component < 192.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **ENABLE**:
0: Disabled: statistics are not accumulated
1: Enabled: statistics are accumulated

39.14.51 DCMIPP Pipe1 statistics window start register (DCMIPP_P1STSTR)

Address offset: 0x85C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	VSTART[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HSTART[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **VSTART[11:0]**: Vertical start, from 0 to 4094 pixels high

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **HSTART[11:0]**: Horizontal start, from 0 to 4094 pixels wide

39.14.52 DCMIPP Pipe1 statistics window size register (DCMIPP_P1STSZR)

Address offset: 0x860

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CROPEN	Res.	Res.	Res.	VSIZE[11:0]											
rw				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HSIZE[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **CROPEN**:

0: Bypass, all pixels are used to compute the statistics

1: Enable, only the rectangle defined by VSTART, HSTART, VSIZE, HSIZE is used for statistics.

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **VSIZE[11:0]**: Vertical size, from 0 to 4094 pixels high

If the value is kept at 0 when enabling the crop with the CROPEN bit, the value is forced internally at the maximum value 0xFFE.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **HSIZE[11:0]**: Horizontal size, from 0 to 4094 pixels wide

If the value is kept at 0 when enabling the crop with the CROPEN bit, the value is forced internally at the maximum value 0xFFE.

39.14.53 DCMIPP Pipe1 statistics 1 status register (DCMIPP_P1ST1SR)

Address offset: 0x864

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACCU[23:16]							
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACCU[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:0 **ACCU[23:0]**: Accumulation result, divided by 256.

39.14.54 DCMIPP Pipe1 statistics 2 status register (DCMIPP_P1ST2SR)

Address offset: 0x868

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACCU[23:16]							
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACCU[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:0 **ACCU[23:0]**: accumulation result, divided by 256.

39.14.55 DCMIPP Pipe1 statistics 3 status register (DCMIPP_P1ST3SR)

Address offset: 0x86C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACCU[23:16]							
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACCU[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:0 **ACCU[23:0]**: accumulation result, divided by 256.

39.14.56 DCMIPP Pipe1 demosaicing configuration register (DCMIPP_P1DMCR)

Address offset: 0x870

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	EDGE[2:0]			Res.	LINEH[2:0]			Res.	LINEV[2:0]			Res.	PEAK[2:0]		
	rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TYPE[1:0]		ENABLE
													rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **EDGE[2:0]**: Strength of the edge detection
0x0: No edge detection, pure linear interpolation
0x1 to 0x7: Algorithm strength, respectively 3, 4, 6, 8, 12, 16, 24

Bit 27 Reserved, must be kept at reset value.

Bits 26:24 **LINEH[2:0]**: Strength of the horizontal line detection
0x0: No horizontal line detection, pure linear interpolation
0x1 to 0x7: Algorithm strength, respectively 3, 4, 6, 8, 12, 16, 24

Bit 23 Reserved, must be kept at reset value.

Bits 22:20 **LINEV[2:0]**: Strength of the vertical line detection
0x0: No vertical line detection, pure linear interpolation
0x1 to 0x7: Algorithm strength, respectively 3, 4, 6, 8, 12, 16, 24

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **PEAK[2:0]**: Strength of the peak detection
0x0: No peak detection, pure linear interpolation
0x1 to 0x7: Algorithm strength, respectively 3, 4, 6, 8, 12, 16, 24

Bits 15:3 Reserved, must be kept at reset value.

Bits 2:1 **TYPE[1:0]**: Raw Bayer type
0x0: RGGB
0x1: GRBG
0x2: GBRG
0x3: BGGR

Bit 0 **ENABLE**:
0: Bypass
1: Enable demosaicing

39.14.57 DCMIPP Pipe1 ColorConv configuration register (DCMIPP_P1CCCR)

Address offset: 0x880

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAMP	TYPE	ENABLE
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **CLAMP**: Clamp the output samples
0: Not activated (clamped to [0;255] by default)
1: Activated, depending on TYPE

Bit 1 **TYPE**: output samples type used while CLAMP is activated
0: Clamped to [16;235] for Y and to [16;240] for U and V
1: Clamped to [16;235] for R, G and B

Bit 0 **ENABLE**:
0: ColorConv is bypassed
1: ColorConv is enabled

39.14.58 DCMIPP Pipe1 ColorConv red coefficient register 1 (DCMIPP_P1CCRR1)

Address offset: 0x884

Reset value: 0x0000 0000

Coefficients of the ColorConv, in signed 2.8 format (1.0 is coded by 0x100). The sign is indicated as complement to 2.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	RG[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	RR[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:16 **RG[10:0]**: Coefficient row 1 column 2 of the matrix

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **RR[10:0]**: Coefficient row 1 column 1 of the matrix

39.14.59 DCMIPP Pipe1 ColorConv red coefficient register 2 (DCMIPP_P1CCRR2)

Address offset: 0x888

Reset value: 0x0000 0000

The coefficients RA[9:0] of the added column, for the ColorConv, are defined as signed 10-bit integer with the sign as complement to 2 (-256 is coded by 0x200).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	RA[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	RB[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **RA[9:0]**: Coefficient row 1 of the added column (signed integer value)

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **RB[10:0]**: Coefficient row 1 column 3 of the matrix

39.14.60 DCMIPP Pipe1 ColorConv green coefficient register 1 (DCMIPP_P1CCGR1)

Address offset: 0x88C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	GG[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	GR[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:16 **GG[10:0]**: Coefficient row 2 column 2 of the matrix

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **GR[10:0]**: Coefficient row 2 column 1 of the matrix

39.14.61 DCMIPP Pipe1 ColorConv green coefficient register 2 (DCMIPP_P1CCGR2)

Address offset: 0x890

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	GA[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	GB[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **GA[9:0]**: Coefficient row 2 of the added column (signed integer value)

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **GB[10:0]**: Coefficient row 2 column 3 of the matrix

39.14.62 DCMIPP Pipex ColorConv blue coefficient register 1 (DCMIPP_P1CCBR1)

Address offset: 0x894

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	BG[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	BR[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:16 **BG[10:0]**: Coefficient row 3 column 2 of the matrix

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **BR[10:0]**: Coefficient row 3 column 1 of the matrix

39.14.63 DCMIPP Pipe1 ColorConv blue coefficient register 2 (DCMIPP_P1CCBR2)

Address offset: 0x898

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	BA[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	BB[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **BA[9:0]**: Coefficient row 3 of the added column (signed integer value)

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **BB[10:0]**: Coefficient row 3 column 3 of the matrix

39.14.64 DCMIPP Pipe1 contrast control register 1 (DCMIPP_P1CTCR1)

Address offset: 0x8A0

Reset value: 0x0000 2000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LUM0[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENABLE
	rw	rw	rw	rw	rw	rw									rw

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:9 **LUM0[5:0]**: Luminance increase for input luminance of 0 (increase is idle with LUMx = 16)

Bits 8:1 Reserved, must be kept at reset value.

Bit 0 **ENABLE**:

0: Bypass, pixels are forwarded idle.

1: Enable, contrast enhancement is applied on pixels.

39.14.65 DCMIPP Pipe1 contrast control register 2 (DCMIPP_P1CTCR2)

Address offset: 0x8A4

Reset value: 0x2020 2020

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	LUM1[5:0]						Res.	Res.	LUM2[5:0]						Res.
	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LUM3[5:0]						Res.	Res.	LUM4[5:0]						Res.
	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	

Bit 31 Reserved, must be kept at reset value.

Bits 30:25 **LUM1[5:0]**: Luminance increase for input luminance of 32 (increase is idle with LUMx = 16)

Bits 24:23 Reserved, must be kept at reset value.

Bits 22:17 **LUM2[5:0]**: Luminance increase for input luminance of 64 (increase is idle with LUMx = 16)

Bits 16:15 Reserved, must be kept at reset value.

Bits 14:9 **LUM3[5:0]**: Luminance increase for input luminance of 96 (increase is idle with LUMx = 16)

Bits 8:7 Reserved, must be kept at reset value.

Bits 6:1 **LUM4[5:0]**: Luminance increase for input luminance of 128 (increase is idle with LUMx = 16)

Bit 0 Reserved, must be kept at reset value.

39.14.66 DCMIPP Pipe1 contrast control register 3 (DCMIPP_P1CTCR3)

Address offset: 0x8A8

Reset value: 0x2020 2020

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	LUM5[5:0]						Res.	Res.	LUM6[5:0]						Res.
	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LUM7[5:0]						Res.	Res.	LUM8[5:0]						Res.
	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	

Bit 31 Reserved, must be kept at reset value.

Bits 30:25 **LUM5[5:0]**: Luminance increase for input luminance of 160 (increase is idle with LUMx = 16)

Bits 24:23 Reserved, must be kept at reset value.

Bits 22:17 **LUM6[5:0]**: Luminance increase for input luminance of 192 (increase is idle with LUMx = 16)

Bits 16:15 Reserved, must be kept at reset value.

Bits 14:9 **LUM7[5:0]**: Luminance increase for input luminance of 224 (increase is idle with LUMx = 16)

Bits 8:7 Reserved, must be kept at reset value.

Bits 6:1 **LUM8[5:0]**: Luminance increase for input luminance of 256 (increase is idle with LUMx = 16)

Bit 0 Reserved, must be kept at reset value.

39.14.67 DCMIPP Pipex flow control configuration register (DCMIPP_PxFCTCR)

Address offset: 0x900 + 0x400 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPT REQ	CPT MODE	FRATE[1:0]	
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **CPTREQ:** Capture requested

0: Capture not requested for next frame

1: Capture requested for next frame

When PIPEN = 1, and when the CPTREQ is set to 1, the pipe waits for the first VSync, then automatically starts a capture and sets CPTACT = 1 to mention it.

In Snapshot mode, the CPTREQ bit is automatically cleared at the start of the first frame received.

In continuous grab mode, the capture remains active and CPTREQ = 1, until the software clears CPTREQ: the capture stops and CPTACT is reset at the end of the ongoing frame.

The DCMI and pipe configuration registers must be correctly programmed before enabling this bit.

Bit 2 **CPTMODE:** Capture mode

0: Continuous grab mode. The received data are transferred into the destination memory through the AXI master.

1: Snapshot mode (single frame). Once activated, the interface waits for the start of frame, and then transfers a single frame through the AXI master. At the end of the frame, the CPTACT bit is automatically reset.

Bits 1:0 **FRATE[1:0]:** Frame capture rate control

These bits define the frequency of frame capture. They are meaningful only in Continuous grab mode, ignored in Snapshot mode.

00: All frames are captured

01: one out of two frames captured (50% bandwidth reduction)

10: one out of four frames captured (75% bandwidth reduction)

11: one out of eight frames captured (87% bandwidth reduction)

39.14.68 DCMIPP Pipex crop window start register (DCMIPP_PxCRSTR)

Address offset: 0x904 + 0x400 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	VSTART[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HSTART[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **VSTART[11:0]**: Vertical start, from 0 to 4094 pixels high

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **HSTART[11:0]**: Horizontal start, from 0 to 4094 pixels wide

39.14.69 DCMIPP Pipex crop window size register (DCMIPP_PxCRSZR)

Address offset: 0x908 + 0x400 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ENABLE	Res.	Res.	Res.	VSIZE[11:0]											
rw				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HSIZE[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **ENABLE**:
0: Bypass, all pixels are transmitted through (no need to set any start and size values)
1: Enable, only the rectangle defined by VSIZE, HSIZE, VSTART, HSTART is transmitted.

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **VSIZE[11:0]**: Vertical size, from 0 to 4094 pixels high. If the value is maintained at 0 when enabling the crop thanks to the ENABLE bit, the value is forced internally at 0xFFE, which is the maximum value.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **HSIZE[11:0]**: Horizontal size, from 0 to 4094 pixels wide. If the value is maintained at 0 when enabling the crop by means of the ENABLE bit, the value is forced internally at 0xFFE, which is the maximum value.

39.14.70 DCMIPP Pipex decimation register (DCMIPP_PxDCCR)

This register controls decimation, prior to downsize (to complement it).

Address offset: 0x90C + 0x400 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VDEC[1:0]		HDEC[1:0]		ENABLE
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:3 **VDEC[1:0]**: Vertical decimation ratio
0x0: all pixels are transmitted, no vertical decimation
0x1: one line out of two transmitted
0x2: one line out of four transmitted
0x3: one line out of eight transmitted

Bits 2:1 **HDEC[1:0]**: Horizontal decimation ratio
0x0: all pixels are transmitted, no horizontal decimation
0x1: one line out of two transmitted
0x2: one line out of four transmitted
0x3: one line out of eight transmitted

Bit 0 **ENABLE**:
0: Bypass: decimation is not active, all pixels are transmitted
1: Enable: one pixel every 1, 2, 4 or 8, in H and V is transmitted, following HDEC and VDEC

39.14.71 DCMIPP Pipex downsize configuration register (DCMIPP_PxDSCR)

Address offset: 0x910 + 0x400 * (x - 1) (x = 1 to 2)

Reset value: 0x0000 0000

The division factor is an unsigned integer based on the source/destination ratio:

div = max(int(1024 / (source/destination)), 1023)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ENABLE	Res.	Res.	Res.	Res.	Res.	VDIV[9:0]									
rw						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	HDIV[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **ENABLE**:
0: Down scaler is bypassed
1: Down scaler is enabled

Bits 30:26 Reserved, must be kept at reset value.

Bits 25:16 **VDIV[9:0]**: Vertical division factor, from 128 (8x) to 1023 (1x)

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:0 **HDIV[9:0]**: Horizontal division factor, from 128 (8x) to 1023 (1x)

39.14.72 DCMIPP Pipex downsize ratio register (DCMIPP_PxDSRTIOR)

Address offset: 0x914 + 0x400 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

The down scale ratio is expressed in unsigned 3.13 format:

ratio = max(int((SourceSize - 1) / (DestinationSize -1)) * 8192**)**, 8 * 8192 - 1**)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
VRATIO[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HRATIO[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **VRATIO[15:0]**: Vertical ratio, from 8192 (1x) to 65535 (8x)

Bits 15:0 **HRATIO[15:0]**: Horizontal ratio, from 8192 (1x) to 65535 (8x)

39.14.73 DCMIPP Pipex downsize destination size register (DCMIPP_PxDSSZR)

Address offset: 0x918 + 0x400 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	VSIZE[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HSIZE[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **VSIZE[11:0]**: Vertical size, from 0 to 4094 pixels high

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **HSIZE[11:0]**: Horizontal size, from 0 to 4094 pixels wide

39.14.74 DCMIPP Pipex common ROI configuration register (DCMIPP_PxCMRICR)

Address offset: 0x920 + 0x400 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ROI8 EN	ROI7 EN	ROI6 EN	ROI5 EN	ROI4 EN	ROI3 EN	ROI2 EN	ROI1 EN
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ROILSZ[1:0]	
														rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **ROI8EN:** Region of interest 8 enable
This bit is written by software or hardware to enable the ROI8 area to be displayed.
0: Disable ROI
1: Enable ROI

Bit 22 **ROI7EN:** Region of interest 7 enable
This bit is written by software or hardware to enable the ROI7 area to be displayed.
0: Disable ROI
1: Enable ROI

Bit 21 **ROI6EN:** Region of interest 6 enable
This bit is written by software or hardware to enable the ROI6 area to be displayed.
0: Disable ROI
1: Enable ROI

Bit 20 **ROI5EN:** Region of interest 5 enable
This bit is written by software or hardware to enable the ROI5 area to be displayed.
0: Disable ROI
1: Enable ROI

Bit 19 **ROI4EN:** Region of interest 4 enable
This bit is written by software or hardware to enable the ROI4 area to be displayed.
0: Disable ROI
1: Enable ROI

Bit 18 **ROI3EN:** Region of interest 3 enable
This bit is written by software or hardware to enable the ROI3 area to be displayed.
0: Disable ROI
1: Enable ROI

Bit 17 **ROI2EN:** Region of interest 2 enable
This bit is written by software or hardware to enable the ROI2 area to be displayed.
0: Disable ROI
1: Enable ROI

Bit 16 **ROI1EN:** Region of interest 1 enable
This bit is written by software or hardware to enable the ROI1 area to be displayed.
0: Disable ROI
1: Enable ROI

Bits 15:2 Reserved, must be kept at reset value.

Bits 1:0 **ROILSZ**[1:0]: Region of interest line size width
00: Line width 1 pixel
01: Line width 2 pixels
10: Line width 4 pixels
11: Line width 8 pixels

39.14.75 DCMIPP Pipe1 ROIx configuration register 1 (DCMIPP_P1RIxCR1)

Address offset: 0x924 + 0x8 * (x - 1), (x = 1 to 8)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	CLR[1:0]		VSTART[11:0]											
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLG[1:0]		CLB[1:0]		HSTART[11:0]											
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **CLR[1:0]:** Color line red
These bits are written and cleared by software or hardware.
The two bits coding the red value are duplicated over 8 bits as 0bCLR1 CLR0 CLR1 CLR0 CLR1 CLR0 CLR1 CLR0.

Bits 27:16 **VSTART**[11:0]: Vertical start, from 0 to 4094 pixels high

Bits 15:14 **CLG[1:0]:** Color line green
These bits are written and cleared by software or hardware.
The two bits coding the green value are duplicated over 8 bits as 0bCLG1 CLG0 CLG1 CLG0 CLG1 CLG0 CLG1 CLG0.

Bits 13:12 **CLB[1:0]:** Color line blue
These bits are written and cleared by software or hardware.
The two bits coding the red value are duplicated over 8 bits as 0bCLB1 CLB0 CLB1 CLB0 CLB1 CLB0 CLB1 CLB0.

Bits 11:0 **HSTART**[11:0]: Horizontal start, from 0 to 4094 pixels wide

39.14.76 DCMIPP Pipe1 ROIx configuration register 2 (DCMIPP_P1RIxCR2)

Address offset: 0x928 + 0x8 * (x - 1), (x = 1 to 8)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	VSIZE[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HSIZE[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **VSIZE**[11:0]: Vertical size, from 0 to 4094 pixels high

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **HSIZE**[11:0]: Horizontal size, from 0 to 4094 pixels wide

39.14.77 DCMIPP Pipex gamma configuration register (DCMIPP_PxGMCR)

Address offset: 0x970 + 0x400 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENABLE
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **ENABLE**:
0: Gamma is bypassed
1: Gamma is enabled

39.14.78 DCMIPP Pipe1 YUVConv configuration register (DCMIPP_P1YUVCR)

Address offset: 0x980

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAMP	TYPE	ENABLE
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **CLAMP**: Clamp the output samples
0: Not activated (clamped to [0;255] by default)
1: Activated, depending on TYPE

Bit 1 **TYPE**: Output samples type used while CLAMP is activated
0: Clamped to [16;235] for Y and to [16;240] for U and V
1: Clamped to [16;235] for R, G and B

Bit 0 **ENABLE**:
0: ColorConv is bypassed
1: ColorConv is enabled

39.14.79 DCMIPP Pipe1 YUVConv red coefficient register 1 (DCMIPP_P1YUVRR1)

Address offset: 0x984

Reset value: 0x0000 0000

Coefficients of the ColorConv, in signed 2.8 format (1.0 is coded by 0x100):

int_value = int(float_value * (1« 8) + 0.5**)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	RG[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	RR[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:16 **RG[10:0]**: Coefficient row 1 column 2 of the matrix

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **RR[10:0]**: Coefficient row 1 column 1 of the matrix

39.14.80 DCMIPP Pipe1 YUVConv red coefficient register 2 (DCMIPP_P1YUVRR2)

Address offset: 0x988

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	RA[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	RB[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **RA[9:0]**: Coefficient row 1 of the added column (signed integer value)

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **RB[10:0]**: Coefficient row 1 column 3 of the matrix

39.14.81 DCMIPP Pipe1 YUVConv green coefficient register 1 (DCMIPP_P1YUVGR1)

Address offset: 0x98C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	GG[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	GR[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:16 **GG[10:0]**: Coefficient row 2 column 2 of the matrix

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **GR[10:0]**: Coefficient row 2 column 1 of the matrix

39.14.82 DCMIPP Pipe1 YUVConv green coefficient register 2 (DCMIPP_P1YUVGR2)

Address offset: 0x990

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	GA[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	GB[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **GA[9:0]**: Coefficient row 2 of the added column (signed integer value)

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **GB[10:0]**: Coefficient row 2 column 3 of the matrix

39.14.83 DCMIPP Pipe1 YUVConv blue coefficient register 1 (DCMIPP_P1YUVBR1)

Address offset: 0x994

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	BG[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	BR[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:16 **BG[10:0]**: Coefficient row 3 column 2 of the matrix

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **BR[10:0]**: Coefficient row 3 column 1 of the matrix

39.14.84 DCMIPP Pipe1 YUV blue coefficient register 2 (DCMIPP_P1YUVBR2)

Address offset: 0x998

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	BA[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	BB[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **BA[9:0]**: Coefficient row 3 of the added column (signed integer value)

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **BB[10:0]**: Coefficient row 3 column 3 of the matrix

39.14.85 DCMIPP Pipe1 pixel packer configuration register (DCMIPP_P1PPCR)

Address offset: 0x9C0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LMAWE	LMAWM[2:0]			DBM
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LINEMULT[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWAP RB	FORMAT[3:0]			
rw	rw	rw									rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **LMAWE**: Line multi address wrapping enable bit.
Enables the wrapping of the address every power-of-2 of written lines.
0: Line multi address wrapping disabled.
1: Line multi address wrapping enabled.

Bits 19:17 **LMAWM[2:0]**: Line multi address wrapping modulo.
Amount of captured lines after which write address is wrapped back to the BaseAddress. It is used to write a full frame onto a circular (ping-pong) buffer.
000: Wraps address after every line
001: Wraps address after two lines
010: Wraps address after four lines
011: Wraps address after eight lines
100: Wraps address after sixteen lines
101: Wraps address after 32 lines
110: Wraps address after 64 lines
111: Wraps address after 128 lines

Bit 16 **DBM:** Double buffer mode
This bit is set and cleared by software.
0: No double buffer mode activated. Pipe1 always drains out the pixels to memory address set by DCMIPP_P1PPM0AR1, and DCMIPP_P1PPM1AR1 as well as DCMIPP_P1PPM2AR1 in case of semi-planar or multi-planar buffer configuration
1: Double buffer mode activated. Output pixels address location switches from DCMIPP_P1PPM0AR1 to DCMIPP_P1PPM0AR2 alternatively on each frame. For the semi- and multi-planar operations, the output pixels switches from/to DCMIPP_P1PPM1AR1 to/from DCMIPP_P1PPM1AR2. Memory buffer switches from/to DCMIPP_P1PPM2AR1 to/from DCMIPP_P1PPM2AR2 for multi-planar operations.

Bits 15:13 **LINEMULT[2:0]**: Amount of capture completed lines for LINE Event and Interrupt
0x0: Event after one line
0x1: Event after two lines
0x2: Event after four lines
0x3: Event after eight lines
0x4: Event after sixteen lines
0x5: Event after 32 lines
0x6: Event after 64 lines
0x7: Event after 128 lines

Bits 12:5 Reserved, must be kept at reset value.

Bit 4 **SWAPRB**: Swaps R-vs-B components if RGB, and U-vs-V components if YUV
0: No swap of R-vs-B (U-vs-V)
1: Swap active

Bits 3:0 **FORMAT[3:0]**: Memory format
0x0: RGB888 or YUV444 1-buffer
0x1: RGB565 1-buffer
0x2: ARGB8888 (with A = 0xFF)
0x3: RGBA8888 (with A = 0xFF)
0x4: monochrome Y8 or G8 1-buffer
0x5: YUV444 1-buffer (32 bpp, FOURCC = AYUV, with A = 0xFF)
0x6: YUV422 1-buffer (16 bpp, FOURCC = YUYV)
0x7: YUV422 2-buffer (16 bpp, FOURCC = none)
0x8: YUV420 2-buffer (12 bpp, FOURCC = NV21), NV12 available with SWAPRB = 1
0x9: YUV420 3-buffer (12 bpp, FOURCC = YV12)
0xA: YUV422 1-buffer (16 bpp, FOURCC = UYVY)
Other formats are reserved and not supported.

39.14.86 DCMIPP Pipe1 pixel packer Memory0 address register 1 (DCMIPP_P1PPM0AR1)

Address offset: 0x9C4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M0A[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M0A[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r	r

Bits 31:0 **M0A[31:0]**: Memory0 address

Base address of memory area 0, to whom data are written. It is assumed to be a multiple of 16, hence its bits 3:0 are always at 0x0.

39.14.87 DCMIPP Pipe1 pixel packer Memory0 address register 2 (DCMIPP_P1PPM0AR2)

Address offset: 0x9C8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M0A[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M0A[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r	r

Bits 31:0 **M0A[31:0]**: Memory0 address

Base address of memory area 0, to whom data are written. It is assumed to be a multiple of 16, hence its bits 3:0 are always at 0x0.

39.14.88 DCMIPP Pipex pixel packer Memory0 pitch register (DCMIPP_PxPPM0PR)

Address offset: 0x9CC + 0x400 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PITCH[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r	r

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:0 **PITCH[14:0]**: Number of bytes between the address of two consecutive lines.

It is assumed to be a multiple of 16, hence its bits 3:0 are always at 0x0. It ranges to 32 Kbytes, to allow a double-width buffer, containing a 4-kpixel wide 32 bpp picture.

39.14.89 DCMIPP Pipex status Memory0 address register (DCMIPP_PxSTM0AR)

Address offset: 0x9D0 + 0x400 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M0A[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M0A[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **M0A[31:0]**: Memory0 address

Base address of memory area 0, to whom data are written. It is assumed to be a multiple of 16, hence its bits 3:0 are always at 0x0. The register returns the status of the last captured frame.

39.14.90 DCMIPP Pipex pixel packer Memory1 address register 1 (DCMIPP_PxPPM1AR1)

Address offset: 0x9D4 + 0x200 * (x - 1), (x = 1 to 1)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M1A[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M1A[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r	r

Bits 31:0 **M1A[31:0]**: Memory1 address

Base address of memory area 1, to whom data are written. It is assumed to be a multiple of 16, hence its bits 3:0 are always at 0x0.

39.14.91 DCMIPP Pipex pixel packer Memory1 address register 2 (DCMIPP_PxPPM1AR2)

Address offset: 0x9D8 + 0x400 * (x - 1), (x = 1 to 1)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M1A[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M1A[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r	r

Bits 31:0 **M1A[31:0]**: Memory1 address

Base address of memory area 1, to whom data are written. It is assumed to be a multiple of 16, hence its bits 3:0 are always at 0x0.

39.14.92 DCMIPP Pipex pixel packer Memory1 pitch register (DCMIPP_PxPPM1PR)

Address offset: 0x9DC + 0x400 * (x - 1), (x = 1 to 1)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PITCH[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r	r

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:0 **PITCH[14:0]**: Number of bytes between the address of two consecutive lines.

It is assumed to be a multiple of 16, hence its bits 3:0 are always at 0x0. It ranges to 32 Kbytes, to allow a double-width buffer, containing a 4-kpixel wide 32 bpp picture. PITCH is used also for Memory2 (there is no need to differentiate the pitch of Memory1/Memory2).

39.14.93 DCMIPP Pipex status Memory1 address register (DCMIPP_PxSTM1AR)

Address offset: 0x9E0 + 0x400 * (x - 1), (x = 1 to 1)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M1A[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M1A[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **M1A[31:0]**: Memory1 address

Base address of memory area 1, to whom data are written. It is assumed to be a multiple of 16, hence its bits 3:0 are always at 0x0. The register returns the status of the last captured frame.

39.14.94 DCMIPP Pipex pixel packer memory2 address register 1 (DCMIPP_PxPPM2AR1)

Address offset: 0x9E4 + 0x400 * (x - 1), (x = 1 to 1)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M2A[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M2A[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r	r

Bits 31:0 **M2A[31:0]**: Memory 2 address

Base address of memory area 2 to whom data are written. It is assumed to be a multiple of 16, hence its bits 3:0 are always at 0x0.

39.14.95 DCMIPP Pipex pixel packer memory2 address register 2 (DCMIPP_PxPPM2AR2)

Address offset: 0x9E8 + 0x400 * (x - 1), (x = 1 to 1)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M2A[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M2A[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r	r

Bits 31:0 **M2A[31:0]**: Memory 2 address

Base address of memory area 2, to whom data are written. It is assumed to be a multiple of 16, hence its bits 3:0 are always at 0x0.

39.14.96 DCMIPP Pipex status Memory2 address register (DCMIPP_PxSTM2AR)

Address offset: 0x9F0 + 0x400 * (x - 1), (x = 1 to 1)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M2A[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M2A[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **M2A[31:0]**: Memory2 address

Base address of memory area 0, to whom data are written. It is assumed to be a multiple of 16, hence its bits 3:0 are always at 0x0. The register returns the status of the last captured frame.

39.14.97 DCMIPP Pipe1 interrupt enable register (DCMIPP_P1IER)

Address offset: 0x9F4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVRIE	Res.	Res.	Res.	Res.	VSYNCIE	FRAMEIE	LINEIE
								rw					rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **OVRIE:** Overrun interrupt enable

0: No interrupt generation

1: An interrupt is generated if the AXI master is unable to transfer the last data before new data (32-bit) are received.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **VSYNCIE:** VSYNC interrupt enable

0: No interrupt generation

1: An interrupt is generated on each VSYNC (captured or not)

Bit 1 **FRAMEIE:** Frame capture completed interrupt enable

0: No interrupt generation

1: An interrupt is generated after the full capture of a cropped frame

Bit 0 **LINEIE:** Multi-line capture completed interrupt enable

0: No interrupt generation when the line is received

1: An interrupt is generated after the full capture of a group of lines (or last line reached)

39.14.98 DCMIPP Pipe1 status register (DCMIPP_P1SR)

Address offset: 0x9F8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPTACT	Res.	Res.	Res.	Res.	Res.	LSTFRM	LSTLINE
								r						r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVRF	Res.	Res.	Res.	Res.	VSYNCF	FRAMEF	LINEF
								r					r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **CPTACT:** Capture immediate status
0: Capture currently inactive
1: Capture currently active
This bit is automatically reset at the end of Frame capture complete event (after all the data of that frame have been captured, usually before the next VSync).

Bits 22:20 Reserved, must be kept at reset value.

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **LSTFRM**: Last frame LSB bit, sampled at frame capture complete event. The information is extracted from the frame data number, which can be delivered by the camera through the CSI2 interface.

Bit 16 **LSTLINE**: Last line LSB bit, sampled at frame capture complete event.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **OVRF:** Overrun raw interrupt status
0: No data buffer overrun occurred
1: A data buffer overrun occurred and this frame data are corrupted.
This bit is cleared by writing 1 to the COVRF bit in DCMIPP_P1FCR.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **VSYNCF:** VSYNC raw interrupt status
This bit is set when the VSYNC signal changes from the inactive state to the active state.
In the case of embedded synchronization, this bit is set only if the CAPTURE bit is set in DCMIPP_CR.
It is cleared by writing 1 to the CVSYNCF bit in DCMIPP_P1FCR.

Bit 1 **FRAMEF:** Frame capture completed raw interrupt status
0: no capture or ongoing capture
1: all data of a frame have been captured
This bit is set when all data of a frame or window have been captured.
In case of a cropped window, this bit is set at the end of line of the last line in the crop, even if the captured frame is empty (for example window cropped outside the frame).
This bit is cleared by writing 1 to the CFRAMEF bit in DCMIPP_P1FCR.

Bit 0 **LINEF:** Multi-line capture completed raw interrupt status

This bit is set when one/more lines have been completed. The periodicity of LINEF event is configured by LINEMULT bits into DCMIPP_P1PPCR. When reaching end of frame, this event is triggered out to allow software action even if the LINEMULT value set is not a multiple of the total lines frame.

In the case of embedded synchronization, this bit is set only if the CAPTURE bit in DCMIPP_CR is set. It is cleared by writing 1 to the CLINEF bit in DCMIPP_P1FCR.

39.14.99 DCMIPP Pipe1 interrupt clear register (DCMIPP_P1FCR)

Address offset: 0x9FC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COVRF	Res.	Res.	Res.	Res.	CVSYNCF	CFRAMEF	CLINEF
								w					w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **COVRF:** Overrun interrupt status clear

Writing 1 into this bit clears the OVRF bit in the DCMIPP_P1SR register.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CVSYNCF:** Vertical synchronization interrupt status clear

Writing 1 into this bit clears the VSYNCF bit in the DCMIPP_P1SR register.

Bit 1 **CFRAMEF:** Frame capture complete interrupt status clear

Writing 1 into this bit clears the FRAMEF bit in the DCMIPP_P1SR register.

Bit 0 **CLINEF:** Multi-line capture complete interrupt status clear

Writing 1 into this bit clears LINEF in the DCMIPP_P1SR register.

39.14.100 DCMIPP Pipe1 current flow selection configuration register (DCMIPP_P1CFSCR)

Address offset: 0xA04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PIPEN	FDTFEN	FDTF[5:0]						Res.	Res.	Res.	VC[1:0]		PIPEDIFF	DTMODE[1:0]	
r	r	r	r	r	r	r	r				r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	DTIDB[5:0]						Res.	Res.	DTIDA[5:0]					
		r	r	r	r	r	r			r	r	r	r	r	r

Bit 31 **PIPEN**: Current activation of PipeN

0: Pipe disabled

1: Pipe enabled, can start capturing with CPTMODE, CPTREQ, CPTACK.

Bit 30 **FDTFEN**: Current force data type format enable

0: Disable force data type format. The data type format used by the pipe is the one defined into DTIDx

1: Enable force data type format. When the data type DTIDx is detected by the pipe, the real pixel format considering by the pipe is the one set into the FDTF[5:0] bit field instead of the physical one received by the CSI2 host. It allows to handle, for instance, 3D sensors camera with user data type in bottom-up approach, and to explicitly define the pixel format to be handled by the pipe.

Bits 29:24 **FDTF[5:0]**: Current force data type format

Forcing the data type format. This data type overwrites the one received by the CSI2-HOST by specifying the pixel format to consider. The value must follow the CSI2 data type format definition in the CSI2 standard. This bit field has no meaning if the FDTFEN bit is cleared.

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:19 **VC[1:0]**: Current flow selection mode

Virtual channel ID of the CSI flow to select for further processing on the pipe.

Bit 18 **PIPEDIFF:** Current differentiates Pipe2 vs. Pipe1

0: Pipe2 gets Pipe1 pixels after their processing in shared Gamma, StatRemoval, Demosaicing. Pipe2 shares the same VC DTID of Pipe1 when a CSI2 camera sensor is plugged.

1: Pipe1 gets pixels from only VC/DTID configured in Pipe1, processed by all including Gamma, StatRemoval, Demosaicing. Pipe2 gets pixels from only VC/DTID configured in Pipe2, processed from Crop only (Gamma common with Pipe1, no StatRem, no demosaicing)

Bits 17:16 **DTMODE[1:0]**: Flow selection mode

00: Only flow DTID A from the selected virtual channel is forwarded in the pipe.

01: Flows DTIDA and DTIDB from the selected virtual channel are forwarded in the pipe. This combination is valid only if the FDTFEN bit is set.

Other values: Reserved.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **DTIDB[5:0]**: Current data type ID B

Data type ID of the CSI flow to select.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **DTIDA[5:0]**: Current data type ID A

Data type ID of the CSI flow to select.

39.14.101 DCMIPP Pipe1 current bad pixel removal register (DCMIPP_P1CBPRCR)

Address offset: 0xA24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRENGTH[2:0]			ENABLE
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:1 **STRENGTH[2:0]**: Current strength (aggressiveness) of the bad pixel detection:
0x0: the filter is fairly tolerant: only a few pixels are declared bad, due to their high noise.
0x7: the filter is very aggressive: many pixels are declared bad, even if only slightly noisy.

Bit 0 **ENABLE**: Current status of enable bit
0: Bypass: bad pixel removal is not active, all pixels are transmitted through
1: Enable: if bad pixel are detected, they are replaced by corrected pixels.
(bad pixel detection must be enabled only for raw Bayer flows, as it would corrupt RGB flows)

39.14.102 DCMIPP Pipe1 current black level calibration control register (DCMIPP_P1CBLCCR)

Address offset: 0xA40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BLCR[7:0]								BLCG[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BLCB[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENABLE
r	r	r	r	r	r	r	r								r

Bits 31:24 **BLCR[7:0]**: Current black level calibration - Red
Value that is subtracted from the red input components, whether it is on 8,10,12,14 bits.

Bits 23:16 **BLCG[7:0]**: Current black level calibration - Green
Value that is subtracted from the green input components, whether it is on 8,10,12,14 bits.

Bits 15:8 **BLCB[7:0]**: Current black level calibration - Blue
Value that is subtracted from the blue input components, whether it is on 8,10,12,14 bits.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **ENABLE**: For current black level calibration
0: Bypass: black level calibration is not active, all pixels are transmitted non-modified.
1: Enable: the BLCR, G, B are subtracted.

39.14.103 DCMIPP Pipe1 current exposure control register 1 (DCMIPP_P1CEXCR1)

Address offset: 0xA44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	SHFR[2:0]			MULTR[7:0]								Res.	Res.	Res.	Res.
	r	r	r	r	r	r	r	r	r	r	r				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENABLE
															r

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **SHFR[2:0]**: Current exposure shift - Red
0x0, 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0x7: exposure of reds is increased, respectively, by a factor 1x, 2x, 4x, 8x, 16x, 32x, 64x, 128x.

Bits 27:20 **MULTR[7:0]**: Current exposure multiplier - Red
Value that is multiplied for the red input components, whether it is on 8,10,12,14 bits.
Multiplying by 64 reduces the exposure (darkens the reds) by a factor 2x.
Multiplying by 128 does not modify the exposure (keeps the reds as-is).
Multiplying by 255 increases the exposure (lightens the reds) by a factor ~2x.

Bits 19:1 Reserved, must be kept at reset value.

Bit 0 **ENABLE**: for exposure control (multiplication and shift)
0: Bypass: exposure multiplier and shift are not applied.
1: Enable: the exposure multiplication and shift is applied.

39.14.104 DCMIPP Pipe1 current exposure control register 2 (DCMIPP_P1CEXCR2)

Address offset: 0xA48

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	SHFG[2:0]			MULTG[7:0]								Res.	Res.	Res.	Res.
	r	r	r	r	r	r	r	r	r	r	r				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SHFB[2:0]			MULTB[7:0]								Res.	Res.	Res.	Res.
	r	r	r	r	r	r	r	r	r	r	r				

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **SHFG[2:0]**: Current exposure shift - Green
0x0, 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0x7: exposure of greens is increased, respectively, by a factor 1x, 2x, 4x, 8x, 16x, 32x, 64x, 128x.

Bits 27:20 **MULTG[7:0]**: Current exposure multiplier - Green
Value that is multiplied for the green input components, whether it is on 8,10,12,14 bits.
Multiplying by 64 reduces the exposure (darkens the greens) by a factor 2x.
Multiplying by 128 does not modify the exposure (keeps the greens as-is).
Multiplying by 255 increases the exposure (lightens the greens) by a factor ~2x.

Bits 19:15 Reserved, must be kept at reset value.

Bits 14:12 **SHFB[2:0]**: Current exposure shift - Blue

0x0, 0x1, 0x2, 0x3, 0x4, 0x5, 0x6, 0x7: exposure of blues is increased, respectively, by a factor 1x, 2x, 4x, 8x, 16x, 32x, 64x, 128x.

Bits 11:4 **MULTB[7:0]**: Current exposure multiplier - Blue

Value that is multiplied for the blue input components, whether it is on 8,10,12,14 bits.
Multiplying by 64 reduces the exposure (darkens the blues) by a factor 2x.
Multiplying by 128 does not modify the exposure (keeps the blues as-is).
Multiplying by 255 increases the exposure (lightens the blues) by a factor ~2x.

Bits 3:0 Reserved, must be kept at reset value.

39.14.105 DCMIPP Pipe1 current statistics 1 control register (DCMIPP_P1CST1CR)

Address offset: 0xA50

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACCU[23:8]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACCU[7:0]								MODE	SRC[2:0]			BINS[1:0]		Res.	ENABLE
r	r	r	r	r	r	r	r	r	r	r	r	r	r		r

Bits 31:8 **ACCU[23:0]**: Current accumulation result, divided by 256.

Bit 7 **MODE**: Current statistics mode

0: Average: accumulates the 8-bit component value of the considered pixel components.
It is typically used to extract the average value of R, G, B, or luminance across the picture, and to drive the exposure control and the white balance.
1: Bins: accumulates 256 for each considered pixel component fitting a bin dynamic.
It is typically used to extract how many pixel components fall into the various dynamic bins.
It is usually used to drive the contrast enhancement.

Bits 6:4 **SRC[2:0]**: Current source of statistics

0x0, 0x1, 0x2, 0x3: components are sampled before the black level calibration and exposure, and sample, respectively, the red, green, blue components, and luminance (computed as R / 4 + G / 2 + B / 4).
0x4, 0x5, 0x6, 0x7: components are sampled after the black level calibration and exposure, and sample, respectively, red, green, blue and luminance.

Bits 3:2 **BINS[1:0]**: Current bin definition

Condition: MODE = Bins
0x0: LowerBins: Accu0 (1, 2) is incremented of 256 if Component <4 (<8, <16)
0x1: LowMid Bins: Accu0 (1, 2) is incremented of 256 if Component <32 (<64, <128)
0x2: UpMid Bins: Accu0 (1, 2) is incremented of 256 if Component >127 (>191, >224)
0x3: UpBins: Accu0 (1, 2) is incremented of 256 if Component >239 (>247, >251)
Condition: MODE = Average
0x0: AllPixels: Accu is incremented of Component, if 0 ≤ Component < 256
0x1: NoExt16: Accu is incremented of Component, if 16 ≤ Component < 240
0x2: NoExt32: Accu is incremented of Component, if 32 ≤ Component < 224
0x3: NoExt64: Accu is incremented of Component, if 64 ≤ Component < 192

Bit 1 Reserved, must be kept at reset value.

Bit 0 **ENABLE**: Current enable bit value

0: Disabled: statistics are not accumulated

1: Enable: statistics are accumulated

39.14.106 DCMIPP Pipe1 current statistics 2 control register (DCMIPP_P1CST2CR)

Address offset: 0xA54

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACCU[23:8]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACCU[7:0]								MODE	SRC[2:0]			BINS[1:0]		Res.	ENABLE
r	r	r	r	r	r	r	r	r	r	r	r	r	r		r

Bits 31:8 **ACCU[23:0]**: Accumulation result, divided by 256.

Bit 7 **MODE**: Statistics mode

0: Average: accumulates the 8-bit component value of the considered pixel components.
It is typically used to extract the average value of R, G, B, or luminance across the picture, and to drive the exposure control and the white balance.

1: Bins: accumulates 256 for each considered pixel component fitting a bin dynamic.
It is typically used to extract how many pixel components fall into the various dynamic bins. It is usually used to drive the contrast enhancement.

Bits 6:4 **SRC[2:0]**: Statistics source

0x0, 0x1, 0x2, 0x3: components are sampled before the black level calibration and exposure, and sample, respectively, the red, green, blue components, and luminance (computed as R / 4 + G / 2 + B / 4).

0x4, 0x5, 0x6, 0x7: components are sampled after the black level calibration and exposure, and sample, respectively, the red, green, blue and luminance.

Bits 3:2 **BINS[1:0]**: Bin definition

Condition: MODE = Bins

0x0: LowerBins: Accu0 (1, 2) is incremented of 256 if Component <4 (<8, <16)

0x1: LowMid Bins: Accu0 (1, 2) is incremented of 256 if Component <32 (<64, <128)

0x2: UpMid Bins: Accu0 (1, 2) is incremented of 256 if Component >127 (>191, >224)

0x3: UpBins: Accu0 (1, 2) is incremented of 256 if Component >239 (>247, >251)

Condition: MODE = Average

0x0: AllPixels: Accu is incremented of Component, if 0 ≤ Component < 256

0x1: NoExt16: Accu is incremented of Component, if 16 ≤ Component < 240

0x2: NoExt32: Accu is incremented of Component, if 32 ≤ Component < 224

0x3: NoExt64: Accu is incremented of Component, if 64 ≤ Component < 192

Bit 1 Reserved, must be kept at reset value.

Bit 0 **ENABLE**:

0: Disabled: statistics are not accumulated

1: Enable: statistics are accumulated

39.14.107 DCMIPP Pipe1 current statistics 3 control register (DCMIPP_P1CST3CR)

Address offset: 0xA58

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACCU[23:8]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACCU[7:0]								MODE	SRC[2:0]			BINS[1:0]		Res.	ENABLE
r	r	r	r	r	r	r	r	r	r	r	r	r	r		r

Bits 31:8 **ACCU[23:0]**: Accumulation result, divided by 256.

Bit 7 **MODE**: Statistics mode

0: Average: accumulates the 8-bit component value of the considered pixel components.
It is typically used to extract the average value of R, G, B, or luminance across the picture, and to drive the exposure control and the white balance.
1: Bins: accumulates 256 for each considered pixel component fitting a bin dynamic.
It is typically used to extract how many pixel components fall into the various dynamic bins. It is usually used to drive the contrast enhancement.

Bits 6:4 **SRC[2:0]**: Statistics source

0x0, 0x1, 0x2, 0x3: components are sampled before the black level calibration and exposure, and sample, respectively, the red, green, blue components, and luminance (computed as R / 4 + G / 2 + B / 4).
0x4, 0x5, 0x6, 0x7: components are sampled after the black level calibration and exposure, and sample, respectively, red, green, blue and luminance.

Bits 3:2 **BINS[1:0]**: Current bin definition

Condition: MODE = Bins
0x0: LowerBins: Accu0 (1, 2) is incremented of 256 if Component <4 (<8, <16)
0x1: LowMid Bins: Accu0 (1, 2) is incremented of 256 if Component <32 (<64, <128)
0x2: UpMid Bins: Accu0 (1, 2) is incremented of 256 if Component >127 (>191, >224)
0x3: UpBins: Accu0 (1, 2) is incremented of 256 if Component >239 (>247, >251)
Condition: MODE = Average
0x0: AllPixels: Accu is incremented of Component, if 0 ≤ Component < 256
0x1: NoExt16: Accu is incremented of Component, if 16 ≤ Component < 240
0x2: NoExt32: Accu is incremented of Component, if 32 ≤ Component < 224
0x3: NoExt64: Accu is incremented of Component, if 64 ≤ Component < 192

Bit 1 Reserved, must be kept at reset value.

Bit 0 **ENABLE**:

0: Disabled: statistics are not accumulated
1: Enable: statistics are accumulated

39.14.108 DCMIPP Pipe1 current statistics window start register (DCMIPP_P1CSTSTR)

Address offset: 0xA5C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	VSTART[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HSTART[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **VSTART[11:0]**: Current vertical start, from 0 to 4094 pixels high

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **HSTART[11:0]**: Current horizontal start, from 0 to 4094 pixels wide

39.14.109 DCMIPP Pipe1 current statistics window size register (DCMIPP_P1CSTSZR)

Address offset: 0xA60

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CROPEN	Res.	Res.	Res.	VSIZE[11:0]											
r				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HSIZE[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **CROPEN**: Current CROPEN bit value

0: Bypass, all pixels are used to compute the statistics.

1: Enable, only the rectangle defined by VSTART, HSTART, VSIZE, HSIZE, are used for statistics.

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **VSIZE[11:0]**: Current vertical size, from 0 to 4094 pixels high

If the value is maintained at 0 when enabling the crop by means of the CROPEN bit, the value is forced internally at 0xFFE, which is the maximum value.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **HSIZE[11:0]**: Current horizontal size, from 0 to 4094 pixels wide

If the value is maintained at 0 when enabling the crop by means of the CROPEN bit, the value is forced internally at 0xFFE, which is the maximum value.

39.14.110 DCMIPP Pipe1 current ColorConv configuration register (DCMIPP_P1CCCCR)

Address offset: 0xA80

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAMP	TYPE	ENABLE
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **CLAMP**: Clamp the output samples
This bit indicates the current value applied.
0: Not activated (clamped to [0;255] by default)
1: Activated, depending on TYPE

Bit 1 **TYPE**: Output samples type used while CLAMP is activated
This bit indicates the current value applied
0: clamped to [16;235] for Y and to [16;240] for U and V
1: clamped to [16;235] for R, G and B

Bit 0 **ENABLE**: Current value applied
0: ColorConv is bypassed
1: ColorConv is enabled

39.14.111 DCMIPP Pipe1 current ColorConv red coefficient register 1 (DCMIPP_P1CCCRR1)

Address offset: 0xA84

Reset value: 0x0000 0000

Coefficients of the ColorConv, in signed 2.8 format (1.0 is coded by 0x100):

int_value = int(float_value * (1 « 8) + 0.5**)**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	RG[10:0]										
					r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	RR[10:0]										
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:16 **RG[10:0]**: Current coefficient row 1 column 2 of the matrix

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **RR[10:0]**: Current coefficient row 1 column 1 of the matrix

39.14.112 DCMIPP Pipe1 current ColorConv red coefficient register 2 (DCMIPP_P1CCCRR2)

Address offset: 0xA88

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	RA[9:0]									
						r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	RB[10:0]										
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **RA[9:0]**: Current coefficient row 1 of the added column (signed integer value)

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **RB[10:0]**: Current coefficient row 1 column 3 of the matrix

39.14.113 DCMIPP Pipe1 current ColorConv green coefficient register 1 (DCMIPP_P1CCCGR1)

Address offset: 0xA8C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	GG[10:0]										
					r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	GR[10:0]										
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:16 **GG[10:0]**: Current coefficient row 2 column 2 of the matrix

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **GR[10:0]**: Current coefficient row 2 column 1 of the matrix

39.14.114 DCMIPP Pipe1 current ColorConv green coefficient register 2 (DCMIPP_P1CCCGR2)

Address offset: 0xA90

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	GA[9:0]									
						r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	GB[10:0]										
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **GA[9:0]**: Current coefficient row 2 of the added column (signed integer value)

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **GB[10:0]**: Current coefficient row 2 column 3 of the matrix

39.14.115 DCMIPP Pipex current ColorConv blue coefficient register 1 (DCMIPP_P1CCCBR1)

Address offset: 0xA94

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	BG[10:0]										
					r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	BR[10:0]										
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:16 **BG[10:0]**: Current coefficient row 3 column 2 of the matrix

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **BR[10:0]**: Current coefficient row 3 column 1 of the matrix

39.14.116 DCMIPP Pipe1 current ColorConv blue coefficient register 2 (DCMIPP_P1CCCBR2)

Address offset: 0xA98

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	BA[9:0]									
						r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	BB[10:0]										
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **BA[9:0]**: Current coefficient row 3 of the added column (signed integer value)

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:0 **BB[10:0]**: Current coefficient row 3 column 3 of the matrix

39.14.117 DCMIPP Pipe1 current contrast control register 1 (DCMIPP_P1CCTCR1)

Address offset: 0xAA0

Reset value: 0x0000 2000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LUM0[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENABLE
	r	r	r	r	r	r									r

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:9 **LUM0[5:0]**: Current luminance increase for input luminance of 0 (increase is idle with LUMx = 16)

Bits 8:1 Reserved, must be kept at reset value.

Bit 0 **ENABLE**: Current ENABLE bit value

0: Bypass, pixels are forwarded idle.

1: Enable, contrast enhancement is applied on pixels.

39.14.118 DCMIPP Pipe1 current contrast control register 2 (DCMIPP_P1CCTCR2)

Address offset: 0xAA4

Reset value: 0x2020 2020

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	LUM1[5:0]						Res.	Res.	LUM2[5:0]						Res.
	r	r	r	r	r	r			r	r	r	r	r	r	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LUM3[5:0]						Res.	Res.	LUM4[5:0]						Res.
	r	r	r	r	r	r			r	r	r	r	r	r	

Bit 31 Reserved, must be kept at reset value.

Bits 30:25 **LUM1[5:0]**: Current luminance increase for input luminance of 32 (increase is idle with LUMx = 16)

Bits 24:23 Reserved, must be kept at reset value.

Bits 22:17 **LUM2[5:0]**: Current luminance increase for input luminance of 64 (increase is idle with LUMx = 16)

Bits 16:15 Reserved, must be kept at reset value.

Bits 14:9 **LUM3[5:0]**: Current luminance increase for input luminance of 96 (increase is idle with LUMx = 16)

Bits 8:7 Reserved, must be kept at reset value.

Bits 6:1 **LUM4[5:0]**: Current luminance increase for input luminance of 128 (increase is idle with LUMx = 16)

Bit 0 Reserved, must be kept at reset value.

39.14.119 DCMIPP Pipe1 current contrast control register 3 (DCMIPP_P1CCTCR3)

Address offset: 0xAA8

Reset value: 0x2020 2020

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	LUM5[5:0]						Res.	Res.	LUM6[5:0]						Res.
	r	r	r	r	r	r			r	r	r	r	r	r	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	LUM7[5:0]						Res.	Res.	LUM8[5:0]						Res.
	r	r	r	r	r	r			r	r	r	r	r	r	

Bit 31 Reserved, must be kept at reset value.

Bits 30:25 **LUM5[5:0]**: Luminance increase for input luminance of 160 (increase is idle with LUMx = 16)

Bits 24:23 Reserved, must be kept at reset value.

Bits 22:17 **LUM6[5:0]**: Luminance increase for input luminance of 192 (increase is idle with LUMx = 16)

Bits 16:15 Reserved, must be kept at reset value.

Bits 14:9 **LUM7[5:0]**: Luminance increase for input luminance of 224 (increase is idle with LUMx = 16)

Bits 8:7 Reserved, must be kept at reset value.

Bits 6:1 **LUM8[5:0]**: Luminance increase for input luminance of 256 (increase is idle with LUMx = 16)

Bit 0 Reserved, must be kept at reset value.

39.14.120 DCMIPP Pipex current flow control configuration register (DCMIPP_PxCFCTCR)

Address offset: 0xB00 + 0x400 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPT REQ	CPT MODE	FRATE[1:0]	
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **CPTREQ:** Capture requested

0: Capture not requested for next frame.

1: Capture requested for next frame.

When PIPEN = 1, and when the CPTREQ is set to 1 the pipe waits for the first VSync, and automatically starts a capture and sets CPTACT= 1 to mention it.

In Snapshot mode, the CPTREQ bit is automatically cleared at the start of the first frame received.

In continuous grab mode, the capture remains active and CPTREQ = 1, until the software clears CPTREQ: the capture stops and CPTACT is reset at the end of the ongoing frame.

The DCMIPP and pipe configuration registers must be correctly programmed before enabling this bit.

Bit 2 **CPTMODE:** Capture mode

0: Continuous grab mode - Received data are transferred into the destination memory through the AXI master.

1: Snapshot mode (single frame) - Once activated, the interface waits for the start of frame and then transfers a single frame through the AXI master. At the end of the frame, the CPTACT bit is automatically reset.

Bits 1:0 **FRATE[1:0]:** Frame capture rate control

These bits define the frequency of frame capture. They are meaningful only in Continuous grab mode, ignored in Snapshot mode.

00: all frames are captured

01: one out of two frames captured (50% bandwidth reduction)

10: one out of four frames captured (75% bandwidth reduction)

11: one out of eight frames captured (87% bandwidth reduction)

39.14.121 DCMIPP Pipex current crop window start register (DCMIPP_PxCCRSTR)

Address offset: 0xB04 + 0x400 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	VSTART[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HSTART[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **VSTART[11:0]**: Current vertical start, from 0 to 4094 pixels high

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **HSTART[11:0]**: Current horizontal start, from 0 to 4094 pixels wide

39.14.122 DCMIPP Pipex current crop window size register (DCMIPP_PxCCRSZR)

Address offset: 0xB08 + 0x400 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ENABLE	Res.	Res.	Res.	VSIZE[11:0]											
r				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HSIZE[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **ENABLE**: Current ENABLE bit value.
0: Bypass, all pixels are transmitted through (no need to set any start and size values)
1: Enable, only the rectangle defined by VSIZE, HSIZE, VSTART, HSTART is transmitted.

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **VSIZE[11:0]**: Current vertical size, from 0 to 4094 pixels high

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **HSIZE[11:0]**: Current horizontal size, from 0 to 4094 pixels wide

39.14.123 DCMIPP Pipex current decimation register (DCMIPP_PxCDCCR)

This register controls decimation, prior to downsize (to complement it).

Address offset: 0xB0C + 0x400 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VDEC[1:0]		HDEC[1:0]		ENABLE
											r	r	r	r	r

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:3 **VDEC[1:0]**: Vertical decimation ratio
- 0x0: All pixels are transmitted, no vertical decimation
- 0x1: one line out of two transmitted
- 0x2: one line out of four transmitted
- 0x3: one line out of eight transmitted

Bits 2:1 **HDEC[1:0]**: Horizontal decimation ratio
- 0x0: All pixels are transmitted, no horizontal decimation
- 0x1: one line out of two transmitted
- 0x2: one line out of four transmitted
- 0x3: one line out of eight transmitted

Bit 0 **ENABLE**:
- 0: Bypass: decimation is not active, all pixels are transmitted
- 1: Enable: one pixel every 1, 2, 4 or 8, in H and V is transmitted, following HDEC and VDEC

39.14.124 DCMIPP Pipex current downsize configuration register (DCMIPP_PxCDSCR)

Address offset: 0xB10 + 0x400 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

The division factor is an unsigned integer based on the source/destination ratio:

div = max(int(024 / (source/destination)), 1023)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ENABLE	Res.	Res.	Res.	Res.	Res.	VDIV[9:0]									
r						r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	HDIV[9:0]									
						r	r	r	r	r	r	r	r	r	r

Bit 31 **ENABLE**: Current value of bit ENABLE
- 0: Down scaler is bypassed
- 1: Down scaler is enabled

Bits 30:26 Reserved, must be kept at reset value.

Bits 25:16 **VDIV[9:0]**: Current vertical division factor, from 128 (8x) to 1023 (1x)

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:0 **HDIV[9:0]**: Current horizontal division factor, from 128 (8x) to 1023 (1x)

39.14.125 DCMIPP Pipex current downsize ratio register (DCMIPP_PxCDSRTIOR)

Address offset: 0xB14 + 0x400 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

The down scale ratio is expressed in unsigned 3.13 format:

ratio = max(int((source / destination) * 8192), 8 * 8192 - 1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
VRATIO[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HRATIO[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **VRATIO[15:0]**: Current vertical ratio, from 8192 (1x) to 65535 (8x)

Bits 15:0 **HRATIO[15:0]**: Current horizontal ratio, from 8192 (1x) to 65535 (8x)

39.14.126 DCMIPP Pipex current downsize destination size register (DCMIPP_PxCDSSZR)

Address offset: 0xB18 + 0x400 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	VSIZE[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HSIZE[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **VSIZE[11:0]**: Current vertical size, from 0 to 4094 pixels high

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **HSIZE[11:0]**: Current horizontal size, from 0 to 4094 pixels wide

39.14.127 DCMIPP Pipex current common ROI configuration register (DCMIPP_PxCCMRICR)

Address offset: 0xB20 + 0x400 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ROI8EN	ROI7EN	ROI6EN	ROI5EN	ROI4EN	ROI3EN	ROI2EN	ROI1EN
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ROILSZ[1:0]	
														r	r

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **ROI8EN:** Current region of interest 8 enable
This bit enables the ROI8 area to be displayed.
0: Disable ROI
1: Enable ROI

Bit 22 **ROI7EN:** Current region of interest 7 enable
This bit enables the ROI7 area to be displayed.
0: Disable ROI
1: Enable ROI

Bit 21 **ROI6EN:** Current region of interest 6 enable
This bit enables the ROI6 area to be displayed.
0: Disable ROI
1: Enable ROI

Bit 20 **ROI5EN:** Current region of interest 5 enable
This bit enables the ROI5 area to be displayed.
0: Disable ROI
1: Enable ROI

Bit 19 **ROI4EN:** Current region of interest 4 enable
This bit enables the ROI4 area to be displayed.
0: Disable ROI
1: Enable ROI

Bit 18 **ROI3EN:** Current region of interest 3 enable
This bit enables the ROI3 area to be displayed.
0: Disable ROI
1: Enable ROI

Bit 17 **ROI2EN:** Current region of interest 2 enable
This bit enables the ROI2 area to be displayed.\
0: Disable ROI
1: Enable ROI

Bit 16 **ROI1EN:** Current region of interest 1 enable
This bit enables the ROI1 area to be displayed.
0: Disable ROI
1: Enable ROI

Bits 15:2 Reserved, must be kept at reset value.

Bits 1:0 **ROILSZ**[1:0]: Current region of interest line size width
00: Line width 1 pixel
01: Line width 2 pixels
10: Line width 4 pixels
11: Line width 8 pixels

39.14.128 DCMIPP Pipe1 current ROIx configuration register 1 (DCMIPP_P1CRIxCR1)

Address offset: 0xB24 + 0x8 * (x - 1), (x = 1 to 8)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	CLR[1:0]		VSTART[11:0]											
		r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLG[1:0]		CLB[1:0]		HSTART[11:0]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **CLR[1:0]:** Current color line red
The two bits coding the red value are duplicated over 8 bits as 0bCLR1 CLR0 CLR1 CLR0 CLR1 CLR0 CLR1 CLR0.

Bits 27:16 **VSTART**[11:0]: Current vertical start, from 0 to 4094 pixels high

Bits 15:14 **CLG[1:0]:** Current color line green
The two bits coding the red value are duplicated over 8 bits as 0bCLG1 CLG0 CLG1 CLG0 CLG1 CLG0 CLG1 CLG0.

Bits 13:12 **CLB[1:0]:** Current color line blue
The two bits coding the red value are duplicated over 8 bits as 0bCLB1 CLB0 CLB1 CLB0 CLB1 CLB0 CLB1 CLB0.

Bits 11:0 **HSTART**[11:0]: Current horizontal start, from 0 to 4094 pixels wide

39.14.129 DCMIPP Pipe1 current ROIx configuration register 2 (DCMIPP_P1CRIxCR2)

Address offset: 0xB28 + 0x8 * (x - 1), (x = 1 to 8)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	VSIZE[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HSIZE[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **VSIZE[11:0]**: Current vertical size, from 0 to 4094 pixels high

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **HSIZE[11:0]**: Current horizontal size, from 0 to 4094 pixels wide

39.14.130 DCMIPP Pipe1 current pixel packer configuration register (DCMIPP_P1CPPCR)

Address offset: 0xBC0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LMAWE	LMAWM[2:0]			DBM
											r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LINEMULT[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWAP RB	FORMAT[3:0]			
r	r	r									r	r	r	r	r

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **LMAWE**: Line multi address wrapping enable bit

Enables the wrapping of the address every power-of-2 of written lines.

0: Line multi address wrapping disabled.

1: Line multi address wrapping enabled.

Bits 19:17 **LMAWM[2:0]**: Line multi address wrapping modulo

Amount of captured lines after which the write address is wrapped back to the BaseAddress. It is used to write a full frame onto a circular (ping-pong) buffer.

000: Wraps address after every line

001: Wraps address after two lines

010: Wraps address after four lines

011: Wraps address after eight lines

100: Wraps address after sixteen lines

101: Wraps address after 32 lines

110: Wraps address after 64 lines

111: Wraps address after 128 lines

Bit 16 **DBM:** Double buffer mode

This bit is set and cleared by software.

0: No double buffer mode activated. Pipe1 always drains out the pixels to memory address set by DCMIPP_P1PPM0AR1, and DCMIPP_P1PPM1AR1 as well as DCMIPP_P1PPM2AR1 in case of semi-planar or multi-planar buffer configuration

1: Double buffer mode activated. Output pixels address location switches from DCMIPP_P1PPM0AR1 to DCMIPP_P1PPM0AR2 alternatively on each frame. For the semi- and multi-planar operations, the output pixels switches from/to DCMIPP_P1PPM1AR1 to/from DCMIPP_P1PPM1AR2. Memory buffer switches from/to DCMIPP_P1PPM2AR1 to/from DCMIPP_P1PPM2AR2 for multi-planar operations.

Bits 15:13 **LINEMULT[2:0]**: Amount of capture completed lines for LINE Event and Interrupt
0x0: Event after one line
0x1: Event after two lines
0x2: Event after four lines
0x3: Event after eight lines
0x4: Event after sixteen lines
0x5: Event after 32 lines
0x6: Event after 64 lines
0x7: Event after 128 lines

Bits 12:5 Reserved, must be kept at reset value.

Bit 4 **SWAPRB**: Swaps R-vs-B components if RGB, and U-vs-V components if YUV
0: No swap of R-vs-B (U-vs-V)
1: Swap active

Bits 3:0 **FORMAT[3:0]**: Memory format
0x0: RGB888 or YUV444 1-buffer
0x1: RGB565 1-buffer
0x2: ARGB8888 (with A = 0xFF)
0x3: RGBA8888 (with A = 0xFF)
0x4: monochrome Y8 or G8 1-buffer
0x5: YUV444 1-buffer (32 bpp, FOURCC = AYUV, with A = 0xFF)
0x6: YUV422 1-buffer (16 bpp, FOURCC = YUYV)
0x7: YUV422 2-buffer (16 bpp, FOURCC = none)
0x8: YUV420 2-buffer (12 bpp, FOURCC = NV21), NV12 available with SWAPRB = 1
0x9: YUV420 3-buffer (12 bpp, FOURCC = YV12)
0xA: YUV422 1-buffer (16 bpp, FOURCC = UYVY)
Other formats are reserved and not supported.

39.14.131 DCMIPP Pipe1 current pixel packer Memory0 address register 1 (DCMIPP_P1CPPM0AR1)

Address offset: 0xBC4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M0A[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M0A[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **M0A[31:0]**: Memory0 address
Base address of current memory area 0, to whom data are written.
It is assumed to be a multiple of 16, hence its bits 3:0 are always at 0x0.

39.14.132 DCMIPP Pipe1 current pixel packer Memory0 address register 2 (DCMIPP_P1CPPM0AR2)

Address offset: 0xBC8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M0A[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M0A[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **M0A[31:0]**: Memory0 address

Base address of current memory area 0, to whom data are written.
It is assumed to be a multiple of 16, hence its bits 3:0 are always at 0x0.

39.14.133 DCMIPP Pipex current pixel packer Memory0 pitch register (DCMIPP_PxCPPM0PR)

Address offset: 0xBCC + 0x400 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PITCH[14:0]														
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:0 **PITCH[14:0]**: Current number of bytes between the address of two consecutive lines.

It is assumed to be a multiple of 16, hence its bits 3:0 are always at 0x0. It ranges to 32 Kbytes, to allow a double-width buffer, containing a 4-kpixel wide 32 bpp picture.

39.14.134 DCMIPP Pipex current pixel packer Memory1 address register 1 (DCMIPP_PxCPPM1AR1)

Address offset: 0xBD4 + 0x400 * (x - 1), (x = 1 to 1)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M1A[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M1A[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **M1A[31:0]**: Memory1 address

Base address of current memory area 1, to whom data are written. It is assumed to be a multiple of 16, hence its bits 3:0 are always at 0x0.

39.14.135 DCMIPP Pipex current pixel packer Memory1 address register 2 (DCMIPP_PxCPPM1AR2)

Address offset: 0xBD8 + 0x400 * (x - 1), (x = 1 to 1)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M1A[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M1A[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **M1A[31:0]**: Memory1 address

Base address of memory area 1, to whom data are written. It is assumed to be a multiple of 16, hence its bits 3:0 are always at 0x0.

39.14.136 DCMIPP Pipex current pixel packer Memory1 pitch register (DCMIPP_PxCPPM1PR)

Address offset: 0xBDC + 0x400 * (x - 1), (x = 1 to 1)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PITCH[14:0]														
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:0 **PITCH[14:0]**: Current number of bytes between the address of two consecutive lines

It is assumed to be a multiple of 16, hence its bits [3:0] are always at 0x0. It ranges to 32 Kbytes, to allow a double-width buffer, containing a 4-kpixel wide 32 bpp picture.

39.14.137 DCMIPP Pipex current pixel packer Memory2 address register 1 (DCMIPP_PxCPPM2AR1)

Address offset: 0xBE4 + 0x400 * (x - 1), (x = 1 to 1)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M2A[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M2A[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **M2A[31:0]**: Memory 2 address

Base address of current memory area 2 where data are written. It is assumed to be a multiple of 16, hence its bits [3:0] are always at 0x0.

39.14.138 DCMIPP Pipex current pixel packer Memory2 address register 1 (DCMIPP_PxCPPM2AR2)

Address offset: 0xBE8 + 0x400 * (x - 1), (x = 1 to 1)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M2A[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M2A[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **M2A[31:0]**: Memory 2 address

Base address of current memory area 2 where data are written. It is assumed to be a multiple of 16, hence its bits [3:0] are always at 0x0.

39.14.139 DCMIPP Pipe2 flow selection configuration register (DCMIPP_P2FSCR)

Address offset: 0xC04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PIPEN	FDTFEN	FDTF[5:0]						Res.	Res.	Res.	VC[1:0]		Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw				rw	rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTIDA[5:0]					
										rw	rw	rw	rw	rw	rw

Bit 31 **PIPEN**: Activation of PipeN

0: PipeN disabled

1: PipeN enabled, can start capturing with CPTMODE, CPTREQ, CPTACK

Bit 30 **FDTFEN**: Force data type format enable

0: Disable force data type format. The data type format used by the pipe is the one defined into DTIDx

1: Enable force data type format. When the data type DTIDA is detected by the pipe, the real pixel format considered by the pipe is the one set into the FDTF[5:0] bit field instead of the physical one received by the CSI2 host.

This bit has no meaning if PIPEDIFF = 0 in the DCMIPP_P1FSCR register.

Bits 29:24 **FDTF[5:0]**: Force data type format

This data type overwrites the one received by the CSI2 Host by specifying the pixel format to consider. The value must follow the CSI2 data type format definition in the CSI2 standard. This bit field has no meaning if the FDTFEN bit is cleared.

These bits have no meaning if PIPEDIFF = 0 in the DCMIPP_P1FSCR register.

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:19 **VC[1:0]**: Flow selection mode

Virtual channel ID of the CSI flow to select for further processing on the pipe.

These bits have no meaning if PIPEDIFF = 0 into DCMIPP_P1FSCR, hence the pipe takes the value VC[1:0] configured into the DMCIPP_P1FSCR register.

Bits 18:6 Reserved, must be kept at reset value.

Bits 5:0 **DTIDA[5:0]**: Data type ID

Data type ID of the CSI flow to select.

These bits have no meaning if PIPEDIFF = 0 into DCMIPP_P1FSCR, hence the pipe takes the value of VC and DTIDA[5:0] / DTIDB[5:0] configured into the DMCIPP_P1FSCR register.

39.14.140 DCMIPP Pipe2 ROIx configuration register 1 (DCMIPP_P2RIxCR1)

Address offset: 0xD24 + 0x8 * (x - 1), (x = 1 to 8)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	CLR[1:0]		VSTART[11:0]											
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLG[1:0]		CLB[1:0]		HSTART[11:0]											
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **CLR[1:0]:** Color line red

These bits are written and cleared by software or hardware. The two bits coding the red value are duplicated over 8 bits as 0bCLR1 CLR0 CLR1 CLR0 CLR1 CLR0 CLR1 CLR0.

Bits 27:16 **VSTART**[11:0]: Vertical start, from 0 to 4094 pixels high

Bits 15:14 **CLG[1:0]:** Color line green

These bits are written and cleared by software or hardware. The two bits coding the green value are duplicated over 8 bits as 0bCLG1 CLG0 CLG1 CLG0 CLG1 CLG0 CLG1 CLG0.

Bits 13:12 **CLB[1:0]:** Color line blue

These bits are written and cleared by software or hardware. The two bits coding the red value are duplicated over 8 bits as 0bCLB1 CLB0 CLB1 CLB0 CLB1 CLB0 CLB1 CLB0.

Bits 11:0 **HSTART**[11:0]: Horizontal start, from 0 to 4094 pixels wide

39.14.141 DCMIPP Pipe2 ROIx configuration register 2 (DCMIPP_P2RIxCR2)

Address offset: 0xD28 + 0x8 * (x - 1), (x = 1 to 8)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	VSIZE[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HSIZE[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **VSIZE**[11:0]: Vertical size, from 0 to 4094 pixels high

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **HSIZE**[11:0]: Horizontal size, from 0 to 4094 pixels wide

39.14.142 DCMIPP Pipe2 pixel packer configuration register (DCMIPP_P2PPCR)

Address offset: 0xDC0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LMAWE	LMAWM[2:0]			DBM
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LINEMULT[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWAP RB	FORMAT[3:0]			
rw	rw	rw									rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **LMAWE**: Line multi address wrapping enable bit

Enables the wrapping of the address every power-of-2 of written lines.

0: Line multi address wrapping disabled.

1: Line multi address wrapping enabled.

Bits 19:17 **LMAWM[2:0]**: Line multi address wrapping modulo
Amount of captured lines after which the write address is wrapped back to the BaseAddress. It is used to write a full frame onto a circular (ping-pong) buffer.
000: Wraps address after every line
001: Wraps address after two lines
010: Wraps address after four lines
011: Wraps address after eight lines
100: Wraps address after sixteen lines
101: Wraps address after 32 lines
110: Wraps address after 64 lines
111: Wraps address after 128 lines

Bit 16 **DBM:** Double buffer mode
This bit is set and cleared by software.
0: No double buffer mode activated. Pipe2 is always dump to memory address set by DCMIPP_P2PPM0AR1
1: Double buffer mode activated. Dump address location switches from DCMIPP_P2PPM0AR1 register to DCMIPP_P2PPM0AR2 register alternatively on each frame

Bits 15:13 **LINEMULT[2:0]**: Amount of capture completed lines for LINE event and interrupt
0x0: Event after every line
0x1: Event after two lines
0x2: Event after four lines
0x3: Event after eight lines
0x4: Event after sixteen lines
0x5: Event after 32 lines
0x6: Event after 64 lines
0x7: Event after 128 lines

Bits 12:5 Reserved, must be kept at reset value.

Bit 4 **SWAPRB**: Swaps R-vs-B components if RGB, and if YUV, swaps U-vs-V components
0: No swap of R-vs-B (U-vs-V)
1: Swap active.

Bits 3:0 **FORMAT[3:0]**: Memory format (only coplanar formats are supported in Pipe2)
0x0: RG/B888 or YUV444 1-buffer
0x1: RGB565 1-buffer
0x2: ARGB8888
0x3: RGBA8888
0x4: monochrome Y8 or G8 1-buffer
0x5: YUV444 1-buffer (FOURCC = AYUV)
0x6: YUV422 1-buffer (FOURCC = YUYV)
Other formats are reserved and not supported.

39.14.143 DCMIPP Pipe2 pixel packer Memory0 address register 1 (DCMIPP_P2PPM0AR1)

Address offset: 0xDC4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M0A[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M0A[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r	r

Bits 31:0 **M0A[31:0]**: Memory0 address

Base address of memory area 0, to whom data are written. It is assumed to be a multiple of 16, hence its bits 3:0 are always at 0x0.

39.14.144 DCMIPP Pipe2 pixel packer Memory0 address register 2 (DCMIPP_P2PPM0AR2)

Address offset: 0xDC8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M0A[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M0A[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r	r

Bits 31:0 **M0A[31:0]**: Memory0 address

Base address of memory area 0, to whom data are written. It is assumed to be a multiple of 16, hence its bits 3:0 are always at 0x0.

39.14.145 DCMIPP Pipe2 interrupt enable register (DCMIPP_P2IER)

Address offset: 0xDF4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVR IE	Res.	Res.	Res.	Res.	VSYNC IE	FRAME IE	LINE IE
								rw					rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **OVRIE:** Overrun interrupt enable
0: No interrupt generation
1: An interrupt is generated if the AXI master is unable to transfer the last data before new data (32-bit) are received

Bits 6:3 Reserved, must be kept at reset value.

Bit 2 **VSYNCIE:** VSYNC interrupt enable
0: No interrupt generation
1: An interrupt is generated on each VSYNC (captured or not)

Bit 1 **FRAMEIE:** Frame capture completed interrupt enable
0: No interrupt generation
1: An interrupt is generated after the full capture of a cropped frame

Bit 0 **LINEIE:** Multi-line capture completed interrupt enable
0: No interrupt generation when the line is received
1: An interrupt is generated after the full capture of a group of lines (or last line reached)

39.14.146 DCMIPP Pipe2 status register (DCMIPP_P2SR)

Address offset: 0xDF8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPTACT	Res.	Res.	Res.	Res.	Res.	LSTFRM	LSTLINE
								r						r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVRF	Res.	Res.	Res.	Res.	VSYNCF	FRAMEF	LINEF
								r					r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **CPTACT:** Capture immediate status
0: Capture currently inactive
1: Capture currently active
This bit is automatically reset at the end of frame capture complete event (after all the data of that frame have been captured, usually before the next VSync).

Bits 22:20 Reserved, must be kept at reset value.

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **LSTFRM**: Last frame LSB bit, sampled at frame capture complete event.
The information is extracted from the frame data number which can be delivered by the camera through the CSI2 interface.

Bit 16 **LSTLINE**: Last line LSB bit, sampled at frame capture complete event.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **OVRF:** Overrun raw interrupt status
0: No data buffer overrun occurred
1: A data buffer overrun occurred and data of this frame are corrupted
This bit is cleared by writing 1 to the COVRF bit in the DCMIPP_P2FCR register.

Bits 6:3 Reserved, must be kept at reset value.

Bit 2 **VSYNCF:** VSYNC raw interrupt status
This bit is set when the VSYNC signal changes from the inactive state to the active state. In the case of embedded synchronization, this bit is set only if the CAPTURE bit is set in the DCMIPP_CR register. It is cleared by writing 1 to the CVSYNCF bit in the DCMIPP_P2FCR register.

Bit 1 **FRAMEF:** Frame capture completed raw interrupt status
0: No capture or ongoing capture
1: All data of a frame have been captured
This bit is set when all data of a frame or window have been captured.
In case of a cropped window, this bit is set at the end of line of the last line in the crop. It is set even if the captured frame is empty (for example window cropped outside the frame).
This bit is cleared by writing 1 to the CFRAMEF bit in DCMIPP_P2FCR.

Bit 0 **LINEF:** Multi-line capture completed raw interrupt status
This bit is set when one/more lines have been completed.
The periodicity of LINEF event is configured bu LINEMULT bits into DCMIPP_P1PPCR register. When reaching end of frame, this event is triggered out to allow software action even if the LINEMULT value set is not a multiple of the total lines frame.
In the case of embedded synchronization, this bit is set only if the CAPTURE bit in the DCMIPP_CR register is set.
It is cleared by writing 1 to the CLINEF bit in the DCMIPP_P2FCR register.

39.14.147 DCMIPP Pipe2 interrupt clear register (DCMIPP_P2FCR)

Address offset: 0xDFC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	C OVRF	Res.	Res.	Res.	Res.	C VSYNCF	C FRAMEF	C LINEF
								w					w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **COVRF:** Overrun interrupt status clear
Writing 1 into this bit clears the OVRF bit in the DCMIPP_P2SR register.

Bits 6:3 Reserved, must be kept at reset value.

Bit 2 **CVSYNCF:** Vertical synchronization interrupt status clear
Writing 1 into this bit clears the VSYNCF bit in the DCMIPP_P2SR register.

Bit 1 **CFRAMEF:** Frame capture complete interrupt status clear
Writing 1 into this bit clears the FRAMEF bit in the DCMIPP_P2SR register.

Bit 0 **CLINEF:** Multi-line capture complete interrupt status clear
Writing 1 into this bit clears LINEF in the DCMIPP_P2SR register.

39.14.148 DCMIPP Pipe2 current flow selection configuration register (DCMIPP_P2CFSCR)

Address offset: 0xE04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PIPEN	FDTFEN	FDTF[5:0]						Res.	Res.	Res.	VC[1:0]		Res.	Res.	Res.
r	r	r	r	r	r	r	r				r	r			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTIDA[5:0]					
										r	r	r	r	r	r

Bit 31 **PIPEN**: Current activation of PipeN
0: Pipe disabled
1: Pipe enabled, can start capturing with CPTMODE, CPTREQ, CPTACK.

Bit 30 **FDTFEN**: Current force data type format enable
0: Disable force data type format. The data type format used by the pipe is the one defined into DTIDx
1: Enable force data type format. When data type DTIDA is detected by the pipe, the real pixel format considering by the pipe is the one set into the FDTF[5:0] bit field instead of the physical one received by the CSI2 host.
This bit has no meaning if PIPEDIFF = 0 in the DCMIPP_P1FSCR register.

Bits 29:24 **FDTF[5:0]**: Current force data type format
Forcing the data type format. This data type overwrites the one received by the CSI2-HOST by specifying the pixel format to consider. The value must follow the CSI2 data type format definition in the CSI2 standard. This bit field has no meaning if the FDTFEN bit is cleared, or if PIPEDIFF = 0 in the DCMIPP_P1FSCR register.

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:19 **VC[1:0]**: Current flow selection mode
Virtual channel ID of the CSI flow to select for further processing on the pipe. These bits have no meaning if PIPEDIFF = 0 into DCMIPP_P1FSCR. For this configuration the pipe takes the value VC[1:0] configured in the DMCIPP_P1FSCR register.

Bits 18:6 Reserved, must be kept at reset value.

Bits 5:0 **DTIDA[5:0]**: Current data type ID
Data type ID of the CSI flow to select. For this configuration the pipe takes the value DTIDA[5:0] configured in the DMCIPP_P1FSCR register.

39.14.149 DCMIPP Pipe2 current ROIx configuration register 1 (DCMIPP_P2CRIxCR1)

Address offset: 0xF24 + 0x8 * (x - 1), (x = 1 to 8)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	CLR[1:0]		VSTART[11:0]											
		r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLG[1:0]		CLB[1:0]		HSTART[11:0]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **CLR[1:0]:** Current color line red

The two bits coding the red value are duplicated over 8 bits as 0bCLR1 CLR0 CLR1 CLR0 CLR1 CLR0 CLR1 CLR0.

Bits 27:16 **VSTART**[11:0]: Current vertical start, from 0 to 4094 pixels high

Bits 15:14 **CLG[1:0]:** Current color line green

The two bits coding the red value are duplicated over 8 bits as 0bCLG1 CLG0 CLG1 CLG0 CLG1 CLG0 CLG1 CLG0.

Bits 13:12 **CLB[1:0]:** Current color line blue

The two bits coding the red value are duplicated over 8 bits as 0bCLB1 CLB0 CLB1 CLB0 CLB1 CLB0 CLB1 CLB0.

Bits 11:0 **HSTART**[11:0]: Current horizontal start, from 0 to 4094 pixels wide

39.14.150 DCMIPP Pipe2 current ROIx configuration register 2 (DCMIPP_P2CRIxCR2)

Address offset: 0xF28 + 0x8 * (x - 1), (x = 1 to 8)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	VSIZE[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HSIZE[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **VSIZE**[11:0]: Current vertical size, from 0 to 4094 pixels high

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **HSIZE**[11:0]: Current horizontal size, from 0 to 4094 pixels wide

39.14.151 DCMIPP Pipe2 current pixel packer configuration register (DCMIPP_P2CPPCR)

Address offset: 0xFC0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LMAWE	LMAWM[2:0]			DBM
											r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LINEMULT[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWAP RB	FORMAT[3:0]			
r	r	r									r	r	r	r	r

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **LMAWE**: Line multi address wrapping enable bit

Enables the wrapping of the address every power-of-2 of written lines.

0: Line multi address wrapping disabled.

1: Line multi address wrapping enabled.

Bits 19:17 **LMAWM[2:0]**: Line multi address wrapping modulo

Amount of captured lines after which the write address is wrapped back to the BaseAddress. It is used to write a full frame onto a circular (ping-pong) buffer.

000: Wraps address after every line

001: Wraps address after two lines

010: Wraps address after four lines

011: Wraps address after eight lines

100: Wraps address after sixteen lines

101: Wraps address after 32 lines

110: Wraps address after 64 lines

111: Wraps address after 128 lines

Bit 16 **DBM:** Double buffer mode

This bit is set and cleared by software.

0: No double buffer mode activated. Pipe2 is always dump to memory address set by DCMIPP_P2PPM0AR1

1: Double buffer mode activated. Dump address location switches from DCMIPP_P2PPM0AR1 register to DCMIPP_P2PPM0AR2 register alternatively on each frame

Bits 15:13 **LINEMULT[2:0]**: Amount of capture completed lines for LINE event and interrupt

0x0: Event after every line

0x1: Event after two lines

0x2: Event after four lines

0x3: Event after eight lines

0x4: Event after sixteen lines

0x5: Event after 32 lines

0x6: Event after 64 lines

0x7: Event after 128 lines

Bits 12:5 Reserved, must be kept at reset value.

Bit 4 **SWAPRB**: Swaps R-vs-B components if RGB, and if YUV, swaps U-vs-V components
0: No swap of R-vs-B (U-vs-V)
1: Swap active.

Bits 3:0 **FORMAT[3:0]**: Memory format (only coplanar formats are supported in Pipe2)
0x0: RG/B888 or YUV444 1-buffer
0x1: RGB565 1-buffer
0x2: ARGB8888
0x3: RGBA8888
0x4: monochrome Y8 or G8 1-buffer
0x5: YUV444 1-buffer (FOURCC = AYUV)
0x6: YUV422 1-buffer (FOURCC = YUYV)
Other formats are reserved and not supported.

39.14.152 DCMIPP Pipe2 current pixel packer Memory0 address register 1 (DCMIPP_P2CPPM0AR1)

Address offset: 0xFC4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M0A[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M0A[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **M0A[31:0]**: Memory0 address
Base address of current memory area 0, to whom data are written. It is assumed to be a multiple of 16, hence its bits 3:0 are always at 0x0.

39.14.153 DCMIPP Pipe2 current pixel packer Memory0 address register 2 (DCMIPP_P2CPPM0AR2)

Address offset: 0xFC8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M0A[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M0A[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **M0A[31:0]**: Memory0 address
Base address of current memory area 0, to whom data are written. It is assumed to be a multiple of 16, hence its bits 3:0 are always at 0x0.

39.15 DCMIPP register map

Table 367. DCMIPP register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**DCMIPP_IPGR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	QOS_MODE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEMORYPAGE[2:0]		
	Reset value								0																						0	1	0
0x004	**DCMIPP_IPGR2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSTART
	Reset value																																0
0x008	**DCMIPP_IPGR3**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDLE
	Reset value																																1
0x00C-0x018	Reserved	Reserved																															
0x01C	**DCMIPP_IPGR8**	IPPID[7:0]								Res.	Res.	Res.	ARCHIID[4:0]					Res.	Res.	Res.	REVID[4:0]					Res.	Res.	DID[5:0]					
	Reset value	1	0	1	0	1	0	1	0				0	0	1	0	0				0	0	0	1	1			0	1	0	1	0	0
0x020	**DCMIPP_IPC1R1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTR[3:0]				Res.	Res.	Res.	Res.	Res.	TRAFFIC[2:0]		
	Reset value																					0	0	0	0						1	0	0
0x028	**DCMIPP_IPC1R3**	Res.	Res.	Res.	Res.	Res.	Res.	DPREGEND[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	DPREGSTART[9:0]									
	Reset value							0	0	0	1	1	1	1	1	1	1							0	0	0	0	0	0	0	0	0	0
0x02C	Reserved	Reserved																															
0x030	**DCMIPP_IPC2R1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTR[3:0]				Res.	Res.	Res.	Res.	Res.	TRAFFIC[2:0]		
	Reset value																					0	0	0	0						1	0	0
0x034	**DCMIPP_IPC2R2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WLRU[3:0]				Res.	Res.	Res.	Res.	SVC MAPPING[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value													0	0	0	1					0	0	0	0								
0x038	**DCMIPP_IPC2R3**	Res.	Res.	Res.	Res.	Res.	Res.	DPREGEND[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	DPREGSTART[9:0]									
	Reset value							0	1	0	0	1	1	1	1	1	1							0	0	1	0	0	0	0	0	0	0
0x03C	Reserved	Reserved																															
0x040	**DCMIPP_IPC3R1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTR[3:0]				Res.	Res.	Res.	Res.	Res.	TRAFFIC[2:0]		
	Reset value																					0	0	0	0						1	0	0
0x044	**DCMIPP_IPC3R2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WLRU[3:0]				Res.	Res.	Res.	Res.	SVC MAPPING[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value													0	0	0	1					0	0	0	0								
0x048	**DCMIPP_IPC3R3**	Res.	Res.	Res.	Res.	Res.	Res.	DPREGEND[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	DPREGSTART[9:0]									
	Reset value							0	1	1	0	0	0	1	1	1	1							0	1	0	1	0	0	0	0	0	0
0x04C	Reserved	Reserved																															
0x050	**DCMIPP_IPC4R1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTR[3:0]				Res.	Res.	Res.	Res.	Res.	TRAFFIC[2:0]		
	Reset value																					0	0	0	0						1	0	0

Table 367. DCMIPP register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x054	**DCMIPP_IPC4R2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WLRU[3:0]								SVC MAPPING [3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value													0	0	0	1					0	0	0	0								
0x058	**DCMIPP_IPC4R3**	Res.	Res.	Res.	Res.	Res.	Res.	DPREGEND[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	DPREGSTART[9:0]								
	Reset value							0	1	1	0	1	1	1	1	1	1							0	1	1	0	0	1	0	0	0	0
0x05C	Reserved	Reserved																															
0x060	**DCMIPP_IPC5R1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OTR[3:0]				Res.	Res.	Res.	Res.	Res.	TRAFFIC [2:0]		
	Reset value																					0	0	0	0						1	0	0
0x064	**DCMIPP_IPC5R2**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WLRU[3:0]								SVC MAPPING [3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value													0	0	0	1					0	0	0	0								
0x068	**DCMIPP_IPC5R3**	Res.	Res.	Res.	Res.	Res.	Res.	DPREGEND[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	DPREGSTART[9:0]								
	Reset value							1	0	0	1	1	1	1	1	1	1							0	1	1	1	0	0	0	0	0	0
0x06C to 0x100	Reserved																																
0x100	**DCMIPP _PRHWCFGR**	PRHWCFGR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x104	**DCMIPP_PRCR**	Res.	Res.	Res.	Res.	Res.	SWAPBITS	SWAPCYCLES	Res.	FORMAT[7:0]								Res.	ENABLE	Res.	EDM [2:0]			Res.	Res.	VSPOL	HSPOL	PCKPOL	ESS	Res.	Res.	Res.	Res.
	Reset value						0	0		0	0	0	0	0	0	0	0		0		0	0	0			0	0	0	0				
0x108	**DCMIPP_PRESCR**	FEC[7:0]								LEC[7:0]								LSC[7:0]								FSC[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10C	**DCMIPP_PRESUR**	FEU[7:0]								LEU[7:0]								LSU[7:0]								FSU[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x110 to 0x1F0	Reserved	Reserved																															
0x1F4	**DCMIPP_PRIER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERRIE	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																										0						
0x1F8	**DCMIPP_PRSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VSYNC	HSYNC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERRF	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value															1	1										0						
0x1FC	**DCMIPP_PRFCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CERRF	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																										0						
0x200	Reserved	Reserved																															
0x204	**DCMIPP_CMCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWAPRB	Res.	Res.	CFC	Res.	PSFC [1:0]		INSEL
	Reset value																									0			0		0	0	0
0x208	**DCMIPP_CMFRCR**	FRMCNT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20C-0x3EC	Reserved	Reserved																															

Table 367. DCMIPP register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x3F0	**DCMIPP_CMIER**	P2OVRIE	Res.	Res.	Res.	Res.	P2VSYNCIE	P2FRAMEIE	P2LINEIE	P1OVRIE	Res.	Res.	Res.	Res.	P1VSYNCIE	P1FRAMEIE	P1LINEIE	P0OVRIE	P0LIMITIE	Res.	Res.	Res.	P0VSYNCIE	P0FRAMEIE	P0LINEIE	Res.	PRERRIE	ATXERRIE	Res.	Res.	Res.	Res.	Res.
	Reset value	0					0	0	0	0					0	0	0	0	0				0	0	0		0	0					
0x3F4	**DCMIPP_CMSR1**	P2CPTACT	Res.	Res.	Res.	Res.	Res.	P2LSTFRM	P2LSTLINE	P1CPTACT	Res.	Res.	Res.	Res.	Res.	P1LSTFRM	P1LSTLINE	P0CPTACT	Res.	Res.	Res.	Res.	Res.	P0LSTFRM	P0LSTLINE	Res.	Res.	Res.	Res.	Res.	Res.	PRVSYNC	PRHSYNC
	Reset value	0						0	0	0						0	0	0						0	0							1	1
0x3F8	**DCMIPP_CMSR2**	P2OVRF	Res.	Res.	Res.	Res.	P2VSYNCF	P2FRAMEF	P2LINEF	P1OVRF	Res.	Res.	Res.	Res.	P1VSYNCF	P1FRAMEF	P1LINEF	P0OVRF	P0LIMITF	Res.	Res.	Res.	P0VSYNCF	P0FRAMEF	P0LINEF	Res.	PRERRF	ATXERRF	Res.	Res.	Res.	Res.	Res.
	Reset value	0					0	0	0	0					0	0	0	0	0				0	0	0		0	0					
0x3FC	**DCMIPP_CMFCR**	CP2OVRF	Res.	Res.	Res.	Res.	CP2VSYNCF	CP2FRAMEF	CP2LINEF	CP1OVRF	Res.	Res.	Res.	Res.	CP1VSYNCF	P1FRAMEF	CP1LINEF	CP0OVRF	CP0LIMITF	Res.	Res.	Res.	CP0VSYNCF	CP0FRAMEF	CP0LINEF	Res.	CPRERRF	CATXERRF	Res.	Res.	Res.	Res.	Res.
	Reset value	0					0	0	0	0					0	0	0	0	0				0	0	0		0	0					
0x400	Reserved	Reserved																															
0x404	**DCMIPP_P0FSCR**	PIPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VC [1:0]		Res.	DTMODE [1:0]		Res.	Res.	DTIDB[5:0]							Res.	Res.	DTIDA[5:0]					
	Reset value	0											0	0		0	0			0	0	0	0	0	0				0	0	0	0	0	0
0x408-0x4FC	Reserved																																
0x500	**DCMIPP_P0FCTCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPTREQ	CPTMODE	FRATE [1:0]	
	Reset value																													0	0	0	0
0x504	**DCMIPP_P0SCSTR**	Res.	Res.	Res.	Res.	VSTART[11:0]												Res.	Res.	Res.	Res.	HSTART[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x508	**DCMIPP_P0SCSZR**	ENABLE	POSNEG	Res.	Res.	VSIZE[11:0]												Res.	Res.	Res.	Res.	HSIZE[11:0]											
	Reset value	0	0			0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x50C-0x5AC	Reserved	Reserved																															
0x5B0	**DCMIPP_P0DCCNTR**	Res.	Res.	Res.	Res.	Res.	Res.	CNT[25:0]																									
	Reset value							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5B4	**DCMIPP_P0DCLMTR**	ENABLE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LIMIT[23:0]																							
	Reset value	0								1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x5B8 - 0x5BC	Reserved	Reserved																															
0x5C0	**DCMIPP_P0PPCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LMAWE	LMAWM[2:0]		DBM	LINEMULT [2:0]			Res.	OELS	LSM	OEBS	BSM[1:0]		HEADEREN	PAD	Res.	Res.	Res.	Res.	SWAPYUV	
	Reset value												0	0	0	0	0	0	0		0	0	0	0	0	0	0					0	

Table 367. DCMIPP register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x5C4	**DCMIPP _P0PPM0AR1**	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5C8	**DCMIPP _P0PPM0AR2**	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5D0	**DCMIPP _P0STM0AR**	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5D4-0x5F0	Reserved	Reserved																															
0x5F4	**DCMIPP_P0IER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVRIE	LIMITIE	Res.	Res.	Res.	VSYNCIE	FRAMEIE	LINEIE
	Reset value																									0	0				0	0	0
0x5F8	**DCMIPP_P0SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPTACT	Res.	Res.	Res.	Res.	Res.	LSTFRM	LSTLINE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVRF	LIMITF	Res.	Res.	Res.	VSYNCF	FRAMEF	LINEF
	Reset value									0						0	0									0	0				0	0	0
0x5FC	**DCMIPP_P0FCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COVRF	CLIMITF	Res.	Res.	Res.	CVSYNCF	CFRAMEF	CLINEF
	Reset value																									0	0				0	0	0
0x600	Reserved	Reserved																															
0x604	**DCMIPP_P0CFSCR**	PIPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VC [1:0]		Res.	DTMODE [1:0]		Res.	Res.	DTIDB[5:0]						Res.	Res.	DTIDA[5:0]						
	Reset value	0											0	0		0	0			0	0	0	0	0	0			0	0	0	0	0	0
0x608-0x6FC	Reserved	Reserved																															
0x700	**DCMIPP_P0CFCTCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPTREQ	CPTMODE	FRATE [1:0]	
	Reset value																													0	0	0	0
0x704	**DCMIPP_P0CSCSTR**	Res.	Res.	Res.	Res.	VSTART[11:0]												Res.	Res.	Res.	Res.	HSTART[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x708	**DCMIPP_P0CSCSZR**	ENABLE	POSNEG	Res.	Res.	VSIZE[11:0]												Res.	Res.	Res.	Res.	HSIZE[11:0]											
	Reset value	0	0			0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x70C-0x7BC	Reserved	Reserved																															
0x7C0	**DCMIPP_P0CPPCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LMAWE	LMAWM[2:0]		DBM	LINEMULT [2:0]			Res.	OELS	LSM	OEBS	BSM[1:0]		HEADEREN	PAD	Res.	Res.	Res.	Res.		SWAPYUV
	Reset value												0	0	0	0	0	0	0		0	0	0	0	0	0	0						0
0x7C4	**DCMIPP _P0CPPM0AR1**	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x7C8	DCMIPP _P0CPPM0AR2	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x7CC - 0x800	Reserved	Reserved																															

Table 367. DCMIPP register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x804	**DCMIPP_P1FSCR**	PIPEN	FDTFEN	FDTF[5:0]						Res.	Res.	Res.	VC [1:0]		PIPEDIFF	DTMODE [1:0]		Res.	Res.	DTIDB[5:0]						Res.	Res.	DTIDA[5:0]					
	Reset value	0	0	0	0	0	0	0	0				0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0
0x808-0x81C	Reserved	Reserved																															
0x820	**DCMIPP_P1SRCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CROPEN	FIRSTLINE DEL[2:0]			LASTLINE[11:0]											
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x824	**DCMIPP_P1BPRCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRENGHT [2:0]			ENABLE
	Reset value																													0	0	0	0
0x828	**DCMIPP_P1BPRSR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BADCNT[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x82C	Reserved	Reserved																															
0x830	**DCMIPP_P1DECR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VDEC [1:0]		HDEC [1:0]		ENABLE
	Reset value																												0	0	0	0	0
0x834-0x83C	Reserved	Reserved																															
0x840	**DCMIPP_P1BLCCR**	BLCR[7:0]								BLCG[7:0]								BLCB[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENABLE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								0
0x844	**DCMIPP_P1EXCR1**	Res.	SHFR [2:0]			MULTR[7:0]									Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENABLE
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0																			0
0x848	**DCMIPP_P1EXCR2**	Res.	SHFG [2:0]			MULTG[7:0]									Res.	Res.	Res.	Res.	Res.	SHFB [2:0]						MULTB[7:0]				Res.	Res.	Res.	Res.
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0				
0x84C	Reserved	Reserved																															
0x850	**DCMIPP_P1ST1CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE	SRC [2:0]			BINS [1:0]		Res.	ENABLE
	Reset value																									0	0	0	0	0	0		0
0x854	**DCMIPP_P1ST2CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE	SRC [2:0]			BINS [1:0]		Res.	ENABLE
	Reset value																									0	0	0	0	0	0		0
0x858	**DCMIPP_P1ST3CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE	SRC [2:0]			BINS [1:0]		Res.	ENABLE
	Reset value																									0	0	0	0	0	0		0
0x85C	**DCMIPP_P1STSTR**	Res.	Res.	Res.	Res.	VSTART[11:0]												Res.	Res.	Res.	Res.	HSTART[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x860	**DCMIPP_P1STSZR**	CROPEN	Res.	Res.	Res.	VSIZE[11:0]												Res.	Res.	Res.	Res.	HSIZE[11:0]											
	Reset value	0				0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0

Table 367. DCMIPP register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x864	DCMIPP_P1ST1SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACCU[23:0]																							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x868	**DCMIPP_P1ST2SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACCU[23:0]																							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x86C	**DCMIPP_P1ST3SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ACCU[23:0]																							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x870	**DCMIPP_P1DMCR**	Res.	EDGE [2:0]			Res.	LINEH [2:0]			Res.	LINEV [2:0]			Res.	PEAK [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TYPE [1:0]		ENABLE
	Reset value		0	0	0		0	0	0		0	0	0		0	0	0														0	0	0
0x874-0x87C	Reserved	Reserved																															
0x880	**DCMIPP_P1CCCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAMP	TYPE	ENABLE
	Reset value																														0	0	0
0x884	**DCMIPP_P1CCRR1**	Res.	Res.	Res.	Res.	Res.	RG[10:0]											Res.	Res.	Res.	Res.	Res.	RR[10:0]										
	Reset value						0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x888	**DCMIPP_P1CCRR2**	Res.	Res.	Res.	Res.	Res.	Res.	RA[9:0]										Res.	Res.	Res.	Res.	Res.	RB[10:0]										
	Reset value							0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x88C	**DCMIPP_P1CCGR1**	Res.	Res.	Res.	Res.	Res.	GG[10:0]											Res.	Res.	Res.	Res.	Res.	GR[10:0]										
	Reset value						0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x890	**DCMIPP_P1CCGR2**	Res.	Res.	Res.	Res.	Res.	Res.	GA[9:0]										Res.	Res.	Res.	Res.	Res.	GB[10:0]										
	Reset value							0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x894	**DCMIPP_P1CCBR1**	Res.	Res.	Res.	Res.	Res.	BG[10:0]											Res.	Res.	Res.	Res.	Res.	BR[10:0]										
	Reset value						0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x898	**DCMIPP_P1CCBR2**	Res.	Res.	Res.	Res.	Res.	Res.	BA[9:0]										Res.	Res.	Res.	Res.	Res.	BB[10:0]										
	Reset value							0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x89C	Reserved	Reserved																															
0x8A0	**DCMIPP_P1CTCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LUM0[5:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENABLE
	Reset value																		0	1	0	0	0	0									0
0x8A4	**DCMIPP_P1CTCR2**	Res.	LUM1[5:0]						Res.	Res.	LUM2[5:0]							Res.	Res.	LUM3[5:0]						Res.	Res.	LUM4[5:0]						Res.
	Reset value		0	1	0	0	0	0			0	1	0	0	0	0			0	1	0	0	0	0			0	1	0	0	0	0	
0x8A8	**DCMIPP_P1CTCR3**	Res.	LUM5[5:0]						Res.	Res.	LUM6[5:0]							Res.	Res.	LUM7[5:0]						Res.	Res.	LUM8[5:0]						Res.
	Reset value		0	1	0	0	0	0			0	1	0	0	0	0			0	1	0	0	0	0			0	1	0	0	0	0	
0x8AC-0x8FC	Reserved	Reserved																															
0x900	**DCMIPP_P1FCTCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPTREQ	CPTMODE	FRATE [1:0]	
	Reset value																													0	0	0	0
0x904	**DCMIPP_P1CRSTR**	Res.	Res.	Res.	Res.	VSTART[11:0]												Res.	Res.	Res.	Res.	HSTART[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0

Table 367. DCMIPP register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x908	**DCMIPP_P1CRSZR**	ENABLE	Res.	Res.	Res.	VSIZE[11:0]												Res.	Res.	Res.	Res.	HSIZE[11:0]											
	Reset value	0				0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x90C	**DCMIPP_P1DCCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VDEC [1:0]		HDEC [1:0]		ENABLE
																													0	0	0	0	0
0x910	**DCMIPP_P1DSCR**	ENABLE	Res.	Res.	Res.	Res.	Res.	VDIV[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	HDIV[9:0]									
	Reset value	0						0	0	0	0	0	0	0	0	0	0							0	0	0	0	0	0	0	0	0	0
0x914	**DCMIPP_P1DSRTIOR**	VRATIO[15:0]																HRATIO[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x918	**DCMIPP_P1DSSZR**	Res.	Res.	Res.	Res.	VSIZE[11:0]												Res.	Res.	Res.	Res.	HSIZE[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x920	**DCMIPP_P1CMRICR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ROI8EN	ROI7EN	ROI6EN	ROI5EN	ROI4EN	ROI3EN	ROI2EN	ROI1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ROILSZ [1:0]	
	Reset value									0	0	0	0	0	0	0	0															0	0
0x924	**DCMIPP_P1RIxCR1**	Res.	Res.	CLR [1:0]		VSTART[11:0]												CLG [1:0]		CLB [1:0]		HSTART[11:0]											
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x928	DCMIPP_P1RIxCR2	Res.	Res.	Res.	Res.	VSIZE[11:0]												Res.	Res.	Res.	Res.	HSIZE[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x92C-0x96C	Reserved	Reserved																															
0x970	**DCMIPP_P1GMCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENABLE
	Reset value																																0
0x974-0x97C	Reserved	Reserved																															
0x980	**DCMIPP_P1YUVCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAMP	TYPE	ENABLE
	Reset value																														0	0	0
0x984	**DCMIPP_P1YUVRR1**	Res.	Res.	Res.	Res.	Res.	Res.	RG[10:0]										Res.	Res.	Res.	Res.	Res.	RR[10:0]										
	Reset value						0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x988	**DCMIPP_P1YUVRR2**	Res.	Res.	Res.	Res.	Res.	Res.	RA[9:0]										Res.	Res.	Res.	Res.	Res.	RB[10:0]										
	Reset value							0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x98C	**DCMIPP_P1YUVGR1**	Res.	Res.	Res.	Res.	Res.	Res.	GG[10:0]										Res.	Res.	Res.	Res.	Res.	GR[10:0]										
	Reset value						0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x990	**DCMIPP_P1YUVGR2**	Res.	Res.	Res.	Res.	Res.	Res.	GA[9:0]										Res.	Res.	Res.	Res.	Res.	GB[10:0]										
	Reset value							0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x994	**DCMIPP_P1YUVBR1**	Res.	Res.	Res.	Res.	Res.	Res.	BG[10:0]										Res.	Res.	Res.	Res.	Res.	BR[10:0]										
	Reset value						0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x998	**DCMIPP_P1YUVBR2**	Res.	Res.	Res.	Res.	Res.	Res.	BA[9:0]										Res.	Res.	Res.	Res.	Res.	BB[10:0]										
	Reset value							0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x99C-0x9BC	Reserved	Reserved																															

Table 367. DCMIPP register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x9C0	**DCMIPP_P1PPCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LMAWE	LMAWM[2:0]			DBM	LINEMULT[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWAPRB	FORMAT[3:0]			
	Reset value												0	0	0	0	0	0	0	0									0	0	0	0	0
0x9C4	**DCMIPP_P1PPM0AR1**	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x9C8	**DCMIPP_P1PPM0AR2**	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x9CC	**DCMIPP_P1PPM0PR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PITCH[14:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x9D0	**DCMIPP_P1STM0A**	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x9D4	**DCMIPP_P1PPM1AR1**	M1A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x9D8	**DCMIPP_P1PPM1AR2**	M1A[31:0]																															
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x9DC	**DCMIPP_P1PPM1PR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PITCH[14:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x9E0	**DCMIPP_P1STM1AR**	M1A[31:0]																															
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x9E4	**DCMIPP_P1PPM2AR1**	M2A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x9E8	**DCMIPP_P1PPM2AR2**	M2A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x9EC	Reserved																																
	Reset value																																
0x9F0	**DCMIPP_P1STM2AR**	M2A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x9F4	**DCMIPP_P1IER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVRIE	Res.	Res.	Res.	Res.	VSYNCIE	FRAMEIE	LINEIE
	Reset value																									0					0	0	0
0x9F8	**DCMIPP_P1SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPTACT	Res.	Res.	Res.	Res.	Res.	LSTFRM	LSTLINE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVRF	Res.	Res.	Res.	Res.	VSYNCF	FRAMEF	LINEF
	Reset value									0						0	0									0					0	0	0
0x9FC	**DCMIPP_P1FCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COVRF	Res.	Res.	Res.	Res.	CVSYNCF	CFRAMEF	CLINEF
	Reset value																									0					0	0	0
0xA00	Reserved	Reserved																															
0xA04	**DCMIPP_P1CFSCR**	PIPEN	FDTFEN	FDTF[5:0]						Res.	Res.	Res.	VC[1:0]		PIPEDIFF	DTMODE[1:0]		Res.	Res.	DTIDB[5:0]						Res.	Res.	DTIDA[5:0]					
	Reset value	0	0	0	0	0	0	0	0				0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0
0xA08-0xA20	Reserved	Reserved																															

Table 367. DCMIPP register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xA24	**DCMIPP_P1CBPRCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STRENGTH [2:0]			ENABLE
	Reset value																													0	0	0	0
0xA28-0xA3C	Reserved	Reserved																															
0xA40	**DCMIPP_P1CBLCCR**	BLCR[7:0]								BLCG[7:0]								BLCB[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENABLE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								0
0xA44	**DCMIPP_P1CEXCR1**	Res.	SHFR [2:0]			MULTR[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENABLE
	Reset value		0	0	0	0	0	0	0	0	0	0	0																				0
0xA48	**DCMIPP_P1CEXCR2**	Res.	SHFG [2:0]			MULTG[7:0]								Res.	Res.	Res.	Res.	Res.	SHFB [2:0]			MULTB[7:0]								Res.	Res.	Res.	Res.
	Reset value		0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0				
0xA4C	Reserved	Reserved																															
0xA50	**DCMIPP_P1CST1CR**	ACCU[23:0]																								MODE	SRC [2:0]			BINS [1:0]		Res.	ENABLE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
0xA54	**DCMIPP_P1CST2CR**	ACCU[23:0]																								MODE	SRC [2:0]			BINS [1:0]		Res.	ENABLE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
0xA58	**DCMIPP_P1CST3CR**	ACCU[23:0]																								MODE	SRC [2:0]			BINS [1:0]		Res.	ENABLE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
0xA5C	**DCMIPP_P1CSTSTR**	Res.	Res.	Res.	Res.	VSTART[11:0]												Res.	Res.	Res.	Res.	HSTART[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0xA60	**DCMIPP_P1CSTSZR**	CROPEN	Res.	Res.	Res.	VSIZE[11:0]												Res.	Res.	Res.	Res.	HSIZE[11:0]											
	Reset value	0				0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0xA64-0xA7C	Reserved	Reserved																															
0xA80	**DCMIPP_P1CCCCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAMP	TYPE	ENABLE
	Reset value																														0	0	0
0xA84	**DCMIPP_P1CCCRR1**	Res.	Res.	Res.	Res.	Res.	RG[10:0]											Res.	Res.	Res.	Res.	Res.	RR[10:0]										
	Reset value						0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0xA88	**DCMIPP_P1CCCRR2**	Res.	Res.	Res.	Res.	Res.	Res.	RA[9:0]										Res.	Res.	Res.	Res.	Res.	RB[10:0]										
	Reset value							0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0xA8C	**DCMIPP_P1CCCGR1**	Res.	Res.	Res.	Res.	Res.	GG[10:0]											Res.	Res.	Res.	Res.	Res.	GR[10:0]										
	Reset value						0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0xA90	**DCMIPP_P1CCCGR2**	Res.	Res.	Res.	Res.	Res.	Res.	GA[9:0]										Res.	Res.	Res.	Res.	Res.	GB[10:0]										
	Reset value							0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0

Table 367. DCMIPP register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xA94	**DCMIPP_P1CCCBR1**	Res.	Res.	Res.	Res.	Res.	BG[10:0]											Res.	Res.	Res.	Res.	Res.	BR[10:0]										
	Reset value						0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0xA98	**DCMIPP_P1CCCBR2**	Res.	Res.	Res.	Res.	Res.	Res.	BA[9:0]										Res.	Res.	Res.	Res.	Res.	BB[10:0]										
	Reset value							0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0xA9C	Reserved	Reserved																															
0xAA0	**DCMIPP_P1CCTCR1**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LUM0[5:0]														ENABLE
	Reset value																		0	1	0	0	0	0									0
0xAA4	**DCMIPP_P1CCTCR2**	Res.	LUM1[5:0]						Res.	Res.	LUM2[5:0]							Res.	Res.	LUM3[5:0]						Res.	Res.	LUM4[5:0]						Res.
	Reset value		0	1	0	0	0	0			0	1	0	0	0	0			0	1	0	0	0	0			0	1	0	0	0	0	
0xAA8	**DCMIPP_P1CCTCR3**	Res.	LUM5[5:0]						Res.	Res.	LUM6[5:0]							Res.	Res.	LUM7[5:0]						Res.	Res.	LUM8[5:0]						Res.
	Reset value		0	1	0	0	0	0			0	1	0	0	0	0			0	1	0	0	0	0			0	1	0	0	0	0	
0xAAC-0xAFC	Reserved	Reserved																															
0xB00	**DCMIPP_P1CFCTCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPTREQ	CPTMODE	FRATE [1:0]	
	Reset value																													0	0	0	0
0xB04	**DCMIPP_P1CCRSTR**	Res.	Res.	Res.	Res.	VSTART[11:0]												Res.	Res.	Res.	Res.	HSTART[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0xB08	**DCMIPP_P1CCRSZR**	ENABLE	Res.	Res.	Res.	VSIZE[11:0]												Res.	Res.	Res.	Res.	HSIZE[11:0]											
	Reset value	0				0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0xB0C	**DCMIPP_P1CDCCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VDEC [1:0]		HDEC [1:0]		ENABLE
																													0	0	0	0	0
0xB10	**DCMIPP_P1CDSCR**	ENABLE	Res.	Res.	Res.	Res.	Res.	VDIV[9:0]										Res.	Res.	Res.	Res.	Res.	HDIV[9:0]										
	Reset value	0						0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0xB14	**DCMIPP_P1CDSRTIOR**	VRATIO[15:0]																HRATIO[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB18	**DCMIPP_P1CDSSZR**	Res.	Res.	Res.	Res.	VSIZE[11:0]												Res.	Res.	Res.	Res.	HSIZE[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0xB20	**DCMIPP_P1CCMRICR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ROI8EN	ROI7EN	ROI6EN	ROI5EN	ROI4EN	ROI3EN	ROI2EN	ROI1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ROILSZ [1:0]	
	Reset value									0	0	0	0	0	0	0	0															0	0
0xB24	**DCMIPP_P1CRIxCR1**	Res.	Res.	CLR [1:0]		VSTART[11:0]												CLG [1:0]		CLB [1:0]		HSTART[11:0]											
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB28	**DCMIPP_P1CRI1CR2**	Res.	Res.	Res.	Res.	VSIZE[11:0]												Res.	Res.	Res.	Res.	HSIZE[11:0]											
						0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0xB2-CxBBC	Reserved	Reserved																															

Table 367. DCMIPP register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xBC0	**DCMIPP_P1CPPCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LMAWE	LMAWM[2:0]			DBM	LINEMULT[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWAPRB	FORMAT[3:0]			
	Reset value												0	0	0	0	0	0	0	0									0	0	0	0	0
0xBC4	**DCMIPP_P1CPPM0AR1**	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xBC8	**DCMIPP_P1CPPM0AR2**	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xBCC	**DCMIPP_P1CPPM0PR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PITCH[14:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xBD0	Reserved																																
0xBD4	**DCMIPP_P1CPPM1AR1**	M1A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xBD8	**DCMIPP_P1CPPM1AR2**	M1A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xBDC	**DCMIPP_P1CPPM1PR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PITCH[14:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xBE0	Reserved																																
0xBE4	**DCMIPP_P1CPPM2AR1**	M2A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xBE8	**DCMIPP_P1CPPM2AR2**	M2A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xBC - 0xC00	Reserved	Reserved																															
0xC04	**DCMIPP_P2FSCR**	PIPEN	FDTFEN	FDTF[5:0]						Res.	Res.	Res.	VC[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTIDA[5:0]					
	Reset value	0	0	0	0	0	0	0	0				0	0															0	0	0	0	0	0
0xC08 - 0xCFC	Reserved	Reserved																															
0xD00	**DCMIPP_P2FCTCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPTREQ	CPTMODE	FRATE[1:0]	
	Reset value																													0	0	0	0
0xD04	**DCMIPP_P2CRSTR**	Res.	Res.	Res.	Res.	VSTART[11:0]												Res.	Res.	Res.	Res.	HSTART[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0
0xD08	**DCMIPP_P2CRSZR**	ENABLE	Res.	Res.	Res.	VSIZE[11:0]												Res.	Res.	Res.	Res.	HSIZE[11:0]											
	Reset value	0				0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0
0xD0C	**DCMIPP_P2DCCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VDEC[1:0]		HDEC[1:0]		ENABLE
																														0	0	0	0	0

Table 367. DCMIPP register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xD10	**DCMIPP_P2DSCR**	ENABLE	Res.	Res.	Res.	Res.	Res.	VDIV[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	HDIV[9:0]									
	Reset value	0						0	0	0	0	0	0	0	0	0	0							0	0	0	0	0	0	0	0	0	0
0xD14	**DCMIPP _P2DSRTIOR**	VRATIO[15:0]																HRATIO[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xD18	**DCMIPP_P2DSSZR**	Res.	Res.	Res.	Res.	VSIZE[11:0]												Res.	Res.	Res.	Res.	HSIZE[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0xD20	**DCMIPP _P1CCMRICR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ROI8EN	ROI7EN	ROI6EN	ROI5EN	ROI4EN	ROI3EN	ROI2EN	ROI1EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ROILSZ [1:0]	
	Reset value									0	0	0	0	0	0	0	0															0	0
0xD24	**DCMIPP_P2RI1CR1**	Res.	Res.	CLR [1:0]		VSTART[11:0]												CLG [1:0]		CLB [1:0]		HSTART[11:0]											
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xD28	**DCMIPP_P2RI1CR2**	Res.	Res.	Res.	Res.	VSIZE[11:0]												Res.	Res.	Res.	Res.	HSIZE[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0xD2C	**DCMIPP_P2RI2CR1**	Res.	Res.	CLR [1:0]		VSTART[11:0]												CLG [1:0]		CLB [1:0]		HSTART[11:0]											
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xD30	**DCMIPP_P2RI2CR2**	Res.	Res.	Res.	Res.	VSIZE[11:0]												Res.	Res.	Res.	Res.	HSIZE[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0xD34	**DCMIPP_P2RI3CR1**	Res.	Res.	CLR [1:0]		VSTART[11:0]												CLG [1:0]		CLB [1:0]		HSTART[11:0]											
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xD38	**DCMIPP_P2RI3CR2**	Res.	Res.	Res.	Res.	VSIZE[11:0]												Res.	Res.	Res.	Res.	HSIZE[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0xD3C	**DCMIPP_P2RI4CR1**	Res.	Res.	CLR [1:0]		VSTART[11:0]												CLG [1:0]		CLB [1:0]		HSTART[11:0]											
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xD40	**DCMIPP_P2RI4CR2**	Res.	Res.	Res.	Res.	VSIZE[11:0]												Res.	Res.	Res.	Res.	HSIZE[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0xD44	**DCMIPP_P2RI5CR1**	Res.	Res.	CLR [1:0]		VSTART[11:0]												CLG [1:0]		CLB [1:0]		HSTART[11:0]											
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xD48	**DCMIPP_P2RI5CR2**	Res.	Res.	Res.	Res.	VSIZE[11:0]												Res.	Res.	Res.	Res.	HSIZE[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0xD4C	**DCMIPP_P2RI6CR1**	Res.	Res.	CLR [1:0]		VSTART[11:0]												CLG [1:0]		CLB [1:0]		HSTART[11:0]											
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xD50	**DCMIPP_P2RI6CR2**	Res.	Res.	Res.	Res.	VSIZE[11:0]												Res.	Res.	Res.	Res.	HSIZE[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0xD54	**DCMIPP_P2RI7CR1**	Res.	Res.	CLR [1:0]		VSTART[11:0]												CLG [1:0]		CLB [1:0]		HSTART[11:0]											
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xD58	**DCMIPP_P2RI7CR2**	Res.	Res.	Res.	Res.	VSIZE[11:0]												Res.	Res.	Res.	Res.	HSIZE[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0

Table 367. DCMIPP register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xD5C	**DCMIPP_P2RI8CR1**	Res.	Res.	CLR[1:0]		VSTART[11:0]												CLG[1:0]		CLB[1:0]		HSTART[11:0]											
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xD60	**DCMIPP_P2RI8CR2**	Res.	Res.	Res.	Res.	VSIZE[11:0]												Res.	Res.	Res.	Res.	HSIZE[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0xD64-0xD6C	Reserved	Reserved																															
0xD70	**DCMIPP_P2GMCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENABLE
	Reset value																																0
0xD64-0xDBC	Reserved	Reserved																															
0xDC0	**DCMIPP_P2PPCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LMAWE	LMAWM[2:0]			DBM	LINEMULT[2:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWAPRB	FORMAT[3:0]		
	Reset value												0	0	0	0	0	0	0	0									0	0	0	0	0
0xDC4	**DCMIPP_P2PPM0AR1**	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xDC8	**DCMIPP_P2PPM0AR2**	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xDCC	**DCMIPP_P2PPM0PR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PITCH[14:0]														
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xDD0	**DCMIPP_P2STM0AR**	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xDD4-0xDF0	Reserved	Reserved																															
0xDF4	**DCMIPP_P2IER**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVRIE	Res.	Res.	Res.	Res.	VSYNCIE	FRAMEIE	LINEIE
	Reset value																									0					0	0	0
0xDF8	**DCMIPP_P2SR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPTACT	Res.	Res.	Res.	Res.	Res.	LSTFRM	LSTLINE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVRF	Res.	Res.	Res.	Res.	VSYNCF	FRAMEF	LINEF
	Reset value									0						0	0									0					0	0	0
0xDFC	**DCMIPP_P2FCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COVRF	Res.	Res.	Res.	Res.	CVSYNCF	CFRAMEF	CLINEF
	Reset value																									0					0	0	0
0xE00	Reserved	Reserved																															
0xE04	**DCMIPP_P2CFSCR**	PIPEN	FDTFEN	FDTF[5:0]						Res.	Res.	Res.	VC[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTIDA[5:0]				
	Reset value	0	0	0	0	0	0	0	0				0	0															0	0	0	0	0	0
0xE08-0xF20	Reserved	Reserved																															

Table 367. DCMIPP register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xF00	**DCMIPP_P2CFCTCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPTREQ	CPTMODE	FRATE [1:0]	
	Reset value																													0	0	0	0
0xF04	**DCMIPP_P2CCRSTR**	Res.	Res.	Res.	Res.	VSTART[11:0]												Res.	Res.	Res.	Res.	HSTART[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0xF08	**DCMIPP_P2CCRSZR**	ENABLE	Res.	Res.	Res.	VSIZE[11:0]												Res.	Res.	Res.	Res.	HSIZE[11:0]											
	Reset value	0				0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0xF0C	**DCMIPP_P2DCCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VDEC [1:0]		HDEC [1:0]		ENABLE
																													0	0	0	0	0
0xF10	**DCMIPP_P2CDSCR**	ENABLE	Res.	Res.	Res.	Res.	Res.	VDIV[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	HDIV[9:0]									
	Reset value	0						0	0	0	0	0	0	0	0	0	0							0	0	0	0	0	0	0	0	0	0
0xF14	**DCMIPP _P2CDSRTIOR**	VRATIO[15:0]																HRATIO[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xF18	**DCMIPP_P2CDSSZR**	Res.	Res.	Res.	Res.	VSIZE[11:0]												Res.	Res.	Res.	Res.	HSIZE[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0xF1C-0xF20	Reserved	Reserved																															
0xF24	**DCMIPP_P2CRI1CR1**	Res.	Res.	CLR [1:0]		VSTART[11:0]												CLG[1:0]		CLB [1:0]		HSTART[11:0]											
				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xF28	**DCMIPP_P2CRI1CR2**	Res.	Res.	Res.	Res.	VSTART[11:0]												Res.	Res.	Res.	Res.	HSTART[11:0]											
						0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0xF2C	**DCMIPP_P2CRI2CR1**			CLR [1:0]		VSTART[11:0]												CLG[1:0]		CLB [1:0]		HSTART[11:0]											
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xF28	**DCMIPP_P2CRI2CR2**	Res.	Res.	Res.	Res.	VSTART[11:0]												Res.	Res.	Res.	Res.	HSTART[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0xF34	**DCMIPP_P2CRI3CR1**			CLR [1:0]		VSTART[11:0]												CLG[1:0]		CLB [1:0]		HSTART[11:0]											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xF38	**DCMIPP_P2CRI3CR2**	Res.	Res.	Res.	Res.	VSTART[11:0]												Res.	Res.	Res.	Res.	HSTART[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0xF3C	**DCMIPP_P2CRI4CR1**			CLR [1:0]		VSTART[11:0]												CLG[1:0]		CLB [1:0]		HSTART[11:0]											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xF40	**DCMIPP_P2CRI4CR2**	Res.	Res.	Res.	Res.	VSTART[11:0]												Res.	Res.	Res.	Res.	HSTART[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0xF44	**DCMIPP_P2CRI5CR1**			CLR [1:0]		VSTART[11:0]												CLG[1:0]		CLB [1:0]		HSTART[11:0]											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 367. DCMIPP register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xF48	**DCMIPP_P2CRI5CR2**	Res.	Res.	Res.	Res.	VSTART[11:0]												Res.	Res.	Res.	Res.	HSTART[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0xF4C	**DCMIPP_P2CRI6CR1**			CLR [1:0]		VSTART[11:0]												CLG[1:0]		CLB [1:0]		HSTART[11:0]											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xF50	**DCMIPP_P2CRI6CR2**	Res.	Res.	Res.	Res.	VSTART[11:0]												Res.	Res.	Res.	Res.	HSTART[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0xF54	**DCMIPP_P2CRI7CR1**			CLR [1:0]		VSTART[11:0]												CLG[1:0]		CLB [1:0]		HSTART[11:0]											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xF58	**DCMIPP_P2CRI7CR2**	Res.	Res.	Res.	Res.	VSTART[11:0]												Res.	Res.	Res.	Res.	HSTART[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0xF5C	**DCMIPP_P2CRI8CR1**			CLR [1:0]		VSTART[11:0]												CLG[1:0]		CLB [1:0]		HSTART[11:0]											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xF60	**DCMIPP_P2CRI8CR2**	Res.	Res.	Res.	Res.	VSTART[11:0]												Res.	Res.	Res.	Res.	HSTART[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0xF64-0xFBC	Reserved	Reserved																															
0xFC0	**DCMIPP_P2CPPCR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LMAWE	LMAWM[2:0]			DBM	LINEMULT [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWAPRB	FORMAT [3:0]			
	Reset value												0	0	0	0	0	0	0	0									0	0	0	0	0
0xFC4	**DCMIPP _P2CPPM0AR1**	M0A[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xFC8	**DCMIPP _P2CPPM0AR2**	M0A[31:0]																															
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

40 CSI-2 Host (CSI)

40.1 CSI introduction

The camera serial interface 2 (CSI-2) is a part of a group of communication protocols defined by the MIPI® Alliance. The MIPI® CSI-2 Host controller is a digital core that implements all protocol functions defined in the MIPI® CSI-2 specification.

It provides an interface between the system and the MIPI® D-PHY, allowing communication with a CSI-2 compliant camera.

40.2 Standard and references

- MIPI Alliance Specification for Camera Serial Interface 2 (CSI-2) v1.3 - 29 May 2014
- MIPI Alliance Specification for D-PHY v1.2 - 01 August 2014

40.3 Glossary

Table 368. Glossary of terms

Term	Definition
D-PHY_RX	D-PHY physical layer receiver
EOF	End of frame
EOT	End of transmission
ESC	Escape mode for the D-PHY_RX
HS	High speed
ISB	Imaging system bus
LLP	Low-level protocol
LP	Low power
PHY	Physical layer
PPI	Phy protocol interface
RX	Receiver
SOF	Start of frame
SOT	Start of transmission

40.4 CSI-2 Host main features

- Compliant with MIPI Alliance standard (see *Section 40.2*)
- Interface with MIPI D-PHY (see *Section 40.2*)
- Up to two D-PHY data lanes supported (up to 2.5 Gbit/s per lane in high-speed (HS) mode and 10 Mbit/s in low-power (LP) mode)
- Data transmission in HS and LP modes supported by the D-PHY
- Escape mode (ESC) and ultra-low-power state mode (ULPS) supported
- CSI-2 lane management layer connected to the D-PHY_RX to merge 1 or 2 data lanes in a single 32-bit ISB-Byte bus
- ECC and error correction capabilities
- CRC check capability
- CSI-2 virtual channel and data type filtering supporting interleaved data
  - Up to 4 virtual channels
  - Support the data formats specified into the MIPI Alliance standard for CSI-2 v1.3 (18 data formats, plus the user defined ones)
- Up to 7 independent data types
- Programmable interrupts:
  - 4 timer interrupts mapped on a selected virtual channel with starting point from SOF (start of frame) or EOF (end of frame)
  - 4 line/byte counter interrupts mapped on a selected virtual channel to trigger an event
- ISB-Byte header generation for internal connection with the DCMIPP peripheral

40.5 CSI-2 Host functional description

40.5.1 General description

The MIPI CSI-2 Host controller offers a means to connect cameras to capture and dispatch a robust header with ECC correction and data payload with checksum.
A D-PHY_RX receiver fills out the role of interfacing the external serial camera devices with the embedded CSI-2 Host controller:

- D-PHY receiver:

  This is a D-PHY_RX receiver composed of two unidirectional data lanes, and one clock lane. It operates in slave mode with HS and LP mode capabilities. The D-PHY drives a PPI interface to communicate with the CSI-2 Host.
- CSI-2 lane merger:

  Up to two data lane inputs to increase data rate while avoiding high clock rates

  The lane manager function collects incoming bytes from N (N = 1 to 2) lanes.
  It consolidates them by merging operations into a complete packet provided to the CSI-2 low-level protocol decoder (LLP), to decompose the meaning of each data packet thanks to the CSI-2 protocol specification.
- CSI-2 clock changer:

  The clock changer moves the data transfer from an input clock domain from D-PHY_RX to a clock_proc domain before being processed by the LLP.

- CSI-2 low-level protocol decoder (LLP)

  The low-level protocol is a byte-orientated packet-based protocol that supports the transport of arbitrary data using short and long packet formats.

  It generates an ISB-byte header data for further processing, with data integrity check or miscellaneous integrity checks.

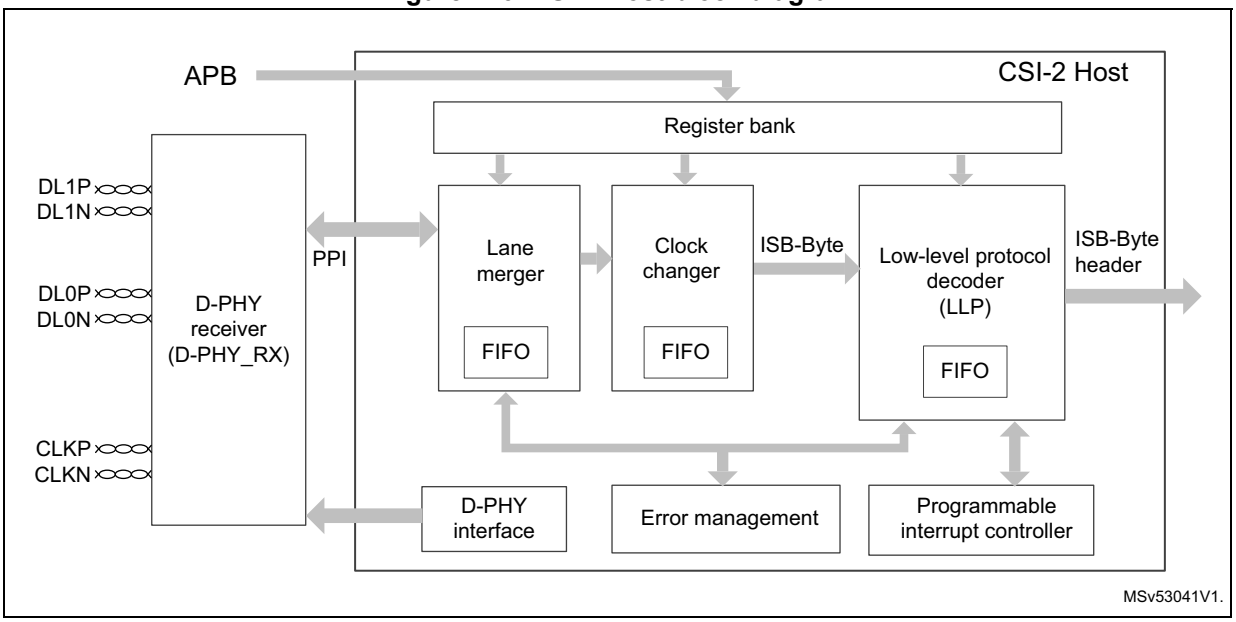
  It supports data interleaving through the support of up to four virtual channels and many data type formats.

  The LLP offers interrupt generation capability, depending on specific events such as line/byte detection within a frame, or a time-base event selected from a SOF or an EOF.

The CSI-2 Host-controller register bank is configured and accessed by the APB bus interface.

A block diagram of the CSI-2 Host is shown in the figure below.

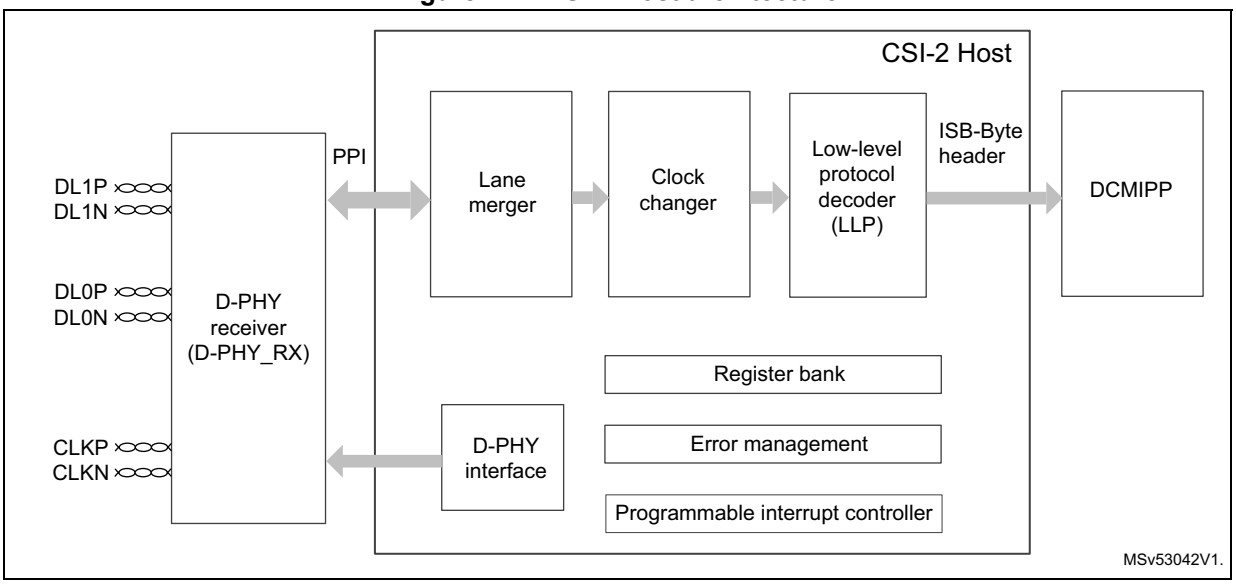
Figure 416. CSI-2 Host block diagram



40.5.2 System level architecture

The figure below shows the CSI-2 Host architecture.

Figure 417. CSI-2 Host architecture



The blocks shown have the following functions:

- D-PHY_RX is a slave receiver device that connects external serial cameras to the CSI-2 Host controller. It is based on unidirectional slave data lanes and clock lanes to handle data transmission from the CSI-2 protocol-based camera devices to subsystems for data processing (handled by the DCMIPP).
- The D-PHY interface manages D-PHY_RX register accesses. It also allows the block to be configured to raise interrupts on the detection of some errors during reception. It highlights the status of the physical lines up to the software layers.
- The lane merger module packs the data coming from the two D-PHY_RX data lanes, if required to do so for increased data rates. It generates a data flow based on an ISB-Byte protocol, which is not linked to any protocol at this point. The data is only identified as a control or a data packet for the clock changer module.
- The clock changer module manages data crossing from the D-PHY_RX clock domain to the clk_proc clock domain.
- The LLP handles data filtering to serve the data-interleaving capability of the DCMIPP. It can filter data from external devices depending on virtual channel identifiers, or on data-type format selection. It also executes, depending on the input pixel format, data reordering in order to optimize the data storage, for instance in memories.
- The CSI-2 Host manages also various errors coming (not exclusively) from the D-PHY_RX physical layer. It can detect error conditions in the received header or a short packet in a frame. ECC is implemented to correct one bit error in the header. A CRC checksum is also implemented to detect any data corruption during the data payload reception. The 16-bit CRC value is located into the packet footer. The CSI-2 Host also ensures that specific timings within the data reception are properly in line with the CSI-2 specification. All error conditions and associated interrupts are detailed in *Section 40.8*.

- A specific module generates interrupts following the detection of selected events in a frame. For a given virtual channel, these events may be triggers based on line- and byte-number detection within the frame, or counters from dedicated timers reaching a number of cycles (clk_proc) from an SOF or an EOF.
- Data is filtered out by the LLP, and sent to the DCMIPP in a format that is quite similar to the CSI-2 protocol, but without CRC (packet footer) and ECC information inside the packet header. The data type and the virtual channel are also sent to the DCMIPP to handle the interleaved transmission of different image data formats within the same video data stream.

40.5.3 CSI-2 Host reset and clocks

There are two reset sources:

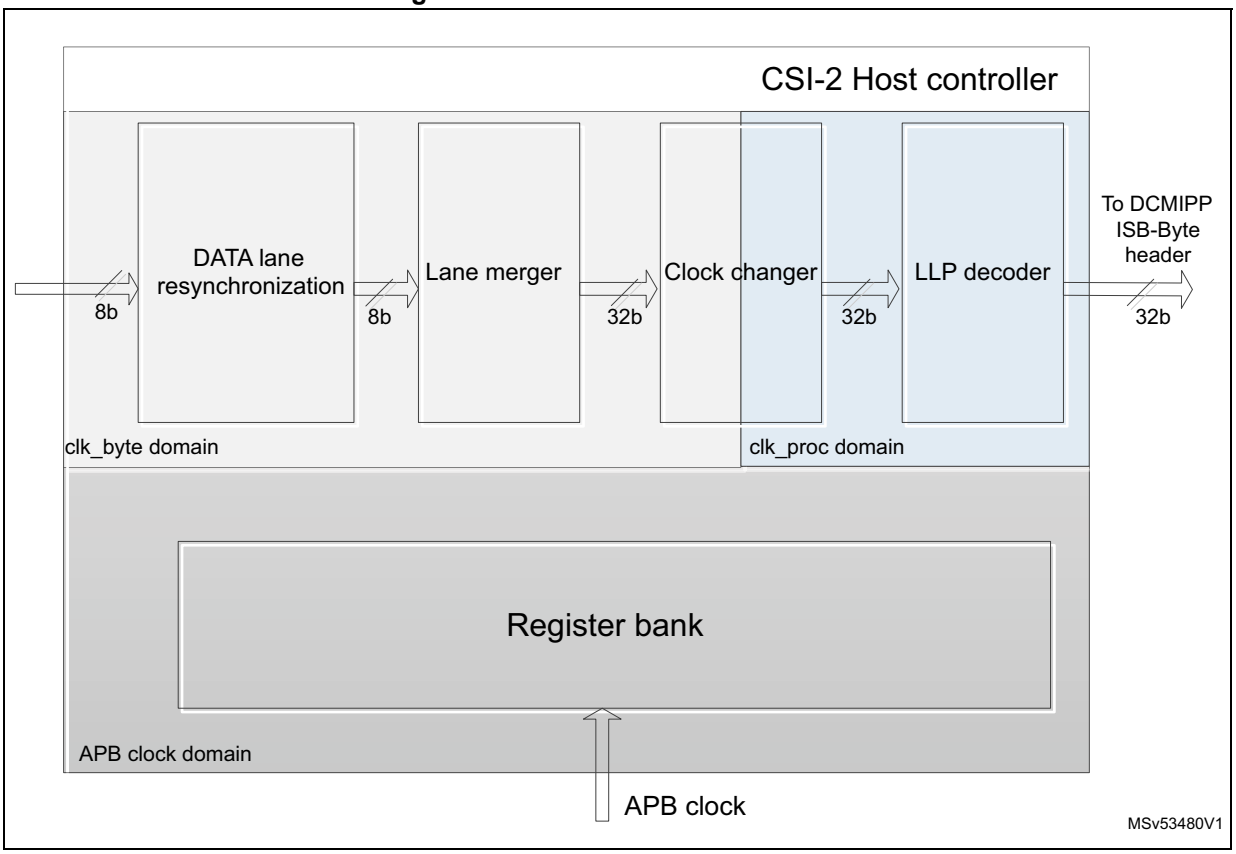
- hardware asynchronous reset which is resynchronized on clk_byte and clk_proc clock domain (processing clock) to reset the flops connected to these clock domains
- APB reset from the reset and clock controller unit to reset mainly all the registers

There are three clocks to serve the CSI-2 Host controller (see *Figure 418*):

- clk_byte: derived from the D-PHY_RX clock lane, and used by the lane merger module and the clock changer module
- clk_proc: used by the kernel of the LLP

  Resynchronization between the clk_byte domain and the clk_proc domain is handled by the clock changer module.
- PCLK clock: APB clock used to access to the register bank

Figure 418. CSI-2 Host clock domain



The D-PHY_RX supports up to 2.5 Gbit/s per data lane: the maximum frequency on the D-PHY_RX clock is 2.5 Gbits / 8 to output a byte = 312 MHz.

40.5.4 Lane merger

Applications requiring more bandwidth than the one provided by a single data lane may expand the data path to more lanes in order to increase the attainable data rate. The lane merger supports then up to two data lanes.

The lane merger module converts the D-PHY_RX PPI protocol into a byte-oriented bus to be sent to the LLP.

The lane merger parallelizes the bytes to 32 bits in parallel to build a 32-bit output as shown in the figures below. The cycle number does not account for module internal latency.

Figure 419. Lane merger input and output (one data lane)

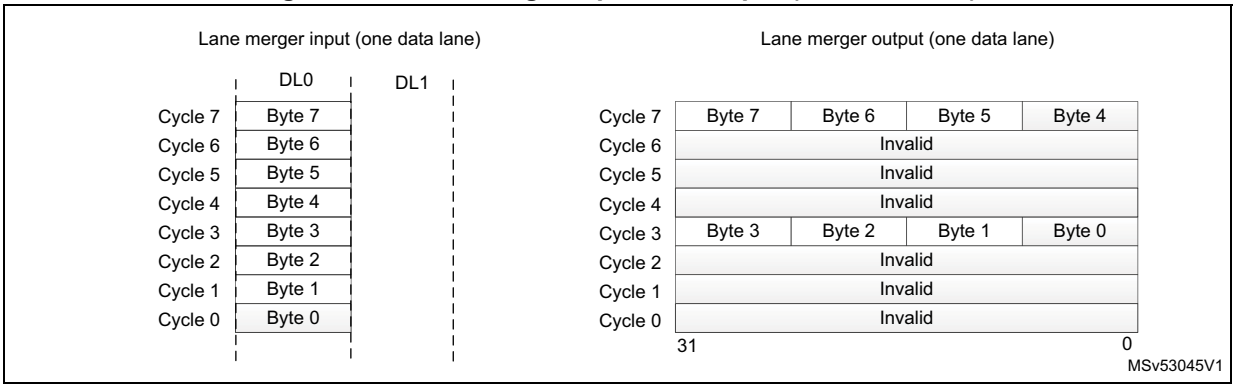
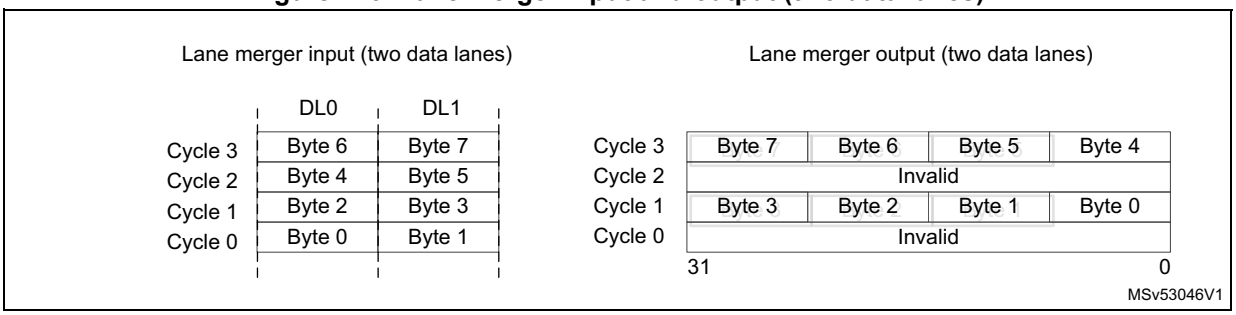
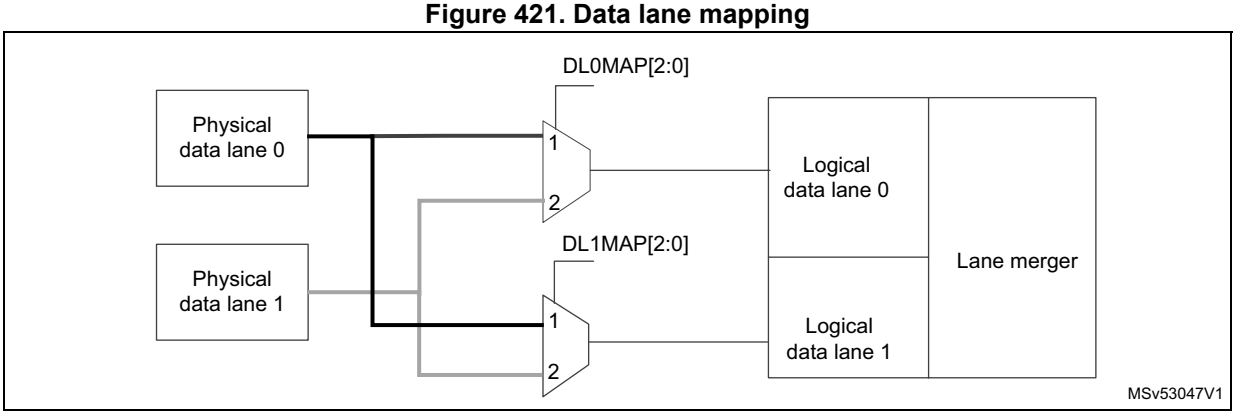


Figure 420. Lane merger input and output (two data lanes)



LANENB[2:0] in CSI_LMCFGR configures the number of data lanes used in the application. The software needs to map the physical data lanes input to the lane merger logical data lanes, using DL0MAP[2:0] and DL1MAP[2:0] as shown in the figure below.

Figure 421. Data lane mapping



40.5.5 Clock changer

The clock changer module changes the clock frequency from the input clock domain linked to the D-PHY_RX PPI protocol, to the output clock domain based on clk_proc clock domain (reference clock for the LLP).

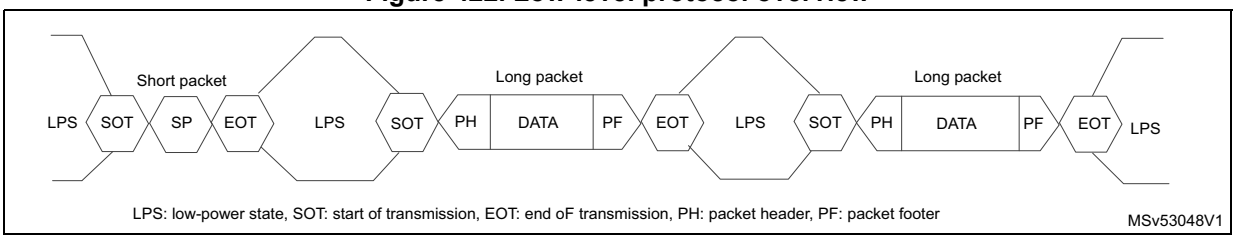
An integrated FIFO receives data from the input clock domain and realigns it to the output clock domain.

40.5.6 Low-level protocol (LLP)

The LPP module retrieves the raw data from the lane merger, and decodes the data flow respect to the CSI-2 low-level protocol. The software selects which data in the data interleaving flow need to be sent to the DCMIPP without being filtered out. The LLP is compliant with the MIPI CSI-2 specification as described in *Section 40.2*.

A data integrity check is done thanks to an ECC code for packet header error correction, and a 16-bit CRC checksum code for payload error detection. All these errors are flagged into specific registers, and may trig interruption for specific software action needs (see *Section 40.8* for more details).

Figure 422. Low-level protocol overview



The CSI-2 low-level protocol is mainly composed by short and long packets, each of them having a well identified role.

Long packet

The long packet format is presented in *Figure 423*. It contains a data ID that is composed of a virtual channel number for which the data packets are attached, and the corresponding data type. The LLP module filters data out by configuring CSI_VCxCFGRy registers (x = 0 to 3, y = 1 to 4), and VCxSTART/VCxSTOP bits in CSI_CR register.

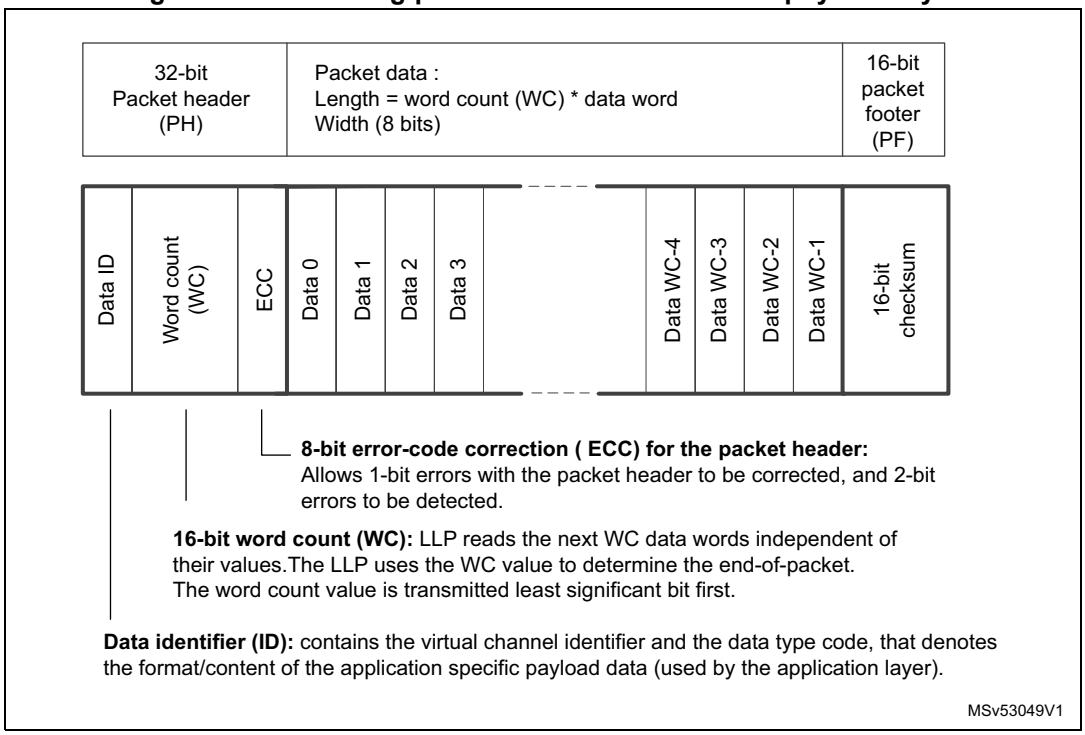
The word-count field allows the LLP to determine the data-payload width before detecting the packet footer and the EOT. This information is used to detect shorter-than-expected packets, and to trigger a corresponding error (refer to *Section 40.8*).

The ECC corrects a single erroneous bit in the packet header in the case of data corruption. An error is raised if there are two errors in total (the LLP decoder immediately stops the current frame), because only one bit can be corrected.

The packet data corresponds to the data payload for a given virtual channel and for an identified data type.

To complete the long-packet reception block, a packet footer that contains a 16-bit CRC checksum is received and compared to that which is internally calculated by the LLP, to check the data integrity of the received data payload.

Figure 423. CSI2- long-packet structure for a D-PHY physical layer

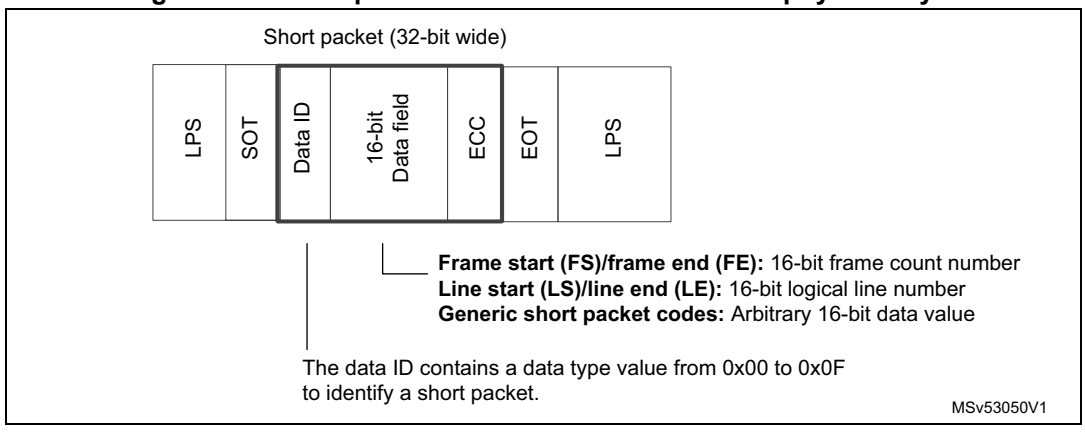


Short packet

The CSI-2 short-packet format for a D-PHY physical layer is shown in *Figure 424*.

It is similar to the long packet header, except that the word count field is replaced by a 16-bit data field. The ECC 8-bit code is present to secure headers having potential bit corruption.

Figure 424. Short-packet structure based on D-PHY physical layer

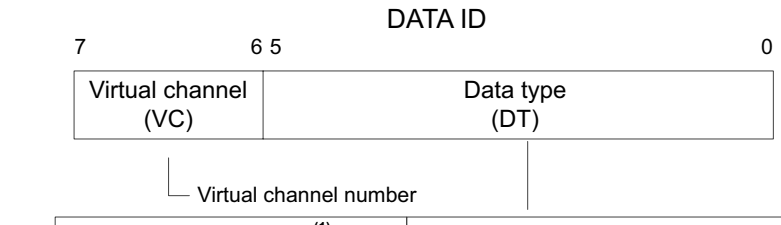


Data ID field

The 8-bit data ID field comprises 2 bits for the virtual channel number coding, and 6 bits to identify the packet data type (short or long packet), and the payload data packet format in the case of long packet reception.

The LLP filters subsequent packets depending on the user configuration indicated in the previously received data ID field (see *Figure 425*).

Figure 425. DATA ID structure in short or long packet format



Data type classes [1]	Description
0x00 to 0x07	Synchronization short packet data types
0x08 to 0x0F	Generic short packet data types
0x10 to 0x17	Generic long packet data types
0x18 to 0x1F	YUV data
0x20 to 0x27	RGB data
0x28 to 0x2F	RAW data
0x30 to 0x37	User defined byte-based data
0x38 to 0x3F	Reserved

(1) Refer to the MIPI Alliance specification for the CSI-2 to have the detail of each intermediate value not described in this table.

MSv53051V1

Data type field

The data type (DT) fields are configured by the application code to determine which data types are processed by the LLP for a given virtual channel. CSI_VCxCFGRy registers are used to configure the data types to be considered by the LLP.

Each virtual channel is managed independently, and up to seven different virtual-channel data types can be declared in parallel. All data type input to the LLP that are not defined for a given virtual channel are filtered out by the LLP module and discarded.

Data types in the video data flow are located:

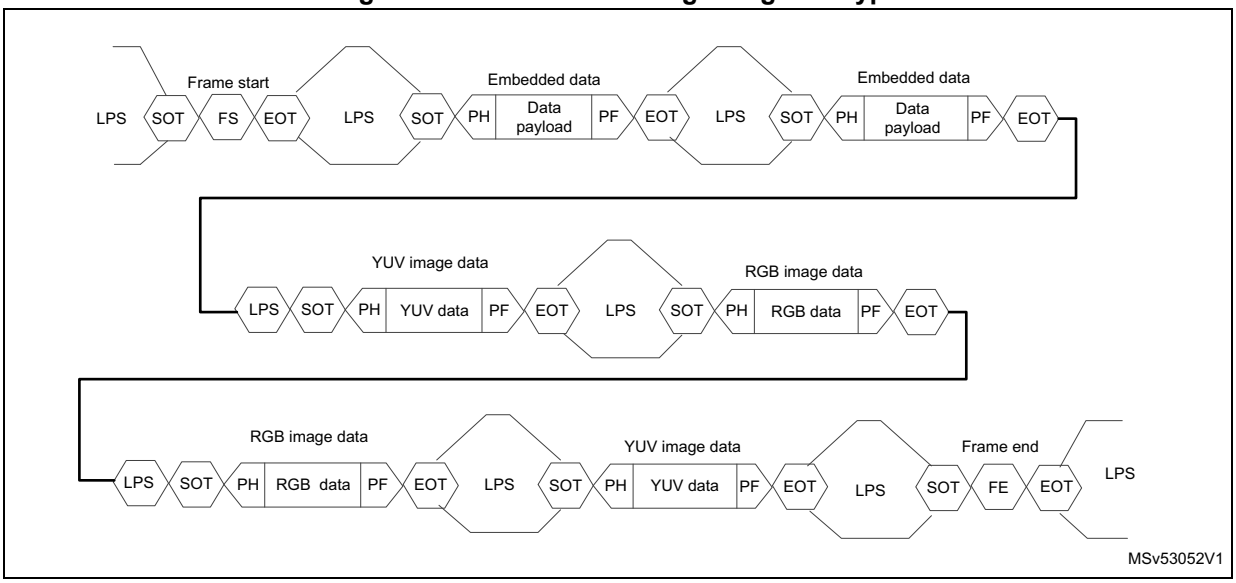
- in the data ID packet for a short packet
- in the packet header (PH) for a long packet format

Data interleaving: data-type based

In the video data flow, the transmitter can merge, on the same physical node, multi-device data transmissions having different data types. The LLP filters out these multi-device flows depending on the application configuration. A typical video data flow interleaving multiple data types is shown in *Figure 426*.

The application software must configure CSI_VCxCFGRy registers to enable data types for a virtual channel, and to set data-type formats to be processed. The LLP reorders the input data in a format that optimizes memory usage (see *Figure 426* to *Figure 434*).

Figure 426. Data interleaving using data type



Data interleaving: virtual channels

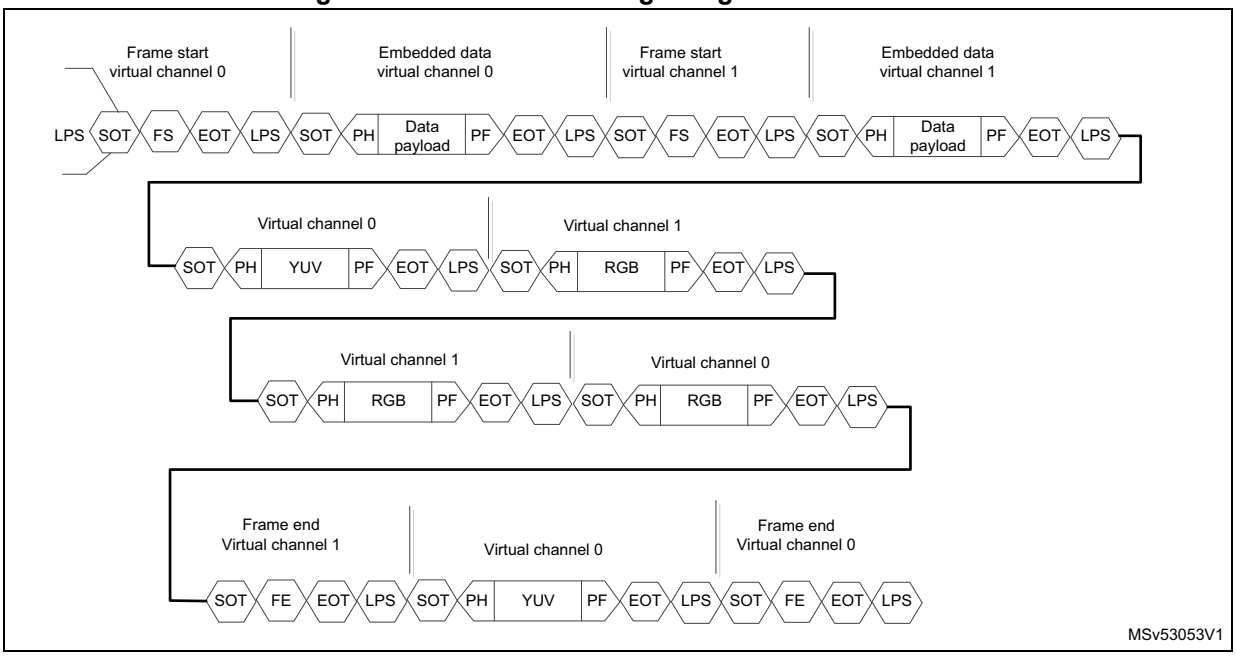
The data interleaving in the video data flow can also be managed by the virtual channel ID. Each external device can have a specific virtual channel identifier in order for the software layer or the DCMIPP to properly handle data packets belonging to a device.

The application code defines which of the virtual channels have to be considered by the LPP without any filtering operation. The CSI_CR register must be configured accordingly by activating the virtual channel (via VCxSTART), or by setting VCxSTOP to deactivate the virtual channel.

The virtual channel number is encapsulated in the data ID packet in short or long packet.

A typical data interleaving case is illustrated in *Figure 427*.

Figure 427. Data interleaving using virtual channels



Data formats

The LLP, in addition to a data-flow filter role, handles data reordering for memory optimization, storage, or post-processing purposes.

The CSI-2 Host controller is compliant with the MIPI Alliance CSI-2 specification. It mainly supports the following data formats:

- BPP6 (for example RAW6, RGB666)
- BPP7 (for example RAW7)
- BPP8 (for example RAW8, YUV 8-bits, RGB444, RGB555, RGB565, RGB888, JPEG)
- BPP10 (for example RAW10, YUV 10-bits)
- BPP12 (for example RAW12)
- BPP14 (for example RAW14)
- BPP16 (for example RAW16)

Figure 428. RAW6 (BPP6) input and output format

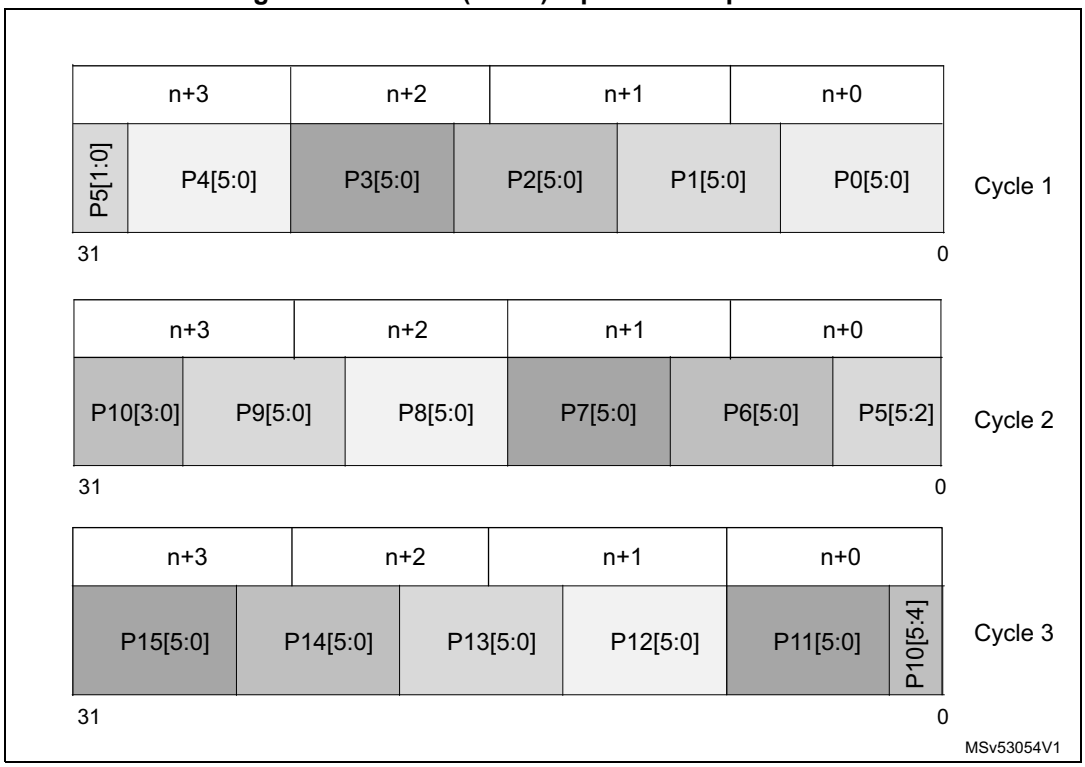


Figure 429. Raw7 (BPP7) input and output format

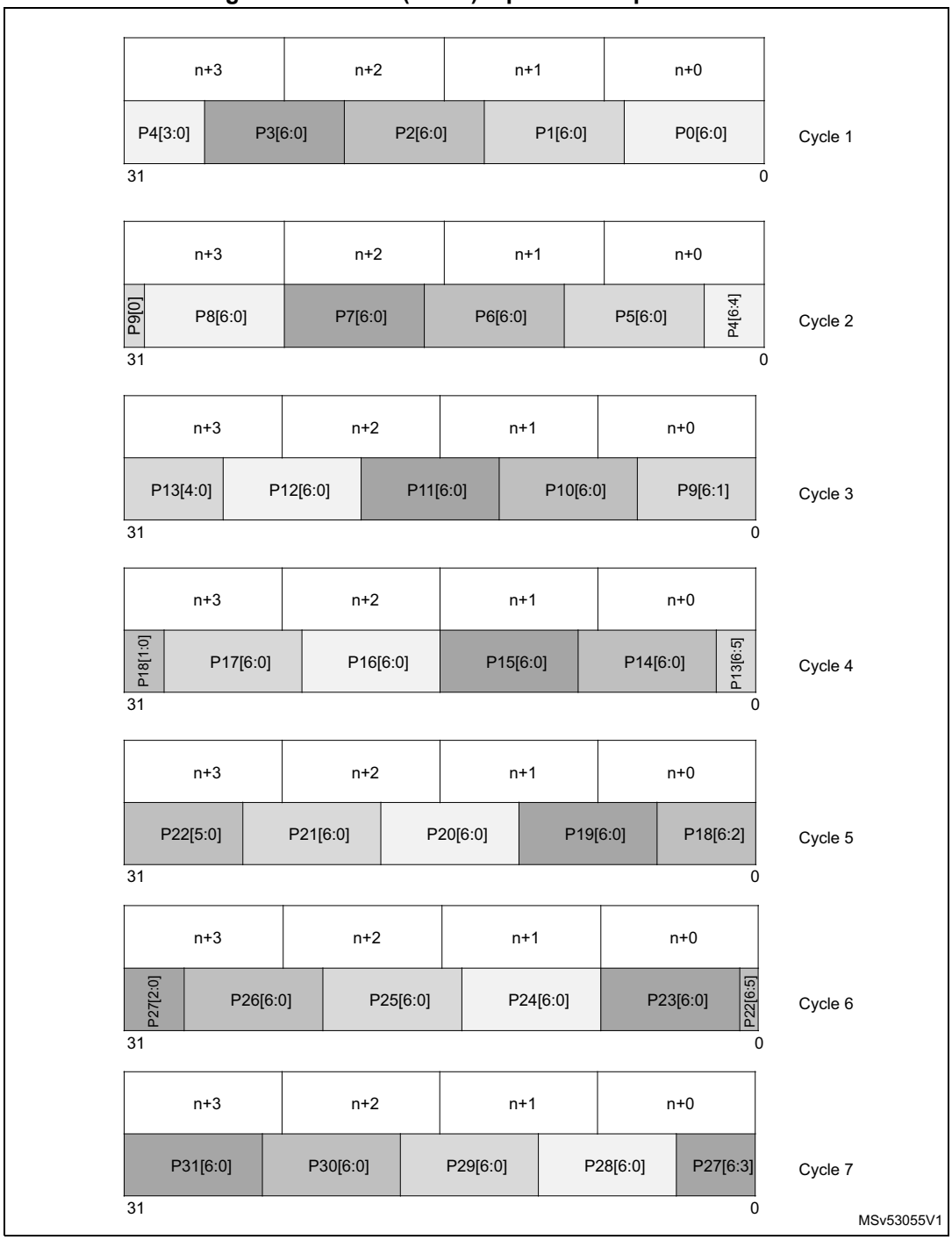


Figure 430. Raw8 (BPP8) input and output format

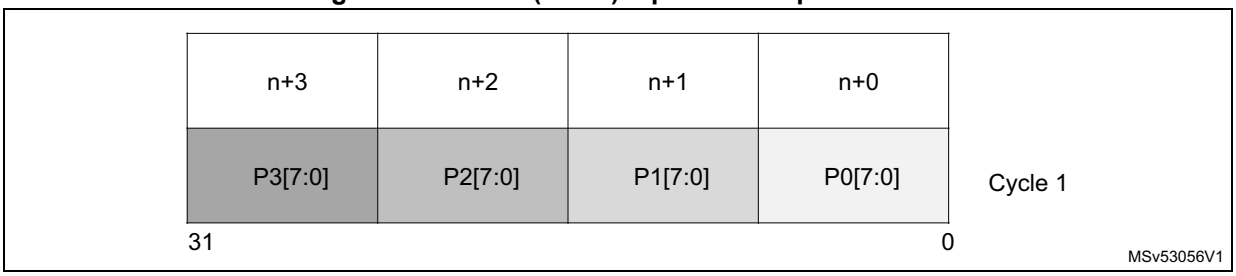


Figure 431. Raw 10 (BPP10) input and output format

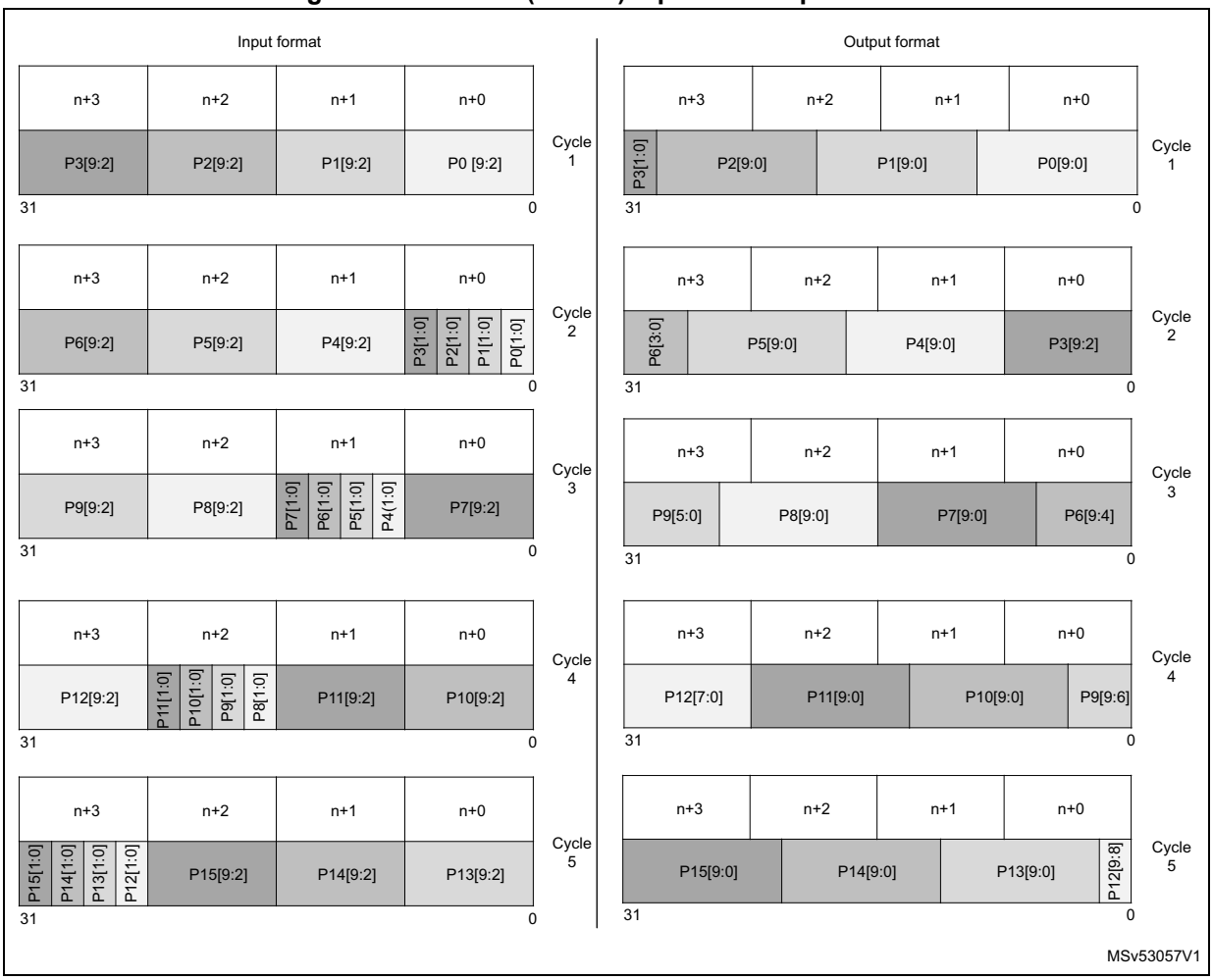


Figure 432. Raw12 (BPP12) input/output format

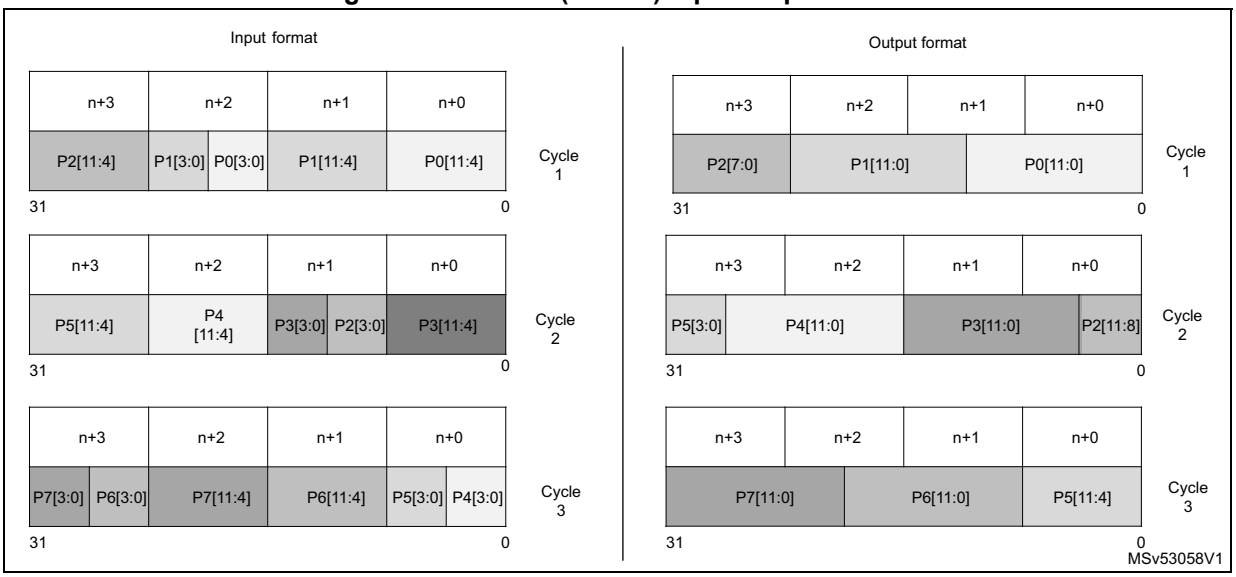


Figure 433. Raw14 (BPP14) input format

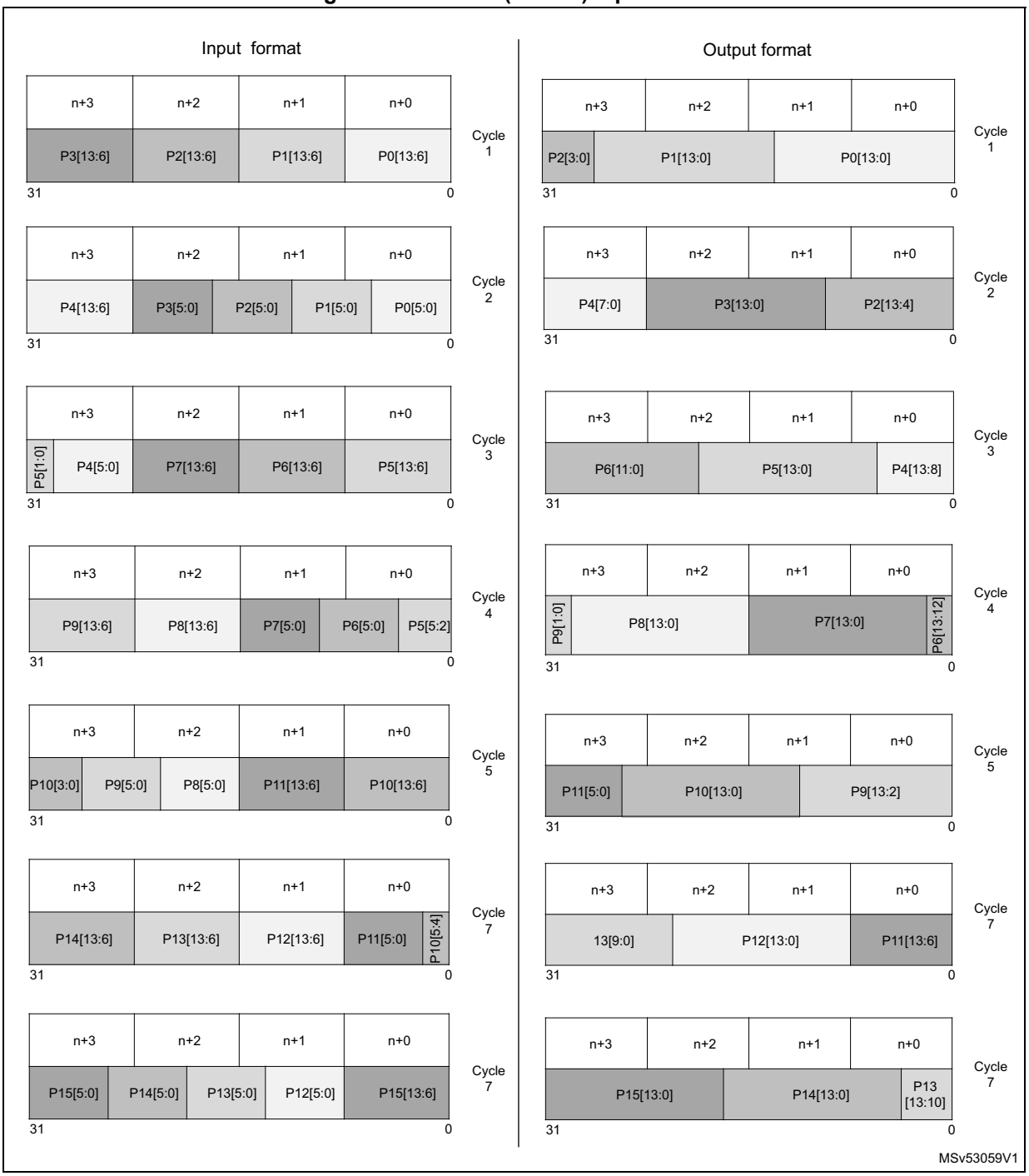
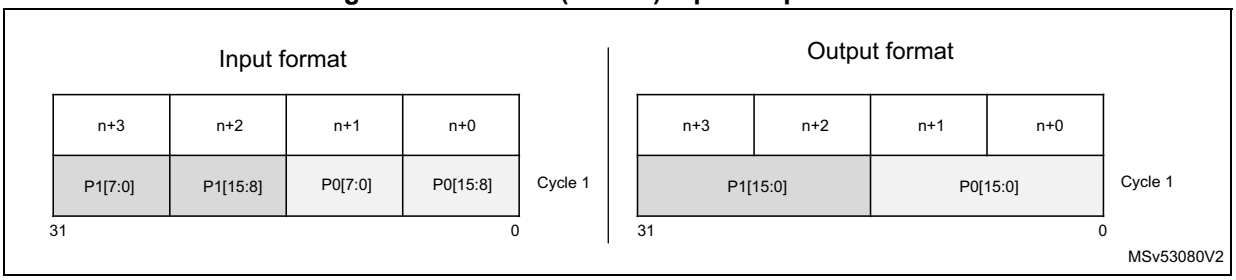


Figure 434. Raw16 (BPP16) input/output format



40.5.7 CSI-2 Host protocol output

The CSI-2 Host controller generates ISB-Byte header data according to CSI_VCxCFGRy. The ISB-Byte header generation respects the following rules:

- ISB-Byte header channel number is identical to the CSI-2 virtual channel and data type (LSB aligned).
- ISB-Byte header data type is identical to CSI-2 data type (LSB aligned).
- ISB-Byte header frame number is identical to the CSI-2 frame number.
- ISB-Byte header count is identical to CSI-2 word count.
- The format is updated according to CSI_VCxCFGRy configuration.

The only differences between the ISB-Byte data flow in input to the LLP and the ISB-Byte header bus at the output are listed below:

- There is no ECC information in the data corresponding to the short packet information.
- The packet footer (containing the CRC-16 value) is not transmitted, or forwarded on the output since the decoder already checked it from the input, and raised an error if the data payload is corrupted.
- The data payload output format is reordered and compared to the input data format of the LLP in order to ease pixels storage in memory for further hardware/software post-processing.
- Only identified virtual channels and their selected data types are generated on the output.

The ISB-Byte header is used by the DCMIPP peripheral to handle post-processing data from the external devices.

40.6 CSI-2 Host programming guide

40.6.1 CSI-2 PHY setup

The following actions are required to setup the CSI-2 PHY:

1. Quit the hardware reset state with PEN = 0x01 in CSI_PRCR.
2. Configure frequency settings with:
   – DLD = 0x00 in CSI_PFCR (to remain in Rx)
   – HSFR in CSI_PFCR that depends on the bit-rate expected on the data lanes
   – CCFR = RoundUp((F - 17) × 4) in CSI_PFCR (F = configuration clock frequency)
3. Configure internal registers via the test interface, with TRSEN = TCKEN = 0x0 in CSI_PTCR0.

40.6.2 CSI-2 Host controller setup and data reception start

The D-PHY RX software initialization is required to set up the CSI-2 Host controller.

Lane-merger configuration (static configuration)

The lane merger must be configured while CSIEN = 0 in CSI_CR, and when the D-PHY_RX is not streaming (when it is not providing a clock).

To map the physical data lane to the logical data lane, the lane merger needs to be configured through DL0MAP and DL1MAP in CSI_LMCFGR. In addition, the number of valid data lanes must be configured through LANENB in the same register.

LLP configuration

1. Configure the watchdog value in CSI_WDR. This corresponds to the maximum number of clock cycles between two packets before an input frame is considered erroneous and the transmission failed. It can be left at 0x0 to disable the watchdog feature.
2. Enable the CSI-2 Host controller with CSIEN = 1 in CSI_CR.
3. Configure the selected virtual channels in CSI_VCxCFGRy.

   If all data types must be captured with the same format, in CSI_VCxCFGR1:
   - Set ALLDT = 0x011.
   - Select the format with CDTFT[4:0].

   If only a few data types are to be captured, or if the format is not identical for all data types, then up to seven data types can be selected:
   - Set ALLDT = 0x0 in CSI_VCxCFGR1.
   - Set DTyEN = 0x1 (y = 0 to 6) in CSI_VCxCFGR1 for the data types used.
   - Set DTy[5:0] in CSI_VCxCFGRy to the selected data type.
   - Set DTyFT[4:0] in CSI_VCxCFGRy to the selected format.
4. Depending on application requirements, specific software actions may detect particular events in a frame, and trigger a corresponding interrupt, with the following:
   - Set LINECNT[15:0] and BYTECNT[15:0] in CSI_LBxCFGR to trigger an event when the data corresponding to a specific byte within a specific line is detected by the LLP. Up to four independent triggers can be programmed and linked to a selected virtual channel through LBxVC[1:0] in CSI_PRGITR. The selected trigger can be enabled by setting LBxEN in the same register.
   - Set COUNT[24:0] in CSI_TIMxCFGR to trigger an event after a specified number of clock cycles (clk_proc) from an SOF or an EO, depending on TIMxEOF and TIMxEN in CSI_PRGITR. Up to four independent timer triggers can be programmed and linked to a selected virtual channel through TIMxVC[1:0] in CSI_PRGITR.
5. Enable interrupt sources for which the code needs to make specific actions, by configuring CSI_IER0 and CSI_IER1 registers.
6. D-PHY_RX lanes can be enabled by setting DL0EN and/or DL1EN, and CLEN in CSI_PCR.
7. Start the virtual channel by setting VCxSTART = 0x1 in CSI_CR.

40.6.3 Data reception stop

To stop data reception, set VCxSTOP = 1 in CSI_CR. The data reception stops at the end of the current frame. The virtual channel status can be retrieved from VCxSTATEF in CSI_SR0.

Note: A virtual channel cannot start before it has been completely stopped (VCxSTATEF = 0 in CSI_SR0).

40.7 CSI-2 Host low-power modes

Table 369. Effect of low-power modes on CSI-2 Host controller

Mode	Description
Sleep	No effect. The CSI-2 host interrupts cause the device to exit the Sleep mode.
Stop	The CSI-2 host is in power-down, and must be reinitialized after exiting Stop mode.
Standby	The CSI-2 host is in power-down, and must be reinitialized after exiting Standby mode.

40.8 CSI-2 Host interrupts and errors

Few events can generate an error flag or an interrupt from the CSI-2 Host controller interrupt handler.

The table below defines the error sources flagged by the CSI-2 HOST controller.
The content of CSI_ERR1 and CSI_ERR2 registers can be used to know exactly from which virtual number and data types a particular error appeared and has been detected. It can be used as debug purpose too.

Table 370. List of errors detected by the CSI-2 Host controller

Error description	Bit name	Register
Virtual channel number on which a data ID information error is detected	IDVCERR[1:0]	CSI_ERR1
Data type for which a data ID information error is detected	IDDTERR[5:0]	
Virtual channel number for which the ECC has corrected one bit in error	CECCVCERR[1:0]	
Data type for which the ECC has corrected one bit in error	CECCDTERR[5:0]	
Virtual channel number for which a CRC error is detected	CRCVCERR[1:0]	
Data type for which a CRC error is detected	CRCDTERR[5:0]	
Virtual channel number on which the invalid synchronization has been detected	SYNCVCERR[1:0]	CSI_ERR2
Virtual channel number on which a watchdog error has been detected	WDVCERR[1:0]	
Virtual channel number on which a shorter packet than expected has been detected	SPKTVCERR[1:0]	
Data type for which the shorter packet than expected error has been detected	SPKTDTERR[5:0]	

The table below shows the interrupt sources that are triggered by the CSI-2 Host controller if the corresponding bit is enabled.

Table 371. List of events that trigger an interrupt

Interrupt flag	Register	Interrupt sources
SYNCERRF	CSI_SR0	Synchronization error. The interrupt is enabled if SYNCERRIE is set in CSI_IER0(1).
WDERRF		Watchdog error. The interrupt is enabled if WDERRIE is set in CSI_IER0(1).
SPKTERRF		Shorter packet than expected error. The interrupt is enabled if SPKTERRIE is set in CSI_IER0 register(1).
IDERRF		Data ID information error. The interrupt is enabled if IDERRIE is set in CSI_IER0(1).
CECCERRF		Corrected ECC error. The interrupt is enabled if CECCERRIE is set in CSI_IER0(1).
ECCERRF		ECC error. The interrupt is enabled if ECCERRIE is set in CSI_IER0(1).
CRCERRF		CRC error. The interrupt is enabled if CRCERRIE is set in CSI_IER0(1).
CCFIFOFF		Clock changer FIFO full event. The interrupt is enabled if CCFIFOFFIE is set in CSI_IER0(2).
SPKTF		Short packet detection. The interrupt is enabled if SPKTIE is set in CSI_IER0(2).
EOF3F		EOF on virtual channel 3. The interrupt is enabled if EOF3IE is set in CSI_IER0(2).
EOF2F		EOF on virtual channel 2. The interrupt is enabled if EOF2IE is set in CSI_IER0(2).
EOF1F		EOF on virtual channel 1. The interrupt is enabled if EOF1IE is set in CSI_IER0(2).
EOF0F		EOF on virtual channel 0. The interrupt is enabled if EOF0IE is set in CSI_IER0(2).
SOF3F		SOF on virtual channel 3. The interrupt is enabled if SOF3IE is set in CSI_IER0(2).
SOF2F		SOF on virtual channel 2. The interrupt is enabled if SOF2IE is set in CSI_IER0(2).
SOF1F		SOF on virtual channel 1. The interrupt is enabled if SOF1IE is set in CSI_IER0(2).
SOF0F		SOF on virtual channel 0. The interrupt is enabled if SOF0IE is set in CSI_IER0(2).
TIM3F		Timer 3 counter value reached. The interrupt is enabled if TIM3IE is set in CSI_IER0(2).
TIM2F		Timer 2 counter value reached. The interrupt is enabled if TIM2IE is set in CSI_IER0(2).
TIM1F		Timer 1 counter value reached. The interrupt is enabled if TIM1IE is set in CSI_IER0(2).
TIM0F		Timer 0 counter value reached. The interrupt is enabled if TIM0IE is set in CSI_IER0(2).
LB3F		Line/byte event 3 reached. The interrupt is enabled if LB3IE is set in CSI_IER0(2).
LB2F		Line/byte event 2 reached. The interrupt is enabled if LB2IE is set in CSI_IER0(2).
LB1F		Line/byte event 1 reached. The interrupt is enabled if LB1IE is set in CSI_IER0(2).
LB0F		Line/byte event 0 reached. The interrupt is enabled if LB0IE is set in CSI_IER0(2).

Table 371. List of events that trigger an interrupt (continued)

Interrupt flag	Register	Interrupt sources
ECTRLDL1F	CSI_SR1	Error control on data line 1. The interrupt is enabled if ECTRLDL1IE is set in CSI_IER1.
ESYNCESCDL1F		Low-power data transmission synchronization error on data line 1. The interrupt is enabled if ESYNCESCDL1IE is set in CSI_IER1.
EESCDL1F		Escape entry error on data line 1. The interrupt is enabled if EESCDL1IE is set in CSI_IER1.
ESOTSYNCDL1F		SOT synchronization error on data line 1. The interrupt is enabled if ESOTSYNCDL1IE is set inCSI_IER1.
ESOTDL1F		SOT error on data line 1. The interrupt is enabled if the bit ESOTDL1IE is set into CSI_IER1.
ECTRLDL0F		Error control on data line 0. The interrupt is enabled if the bit ECTRLDL0IE is set into CSI_IER1.
ESYNCESCDL0F		Low-power data transmission synchronization error on data line 0. The interrupt is enabled if the bit ESYNCESCDL0IE is set into CSI_IER1.
EESCDL0F		Escape entry error on data line 0. The interrupt is enabled if EESCDL0IE is set in CSI_IER1.
ESOTSYNCDL0F		SOT synchronization error on data line 0. The interrupt is enabled if ESOTSYNCDL0IE is set in CSI_IER1.
ESOTDL0F		SOT error on data line 0. The interrupt is enabled if ESOTDL0IE is set in CSI_IER1.

1. Refer to *Section 40.8.1: Error handling*.
2. Refer to *Section 40.8.2: Interrupt handling*.

40.8.1 Error handling

Details of the errors are given in CSI_ERR1 and CSI_ERR2.

Note: Except for data ID information, all other errors can occur during the product life, and must be handled by the software.

Corrected ECC error

When this error is detected, the CECCERR interrupt is triggered if CECCERRIE = 1 in CSI_IER0. This is an information event rather than a real error, as the CSI-2 Host controller successfully recovered the ECC error, so the frame is processed properly. Some details of the error are located in CECCVCERR and CECCDTERR in CSI_ERR1:

ECC error

When a multi-bit ECC error is detected, the ECCERR interrupt is triggered if ECCERRIE = 1 in CSI_IER0. As all virtual channels are potentially corrupted, the LPP immediately stops the current frames.

The CSI-2 Host controller restarts each enabled virtual channel on the next SOF packet of each channel. The software must perform the necessary error management (such as resetting some peripherals after this error detection). No details are available for the error as the relevant information is corrupted.

CRC error

When a CRC error is detected, the CRCERR interrupt is triggered if CRCERRIE = 1 in CSI_IER0. As the frame envelope is not corrupted, the CSI-2 Host controller does not stop the frame processing. The software must perform the necessary error management (such as reseting some peripherals, possibly including the CSI-2 Host controller). Details are available for this error in CRCVCERR and CRCDTERR in CSI_ERR1.

Data ID information

When a reserved data type is detected, the IDERR interrupt is triggered if IDERRIE = 1 in CSI_IER0. This is an information event rather than a real error, as the hardware processes this packet as for any non-erroneous packet.Details for the error are available in IDVCERR and IDDTERR in CSI_ERR1.

Invalid synchronization

When an invalid synchronization is detected, the SYNCERR interrupt is triggered if SYNCERRIE = 1 in CSI_IER0. An invalid synchronization is either an SOF short packet during a frame, or an EOF short packet during an inter-frame. It is an information event rather than a real error, as the hardware is able to handle such an error:

- An SOF short packet during a frame is interpreted as back-to-back EOF and SOF. It behaves accordingly (ISB-Byte header EOF, ISB-Byte header SOF, and so on). The SOFx interrupt is generated in addition to the SYNCERR one if SOFxIE = 1 in CSI_IER0.
- An EOF short packet during an inter-frame is ignored (no EOFx interrupt is generated whatever the state of EOFx in CSI_IER0). Details for the error are available in SYNCVCERR in CSI_ERR2.

Shorter than expected packet

When a shorter-than-expected packet is received, the SPKTERR interrupt is triggered if SPKTERRIE = 1 in CSI_IER0. The packet is padded up to its theoretical size, and the frame is processed as if no error occurred. The software must perform the necessary error management (such as reseting some peripherals, possibly including the CSI decoder). Details for the error are available in SPKTVCERR and SPKTDTERR in CSI_ERR2.

Watchdog

When the time between two packets of a given virtual channel is longer than the programmed watchdog (CNT[31:0] in CSI_WDR), the WDERR interrupt is triggered if WDERRIE = 1 in CSI_WDR. As the corresponding virtual channel stream is corrupted, the CSI2 Host controller immediately stops processing this virtual channel. The software must perform the necessary error management (such as reseting some peripherals after this error detection), before restarting the CSI-2 Host. Details for the error are available in WDVCERR in CSI_ERR2.

40.8.2 Interrupt handling

The CSI-2 Host controller offers many interrupt sources that can be activated if the associated enable bit is set. Some interrupt sources are linked to an error detection from the D-PHY physical layer or the CSI-2 Host controller (refer to *Section 40.8.1*).

Short packet interrupt

This interrupt notifies the software that a generic short packet has been received. The software can retrieve the associated data in CSI_SPDFR. This register value is overwritten with each new generic short packet.

SOF and EOF interrupts

The CSI-2 Host controller generates an SOF interrupt for each SOF short packet received. It also generates an EOF interrupt for each EOF short packet, independently for all four virtual channels. These interrupts are generated even if the virtual channel is disabled. The software must mask or ignore these interrupts if they are not needed.

Line/byte programmable interrupts

The CSI-2 Host controller provides four interrupts that can be programmed to be triggered at any point during the active data of a frame. In order to set a programmable line/byte interrupt, the software must follow these steps:

1. Select a virtual channel by setting LBxVC[1:0] in CSI_PRGITR.
2. Select the frame location at which the interrupt is triggered by setting LINECNT and BYTECNT in CSI_LBxCFGRy.
3. Enable the line/byte counting by setting LBxEN in the CSI_PGRITR register. This bit must be set only when the CSI-2 is enabled.
4. Enable the interrupt by setting LBxIE in CSI_IER0.
5. The interrupt is generated starting from the next frame on the selected virtual channel.

In order to stop a programmable interrupt, the software must reset the LBxIE bit in the CSI_IER0 register. It immediately stops the interrupt.

Note: All four line/byte programmable interrupts are independent.

Timer-based programmable interrupts

The CSI-2 Host controller provides four interrupts that can be programmed to be triggered based on a timer. In order to set a programmable timer based interrupt, the software must select a virtual channel by setting TIMxVC[1:0] in CSI_PRGITR. The following steps must be followed, while the CSI-2 is disabled:

1. Select the starting event of the timer with TIMxEOF in CSI_PRGITR.
2. Select the timer duration by setting COUNT[24:0] in CSI_TIMxCFGR.
3. Select and enable counting by setting TIMxEN in CSI_PRGITR.
4. Enable the interrupt by setting TIMxIE in CSI_IER0.
5. The interrupt is generated starting from the next frame on the selected virtual channel.

In order to stop a programmable interrupt, the software must reset the TIMxIE bit in the CSI_IER0 register. It immediately stops the interrupt.

Note: All four timer-based programmable interrupts are independent.

If the timer is set to a value longer than the frame period, the interrupt may not be generated for all frames.

Clock-changer FIFO full interrupt

The clock changer is also built around a FIFO which is never full in normal conditions. If the overall system works very slowly (for example due to the CSI-2 and DCMIPP processing clock frequency being set to an inadequate frequency, system-bus bandwidth availability, and so on), the clock-changer FIFO may become full. In this case, an interrupt is set if CCFIFOFIE = 1 in CSI_IER0.

40.9 CSI-2 Host registers

40.9.1 CSI-2 Host control register (CSI_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VC3 STOP	VC3 START	Res.	Res.	VC2 STOP	VC2 START	Res.	Res.	VC1 STOP	VC1 START	Res.	Res.	VC0 STOP	VC0 START	Res.	CSIEN
w	w			w	w			w	w			w	w		rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **VC3STOP**: Virtual channel 3 stop

The CSI-2 decoder stops the virtual channel 3.

0: No effect

1: Pulse generation to stop the virtual channel 3 processing

Note: The CSI-2 decoder stops processing the virtual channel 3 after the next EOF event (or immediately if the Stop command is issued during frame blanking).

A write operation into this bit must not be simultaneous with a write into VC3START.

Reading this bit returns 0.

Bit 14 **VC3START**: Virtual channel 3 start

The CSI-2 decoder processes the virtual channel 3 according to CSI_VC3CFGRx (x = 0..3).

0: No effect

1: Pulse generation to start the virtual channel 3 processing

Note: The CSI-2 decoder starts processing the virtual channel 3 after the next SOF event.

A write operation into this bit must not be simultaneous with a write into VC3STOP.

Reading this bit returns 0.

Bits 13:12 Reserved, must be kept at reset value.

Bit 11 **VC2STOP**: Virtual channel 2 stop
The CSI-2 decoder stops the virtual channel 2.
0: No effect
1: Pulse generation to stop the virtual channel 2 processing
Note: The CSI-2 decoder stops processing the virtual channel 2 after the next EOF event (or immediately if the Stop command is issued during frame blanking).
A write operation into this bit must not be simultaneous with a write into VC2START.
Reading this bit returns 0.

Bit 10 **VC2START**: Virtual channel 2 start
The CSI-2 decoder processes the virtual channel 2 according to CSI_VC2CFGRx (x = 0..3).
0: No effect
1: Pulse generation to start the virtual channel 2 processing
Note: The CSI-2 decoder starts processing the virtual channel 2 after the next SOF event.
A write operation into this bit must not be simultaneous with a write into VC2STOP.
Reading this bit returns 0.

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **VC1STOP**: Virtual channel 1 stop
The CSI-2 decoder stops the virtual channel 1.
0: No effect
1: Pulse generation to stop the virtual channel 1 processing
Note: The CSI-2 decoder stops processing the virtual channel 1 after the next EOF event (or immediately if the Stop command is issued during frame blanking).
A write operation into this bit must not be simultaneous with a write into VC1START.
Reading this bit returns 0.

Bit 6 **VC1START**: Virtual channel 1 start
The CSI-2 decoder processes the virtual channel 1 according to CSI_VC1CFGRx (x = 0..3).
0: No effect
1: Pulse generation to start the virtual channel 1 processing
Note: The CSI-2 decoder starts processing the virtual channel 1 after the next SOF event.
A write operation into this bit must not be simultaneous with a write into VC1STOP.
Reading this bit returns 0.

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **VC0STOP**: Virtual channel 0 stop
The CSI-2 decoder stops the virtual channel 0.
0: No effect
1: Pulse generation to stop the virtual channel 0 processing
Note: The CSI-2 decoder stops processing the virtual channel 0 after the next EOF event (or immediately if the Stop command is issued during frame blanking).
A write operation into this bit must not be simultaneous with a write into VC0START.
Reading this bit returns 0.

Bit 2 **VC0START**: Virtual channel 0 start
he CSI-2 decoder processes the virtual channel 0 according to CSI_VC 0CFGRx (x = 0..3).
0: No effect
1: Pulse generation to start the virtual channel 0 processing
Note: The CSI-2 decoder starts processing the virtual channel 0after the next SOF event.
A write operation into this bit must not be simultaneous with a write into VC0STOP.
Reading this bit returns 0.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CSIEN**: CSI-2 enable

This bit enables the CSI-2 peripheral for processing the data flow coming from the camera through the D-PHY_RX.

0: CSI-2 disabled

1: CSI-2 enabled

40.9.2 CSI-2 Host DPHY_RX control register (CSI_PCR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DL1EN	DL0EN	CLEN	PWR DOWN
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **DL1EN**: D-PHY_RX data lane 1 enable

This bit enables the data lane 1 for the D-PHY_RX.

0: Data lane 1 module is in shutdown mode.

1: Enable the data lane 1 module.

Bit 2 **DL0EN**: D-PHY_RX data lane 0 enable

This bit enables the data lane 0 for the D-PHY_RX.

0: Data lane 0 module is in shutdown mode.

1: Enable the data lane 0 module.

Bit 1 **CLEN**: Clock lane enable

This bit enables the clock lane module of the D-PHY_RX.

0: Clock lanes disabled

1: Clock lanes enabled

Bit 0 **PWRDOWN**: Power down

This bit keeps the D-PHY_RX in power-down state.

0: Active state

1: Power down the D-PHY_RX

40.9.3 CSI-2 Host virtual channel x configuration register 1 (CSI_VCxCFGR1)

Address offset: 0x10 + 0x10 * x, (x = 0 to 3)

Reset value: 0x0000 0000

Any change in this register while the virtual channel x is running is ignored.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	DT0FT[4:0]					Res.	Res.	DT0[5:0]					
			rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CDTFT[4:0]					DT6EN	DT5EN	DT4EN	DT3EN	DT2EN	DT1EN	DT0EN	ALLDT
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **DT0FT[4:0]**: Data type 0 format

These bits configure the input pixel format for the virtual channel x on data type 0.
00000: (BPP6) 6-bit words (for example RAW6, RGB666)
00001: (BPP7) 7-bit words (for example RAW7)
00010: (BPP8) 8-bit words (for example RAW8, YUV 8 bits, RGB888, RGB444, RGB555, RGB565, JPEG)
00011: (BPP10) 10-bit words (for example RAW10, YUV 10 bits)
00100: (BPP12) 12-bit words (for example RAW12)
00101: (BPP14) 14-bit words (for example RAW14)
00110: (BPP16) 16-bit words (for example RAW16)
Others: reserved and must not be used

Note: The format selection is valid only if ALLDT = 0 and DT0EN = 1. These bits have no effect if ALLDT = 1.

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **DT0[5:0]**: Data type 0 class selection for virtual channel x

These bits select the data type 0 class and subclass for the virtual channel x according to the MIPI specification for CSI-2.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **CDTFT[4:0]**: Common format for all data types

These bits are configured to select which input pixel format is applied for all the seven data type on virtual channel x.
00000: (BPP6) 6-bit words (for example RAW6, RGB666)
00001: (BPP7) 7-bit words (for example RAW7)
00010: (BPP8) 8-bit words (for example RAW8, YUV 8 bits, RGB888, RGB444, RGB555, RGB565, JPEG)
00011: (BPP10) 10-bit words (for example RAW10, YUV 10 bits)
00100: (BPP12) 12-bit words (for example RAW12)
00101: (BPP14) 14-bit words (for example RAW14)
00110: (BPP16) 16-bit words (for example RAW16)
Others: Reserved

Note: The format selection is valid only if ALLDT = 1. These bits have no effect if ALLDT = 0.

Bit 7 **DT6EN**: Data type 6 enable

This bit enables the data type 6 for the virtual channel x. The data type format is applied with respect to the configuration of DT6[5:0] and DT6FT[4:0] in CSI_VCxCFGR4.

0: Data type 6 for virtual channel x is disabled.

1: Data type 6 for virtual channel x is enabled.

Note: This bit has no effect if ALLDT = 1.

Bit 6 **DT5EN**: Data type 5 enable

This bit enables the data type 5 for the virtual channel x. The data type format is applied with respect to the configuration of DT5[5:0] and DT5FT[4:0] in CSI_VCxCFGR4.

0: Data type 5 for virtual channel x is disabled.

1: Data type 5 for virtual channel x is enabled.

Note: This bit has no effect if ALLDT = 1.

Bit 5 **DT4EN**: Data type 4 enable

This bit enables the data type 4 for the virtual channel x. The data type format is applied with respect to the configuration of DT4[5:0] and DT4FT[4:0] in CSI_VCxCFGR3.

0: Data type 4 for virtual channel x is disabled.

1: Data type 4 for virtual channel x is enabled.

Note: This bit has no effect if ALLDT = 1.

Bit 4 **DT3EN**: Data type 3 enable

This bit enables the data type 3 for the virtual channel x. The data type format is applied with respect to the configuration of DT3[5:0] and DT3FT[4:0] in CSI_VCxCFGR3.

0: Data type 3 for virtual channel x is disabled.

1: Data type 3 for virtual channel x is enabled.

Note: This bit has no effect if ALLDT = 1.

Bit 3 **DT2EN**: Data type 2 enable

This bit enables the data type 2 for the virtual channel x. The data type format is applied with respect to the configuration of DT2[5:0] and DT2FT[4:0] in CSI_VCxCFGR2.

0: Data type 2 for virtual channel x is disabled.

1: Data type 2 for virtual channel x is enabled.

Note: This bit has no effect if ALLDT = 1.

Bit 2 **DT1EN**: Data type 1 enable

This bit enables the data type 1 for the virtual channel x. The data type format is applied with respect to the configuration of DT1[5:0] and DT1FT[4:0] in CSI_VCxCFGR2.

0: Data type 1 for virtual channel x is disabled.

1: Data type 1 for virtual channel x is enabled.

Note: This bit has no effect if ALLDT = 1.

Bit 1 **DT0EN**: Data type 0 enable

This bit enables the data type 0 for the virtual channel x. The data type format is applied with respect to the configuration of DT0[5:0] and DT0FT[4:0] in CSI_VCxCFGR1.

0: Data type 0 for virtual channel x is disabled.

1: Data type 0 for virtual channel x is enabled.

Note: This bit has no effect if ALLDT = 1.

Bit 0 **ALLDT**: All data types enable for the virtual channel x

This bit allows to capture all data types according to CDTFT[4:0] for the virtual channel x.

0: Data type capture managed individually

1: All data types captured for virtual channel x

40.9.4 CSI-2 Host virtual channel x configuration register 2 (CSI_VCxCFGR2)

Address offset: 0x14 + 0x10 * x, (x = 0 to 3)

Reset value: 0x0000 0000

Any change in this register while the virtual channel x is running is ignored.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	DT2FT[4:0]					Res.	Res.	DT2[5:0]					
			rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DT1FT[4:0]					Res.	Res.	DT1[5:0]					
			rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **DT2FT[4:0]**: Data type 2 format

These bits configure the input pixel format for the virtual channel x on data type 2.
00000: (BPP6) 6-bit words (for example RAW6, RGB666)
00001: (BPP7) 7-bit words (for example RAW7)
00010: (BPP8) 8-bit words (for example RAW8, YUV 8 bits, RGB888, RGB444, RGB555, RGB565, JPEG)
00011: (BPP10) 10-bit words (for example RAW10, YUV 10 bits)
00100: (BPP12) 12-bit words (for example RAW12)
00101: (BPP14) 14-bit words (for example RAW14)
00110: (BPP16) 16-bit words (for example RAW16)
Others: Reserved and must not be used.

Note: The format selection is valid only if ALLDT = 0 and DT2EN = 1. These bits have no effect if ALLDT = 1.

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **DT2[5:0]**: Data type 2 class selection for virtual channel x

These bits select the data type 2 class and subclass for the virtual channel x according to the MIPI specification for CSI- 2.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DT1FT[4:0]**: Data type 1 format

These bits configure the input pixel format for the virtual channel x on data type 1.
00000: (BPP6) 6-bit words (for example RAW6, RGB666)
00001: (BPP7) 7 -it words (for example RAW7)
00010: (BPP8) 8 -it words (for example RAW8, YUV 8 bits, RGB888, RGB444, RGB555, RGB565, JPEG)
00011: (BPP10) 10-bit words (for example RAW10, YUV 10 bits)
00100: (BPP12) 12 -it words (for example RAW12)
00101: (BPP14) 14 -it words (for example RAW14)
00110: (BPP16) 16 -it words (for example RAW16)
Others: Reserved and must not be used.

Note: The format selection is valid only if ALLDT = 0 and DT1EN = 1. These bits have no effect if ALLDT = 1.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **DT1[5:0]**: Data type 1 class selection for virtual channel x
These bits select the data type 1 class and subclass for the virtual channel x according to the MIPI specification for CSI- 2.

40.9.5 CSI-2 Host virtual channel x configuration register 3 (CSI_VCxCFGR3)

Address offset: 0x18 + 0x10 * x, (x = 0 to 3)

Reset value: 0x0000 0000

Any change in this register while the virtual channel x is running is ignored.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	DT4FT[4:0]					Res.	Res.	DT4[5:0]					
			rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DT3FT[4:0]					Res.	Res.	DT3[5:0]					
			rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **DT4FT[4:0]**: Data type 4 format
These bits configure the input pixel format for the virtual channel x on data type 4.
00000: (BPP6) 6-bit words (for example RAW6, RGB666)
00001: (BPP7) 7-bit words (for example RAW7)
00010: (BPP8) 8-bit words (for example RAW8, YUV 8 bits, RGB888, RGB444, RGB555, RGB565, JPEG)
00011: (BPP10) 10-bit words (for example RAW10, YUV 10 bits)
00100: (BPP12) 12-bit words (for example RAW12)
00101: (BPP14) 14-bit words (for example RAW14)
00110: (BPP16) 16-bit words (for example RAW16)
Others: Reserved and must not be used.

Note: The format selection is valid only if ALLDT = 0 and DT4EN = 1. These bits have no effect if ALLDT = 1.

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **DT4[5:0]**: Data type 4 class selection for virtual channel x
These bits select the data type 4 class and subclass for the virtual channel x according to the MIPI specification for CSI-2.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DT3FT[4:0]**: Data type 3 format

These bits configure the input pixel format for the virtual channel x on data type 3.
00000: (BPP6) 6-bit words (for example RAW6, RGB666)
00001: (BPP7) 7-bit words (for example RAW7)
00010: (BPP8) 8-bit words (for example RAW8, YUV 8 bits, RGB888, RGB444, RGB555, RGB565, JPEG)
00011: (BPP10) 10-bit words (for example RAW10, YUV 10 bits)
00100: (BPP12) 12-bit words (for example RAW12)
00101: (BPP14) 14-bit words (for example RAW14)
00110: (BPP16) 16-bit words (for example RAW16)
Others: Reserved and must not be used.

Note: The format selection is valid only if ALLDT = 0 and DT3EN = 1. These bits have no effect if ALLDT = 1.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **DT3[5:0]**: Data type 3 class selection for virtual channel x

These bits select the data type 3 class and subclass for the virtual channel x according to the MIPI specification for CSI-2.

40.9.6 CSI-2 Host virtual channel x configuration register 4 (CSI_VCxCFGR4)

Address offset: 0x1C + 0x10 * x, (x = 0 to 3)

Reset value: 0x0000 0000

Any change in this register while the virtual channel x is running is ignored.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	DT6FT[4:0]					Res.	Res.	DT6[5:0]					
			rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DT5FT[4:0]					Res.	Res.	DT5[5:0]					
			rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:24 **DT6FT[4:0]**: Data type 6 format

These bits configure the input pixel format for the virtual channel x on data type 6.
00000: (BPP6) 6-bit words (for example RAW6, RGB666)
00001: (BPP7) 7-bit words (for example RAW7)
00010: (BPP8) 8-bit words (for example RAW8, YUV 8 bits, RGB888, RGB444, RGB555, RGB565, JPEG)
00011: (BPP10) 10-bit words (for example RAW10, YUV 10 bits)
00100: (BPP12) 12-bit words (for example RAW12)
00101: (BPP14) 14-bit words (for example RAW14)
00110: (BPP16) 16-bit words (for example RAW16)
Others: Reserved and must not be used.

Note: The format selection is valid only if ALLDT = 0 and DT6EN = 1. These bits have no effect if ALLDT = 1.

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **DT6[5:0]**: Data type 6 class selection for virtual channel x

These bits select the data type 6 class and subclass for the virtual channel x according to the MIPI specification for CSI-2.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DT5FT[4:0]**: Data type 5 format

These bits configure the input pixel format for the virtual channel x on data type 5.
00000: (BPP6) 6-bit words (for example RAW6, RGB666)
00001: (BPP7) 7-bit words (for example RAW7)
00010: (BPP8) 8-bit words (for example RAW8, YUV 8 bits, RGB888, RGB444, RGB555, RGB565, JPEG)
00011: (BPP10) 10-bit words (for example RAW10, YUV 10 bits)
00100: (BPP12) 12-bit words (for example RAW12)
00101: (BPP14) 14-bit words (for example RAW14)
00110: (BPP16) 16-bit words (for example RAW16)
Others: Reserved.

Note: The format selection is valid only if ALLDT = 0 and DT5EN = 1. These bits have no effect if ALLDT = 1.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **DT5[5:0]**: Data type 5 class selection for virtual channel x

These bits select the data type 5 class and subclass for the virtual channel x according to the MIPI specification for Camera Serial Interface 2.

40.9.7 CSI-2 Host line byte x configuration register (CSI_LBxCFGR)

Address offset: 0x50 + 0x04 * x, (x = 0 to 3)

Reset value: 0x0000 0000

Any change in this register while LBxEN bit is set into the CSI_PRGITR is ignored.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LINECNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BYTECNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **LINECNT[15:0]**: Line counter

These bits select the line number, associated to LINECNT[15:0], to trigger, if enabled, the corresponding line/byte interruption flag based on the selected virtual channel identified in CSI_PRGITR.

Bits 15:0 **BYTECNT[15:0]**: Byte counter

These bits select the byte number, associated to BYTECNT[15:0], to trigger, if enabled, the corresponding line/byte interruption flag based on the selected virtual channel identified in CSI_PRGITR.

40.9.8 CSI-2 Host timer x configuration register (CSI_TIMxCFGR)

Address offset: 0x60 + 0x04 * x, (x = 0 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	COUNT[24:16]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COUNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:0 **COUNT[24:0]**: Clock cycle counter

This register is used to specify how many clock cycles (clk_proc) the timer needs to count from a SOF on the considered virtual channel, before triggering the associated interrupt flag if enabled in CSI_PRGITR. The interruption is generated if TIMxIE is set in CSI_IER0.

40.9.9 CSI-2 Host lane merger configuration register (CSI_LMCFGR)

Address offset: 0x70

Reset value: 0x4321 0200

This register must not be modified while the CSI-2 is enabled.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DL1MAP[2:0]			Res.	DL0MAP[2:0]		
									rw	rw	rw		rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	LANENB[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
					rw	rw	rw								

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:20 **DL1MAP[2:0]**: Physical mapping of logical data lane 1

These bits are used to map the physical data lane 1 on the logical data lane.
001: Physical data lane 1 connected to logical data lane 0
010: Physical data lane 1 connected to logical data lane 1
Others: Reserved

Note: This field value must be different from DL0MAP[2:0].

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **DL0MAP[2:0]**: Physical mapping of logical data lane 0

These bits are used to map the physical data lane 0 on the logical data lane.
001: Physical data lane 0 connected to logical data lane 0
010: Physical data lane 0 connected to logical data lane 1
Others: Reserved

Note: This field value must be different from DL1MAP[2:0].

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:8 **LANENB[2:0]**: Number of lanes

These bits are used to specify the number of lanes used for the data reception.
001: 1 lane for the reception
010: 2 lanes for the reception
Others: Reserved

Bits 7:0 Reserved, must be kept at reset value.

40.9.10 CSI-2 Host program interrupt register (CSI_PRGITR)

Address offset: 0x74

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIM3 EN	TIM3 EOF	TIM3VC[1:0]		TIM2 EN	TIM2 EOF	TIM2VC[1:0]		TIM1 EN	TIM1 EOF	TIM1VC[1:0]		TIM0 EN	TIM0 EOF	TIM0VC[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LB3 EN	Res.	LB3VC[1:0]		LB2 EN	Res.	LB2VC[1:0]		LB1 EN	Res.	LB1VC[1:0]		LB0 EN	Res.	LB0VC[1:0]	
rw		rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw

Bit 31 **TIM3EN**: TIM3 base time enable

This bit is used to enable the mechanism of event generation starting from SOF or EOF depending on TIM3EOF. An interrupt is generated if TIM3IE is set int CSI_IER0.
0: Timer 3 clock cycle counter not active
1: Timer 3 clock cycle counter active

Note: Enabling this bit locks TIM3EOF and TIM3VC[1:0] values.
The counting starts from the next frame after setting this bit.
This bit must be written only when the CSI-2 is disabled.

Bit 30 **TIM3EOF**: TIM3 base time starting from the EOF

This bits is used to select the starting point of the counter on the selected virtual channel.
0: SOF selected as starting point for the counter
1: EOF selected as starting point for the counter

Note: The write access to this bit is ignored if TIM3EN = 1 in CSI_PRGITR.
This bit must be written only when the CSI-2 is disabled.

Bits 29:28 **TIM3VC[1:0]**: TIM3 base time linked to a virtual channel

These bits are used to select from which of the virtual channels the TIM3 counter is linked.
00: Virtual channel 0 used to generate the TIM3 event
01: Virtual channel 1 used to generate the TIM3 event
10: Virtual channel 2 used to generate the TIM3 event
11: Virtual channel 3 used to generate the TIM3 event

Note: The write access to these bits is ignored if TIM3EN = 1 in CSI_PRGITR.
These bits must be written only when the CSI-2 is disabled.

Bit 27 **TIM2EN**: TIM2 base time enable

This bit is used to enable the mechanism of event generation starting from SOF or EOF depending on TIM2EOF. An interrupt is generated if TIM2IE is set in CSI_IER0.

0: Timer 2 clock cycle counter not active.

1: Timer 2 clock cycle counter active

Note: Enabling this bit locks TIM2EOF and TIM2VC[1:0] values.

The counting starts from the next frame after setting this bit.

This bit must be written only when the CSI-2 is disabled.

Bit 26 **TIM2EOF**: TIM2 base time starting from the EOF

This bits is used to select the starting point of the counter on the selected virtual channel.

0: SOF selected as starting point for the counter

1: EOF selected as starting point for the counter

Note: The write access to this bit is ignored if TIM2EN = 1 in CSI_PRGITR.

This bit must be written only when the CSI-2 is disabled.

Bits 25:24 **TIM2VC[1:0]**: TIM2 base time linked to a virtual channel

These bits are used to select from which of the virtual channels the TIM2 counter is linked.

00: Virtual channel 0 used to generate the TIM2 event

01: Virtual channel 1 used to generate the TIM2 event

10: Virtual channel 2 used to generate the TIM2 event

11: Virtual channel 3 used to generate the TIM2 event

Note: The write access to these bits is ignored if TIM2EN = 1 in CSI_PRGITR.

These bits must be written only when the CSI-2 is disabled.

Bit 23 **TIM1EN**: TIM1 base time enable

This bit is used to enable the mechanism of event generation starting from SOF or EOF depending on TIM1EOF. An interrupt is generated if TIM1IE is set in CSI_IER0.

0: Timer 1 clock cycle counter not active.

1: Timer 1 clock cycle counter active

Note: Enabling this bit locks TIM1EOF and TIM1VC[1:0] values.

The counting starts from the next frame after setting this bit.

This bit must be written only when the CSI-2 is disabled.

Bit 22 **TIM1EOF**: TIM1 base time starting from the EOF

This bits is used to select the starting point of the counter on the selected virtual channel.

0: SOF selected as starting point for the counter

1: EOF selected as starting point for the counter

Note: The write access to this bit is ignored if TIM1EN = 1 in CSI_PRGITR.

This bit must be written only when the CSI-2 is disabled.

Bits 21:20 **TIM1VC[1:0]**: TIM1 base time linked to a virtual channel

These bits are used to select from which of the virtual channels the TIM1 counter is linked.

00: Virtual channel 0 used to generate the TIM1 event

01: Virtual channel 1 used to generate the TIM1 event

10: Virtual channel 2 used to generate the TIM1 event

11: Virtual channel 3 used to generate the TIM1 event

Note: The write access to these bits is ignored if TIM1EN = 1 in CSI_PRGITR.

These bits must be written only when the CSI-2 is disabled.

Bit 19 **TIM0EN**: TIM0 base time enable

This bit is used to enable the mechanism of event generation starting from SOF or EOF depending on TIM0EOF. An interrupt is generated if TIM0IE is set in CSI_IER0.

0: Timer 0 clock cycle counter not active

1: Timer 0 clock cycle counter active

Note: Enabling this bit locks TIM0EOF and TIM0VC[1:0] values.

The counting starts from the next frame after setting this bit.

This bit must be written only when the CSI-2 is disabled.

Bit 18 **TIM0EOF**: TIM0 base time starting from the EOF

This bits is used to select the starting point of the counter on the selected virtual channel.

0: SOF selected e as starting point for the counter

1: EOF selected as starting point for the counter

Note: The write access to this bit is ignored if TIM0EN = 1 in CSI_PRGITR.

This bit must be written only when the CSI-2 is disabled.

Bits 17:16 **TIM0VC[1:0]**: TIM0 base time linked to a virtual channel

These bits are used to select from which of the virtual channels the TIM0 counter is linked.

00: Virtual channel 0 used to generate the TIM0 event

01: Virtual channel 1 used to generate the TIM0 event

10: Virtual channel 2 used to generate the TIM0 event

11: Virtual channel 3 used to generate the TIM0 event

Note: The write access to these bits is ignored if TIM0EN = 1 in CSI_PRGITR.

These bits must be written only when the CSI-2 is disabled.

Bit 15 **LB3EN**: Line/byte 3 counter enable

This bit is used to enable the mechanism of event generation at any location in the frame corresponding to the value set in CSI_LB3CFGR. An interrupt is generated if LB3IE is set in CSI_IER0.

0: Line/byte 3 counter stopped

1: Line/byte 3 counter active

Note: Enabling this bit locks LB3VC[1:0] value.

The counter starts from the next frame after setting this bit.

This bit must be set only when the CSI-2 is enabled.

Bit 14 Reserved, must be kept at reset value.

Bits 13:12 **LB3VC[1:0]**: Line/byte counter 3 linked to a virtual channel

These bits are used to select from which of the virtual channels the line/byte counter 3 is linked.

00: Virtual channel 0 used to generate the line/byte counter 3 event

01: Virtual channel 1 used to generate the line/byte counter 3 event

10: Virtual channel 2 used to generate the line/byte counter 3 event

11: Virtual channel 3 used to generate the line/byte counter 3 event

Note: The write access to these bits is ignored if LB3EN = 1 in CSI_PRGITR.

Bit 11 **LB2EN**: Line/byte 2 counter enable

This bit is used to enable the mechanism of event generation at any location in the frame corresponding to the value set in CSI_LB2CFGR. An interrupt is generated if LB2IE is set in CSI_IER0.

0: Line/byte 2 counter stopped

1: Line/byte 2 counter active

Note: Enabling this bit locks LB2VC[1:0] value.

The counter starts from the next frame after setting this bit.

This bit must be set only when the CSI-2 is enabled.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **LB2VC[1:0]**: Line/byte counter 2 linked to a virtual channel

These bits are used to select from which of the virtual channels the line/byte counter 2 is linked.

00: Virtual channel 0 used to generate the line/byte counter 2 event

01: Virtual channel 1 used to generate the line/byte counter 2 event

10: Virtual channel 2 used to generate the line/byte counter 2 event

11: Virtual channel 3 used to generate the line/byte counter 2 event

Note: The write access to these bits is ignored if LB2EN = 1 in CSI_PRGITR.

Bit 7 **LB1EN**: Line/byte 1 counter enable

This bit is used to enable the mechanism of event generation at any location in the frame corresponding to the value set in CSI_LB1CFGR. An interrupt is generated if LB1IE is set in CSI_IER0.

0: Line/byte 1 counter stopped

1: Line/byte 1 counter active

Note: Enabling this bit locks LB1VC[1:0] value.

The counter starts from the next frame after setting this bit.

This bit must be set only when the CSI-2 is enabled.

Bit 6 Reserved, must be kept at reset value.

Bits 5:4 **LB1VC[1:0]**: Line/byte counter 1 linked to a virtual channel

These bits are used to select from which of the virtual channels the line/byte counter 1 is linked.

00: Virtual channel 0 used to generate the line/byte counter 1 event

01: Virtual channel 1 used to generate the line/byte counter 1 event

10: Virtual channel 2 used to generate the line/byte counter 1 event

11: Virtual channel 3 used to generate the line/byte counter 1 event

Note: The write access to these bits is ignored if LB1EN = 1 in CSI_PRGITR.

Bit 3 **LB0EN**: Line/byte 0 counter enable

This bit is used to enable the mechanism of event generation at any location in the frame corresponding to the value set in CSI_LB0CFGR. An interrupt is generated if LB0IE is set in CSI_IER0.

0: Line/byte 0 counter stopped

1: Line/byte 0 counter active

Note: Enabling this bit locks LB0VC[1:0] value.

The counter starts from the next frame after setting this bit.

This bit must be set only when the CSI-2 is enabled.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **LB0VC[1:0]**: Line/byte counter 0 linked to a virtual channel

These bits are used to select from which of the virtual channels the line/byte counter 0 is linked.

00: Virtual channel 0 used to generate the line/byte counter 0 event

01: Virtual channel 1 used to generate the line/byte counter 0 event

10: Virtual channel 2 used to generate the line/byte counter 0 event

11: Virtual channel 3 used to generate the line/byte counter 0 event

Note: The write access to these bits is ignored if LB0EN = 1 in CSI_PRGITR.

40.9.11 CSI-2 Host watchdog register (CSI_WDR)

Address offset: 0x78

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNT[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CNT[31:0]**: Watchdog counter

These bits contains the maximum number of clock cycles (clk_proc) between an EOF and a SOF for a given channel before triggering an interrupt if WDERRIE = 1 in CSI_IER0.

The value 0x000000 disables the watchdog counter.

40.9.12 CSI-2 Host interrupt enable register 0 (CSI_IER0)

Address offset: 0x80

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	SYNC ERRIE	WD ERRIE	SPKT ERRIE	ID ERRIE	CECC ERRIE	ECC ERRIE	CRC ERRIE	Res.	Res.	CCFIF OFIE	Res.	Res.	Res.	Res.	SPKTIE
	rw	rw	rw	rw	rw	rw	rw			rw					rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EOF3 IE	EOF2 IE	EOF1 IE	EOF0 IE	SOF3 IE	SOF2 IE	SOF1 IE	SOF0 IE	TIM3 IE	TIM2 IE	TIM1 IE	TIM0 IE	LB3 IE	LB2 IE	LB1 IE	LB0 IE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **SYNCERRIE**: Invalid synchronization error interrupt enable

This bit enables the interrupt generation when an invalid synchronization is detected. It can be a SOF short packet during a frame, or and EOF short packet during an inter-frame.

0: Interrupt on invalid synchronization error disabled

1: Interrupt on invalid synchronization error enabled

Bit 29 **WDERRIE**: Watchdog error interrupt enable
This bit enables the interrupt generation when the time between two packets for a virtual channel is longer than the programmed watchdog.
0: Interrupt on watchdog error disabled
1: Interrupt on watchdog error enabled

Bit 28 **SPKTERRIE**: Short packet error interrupt enable
This bit enables the interrupt generation based on a reception of a shorter packet than expected.
0: Interrupt on short packet error disabled
1: Interrupt on short packet error enabled

Bit 27 **IDERRIE**: Data type ID error interrupt enable
This bit enables the interrupt generation based on a detection of a reserved data type.
0: Interrupt on reception of reserved data type disabled
1: Interrupt on reception of reserved data type enabled

Bit 26 **CECCERRIE**: Corrected ECC error interrupt enable
This bit enables the interrupt generation based on a corrected ECC error detection.
0: Interrupt on corrected ECC error detection disabled
1: Interrupt on corrected ECC error detection enabled

Bit 25 **ECCERRIE**: ECC error interrupt enable
This bit enables the interrupt generation based on an ECC error detection.
0: Interrupt on ECC error detection disabled
1: Interrupt on ECC error detection enabled

Bit 24 **CRCERRIE**: CRC error interrupt enable
This bit enables the interrupt generation based on CRC error detection on a long packet reception.
0: Interrupt on CRC error detection disabled
1: Interrupt on CRC error detection enabled

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **CCFIFOFIE**: Clock changer FIFO full interrupt enable
This bit enables the interrupt generation based on clock changer FIFO full detection.
0: Interrupt on clock changer FIFO full detection disabled
1: Interrupt on clock changer FIFO full detection enabled

Bits 20:17 Reserved, must be kept at reset value.

Bit 16 **SPKTIE**: Short packet interrupt enable
This bit generates an interrupt which notifies that a generic short packet has been received by the CSI-2 Host.
0: Short packet detection interrupt disabled
1: Short packet detection interrupt enabled

Bit 15 **EOF3IE**: EOF for virtual channel 3 interrupt enable
This bit enables the interrupt generation when the CSI-2 Host is decoding an EOF on the virtual channel 3.
0: EOF on virtual channel 3 interrupt disabled
1: EOF on virtual channel 3 interrupt enabled

Bit 14 **EOF2IE**: EOF for virtual channel 2 interrupt enable
This bit enables the interrupt generation when the CSI-2 Host is decoding an EOF on the virtual channel 2.
0: EOF on virtual channel 2 interrupt disabled
1: EOF on virtual channel 2 interrupt enabled

Bit 13 **EOF1IE**: EOF for virtual channel 1 interrupt enable
This bit enables the interrupt generation when the CSI-2 Host is decoding an EOF on the virtual channel 1.
0: EOF on virtual channel 1 interrupt disabled
1: EOF on virtual channel 1 interrupt enabled

Bit 12 **EOF0IE**: EOF for virtual channel 0 interrupt enable
This bit enables the interrupt generation when the CSI-2 Host is decoding an EOF on the virtual channel 0.
0: EOF on virtual channel 0 interrupt disabled
1: EOF on virtual channel 0 interrupt enabled

Bit 11 **SOF3IE**: SOF for virtual channel 3 interrupt enable
This bit enables the interrupt generation when the CSI-2 Host is decoding a SOF on the virtual channel 3.
0: SOF on virtual channel 3 interrupt disabled
1: SOF on virtual channel 3 interrupt enabled

Bit 10 **SOF2IE**: SOF for virtual channel 2 interrupt enable
This bit enables the interrupt generation when the CSI-2 Host is decoding a SOF on the virtual channel 2.
0: SOF on virtual channel 2 interrupt disabled
1: SOF on virtual channel 2 interrupt enabled

Bit 9 **SOF1IE**: SOF for virtual channel 1 interrupt enable
This bit enables the interrupt generation when the CSI-2 Host is decoding a SOF on the virtual channel 1.
0: SOF on virtual channel 1 interrupt disabled.
1: SOF on virtual channel 1 interrupt enabled.

Bit 8 **SOF0IE**: SOF for virtual channel 0 interrupt enable
This bit enables the interrupt generation when the CSI-2 Host is decoding a SOF on the virtual channel 0.
0: SOF on virtual channel 0 interrupt disabled
1: SOF on virtual channel 0 interrupt enabled

Bit 7 **TIM3IE**: Timer 3 interrupt enable
This bit enables the interrupt generation from the timer 3 linked to the virtual channel number configured by TIM3VC in CSI_PRGITR. The interrupt raises when the timer 3 value reaches the value configured in CSI_TIM3CFGR.
0: Timer 3 interrupt disabled
1: Timer 3 interrupt enabled

Bit 6 **TIM2IE**: Timer 2 interrupt enable
This bit enables the interrupt generation from the timer 2 linked to the virtual channel number configured by TIM2VC in CSI_PRGITR. The interrupt raises when the timer 2 value reaches the value configured in CSI_TIM2CFGR.
0: Timer 2 interrupt disabled
1: Timer 2 interrupt enabled

Bit 5 **TIM1IE**: Timer 1 interrupt enable

This bit enables the interrupt generation from the timer 1 linked to the virtual channel number configured by TIM1VC in CSI_PRGITR. The interrupt raises when the timer 1 value reaches the value configured in CSI_TIM1CFGR.

0: Timer 1 interrupt disabled

1: Timer 1 interrupt enabled

Bit 4 **TIM0IE**: Timer 0 interrupt enable

This bit enables the interrupt generation from the timer 0 linked to the virtual channel number configured by TIM0VC in CSI_PRGITR. The interrupt raises when the timer 0 value reaches the value configured in CSI_TIM0CFGR.

0: Timer 0 interrupt disabled

1: Timer 0 interrupt enabled

Bit 3 **LB3IE**: Line/byte counter 3 interrupt enable

This bit enables the interrupt generation when the active frame of the virtual channel configured by LB3VC in CSI_PRGITR reaches the line/byte number programmed in CSI_LB3CFGR.

Bit 2 **LB2IE**: Line/byte counter 2 interrupt enable

This bit enables the interrupt generation when the active frame of the virtual channel configured by LB2VC in CSI_PRGITR reaches the line/byte number programmed in CSI_LB2CFGR.

Bit 1 **LB1IE**: Line/byte counter 1 interrupt enable

This bit enables the interrupt generation when the active frame of the virtual channel configured by LB1VC in CSI_PRGITR reaches the line/byte number programmed in CSI_LB1CFGR.

Bit 0 **LB0IE**: Line/byte counter 0 interrupt enable

This bit enables the interrupt generation when the active frame of the virtual channel configured by LB0VC in CSI_PRGITR reaches the line/byte number programmed in CSI_LB0CFGR.

40.9.13 CSI-2 Host interrupt enable register 1 (CSI_IER1)

Address offset: 0x84

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	ECTRLDL1IE	ESYNCESCDL1IE	EESCDL1IE	ESOTSYNCDL1IE	ESOTDL1IE	Res.	Res.	Res.	ECTRLDL0IE	ESYNCESCDL0IE	EESCDL0IE	ESOTSYNCDL0IE	ESOTDL0IE
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **ECTRLDL1IE**: D-PHY_RX lane 1 control error interrupt enable
This bit enables the D-PHY_RX control error interrupt detected on lane 1 consecutive to an incorrect line state sequence.
0: Lane 1 control error interrupt disabled
1: Lane 1 control error interrupt enabled

Bit 11 **ESYNCESCDL1IE**: D-PHY_RX lane 1 low-power data transmission synchronization error interrupt enable
This bit enables the interrupt generation when the number of bits received during a low-power data transmission is not a multiple of eight when the transmission ends on lane 1.
0: Lane 1 low-power data transmission interrupt error disabled
1: Lane 1 low-power data transmission Interrupt error enabled

Bit 10 **EESCDL1IE**: D-PHY_RX lane 1 escape entry error interrupt enable
This bit enables the interrupt generation when an unrecognized escape entry command is received on lane 1.
0: Lane 1 unrecognized escape entry command interrupt error disabled
1: Lane 1 unrecognized escape entry command interrupt error enabled

Bit 9 **ESOTSYNCDL1IE**: SOT synchronization interrupt error enable on lane 1
This bit enables the interrupt generation when the high-speed SOT leader sequence is corrupted in a such way that the proper synchronization cannot be expected.
0: Lane 1 SOT synchronization interrupt error disabled
1: Lane 1 SOT synchronization interrupt error enabled

Bit 8 **ESOTDL1IE**: SOT error interrupt enable on lane 1
This bit enables the interrupt generation when the high-speed SOT leader sequence is corrupted in a such way that the proper synchronization can still be achieved.
0: Lane 1 SOT interrupt error disabled
1: Lane 1 SOT interrupt error enabled

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **ECTRLDL0IE**: D-PHY_RX lane 0 control error interrupt enable
This bit enables the D-PHY_RX control error interrupt detected on lane 0 consecutive to an incorrect line state sequence.
0: Lane 0 control error interrupt disabled
1: Lane 0 control error interrupt enabled

Bit 3 **ESYNCESCDL0IE**: D-PHY_RX lane 0 low power data transmission synchronization error interrupt enable
This bit enables the interrupt generation when the number of bits received during a low-power data transmission is not a multiple of eight when the transmission ends on lane 1.
0: Lane 0 low-power data transmission interrupt error disabled
1: Lane 0 low-power data transmission Interrupt error enabled

Bit 2 **EESCDL0IE**: D-PHY_RX lane 0 escape entry error interrupt enable
This bit enables the interrupt generation when an unrecognized escape entry command is received on lane 0.
0: Lane 0 unrecognized escape entry command interrupt error disabled
1: Lane 0 unrecognized escape entry command interrupt error enabled

Bit 1 **ESOTSYNCDL0IE**: SOT synchronization interrupt error enable on lane 0
This bit enables the interrupt generation when the high-speed SOT leader sequence is corrupted in a such way that the proper synchronization cannot be expected.
0: Lane 0 SOT synchronization interrupt error disabled
1: Lane 0 SOT synchronization interrupt error enabled

Bit 0 **ESOTDL0IE**: SOT error interrupt enable on lane 0

This bit enables the interrupt generation when the high-speed SOT leader sequence is corrupted in a such way that the proper synchronization can still be achieved.

0: Lane 0 SOT interrupt error disabled

1: Lane 0 SOT interrupt error enabled

40.9.14 CSI-2 Host status register 0 (CSI_SR0)

Address offset: 0x90

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	SYNCERRF	WDERRF	SPKTERRF	IDERRF	CECCERRF	ECCERRF	CRCERRF	Res.	Res.	CCFIFOFF	VC3STATEF	VC2STATEF	VC1STATEF	VC0STATEF	SPKTF
	r	r	r	r	r	r	r			r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EOF3F	EOF2F	EOF1F	EOF0F	SOF3F	SOF2F	SOF1F	SOF0F	TIM3F	TIM2F	TIM1F	TIM0F	LB3F	LB2F	LB1F	LB0F
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bit 30 **SYNCERRF**: Invalid synchronization error flag

This bit indicates an invalid synchronization is detected. It can be a SOF short packet during a frame, or and EOF short packet during an inter-frame.

Bit 29 **WDERRF**: Watchdog error flag

This bit indicates that the time between two packets for a virtual channel is longer than the programmed watchdog.

Bit 28 **SPKTERRF**: Short packet error flag

This bit indicates a reception of a shorter packet than expected.

Bit 27 **IDERRF**: Data type ID error flag

This bit indicates a detection of a reserved data type.

Bit 26 **CECCERRF**: Corrected ECC error flag

This bit indicates a corrected ECC error detection took place in the frame.

Bit 25 **ECCERRF**: ECC error flag

This bit indicates that an ECC error has been detected.

Bit 24 **CRCERRF**: CRC error flag

This bit indicates a CRC error detection on a long packet reception.

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **CCFIFOFF**: Clock changer FIFO full flag

This bit indicates a FIFO full detection in the clock changer.

Bit 20 **VC3STATEF**: Virtual channel 3 state flag

This bit indicates the state of the virtual channel 3.

0: Virtual channel inactive

1: Virtual channel active

Bit 19 **VC2STATEF**: Virtual channel 2 state flag
This bit indicates the state of the virtual channel 2.
0: Virtual channel inactive
1: Virtual channel active

Bit 18 **VC1STATEF**: Virtual channel 1 state flag
This bit indicates the state of the virtual channel 1.
0: Virtual channel inactive
1: Virtual channel active

Bit 17 **VC0STATEF**: Virtual channel 0 state flag
This bit indicates the state of the virtual channel 0.
0: Virtual channel inactive
1: Virtual channel active

Bit 16 **SPKTF**: Short packet flag
This bit notifies that a generic short packet has been received by the CSI-2 Host.

Bit 15 **EOF3F**: EOF flag for virtual channel 3
This bit notifies that the CSI-2 Host decodes an EOF on the virtual channel 3.

Bit 14 **EOF2F**: EOF flag for virtual channel 2
This bit notifies that the CSI-2 Host is decoding an EOF on the virtual channel 2.

Bit 13 **EOF1F**:EOF flag for virtual channel 1
This bit notifies that the CSI-2 Host is decoding an EOF on the virtual channel 1.

Bit 12 **EOF0F**: EOF flag for virtual channel 0
This bit notifies that the CSI-2 Host is decoding an EOF on the virtual channel 0.

Bit 11 **SOF3F**: SOF flag for virtual channel 3
This bit notifies that the CSI-2 Host is decoding a SOF on the virtual channel 3.

Bit 10 **SOF2F**: SOF flag for virtual channel 2
This bit notifies that the CSI-2 Host is decoding a SOF on the virtual channel 2.

Bit 9 **SOF1F**: SOF flag for virtual channel 1
This bit notifies that the CSI-2 Host is decoding a sSOF on the virtual channel 1.

Bit 8 **SOF0F**: SOF flag for virtual channel 0
This bit notifies that the CSI-2 Host is decoding a SOF on the virtual channel 0.

Bit 7 **TIM3F**: Timer 3 flag
This bit indicates that the timer 3 linked to the virtual channel configured by TIM3VC in CSI_PRGITR reaches the value programmed in CSI_TIM3CFGR.

Bit 6 **TIM2F**: Timer 2 flag
This bit indicates that the timer 2 linked to the virtual channel configured by TIM2VC in CSI_PRGITR reaches the value programmed in CSI_TIM2CFGR.

Bit 5 **TIM1F**: Timer 1 flag
This bit indicates that the timer 1 linked to the virtual channel configured by TIM1VC in CSI_PRGITR reaches the value programmed in CSI_TIM1CFGR.

Bit 4 **TIM0F**: Timer 0 flag
This bit indicates that the timer 0 linked to the virtual channel configured by TIM0VC in CSI_PRGITR reaches the value programmed in CSI_TIM0CFGR.

Bit 3 **LB3F**: Line/byte counter 3 flag
This bit notifies that the active frame of the virtual channel configured by LB3VC in CSI_PRGITR has reached the line/byte number programmed in th CSI_LB3CFGR.

Bit 2 **LB2F**: Line/byte counter 2 flag
This bit notifies that the active frame of the virtual channel configured by LB2VC in CSI_PRGITR has reached the line/byte number programmed in CSI_LB2CFGR.

Bit 1 **LB1F**: Line/byte counter 1 flag
This bit notifies that the active frame of the virtual channel configured by LB1VC in CSI_PRGITR has reached the line/byte number programmed in CSI_LB1CFGR.

Bit 0 **LB0F**: Line/byte counter 0 flag
This bit notifies that the active frame of the virtual channel configured by LB0VC in CSI_PRGITR has reached the line/byte number programmed in CSI_LB0CFGR.

40.9.15 CSI-2 Host status register 1 (CSI_SR1)

Address offset: 0x94

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACTCLF	ULPNCLF	ULPNACTF	STOPCLF	Res.	ULPNDL1F	STOPDL1F	SKCALDL1F	SYNCDL1F	ACTDL1F	Res.	ULPNDL0F	STOPDL0F	SKCALDL0F	SYNCDL0F	ACTDL0F
r	r	r	r		r	r	r	r	r		r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	ECTRLDL1F	ESYNCESCDL1F	EESCDL1F	ESOTSYNCDL1F	ESOTDL1F	Res.	Res.	Res.	ECTRLDL0F	ESYNCESCDL0F	EESCDL0F	ESOTSYNCDL0F	ESOTDL0F
			r	r	r	r	r				r	r	r	r	r

Bit 31 **ACTCLF**: D-PHY_RX receiver clock active flag
This bit indicates the status of RxClkactiveHs D-PHY_RX signal.

Bit 30 **ULPNCLF**: D-PHY_RX receiver Ultra-Low power state (not) on clock lane.
This bit indicates the status of RxUlpsClkNot D-PHY_RX signal.

Bit 29 **ULPNACTF**: D-PHY_RX receiver ULP state (not) active
This bit indicates the status of UlpsactiveNotC D-PHY_RX signal.

Bit 28 **STOPCLF**: D-PHY_RX receiver in stop state for the clock lane
This bit indicates the status of StopStateC D-PHY_RX signal.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **ULPNDL1F**: D-PHY_RX receiver ultra-low-power state (not) active on data lane 1
This bit indicates the status of UlpsActiveNot_D_lane1 D-PHY_RX signal.

Bit 25 **STOPDL1F**: D-PHY_RX receiver data lane 1 in stop state
This bit indicates the status of StopState_D_lane1 D-PHY_RX signal.

Bit 24 **SKCALDL1F**: D-PHY_RX lane 1 high-speed skew calibration
This bit indicates the status of RxSkewCalHS_lane1 D-PHY_RX signal.

Bit 23 **SYNCDL1F**: D-PHY_RX lane 1 receiver synchronization observed
This bit indicates the status of RXSyncHS_lane1 D-PHY_RX signal.

Bit 22 **ACTDL1F**: D-PHY_RX lane 1 high-speed reception active
This bit indicates the status of RxActiveHS_lane1 D-PHY_RX signal.

Bit 21 Reserved, must be kept at reset value.

Bit 20 **ULPNDL0F**: D-PHY_RX receiver ultra-low-power state (not) active on data lane 0
This bit indicates the status of UlpsActiveNot_D_lane0 D-PHY_RX signal.

Bit 19 **STOPDL0F**: D-PHY_RX receiver data lane 0 in stop state
This bit indicates the status of StopState_D_lane0 D-PHY_RX signal.

Bit 18 **SKCALDL0F**: D-PHY_RX lane 0 high-speed skew calibration
This bit indicates the status of RxSkewCalHS_lane0 D-PHY_RX signal.

Bit 17 **SYNCDL0F**: D-PHY_RX lane 0 receiver synchronization observed
This bit indicates the status of RXSyncHS_lane0 D-PHY_RX signal.

Bit 16 **ACTDL0F**: D-PHY_RX lane 0 high-speed reception active
This bit indicates the status of RxActiveHS_lane0 D-PHY_RX signal.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **ECTRLDL1F**: D-PHY_RX lane 1 control error flag
This bit indicates that the D-PHY_RX detected on lane 1 an incorrect line state sequence.

Bit 11 **ESYNCESCDL1F**: D-PHY_RX lane 1 low-power data transmission synchronization error flag
This bit notifies a synchronization error when the number of bits received during a low-power data transmission is not a multiple of eight when the transmission ends on lane 1.

Bit 10 **EESCDL1F**: D-PHY_RX lane 1 escape entry error flag
This bit notifies an escape entry error when an unrecognized escape entry command is received on lane 1.

Bit 9 **ESOTSYNCDL1F**: SOT synchronization error flag on lane 1
This bit indicates a SOT synchronization error when the high-speed SOT leader sequence is corrupted in a such way that the proper synchronization cannot be expected.

Bit 8 **ESOTDL1F**: SOT error flag on lane 1
This bit notifies a SOT error when the high-speed SOT leader sequence is corrupted in a such way that the proper synchronization can still be achieved.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **ECTRLDL0F**: D-PHY_RX lane 0 control error flag
This bit indicates that the D-PHY_RX detected on lane 0 an incorrect line state sequence.

Bit 3 **ESYNCESCDL0F**: D-PHY_RX lane 0 low-power data transmission synchronization error flag
This bit notifies a synchronization error when the number of bits received during a low-power data transmission is not a multiple of eight when the transmission ends on lane 0.

Bit 2 **EESCDL0F**: D-PHY_RX lane 0 escape entry error flag
This bit notifies an escape entry error when an unrecognized escape entry command is received on lane 0.

Bit 1 **ESOTSYNCDL0F**: SOT synchronization error flag on lane 0
This bit indicates a SOT synchronization error when the high-speed SOT leader sequence is corrupted in a such way that the proper synchronization cannot be expected.

Bit 0 **ESOTDL0F**: SOT error flag on lane 0
This bit notifies a SOT error when the high-speed SOT leader sequence is corrupted in a such way that the proper synchronization can still be achieved.

40.9.16 CSI-2 Host flag clear register 0 (CSI_FCR0)

Address offset: 0x100

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	CSYNCERRF	CWDERRF	CSPKTERRF	CIDERRF	CCECCERRF	CECCERRF	CCRCERRF	Res.	Res.	CCCFIFOFF	Res.	Res.	Res.	Res.	CSPKTF
	w	w	w	w	w	w	w			w					w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CEOF3F	CEOF2F	CEOF1F	CEOF0F	CSOF3F	CSOF2F	CSOF1F	CSOF0F	CTIM3F	CTIM2F	CTIM1F	CTIM0F	CLB3F	CLB2F	CLB1F	CLB0F
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit 31 Reserved, must be kept at reset value.

Bit 30 **CSYNCERRF**: Clear invalid synchronization error flag
Write 1 clears SYNCERRF flag in CSI_SR0.

Bit 29 **CWDERRF**: Clear watchdog error flag
Write 1 clears WDERRF flag in CSI_SR0.

Bit 28 **CSPKTERRF**: Clear short packet error flag
Write 1 clears SPKTERRF flag in CSI_SR0.

Bit 27 **CIDERRF**: Clear data type ID error flag
Write 1 clears CIDERRF flag in CSI_SR0.

Bit 26 **CCECCERRF**: Clear corrected ECC error flag
Write 1 clears CCECCERRF flag in CSI_SR0.

Bit 25 **CECCERRF**: Clear ECC error flag
Write 1 clears CECCERRF flag in CSI_SR0.

Bit 24 **CCRCERRF**: Clear CRC error flag
Write 1 clears CRCERRF flag in CSI_SR0.

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **CCCFIFOFF**: Clear clock changer FIFO full flag
Write 1 clears CCFIFOFF flag in CSI_SR0.

Bits 20:17 Reserved, must be kept at reset value.

Bit 16 **CSPKTF**: Clear short packet flag
Write 1 clears SPKTF flag in CSI_SR0.

Bit 15 **CEOF3F**: Clear EOF flag for virtual channel 3
Write 1 clears EOF3F flag in CSI_SR0.

Bit 14 **CEOF2F**: Clear EOF flag for virtual channel 2
Write 1 clears EOF2F flag in CSI_SR0.

Bit 13 **CEOF1F**: Clear EOF flag for virtual channel 1
Write 1 clears EOF1F flag in CSI_SR0.

Bit 12 **CEOF0F**: Clear EOF flag for virtual channel 0
Write 1 clears EOF0F flag in CSI_SR0.

Bit 11 **CSOF3F**: Clear SOF flag for virtual channel 3
Write 1 clears SOF3F flag in CSI_SR0.

Bit 10 **CSOF2F**: Clear SOF flag for virtual channel 2
Write 1 clears SOF2F flag in CSI_SR0.

Bit 9 **CSOF1F**: Clear SOF flag for virtual channel 1
Write 1 clears SOF1F flag in CSI_SR0.

Bit 8 **CSOF0F**: Clear SOF flag for virtual channel 0
Write 1 clears SOF0F flag in CSI_SR0.

Bit 7 **CTIM3F**: Clear timer 3 flag
Write 1 clears TIM3F flag in CSI_SR0.

Bit 6 **CTIM2F**: Clear timer 2 flag
Write 1 clears TIM2F flag in CSI_SR0.

Bit 5 **CTIM1F**: Clear timer 1 flag
Write 1 clears TIM1F flag in CSI_SR0.

Bit 4 **CTIM0F**: Clear timer 0 flag
Write 1 clears TIM0F flag in CSI_SR0.

Bit 3 **CLB3F**: Clear line/byte counter 3 flag
Write 1 clears LB3F flag in CSI_SR0.

Bit 2 **CLB2F**: Clear line/byte counter 2 flag
Write 1 clears LB2F flag in CSI_SR0.

Bit 1 **CLB1F**: Clear line/byte counter 1 flag
Write 1 clears LB1F flag in CSI_SR0.

Bit 0 **CLB0F**: Clear line/byte counter 0 flag
Write 1 clears LB0F flag in CSI_SR0.

40.9.17 CSI-2 Host flag clear register 1 (CSI_FCR1)

Address offset: 0x104

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CECTRLDL1F	CESYNCESCDL1F	CEESCDL1F	CESOTSYNCDL1F	CESOTDL1F	Res.	Res.	Res.	CECTRLDL0F	CESYNCESCDL0F	CEESCDL0F	CESOTSYNCDL0F	CESOTDL0F
			w	w	w	w	w				w	w	w	w	w

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **CECTRLDL1F**: Clear D-PHY_RX lane 1 control error flag
Write 1 clears ECTRLDL1F flag in CSI_SR1.

Bit 11 **CESYNCESCDL1F**: Clear D-PHY_RX lane 1 low-power data transmission synchronization error flag
Write 1 clears ESYNCESCDL1F flag in CSI_SR1.

Bit 10 **CEESCDL1F**: Clear D-PHY_RX lane 1 escape entry error flag
Write 1 clears the EESCDL1F flag in the CSI_SR1 register

Bit 9 **CESOTSYNCDL1F**: Clear SOT synchronization error flag on lane 1
Write 1 clears ESOTSYNCDL1F flag in CSI_SR1.

Bit 8 **CESOTDL1F**: Clear SOT error flag on lane 1
Write 1 clears ESOTDL1F flag in CSI_SR1.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **CECTRLDL0F**: Clear D-PHY_RX lane 0 control error flag
Write 1 clears ECTRLDL0F in CSI_SR1.

Bit 3 **CESYNCESCDL0F**: Clear D-PHY_RX lane 0 low-power data transmission synchronization error flag
Write 1 clears ESYNCESCDL0F flag in CSI_SR1.

Bit 2 **CEESCDL0F**: Clear D-PHY_RX lane 0 escape entry error flag
Write 1 clears EESCDL0F flag in CSI_SR1.

Bit 1 **CESOTSYNCDL0F**: Clear SOT synchronization error flag on lane 0
Write 1 clears ESOTSYNCDL0F flag in CSI_SR1.

Bit 0 **CESOTDL0F**: Clear SOT error flag on lane 0
Write 1 clears ESOTDL0F flag in CSI_SR1.

40.9.18 CSI-2 Host short packet data field register (CSI_SPDFR)

Address offset: 0x110

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VCHANNEL[1:0]		DATATYPE[5:0]					
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAFIELD[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:22 **VCHANNEL[1:0]**: Virtual channel
This bit indicates which is the virtual channel number decoded in the last generic short packet data field reception. Write to this bit has no effect.

Bits 21:16 **DATATYPE[5:0]**: Data type class
This bit indicates which is the data type class and subclass (according to the MIPI specification for CSI-2) decoded in the last generic short packet data field reception. Write to this bit has no effect.

Bits 15:0 **DATAFIELD[15:0]**: Data field
This bit indicates which is the current data field decoded in the last generic short packet data field reception. Write to this bit has no effect.

40.9.19 CSI-2 Host error register 1 (CSI_ERR1)

Address offset: 0x114

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDVCERR[1:0]		IDDTERR[5:0]					
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CECCVCERR [1:0]		CECCDTERR[5:0]						CRCVCERR [1:0]		CRCDTERR [5:0]					
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:22 **IDVCERR[1:0]**: Virtual channel having ID error
This bit indicates for which virtual channel an ID error has been detected.
IDERRF is set in CSI_SR0 to highlight the error.

Bits 21:16 **IDDTERR[5:0]**: Data type in error
This bit indicates for which data type an ID error has been detected.
IDERRF is set in CSI_SR0 to highlight the error.

Bits 15:14 **CECCVCERR[1:0]**: Virtual channel having a corrected ECC error
This bit indicates for which virtual channel a corrected ECC error has been detected.
CECCERRF is set in CSI_SR0 to highlight the error.

Bits 13:8 **CECCDTERR[5:0]**: Data type having a corrected ECC error
This bit indicates for which data type a corrected ECC error has been detected.
CECCERRF is set in CSI_SR0 to highlight the error.

Bits 7:6 **CRCVCERR[1:0]**: Virtual channel having a CRC error
This bit indicates for which virtual channel a CRC error has been detected.
CRCERRF is set in CSI_SR0 to highlight the error.

Bits 5:0 **CRCDTERR[5:0]**: Data type having a CRC error
This bit indicates for which data type a CRC error has been detected.
CRCERRF is set in CSI_SR0 to highlight the error.

40.9.20 CSI-2 Host error register 2 (CSI_ERR2)

Address offset: 0x118

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYNCVCERR [1:0]		WDVCERR [1:0]	
												r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPKTVCERR [1:0]		SPKTDTERR [5:0]					
								r	r	r	r	r	r	r	r

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:18 **SYNCVCERR[1:0]**: Virtual channel having synchronization error
This bit indicates for which virtual channel a synchronization error has been detected. SYNCERRF is set in CSI_SR0 to highlight the error.

Bits 17:16 **WDVCERR[1:0]**: Virtual channel having a watchdog error
This bit indicates for which virtual channel a watchdog error has been detected. WDERRF is set in CSI_SR0 to highlight the error.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:6 **SPKTVCERR[1:0]**: Virtual channel having a short packet error
This bit indicates for which virtual channel a short packet error has been detected. SPKTERRF is set in CSI_SR0 to highlight the error.

Bits 5:0 **SPKTDTERR[5:0]**: Data type having a short packet error
This bit indicates for which data type a shorter packet than expected error has been detected. SPKTERRF is set in CSI_SR0 to highlight the error.

40.10 CSI PHY registers

40.10.1 CSI PHY reset control register (CSI_PRCR)

Address offset: 0x1000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PEN	Res.
														rw	

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **PEN**:
When set to 0, this bit places the digital section of the D-PHY in the reset state.
0: PHY is disabled (in reset state).
1: PHY is enabled (out of reset state).

Bit 0 Reserved, must be kept at reset value.

40.10.2 CSI PHY mode control register (CSI_PMCR)

Address offset: 0x1004

Reset value: 0x0001 1003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TUEXDL0
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	TUESDL0	Res.	Res.	Res.	RTDL0	Res.	Res.	Res.	DTDL	Res.	FTXSMDL0	FRXMDL1	FRXMDL0
			rw				rw				rw		rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **TUEXDL0**: Tx ULP exit sequence data lane 0
This bit initiates the transmission of the ULP exit sequence.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **TUESDL0**: Tx ULP escape-mode data lane 0

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **RTDL0**: Turn-around request data lane 0
This bit is used to indicate that the protocol layer needs to turn the lane around, allowing from the other side to start transmitting the data.
0: No request
1: Request for turn-around for DL0

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **DTDL**: Disable turn-around data lane 0
This bit is used to prevent the bi-directional lane from processing a turn-around request in the lane interconnect
0: Disabled
1: Enabled

Bit 3 Reserved, must be kept at reset value.

Bit 2 **FTXSMDL0**: Force to Tx Stop mode the data lane 0
0: Disabled
1: Enabled

Bit 1 **FRXMDL1**: Force to Rx mode the data lane 1
Module 0 into receive mode/wait for stop state.
0: Disabled
1: Enabled

Bit 0 **FRXMDL0**: Force to Rx mode the data lane 0
Module 0 into receive mode/wait for stop state.
0: Disabled
1: Enabled

40.10.3 CSI PHY frequency control register (CSI_PFCR)

Address offset: 0x1008

Reset value: 0x0000 0017

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DLD
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	HSFR[6:0]							Res.	Res.	CCFR[5:0]					
	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **DLD**: Data lane direction of lane 0
0: Rx
1: Tx

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **HSFR[6:0]**: PHY high-speed frequency range selection
Must be set when the CSI PHY is in power-down, and must not change during operation.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **CCFR[5:0]**: Configuration clock frequency range selection
Must be set at value = round (Fcfg_clk(MHz) - 17) * 4.

40.10.4 CSI PHY test control register 0 (CSI_PTCR0)

Address offset: 0x1010

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRSEN	TCKEN
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **TRSEN**: Test-interface reset enable for the TDI bus into the PHY

Bit 0 **TCKEN**: Test-interface clock enable for the TDI bus into the PHY

40.10.5 CSI PHY test control register 1 (CSI_PTCR1)

Address offset: 0x1014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TWM
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDI[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **TWM**: Test-interface write mode selector
0: Data write operation is set on the rising edge of the testclk signal.
1: Address write operation is set on the falling edge of the testclk signal.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **TDI[7:0]**: Test-interface data in

40.10.6 CSI PHY test status register (CSI_PTSR)

Address offset: 0x1018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDO[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TDO[7:0]**: CSI PHY test interface data output bus for read-back and internal probing functionalities

40.10.7 CSI-2 Host and PHY register map

Table 372. CSI-2 Host and PHY register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	CSI_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VC3STOP	VC3START	Res.	Res.	VC2STOP	VC2START	Res.	Res.	VC1STOP	VC1START	Res.	Res.	VC0STOP	VC0START	Res.	CSIEN.
	Reset value																	0	0			0	0			0	0			0	0		0

Table 372. CSI-2 Host and PHY register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x04	CSI_PCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DL1EN	DL0EN	CLEN	PWRDOWN
	Reset value																													0	0	0	0
0x08-0x0C	Reserved	Reserved																															
0x10 + 0x10*x (x=0 t0 3)	CSI_ VCxCFGR1	Res.	Res.	Res.	DT0FT[4:0]					Res.	Res.	DT0[5:0]						Res.	Res.	Res.	CDTFT[4:0]					DT6EN	DT5EN	DT4EN	DT3EN	DT2EN	DT1EN	DT0EN	ALLDT
	Reset value				0	0	0	0	0			0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14 + 0x10*x (x=0 t0 3)	CSI_ VCxCFGR2	Res.	Res.	Res.	DT2FT[4:0]					Res.	Res.	DT2[5:0]						Res.	Res.	Res.	DT1FT[4:0]					Res.	Res.	DT1[5:0]					
	Reset value				0	0	0	0	0			0	0	0	0	0	0				0	0	0	0	0			0	0	0	0	0	0	0
0x18+ 0x10*x (x=0 t0 3)	CSI_ VCxCFGR3	Res.	Res.	Res.	DT4FT[4:0]					Res.	Res.	DT4[5:0]						Res.	Res.	Res.	DT3FT[4:0]					Res.	Res.	DT3[5:0]					
	Reset value				0	0	0	0	0			0	0	0	0	0	0				0	0	0	0	0			0	0	0	0	0	0	0
0x1C + 0x10*x (x=0 t0 3)	CSI_ VCxCFGR4	Res.	Res.	Res.	DT6FT[4:0]					Res.	Res.	DT6[5:0]						Res.	Res.	Res.	DT5FT[4:0]					Res.	Res.	DT5[5:0]					
	Reset value				0	0	0	0	0			0	0	0	0	0	0				0	0	0	0	0			0	0	0	0	0	0	0
0x50 + 0x04*x (x=0 t0 3)	CSI_LBxCFGR	LINECNT[15:0]																BYTECNT[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x60 + 0x04*x (x=0 t0 3)	CSI_ TIMxCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COUNT[24:0]																								
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x70	CSI_LMCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DL1MAP[2:0]			Res.	DL0MAP[2:0]			Res.	Res.	Res.	Res.	Res.	LANENB[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value										0	1	0	0	0	0	1					0	0	1	0								
0x74	CSI_PRGITR	TIM3EN	TIM3EOF	TIM3VC[1:0]		TIM2EN	TIM2EOF	TIM2VC[1:0]		TIM1EN	TIM1EOF	TIM1VC[1:0]		TIM0EN	TIM0EOF	TIM0VC[1:0]		LB3EN	Res.	LB3VC[1:0]		LB2EN	Res.	LB2VC[1:0]		LB1EN	Res.	LB1VC[1:0]		LB0EN	Res.	LB0VC[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0		0	0	0		0	0	0		0	0
0x78	CSI_WDR	CNT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x7C	Reserved	Reserved																															
0x80	CSI_IER0	Res.	SYNCERRIE	WDERRIE	SPKTERRIE	IDERRIE	CECCERRIE	ECCERRIE	CRCERRIE	Res.	Res.	CCFIFOFIE	Res.	Res.	Res.	Res.	SPKTIE	EOF4IE	EOF3IE	EOF2IE	EOF1IE	SOF3IE	SOF2IE	SOF1IE	SOF0IE	TIM3IE	TIM2IE	TIM1IE	TIM0IE	LB3IE	LB2IE	LB1IE	LB0IE
	Reset value		0	0	0	0	0	0	0			0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 372. CSI-2 Host and PHY register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x84	CSI_IER1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECTRLDL1IE	ESYNCESCDL1IE	EESCDL1IE	ESOTSYNCDL1IE	ESOTDL1IE	Res.	Res.	Res.	ECTRLDL0IE	ESYNCESCDL0IE	EESCDL0IE	ESOTSYNCDL0IE	ESOTDL0IE
	Reset value																				0	0	0	0	0				0	0	0	0	0
0x88-0x8C	Reserved	Reserved																															
0x90	CSI_SR0	Res.	SYNCERRF	WDERRF	SPKTERRF	IDERRF	CECCERRF	ECCERRF	CRCERRF	Res.	Res.	CCFIFOFF	VC3STATEF	VC2STATEF	VC1STATEF	VC0STATEF	SPKTF	EOF3F	EOF2F	EOF1F	EOF0F	SOF3F	SOF2F	SOF1F	SOF0F	TIM3F	TIM2F	TIM1F	TIM0F	LB3F	LB2F	LB1F	LB0F
	Reset value		0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x94	CSI_SR1	ACTCLF	ULPNCLF	ULPNACTF	STOPCLF	Res.	ULPNDL1F	STOPDL1F	SKCALDL1F	SYNCDL1F	ACTDL1F	Res.	ULPNDL0F	STOPDL0F	SKCALDL0F	SYNCDL0F	ACTDL0F	Res.	Res.	Res.	ECTRLDL1F	ESYNCESCDL1F	EESCDL1F	ESOTSYNCDL1F	ESOTDL1F	Res.	Res.	Res.	ECTRLDL0F	ESYNCESCDL0F	EESCDL0F	ESOTSYNCDL0F	ESOTDL0F
	Reset value	0	0	0	0		0	0	0	0	0		0	0	0	0	0				0	0	0	0	0				0	0	0	0	0
0x98-0x9C	Reserved	Reserved																															
0x100	CSI_FCR0	Res.	CSYNCERRF	CWDERRF	CSPKTERRF	CIDERRF	CCECCRRF	CECCERRF	CCRCERRF	Res.	Res.	CCCFIFOFF	Res.	Res.	Res.	Res.	CSPKTF	CEOF3F	CEOF2F	CEOF1F	CEOF0F	CSOF3F	CSOF2F	CSOF1F	CSOF0F	CTIM3F	CTIM2F	CTIM1F	CTIM0F	CLB3F	CLB2F	CLB1F	CLB0F
	Reset value		0	0	0	0	0	0	0			0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x104	CSI_FCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CECTRLDL1F	CESYNCESCDLF	CEESCDL1F	CESOTSYNCDL1F	CESOTDL1F	Res.	Res.	Res.	CECTRLDL0F	CESYNCESCDL0F	CEESCDL0F	CESOTSYNCDL0F	CESOTDL0F
	Reset value																				0	0	0	0	0				0	0	0	0	0
0x108-0x10C	Reserved	Reserved																															
0x110	CSI_SPDFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VCHANNEL[1:0]		DATATYPE[5:0]						DATAFIELD[15:0]																
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x114	CSI_ERR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDVCERR[1:0]		IDDTERR[5:0]						CECCVCERR[1:0]		CECCDTERR[5:0]						CRCVCERR[1:0]		CRCDTERR[5:0]					
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 372. CSI-2 Host and PHY register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x118	CSI_ERR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYNCVCERR[1:0]		WDVCERR[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPKTVCERR[1:0]		SPKTDTERR[5:0]					
	Reset value													0	0	0	0									0	0	0	0	0	0	0	0
0x11C-0xFFC	Reserved	Reserved																															
0x1000	CSI_PRCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PEN	Res.
	Reset value																															0	
0x1004	CSI_PMCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TUEXDL0	Res.	Res.	Res.	TUESDL0	Res.	Res.	Res.	RTDL0	Res.	Res.	Res.	DTDL	Res.	FTXSMDL0	FRXMDL1	FRXMDL0
	Reset value																1				1				0				0		0	1	1
0x1008	CSI_PFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DLD	Res.	HSFR[6:0]							Res.	Res.	CCFR[5:0]					
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1
0x100C	Reserved	Reserved																															
0x1010	CSI_PTCR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRSEN	TCKEN
	Reset value																															0	1
0x1014	CSI_PTCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TWM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDI[7:0]							
	Reset value																0									0	0	0	0	0	0	0	0
0x1018	CSI_PTSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDO[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

Refer to *Section 2.3* for the register boundary addresses.

41 Parallel synchronous slave interface (PSSI)

The PSSI peripheral and the DCMI (digital camera interface) use the same circuitry. As a result, these two peripherals cannot be used at the same time: when using the PSSI, the DCMI registers cannot be accessed, and vice-versa.

In addition, the PSSI and the DCMI share the same alternate functions and interrupt vector (see *Section 41.3.2: PSSI pins and internal signals*).

41.1 PSSI introduction

The PSSI is a generic synchronous 8/16-bit parallel data input/output slave interface. It enables the transmitter to send a data valid signal that indicates when the data is valid, and the receiver to output a flow control signal that indicates when it is ready to sample the data.

41.2 PSSI main features

The PSSI peripheral main features are the following:

- Slave mode operation
- 8-bit or 16-bit parallel data input or output
- 8-word (32-byte) FIFO
- Data enable (PSSI_DE) and Ready (PSSI_RDY) alternate function

  When selected, these signals can either enable the transmitter to indicate when the data is valid, allow the receiver to indicate when it is ready to sample the data, or both.
- Clock out mode

41.3 PSSI functional description

The PSSI is a synchronous parallel slave interface that can send or receive high-speed data flows. It consists of up to 16 data lines (PSSI_D[15:0]) plus a clock line (PSSI_PDCK). The clock polarity can be configured so that data can be captured or transmitted on either the clock rising or falling edge.

Usually, a general-purpose DMA channel is used to pass 32-bit packed data via the data register (PSSI_DR).

The data flow can either be continuous or synchronized by hardware using the optional PSSI_DE (Data enable), and PSSI_RDY (Ready) signals.

Figure 435 shows the PSSI block diagram.

41.3.1 PSSI block diagram

Figure 435. PSSI block diagram

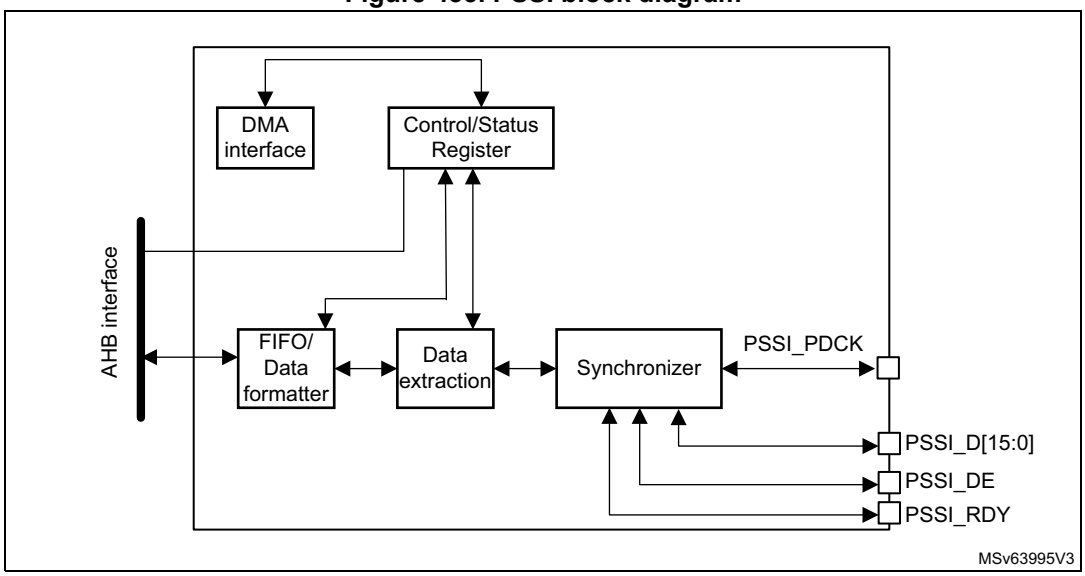
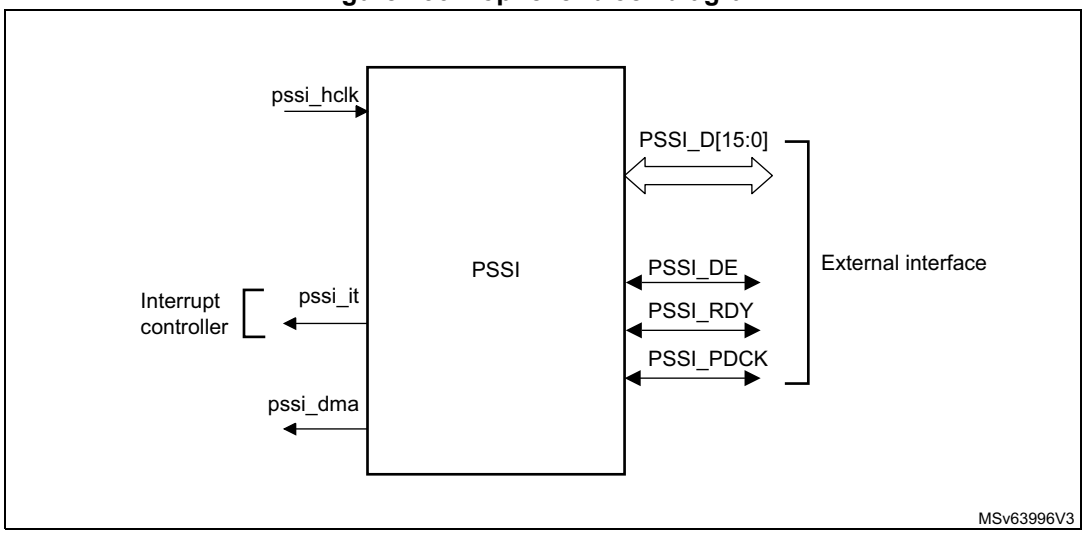


Figure 436. Top-level block diagram



41.3.2 PSSI pins and internal signals

The PSSI interface is composed of 19 pins, though nine signals are enough to transfer parallel data. *Table 373* shows the PSSI pins.

When the PSSI ENABLE bit (bit 14 of PSSI_CR) is set to 1, the alternate functions and the interrupt vector are associated with the PSSI. Otherwise, they are associated with the DCMI. The DCMI ENABLE bit (bit 15 of DCMI_CR) and the PSSI ENABLE bit (bit 14 of PSSI_CR) must not be set to 1 at the same time. As an example, if a GPIO is configured to use the alternate function PSSI_PDCK/DCMI_PIXCK, it is the PSSI_PDCK function which becomes active if PSSI_CR/ENABLE is set to 1.

Table 373. PSSI input/output pins

PSSI signal name	DCMI signal it is shared with	Signal type	Description
PSSI_PDCK	DCMI_PIXCK	Input/output	Parallel data clock input/output
PSSI_D[15:0]	DCMI_D[13:0]	Input/output	Data output when transmitting, data input when receiving
PSSI_DE	DCMI_HSYNC	Input/output	Data enable signal
PSSI_RDY	DCMI_VSYNC	Input/output	Ready signal

Table 374 shows the PSSI internal input/output signals.

Table 374. PSSI internal input/output signals

Internal signal name	Signal type	Description
pssi_it	Output	Interrupt
pssi_dma	Output	DMA request
pssi_hclk	Input	AHB clock

41.3.3 PSSI clock

The AHB clock frequency must be at least 2.5 times higher than the PSSI_PDCK frequency. At frequency ratios lower than 2.5, data might be corrupted or lost during transfers.

Data transfers are synchronous with PSSI_PDCK. The PSSI_PDCK polarity can be configured as follows, through CKPOL bit (bit 5 of PSSI_CR):

- When CKPOL = 0
  - Input pins are sampled on PSSI_PDCK falling edge
  - Output pins are driven on PSSI_PDCK rising edge
- When CKPOL = 1
  - Input pins are sampled on PSSI_PDCK rising edge
  - Output pins are driven on PSSI_PDCK falling edge

The PSSI_PDCK can be configured in input mode (default) or in output mode. When configured in output mode, the clock is generated by the device RCC.

41.3.4 PSSI data management

Data direction

The direction of data transfers is configured through the OUTEN control bit (bit 31 of PSSI_CR):

- When OUTEN is cleared to 0 (default setting), the PSSI operates in receive mode and the data is input on the data pins.
- When OUTEN is set to 1, the peripheral operates in transmit mode and the data is output on the data pins.

OUTEN can be modified only when the ENABLE bit is cleared to 0.

Data register and DMA

Data are transferred from/to the FIFO using the PSSI_DR data register:

- In receive mode, data must be read from the FIFO by reading PSSI_DR.
- In transmit mode, data must be written to the FIFO by writing into PSSI_DR.

Word (32-bit) accesses to PSSI_DR and half-word (16-bit) accesses to PSSI_DR[15:0] are permitted in all modes. Byte (8-bit) accesses to PSSI_DR[7:0] are permitted only when the PSSI is configured to transfer 8 bits at a time (EDM=00 in the PSSI_CR register).

To reduce the load on the CPU, it is recommended to use the DMA to transfer data from/to the PSSI FIFO. When it is used, the DMA must be configured to transfer data via the PSSI_DR register. Using 32-bit transfers optimizes bandwidth and reduces the bus load. However, 8-bit and 16-bit transfers are also permitted.

To use the DMA, set the PSSI DMA enable bit (DMAEN in PSSI_CR) to 1 (default setting). When DMAEN is set to 1, a DMA transfer is initiated when the FIFO is ready for a 32-bit transfer (four valid bytes in receive mode or four empty bytes in transmit mode). As a result, in receive mode, no DMA transfers are initiated if there are three bytes or fewer in the FIFO, even if the DMA is configured to perform 8-bit transfers.

The RTT4B and RTT1B status bits (PSSI_SR) are useful when the CPU directly perform transfers to and from the FIFO. RTT4B set to 1 indicates that the FIFO is ready to transfer four bytes: at least four valid bytes in the FIFO in receive mode or at least four free bytes in transmit mode. RTT1B set to 1 indicates that the FIFO is ready to transfer one byte: at least one valid byte in the FIFO in receive mode or at least one free byte in transmit mode.

8-bit data

The PSSI parallel interface can transfer either 8-bit (using D[7:0]) or 16-bit data (using D[15:0]) depending on the EDM[1:0] control bits (bits 11:10 of PSSI_CR). If the 8-bit configuration is selected (EDM[1:0] set to 00), the unused D[15:0] pins can be used for GPIO or other functions.

When EDM[1:0] in PSSI_CR are programmed to 00, the interface transfers 8 bits using the D[7:0] pins. In this case, D[15:8] are not used and four PSSI_PDCK cycles are required to transfer a 32-bit word.

The least-significant byte (bits 7:0) correspond to the first byte transferred, and the most-significant byte (bits 31:28) corresponds to the forth byte transferred. *Table 375* illustrates the positioning of the data bytes in two 32-bit words.

Table 375. Positioning of captured data bytes in 32-bit words (8-bit width)

Byte address	31:24	23:16	15:8	7:0
0	D_{n+3}[7:0]	D_{n+2}[7:0]	D_{n+1}[7:0]	D_n[7:0]
4	D_{n+7}[7:0]	D_{n+6}[7:0]	D_{n+5}[7:0]	D_{n+4}[7:0]

16-bit data

When EDM[1:0] in PSSI_CR are programmed to 11, the interface transfers 16 bits using the D[15:0] pins. In this case, two PSSI_PDCK cycles are required to transfer a 32-bit word.

The least-significant half word (bits 15:0) correspond to the first half word transferred, and the most-significant half-word (bits 31:16) corresponds to the second half word transferred. *Table 376* illustrates the positioning of the data in two 32-bit words.

Table 376. Positioning of captured data bytes in 32-bit words (16-bit width)

Byte address	31:16	15:0
0	D_{n+1}[15:0]	D_n[15:0]
4	D_{n+3}[15:0]	D_{n+2}[15:0]

FIFO data buffer and error conditions

An eight-word FIFO helps improving performance and avoids overruns and underruns.

If the ready signal (PSSI_RDY) is disabled in receive mode, an overrun error is generated when a clock active edge occurs when the FIFO is full. In this case, the input data is lost.

If the data enable signal (PSSI_DE) is disabled in transmit mode, an underrun error is generated when a clock active edge occurs when the FIFO is empty. In this case, unpredictable data are output.

The OVR_RIS status bit indicates that either an overrun or an underrun occurred. An interrupt can be generated when these events occur.

41.3.5 PSSI optional control signals

Data Enable (PSSI_DE) alternate function input

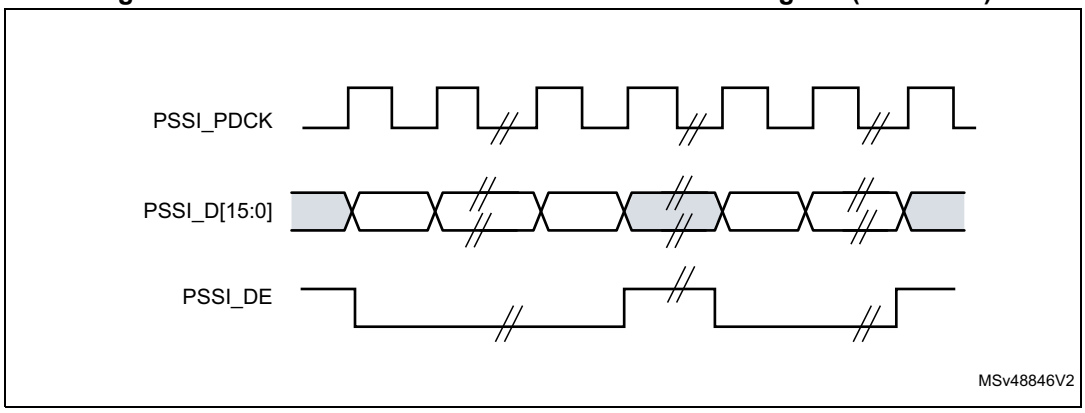
The data enable signal, PSSI_DE, is an optional signal. It is driven by the data source/transmitter in order to indicate that the data is valid to be transferred during the current cycle. When PSSI_DE is inactive, it means that the data must not be sampled by the receiver at the next clock edge.

This alternate function signal can be enabled using the DERDYCFG (bits 20:18 of PSSI_CR) control bits. PSSI_DE polarity is configured through DEPOL control bit (bit 6 of PSSI_CR). PSSI_DE is active low when DEPOL is cleared to 0, and high when DEPOL is et to 1.

The direction of the PSSI_DE signal is defined by the OUTEN value. It is the same as the data direction.

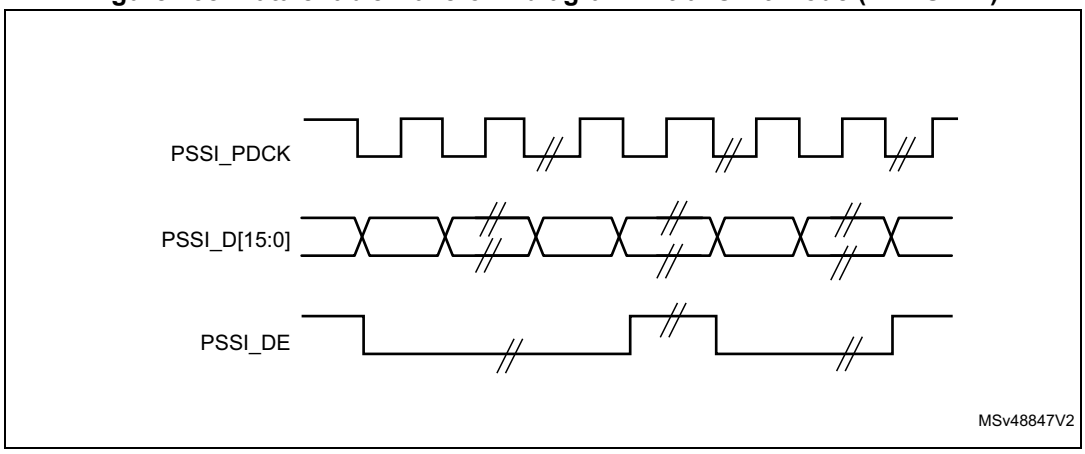
If the PSSI_DE alternate function input is enabled (through DERDYCFG) in receive mode (OUTEN cleared to 0), the PSSI samples PSSI_DE on the same PSSI_PDCK edge as the one used for sampling the data (D[15:0]). If PSSI_DE is active, the sampled data is saved in the FIFO. Otherwise, the sampled data is considered invalid and discarded. The transmitting device can use PSSI_DE as a data valid signal, driving it inactive when the data in the current cycle is not valid. This flow control function allows avoiding underrun errors.

Figure 437. Data enable in receive mode waveform diagram (CKPOL=0)



If the PSSI_DE alternate output function is enabled (through DERDYCFG) in transmit mode (OUTEN=1), the PSSI drives PSSI_DE on the same PSSI_PDCK edge that the one used to drive the data (D[15:0]). If a new 8 or 16-bit data (as programmed in the EDM[1:0] control bits in PSSI_CR) is available for transmission in the internal FIFO, this data is output on the data outputs (D[15:0]) and the PSSI_DE output becomes active on the current PSSI_PDCK edge. Otherwise (if the TX FIFO is empty), the D[15:0] outputs remains unchanged on the next clock edge and the PSSI_DE output becomes inactive.

Figure 438. Data enable waveform diagram in transmit mode (CKPOL=1)



Ready (PSSI_RDY) alternate function output

The ready signal, PSSI_RDY, is an optional signal. It is driven by the receiving device and indicates whether data is being accepted in the current cycle. When PSSI_RDY is inactive, it means that the data must not be sampled by the receiver at the next clock edge.

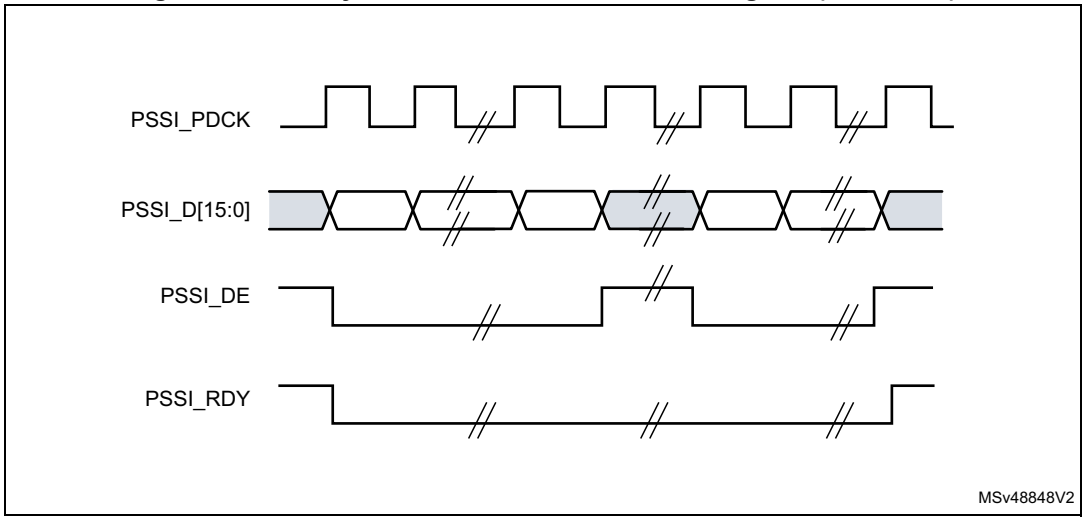
This alternate function signal can be enabled using the DERDYCFG control bits (bits 20:18 of PSSI_CR). PSSI_RDY polarity is configured through the RDYPOL control bit (bit 6 of PSSI_CR). PSSI_RDY is active low when RDYPOL is cleared to 0, and high when RDYPOL set to 1.

The direction of the PSSI_RDY signal is defined by the OUTEN (bit 31 of PSSI_CR). It is set in the opposite direction compared to the PSSI_DE and data signals.

If the PSSI_RDY alternate output function is enabled (through DERDYCFG) in receive mode (OUTEN=0), the PSSI drives PSSI_RDY one PSSI_PDCK half cycle after it samples

the data (D[15:0]). If the FIFO has enough free space to receive more data, the PSSI drives the PSSI_RDY signal active. Otherwise, if the FIFO is full and cannot accept more data, the PSSI drives the PSSI_RDY signal inactive. The transmitting device must repeat the current data in the next cycle when it detects that PSSI_RDY is inactive. This flow control function allows the PSSI to avoid overrun errors when the system (via the DMA) is unable to keep up with the data flow.

Figure 439. Ready in receive mode waveform diagram (CKPOL=0)



If the PSSI_RDY alternate input function is enabled (through DERDYCFG) in transmit mode (OUTEN=1), the PSSI samples the PSSI_RDY signal on the opposite PSSI_PDCK edge to the one at which D[15:0] are driven. If the PSSI_RDY signal is inactive, the PSSI keeps the same data (D[15:0]) and PSSI_DE signals that valid data are available during the next PSSI_PDCK clock cycle. Otherwise, if PSSI_RDY signal is sampled as active, the next data from the TX FIFO (if available) is output on the data outputs (D[15:0]). If no new data are available in the TX FIFO, the PSSI keeps the data output values and outputs the PSSI_DE signal as inactive (if enabled).

The receiving device uses the PSSI_RDY to control the data flow and avoid overrun errors when the system (via the DMA) is unable to keep up with the data flow.

Bidirectional PSSI_DE/PSSI_RDY signal

A single pin can be used for both data enable (PSSI_DE) and ready (PSSI_RDY) functions if DEPOL and RDYPOL are both set to 1 and DERDYCFG is set to 111 or 100 in the PSSI_CR register. In this case, the GPIO corresponding to selected alternate function (PSSI_DE when DERDYCFG=111 or PSSI_RDY when DERDYCFG=100) must be configured as open-drain. The other device must also be configured to drive the line as open-drain, and a weak pull-up must be applied to the line.

The signal thus becomes bidirectional. If either the sender drives the line low (to indicate that the data is not valid) or the receiver drives the line low (to indicate that it is not sampling the current data), then both devices know that the data is not being transferred in the current cycle.

Figure 440. Bidirectional PSSI_DE/PSSI_RDY waveform

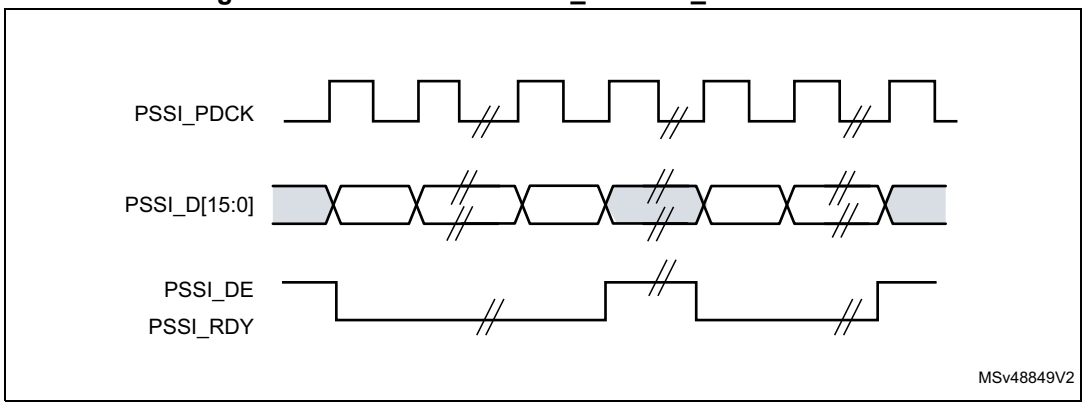
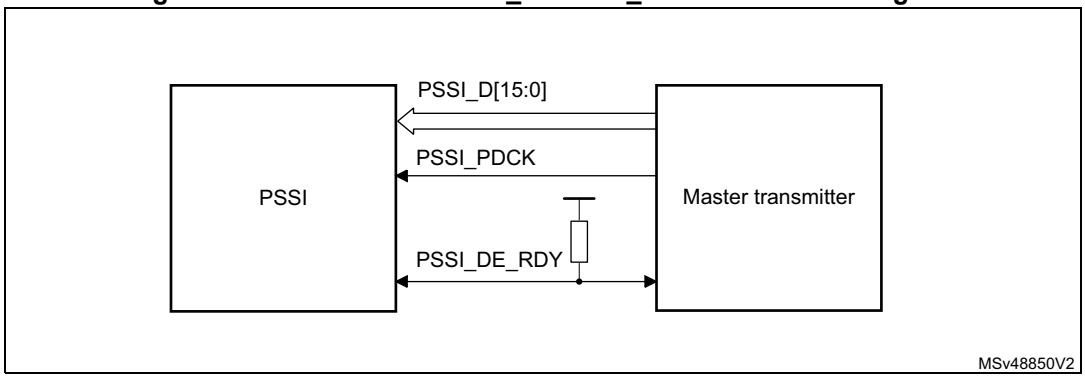


Figure 441. Bidirectional PSSI_DE/PSSI_RDY connection diagram



41.4 PSSI interrupts

The PSSI generates only one interrupt (IT_OVR). It is consequently equivalent to the global interrupt (pssi_it). Refer to *Table 377* for the list of interrupts.

The PSSI and the DCMI share the same interrupt vector. When the PSSI ENABLE bit (bit 14 of PSSI_CR) is set to 1, these interrupts are triggered by the PSSI. Otherwise, they are controlled by the DCMI.

The DCMI ENABLE bit (bit 14 of DCMI_CR) and PSSI ENABLE bit must not be set to 1 at the same time.

Table 377. PSSI interrupt requests

Interrupt acronym	Shared with DCMI	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from low-power mode
IT_OVR	IT_OVR	indicates overrun in receive mode or underrun in transmit mode	OVR_RIS	OVR_IE	OVR_ISC	NA

41.5 PSSI registers

An 8-bit write or a 16-bit write operation to any PSSI register besides PSSI_DR, results in a bus error. 32-bit read and write operations are permitted.

41.5.1 PSSI control register (PSSI_CR)

Address offset: 0x00

Reset value: 0x4000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OUTEN	DMAEN	CKSRC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DERDYCFG[2:0]			Res.	Res.
rw	rw	rw									rw	rw	rw		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	ENABLE	Res.	Res.	EDM[1:0]		Res.	RDYPOL	Res.	DEPOL	CKPOL	Res.	Res.	Res.	Res.	Res.
	rw			rw	rw		rw		rw	rw					

Bit 31 **OUTEN:** Data direction selection bit
0: Receive mode: data is input synchronously with PSSI_PDCK
1: Transmit mode: data is output synchronously with PSSI_PDCK

Bit 30 **DMAEN:** DMA enable bit
0: DMA transfers are disabled. The user application can directly access the PSSI_DR register when DMA transfers are disabled.
1: DMA transfers are enabled (default configuration). A DMA channel in the general-purpose DMA controller must be configured to perform transfers from/to PSSI_DR.

Bit 29 **CKSRC**: Clock source
This bit configures the clock source of the PSSI_PDCK.
0: External clock (PSSI_PDCK in input)
1: Internal clock (PSSI_PDCK in output)

Bits 28:21 Reserved, must be kept at reset value.

Bits 20:18 **DERDYCFG[2:0]:** Data enable and ready configuration
000: PSSI_DE and PSSI_RDY both disabled
001: Only PSSI_RDY enabled
010: Only PSSI_DE enabled
011: Both PSSI_RDY and PSSI_DE alternate functions enabled
100: Both PSSI_RDY and PSSI_DE features enabled - bidirectional on PSSI_RDY pin (see *Bidirectional PSSI_DE/PSSI_RDY signal on page 2080*)
101: Only PSSI_RDY function enabled, but mapped to PSSI_DE pin
110: Only PSSI_DE function enabled, but mapped to PSSI_RDY pin
111: Both PSSI_RDY and PSSI_DE features enabled - bidirectional on PSSI_DE pin (see *Bidirectional PSSI_DE/PSSI_RDY signal on page 2080*)
When the PSSI_RDY function is mapped to the PSSI_DE pin (settings 101 or 111), it is still the RDYPOL bit which determines its polarity. Similarly, when the PSSI_DE function is mapped to the PSSI_RDY pin (settings 110 or 111), it is still the DEPOL bit which determines its polarity.

Bits 17:15 Reserved, must be kept at reset value.

Bit 14 **ENABLE:** PSSI enable

0: PSSI disabled

1: PSSI enabled

The contents of the FIFO are flushed when ENABLE is cleared to 0.

Note: When ENABLE=1, the content of PSSI_CR must not be changed, except for the ENABLE bit itself. All configuration bits can change as soon as ENABLE changes from 0 to 1.

The DMA controller and all PSSI configuration registers must be programmed correctly before setting the ENABLE bit to 1.

The ENABLE bit and the DCMI ENABLE bit (bit 15 of DCMI_CR) must not be set to 1 at the same time.

Bits 13:12 Reserved, must be kept at reset value.

Bits 11:10 **EDM[1:0]:** Extended data mode

00: Interface captures 8-bit data on every parallel data clock

01: Reserved, must not be selected

10: Reserved, must not be selected

11: The interface captures 16-bit data on every parallel data clock

Bit 9 Reserved, must be kept at reset value.

Bit 8 **RDYPOL:** Ready (PSSI_RDY) polarity

This bit indicates the level on the PSSI_RDY pin when the data are not valid on the parallel interface.

0: PSSI_RDY active low (0 indicates that the receiver is ready to receive)

1: PSSI_RDY active high (1 indicates that the receiver is ready to receive)

Bit 7 Reserved, must be kept at reset value.

Bit 6 **DEPOL:** Data enable (PSSI_DE) polarity

This bit indicates the level on the PSSI_DE pin when the data are not valid on the parallel interface.

0: PSSI_DE active low (0 indicates that data is valid)

1: PSSI_DE active high (1 indicates that data is valid)

Bit 5 **CKPOL:** Parallel data clock polarity

This bit configures the capture edge of the parallel clock or the edge used for driving outputs, depending on OUTEN.

0: Falling edge active for inputs or rising edge active for outputs

1: Rising edge active for inputs or falling edge active for outputs.

Bits 4:0 Reserved, must be kept at reset value.

41.5.2 PSSI status register (PSSI_SR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTT1B	RTT4B	Res.	Res.
												r	r		

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **RTT1B:** FIFO is ready to transfer one byte

1: FIFO is ready for a one byte transfer. In receive mode, this means that at least one valid data byte is in the FIFO. In transmit mode, this means that there is at least one byte free in the FIFO.

0: FIFO is not ready for a 1-byte transfer

Bit 2 **RTT4B:** FIFO is ready to transfer four bytes

1: FIFO is ready for a four-byte (32-bit) transfer. In receive mode, this means that at least four valid data bytes are in the FIFO. In transmit mode, this means that there are at least four bytes free in the FIFO.

0: FIFO is not ready for a four-byte transfer

Bits 1:0 Reserved, must be kept at reset value.

41.5.3 PSSI raw interrupt status register (PSSI_RIS)

Address offset: 0x08

Reset value: 0x0000 0000

PSSI_RIS gives the raw interrupt status. This register is read-only. When read, it returns the status of the corresponding interrupt before masking with the PSSI_IER register value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVR_RIS	Res.
														r	

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **OVR_RIS:** Data buffer overrun/underrun raw interrupt status

0: No overrun/underrun occurred

1: An overrun/underrun occurred: overrun in receive mode, underrun in transmit mode. This bit is cleared by writing a 1 to the OVR_ISC bit in PSSI_ICR.

Bit 0 Reserved, must be kept at reset value.

41.5.4 PSSI interrupt enable register (PSSI_IER)

Address offset: 0x0C

Reset value: 0x0000 0000

The PSSI_IER register is used to enable interrupts. When one of the PSSI_IER bits is set, the corresponding interrupt is enabled. This register is accessible both in read and write modes.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVR_IE	Res.
														rw	

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **OVR_IE:** Data buffer overrun/underrun interrupt enable
0: No interrupt generation
1: An interrupt is generated if either an overrun or an underrun error occurred.

Bit 0 Reserved, must be kept at reset value.

41.5.5 PSSI masked interrupt status register (PSSI_MIS)

This PSSI_MIS register is read-only. When read, it returns the current masked status value of the corresponding interrupt (depending on the value in PSSI_IER). A bit in this register is set if the corresponding enable bit in PSSI_IER is set and the corresponding bit in PSSI_RIS is set.

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVR_MIS	Res.
														r	

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **OVR_MIS:** Data buffer overrun/underrun masked interrupt status
This bit is set to 1 only when PSSI_IER/OVR_IE and PSSI_RIS/OVR_RIS are both set to 1.
0: No interrupt is generated when an overrun/underrun error occurs
1: An interrupt is generated if there is either an overrun or an underrun error and the OVR_IE bit is set in PSSI_IER.

Bit 0 Reserved, must be kept at reset value.

41.5.6 PSSI interrupt clear register (PSSI_ICR)

Address offset: 0x14

Reset value: 0x0000 0000

The PSSI_ICR register is write-only. Writing a 1 into a bit of this register clears the corresponding bit in the PSSI_RIS and PSSI_MIS registers. Writing a 0 has no effect. Reading this register always gives zeros.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVR_ISC	Res.
														w	

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **OVR_ISC:** Data buffer overrun/underrun interrupt status clear
Writing this bit to 1 clears the OVR_RIS bit in PSSI_RIS.

Bit 0 Reserved, must be kept at reset value.

41.5.7 PSSI data register (PSSI_DR)

Address offset: 0x28

Reset value: 0x0000 0000

In receive mode (OUTEN = 0), the DMA controller must read the received data from this register. Write operations to PSSI_DR result in an error response. When more bytes than the number of valid bytes are read in the FIFO, the invalid bytes return zeros.

In transmit mode (OUTEN = 1), the DMA controller must write the data to be transmitted into this register. Read operations to PSSI_DR result in an error response.

32-bit, 16-bit, and 8-bit accesses are all supported for PSSI_DR. For instance, 16-bit read/write operations remove/add two bytes from/to the FIFO. However, 8-bit accesses are permitted only when the PSSI is configured to transfer 8 data bits at a time (EDM=00 in PSSI_CR). 8-bit accesses to PSSI_DR when EDM is not set to 0 result in an error response.

All accesses must include byte 0: 8-bit accesses must be performed to bits 7 to 0 and 16-bit accesses from bits 15 to 0. Accesses that do not include byte 0 results in an error response.

Accessing PSSI_DR when ENABLE bit in PSSI_CR is set to 0 results in an error response.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BYTE3[7:0]								BYTE2[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BYTE1[7:0]								BYTE0[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **BYTE3[7:0]**: Data byte 3

Bits 23:16 **BYTE2[7:0]**: Data byte 2

Bits 15:8 **BYTE1[7:0]**: Data byte 1

Bits 7:0 **BYTE0[7:0]**: Data byte 0

41.5.8 PSSI register map

Table 378. PSSI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	PSSI_CR	OUTEN	DMAEN	CKSRC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DERDYCFG			Res.	Res.	Res.	ENABLE[2:0]			EDM		Res.	RDYPOL	Res.	DEPOL	CKPOL	Res.	Res.	Res.	Res.	Res.
	Reset value	0	1	0									0	0	0				0			0	0		0		0	0					
0x04	PSSI_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RTT1B	RTT4B	Res.	Res.
	Reset value																													0	0		
0x08	PSSI_RIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVR_RIS	Res.
	Reset value																															0	
0x0C	PSSI_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.am	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVR_IE	Res.
	Reset value																															0	
0x10	PSSI_MIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVR_MIS	Res.
	Reset value																															0	
0x14	PSSI_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVR_ISC	Res.
	Reset value																															0	
0x18-0x24	Reserved	Res.																															
0x28	PSSI_DR	BYTE3[7:0]								BYTE2[7:0]								BYTE1[7:0]								BYTE0[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

42 Display subsystem

42.1 Introduction

The display subsystem is targeted to drive up to 1080p60 display panels, through a parallel interface. It supports some on-the-fly compositions to offload the GPU and optimize the use of the system bandwidth: up to two layers, color conversion, blend, mirror, and a final YUV conversion.

The display subsystem can display a secure layer with data that cannot be read by nonsecure application, and with the display guaranteed stable.

42.2 Display subsystem main features

The display subsystem is built around LTDC:

- LTDC: handles display composition
  - Composition: two layers
  - Input pixel format: flexible format, including YUV420 full-planar on layer L1
  - Secure layer: protected access to buffer and configuration registers.
  - Mirror: horizontal and vertical
  - Miscellaneous: color lookup-table, color keying, Gamma
  - 1080p60 max performance
- LTDC parallel interface (integrated inside the LTDC):
  - Standard: 24 bpp + Hs, Vs, De (DataEnable) synchronization signals with ReTime
  - Output rate: 150 Mpixel/s
  - Resolution: 1920x1080 at 60 fps max, with HDMI blankings
  - Pixel formats: RGB888, 666, 565, YUV422-16 bits/BT601/709

42.3 Display subsystem implementation

The implementation is described in *Section 43: LCD-TFT display controller (LTDC)*.

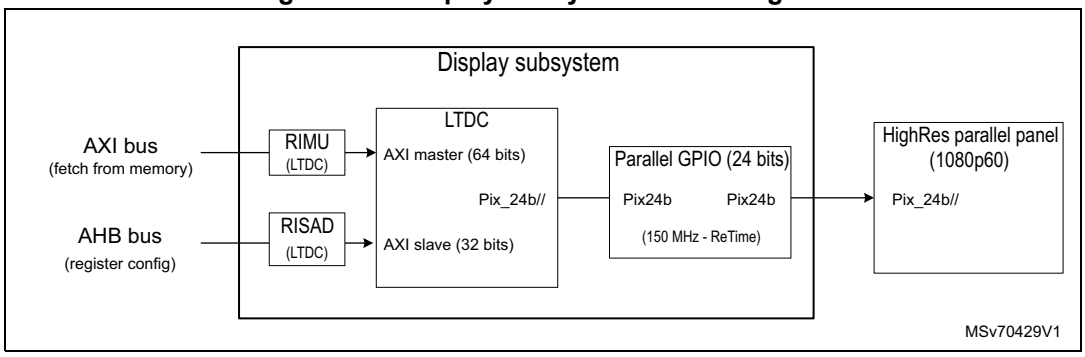
42.4 Display subsystem functional description

42.4.1 Display subsystem block diagram

Figure 442 shows the functional blocks in the display subsystem, with their instance/design names:

- LTDC: composition engine, includes a parallel interface
- Parallel GPIO: Parallel I/O-ring of the device with up to 24 bpp + H/V synchronization
- RIMU/RISAD: RIF security logic, configured for the specific display needs

Figure 442. Display subsystem block diagram

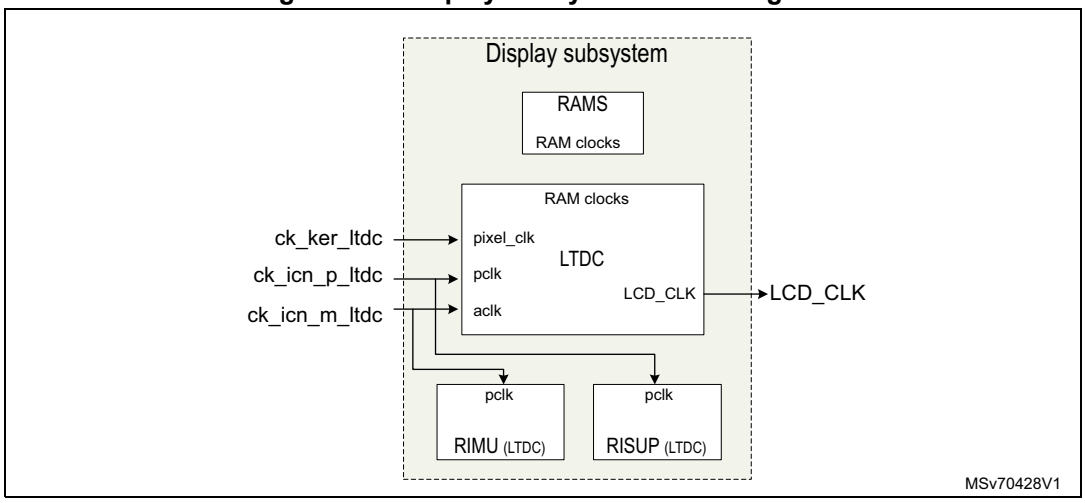


42.4.2 Display subsystem pins and external signal interface

The details of the pins and external signal interface of the display subsystem can be found in the respective functional block sections in this reference manual.

42.4.3 Display subsystem clocks

Figure 443. Display subsystem clock diagram



The display subsystem is implemented with the following clock max frequencies:

- LTDC pixel clock (pixel_clk): 150 MHz
- LTDC parallel interface (lcd_clk): 150 MHz (I/O ReTime mandatory)
- Register clocks (ck_icn_p_*): 200 MHz (all APB/AHB register files)
- AXI-Master clock (ck_icn_m_ltdc): 400 MHz

42.4.4 Display subsystem security

This section is an application of RIF security architecture (see *Section 3.5.3: RIF infrastructure*). The security of the display subsystem is based on the protection of its register map and its master access to the system interconnect, based on default RISUP and RIMU respectively.

The RISUP differentiates the access right of accesses performed toward the following RIF protected peripheral ID (see *Table 20: RISUP indexes*):

- LTDC common: LTDC common registers, about panel info, synchronization, interface
- LTDC_L1 (layer 1): for the window of a default application
- LTDC_L2 (layer 2): LTDC layer 2, for the window of a potentially secure application

The RIMU (RIMU_AXI_LTDC) differentiates the bus transactions emitted by the following AXI masters:

- RIMU_L1: read access only for layer 1, always non-protected
- RIMU_L2: read access only for layer 2, potentially protected

Security configuration of devices with never-secure display content

The security setup of devices that are never required to display any secure content is detailed below (the display subsystem works by default in a non-protected mode):

- The display subsystem configuration space is set as non-protected.

  The default operating system (OS) software has a full access and can configure/handle all peripherals of the display subsystem.
- The layer 2 (which has a hardware secure capability) can be reused as a default layer, like layer 1, and can display the content of any application.

Note: Layer 2 is not fully featured, and lacks the resize and multi-planar capabilities of layer 1.

- The LTDC AXI-master emits by default non-protected accesses.

  The AXI CID = DISP S and P can be associated to the two AXI clients, via the RIMU, to tag that the AXI transaction contains pixels.

Security configuration of devices with a temporary secure content

For devices that temporarily display a secure content, it is expected that the secure content is displayed during short periods.

The working principle of the display subsystem is to segregate configuration spaces:

- The secure layer 2 is always protected.
- Never-secure layer 1 is always non-protected.
- The intermediate common features (such as panel data, synchronization) are usually non-protected and handled by the OS (when no secure content is displayed), but temporarily protected (with no OS access) during the short period when a secure content is displayed via layer 2. Removing the access of the non-protected OS guarantees that a malware does not impact the stability of the displayed protected content.
- A few minor items get specifically protected or open, as detailed below.

The configuration of the display subsystem for a secure content is the following:

- Layer 2 is set permanently protected:
  - RIF LTDC_L2 RISUP peripheral ID must be set as protected (S and/or P flags).
  - RIF LTDC_L2 RIMU AXI-Master emits protected transactions (tagged as S a/o P), which allow it to retrieve the protected layer 2 pixels from their protected memory

buffer. The RIMU must be updated by a trusted domain application using RIFSC_RIMC_ATTR11.

The recommendation is to use a secure, privileged (optional), and CID = 0x4 setup. With this secure setup, the application must make sure that CID4 has access to the secure pixels area.

  - Only a secure software is able to configure the layer 2 parameters (such as width, height, buffer address), so that the characteristics of the layer 2 are guaranteed.
  - The interrupt bank B is associated automatically with the layer 2, and gets permanently protected and devoted to the secure software handling layer 2. The secure software must thus handle and be driven by that interrupt bank B.
  - The LTDC features that must allow data-evasion get also protected (automatically, as associated with the layer 2 rights), with an access allowed only to the secure software. These features are the CRC (which allows the displayed frames to be signed mainly for test purposes), the blend-order (which is used to hide the protected content below some malware layer faking the protected content).
- Layer 1 is set permanently as non-protected:
  - RIF LTDC_L1 RISUP peripheral ID is set as not protected.
  - RIF LTDC_L1 RIMU AXI-master keeps its initial setup in RIFSC_RIMC_ATTR10, and emits only non-protected transactions which is enough to retrieve pixels from any application buffer.
  - The default OS drivers handle the layer 1 parameters (such as width, height, buffer address).
  - The interrupt bank A is associated automatically with layer 1, and keeps an non-protected open access. The OS must then handle and be driven by that Interrupt bank A to globally drive the display.
- Common registers must be set at boot as non-protected, temporarily:
  - RIF LTDC common non-layer peripheral indexes is concerned.
  - This allows the default OS to boot the display subsystem, and to handle it.
- Common registers must be set as protected when displaying a protected content:
  - The RIF LTDC common must be set protected before starting to display a protected content, and be unprotected after the protected content has finished to be updated. The LTDC common configuration spaces keep a permanent read access for nonsecure or unprivileged software.
  - The protection restricts the write accesses only: it removes the access of the non-protected OS, and so guarantees that a malware does not impact the stability of the displayed protected content. The LTDC common configuration spaces keep a permanent read-access for nonsecure or unprivileged software.
  - The protection does not restrict the read accesses: the OS is usually unaware of the status of the protected layer 2 that is handled by a different, secure, software. The read access capability for the OS allow the display drivers to remain. somehow stable, although the write access of the display drivers is not performed.

Table 379 lists vertically all the functional blocks of the display subsystem, and horizontally lists the associated RIF peripheral ID.

Note: Table color legend for recommended settings:

- green for a permanently non-protected setting

- blue for a temporarily protected versus non-protected

- purple for a permanently protected state

Table 379. Display subsystem RIF peripheral ID

All in display subsystem					RISUP-slave		
					102	103	104
Blocks	Address range	(LTDC) address x1	Start address x4	End address x4	LTDC common	LTDC_L1	LTDC_L2
LTDC	Common CFG0	0x0	0x0	0x30	Y	-	-
	Interrupt-A	0xD	0x34	0x3C	-	Y	-
	Common CFG1	0x10	0x40	0x60	Y	-	-
	Interrupt-B	0x19	0x64	0x70	-	-	Y
	(empty)	0x1D	0x74	0x74	Y	-	-
	CRC	0x1E	0x78	0x7C	-	-	Y
	Common CFG2	0x23	0x8C	0xFC	Y	-	-
	L1-CFG0 (typ: video)	0x40	0x100	0x100	-	Y	-
	L1-CTRL	0x41	0x104	0x104	-	Y	-
	L1-CFG1	0x42	0x108	0x1FC	-	Y	-
	L2-CFG0 (typ: UI)	0x80	0x200	0x200	-	-	Y
	L2-CTRL	0x81	0x204	0x204	-	-	Y
	L2-CFG1	0x82	0x208	0x2FC	-	-	Y

42.5 Display subsystem programmable parameters

Steps to configure the various main use cases supported by the display subsystem are detailed below. They provide the global picture and settings of the display subsystem (not repeating information available in the local sections).

The display subsystem drives a display panel or bridge through the parallel interface. It allows panels to be driven up to 1080p60 (150 Mpixel/s), with either RGB888 or YUV422.

The RCC provides a pixel clock that drives both the LTDC and its synchronous parallel interface. Aside, the LTDC needs an AXI-Master clock to fetch pixels, and an APB-Slave clock to get configured.

The steps to follow are detailed below:

1. Setup security settings, with or without a secure layer depending on application needs, as described in *Figure 442*.
2. Clock and reset:
   - ck_ker_ltdc clock, provided by the RCC, as output pixel clock.

     One PLL of the RCC needs to be allocated to the display subsystem to provide the clock that is exactly at the wanted pixel rate for the output display. For instance, a standard 1080p60 requires a 148,50 MHz clock.
   - LTDC_CLK is driven automatically by the LTDC (after some gating) from its above provided pixel_clock (ck_ker_ltdc from the RCC).
   - ck_icn_p_ltdc is provided by the RCC to allow the LTDC configuration (default 200 MHz).
   - ck_icn_m_ltdc is provided by the RCC to allow the LTDC AXI to work (default 400 MHz).

   See *Section 14: Reset and clock control (RCC)*for more details.
3. LTDC configuration:
   - Setup the LTDC (such as input layers, display information, parallel interface, AXI settings) as described in the LTDC section.
4. Signal polarity:

   The polarities of the signals are impacted by the fact that the LTDC parallel interface signals are reclocked in the IO-ReTime logic, downstream of the LTDC. It results the following when configuring the polarities to match the panel on the parallel interface:
   - The clock polarity (rising-vs-falling edge) must be configured in the IO-ReTime only (see *Section 15: General-purpose I/Os (GPIO)*), and not in the LTDC output logic. The clock polarity logic in the LTDC must remain in its default post-reset state.
   - The polarity of the Vs, Hs, and De synchronization signals, unlike for the clock polarity, can be configured directly in the LDTC output stage (see *Section 43: LCD-TFT display controller (LTDC)*).
5. I/O configuration:

   Allocate an alternate GPIO bank to the display subsystem, with usually 28 pins needed: 24 pins for RGB888 and four others for the ancillary: Hs, Vs, De, and clock.

42.6 Display subsystem interrupts

The interrupts are described in sections of peripherals in this reference manual.

42.7 Display subsystem registers

All registers are described in *Section 43: LCD-TFT display controller (LTDC)*.

43 LCD-TFT display controller (LTDC)

43.1 LTDC introduction

The LTDC (liquid crystal display - thin film transistor) display controller provides a parallel digital RGB (Red, Green, Blue) and signals for horizontal, vertical synchronization, pixel clock and data enable as output to interface directly to a variety of LCD-TFT panels.

43.2 LTDC main features

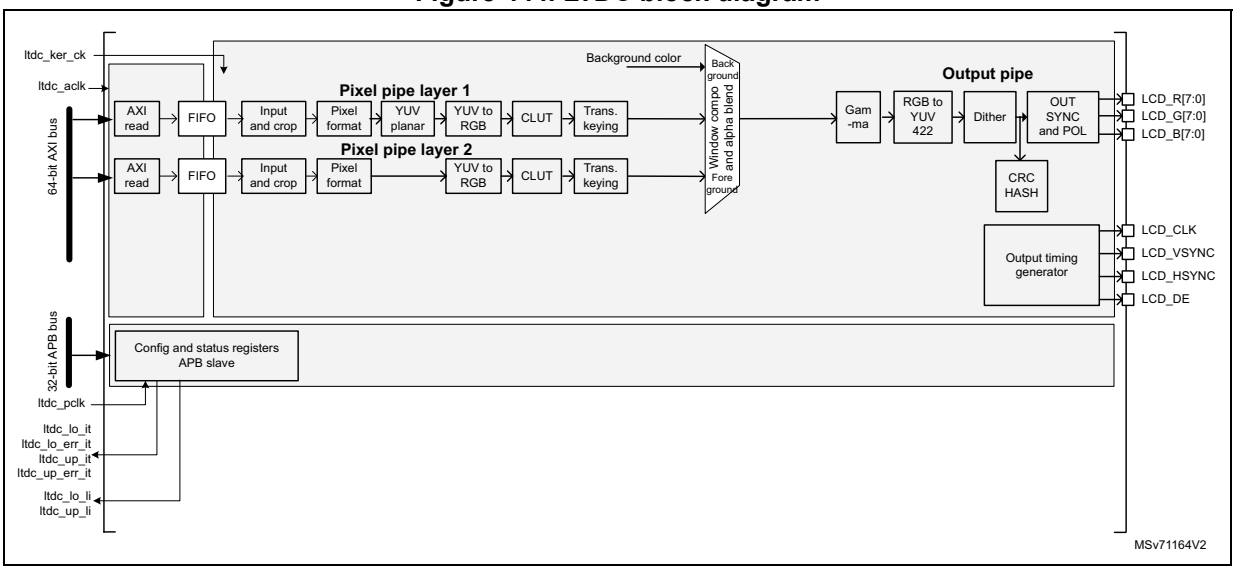
- 2 input layers blended together to compose the display
- Cropping of layers from any input size and location
- Multiple input pixel formats:
  - Predefined ARGB formats: ARGB8888, ABGR8888, RGBA8888, BGRA8888, RGB565, BGR565, and RGB888packed
  - Flexible ARGB, allowing any width and location for A,R,G, and B components
  - Predefined YUV formats: YUV422-1L (FourCC: YUYV, Interleaved), YUV420-2L (FourCC: NV12, semi planar), YUV420-3L (FourCC: Yxx I420, full planar) with some flexibility on the component sequence
- Color look-up table (CLUT) up to 256 colors (256 x 24 bits) per layer
- Color transparency keying
- Composition with flexible window position and size versus output display
- Blending with flexible layer order and alpha value (per pixel or constant)
- Background underlying color
- Gamma with non-linear configurable table
- Dithering for output with less bits per component (pseudo-random on 2 bits)
- Polarity inversion for HSync, VSync, and DataEnable outputs
- Output as RGB888 24 bpp or YUV422 16 bpp
- Secure layer (using layer2) capability, with grouped regs and additional interrupt set
- Interrupts based on 7 different events
- AXI master interface with long efficient bursts (64 or 128 bytes)

Note: The above mentioned FourCC pixel formats are described at www.fourcc.org.

43.3 LTDC functional description

43.3.1 LTDC block diagram

Figure 444. LTDC block diagram



43.3.2 LTDC pins and internal signals

Table 380. LTDC pins

Pin name	Pin type	Description
LCD_CLK	Output	Clock output
LCD_HSYNC		Horizontal synchronization
LCD_VSYNC		Vertical synchronization
LCD_DE		Not data enable
LCD_R[7:0]		8-bit Red data (or U/V component, if output is YUV422)
LCD_G[7:0]		8-bit Green data (or Y component if output is YUV422)
LCD_B[7:0]		8-bit Blue data

The LTDC pins must be configured by the user application. The unused pins can be used for other purposes.

Using less pins than the default 24 bpp

The LTDC outputs the pixels on 24 pins by default (RGB888). The user can however connect the display panel using less wires, for instance 16 wires (if RGB565) or 18 wires (RGB666). If so, R (resp. G, B) wires must be connected to the MSB of the LTDC R (resp. G, B) output pins (see *Section 43.3.2*).

Example: LTDC interfacing with a RGB565 16-bit display panel. Display panel R[4:0], G[5:0] and B[4:0] wires and input pins must be connected to LTDC_R[7:3], LCD_G[7:2], and LCD_B[7:3] output pins.

The following table shows how the LTDC is connected.

Table 381. LTDC trigger interconnections

Trigger name	Direction	Trigger source/destination
ltdc_lo_li	Output	hpdma1_trigsel[20] gpdma1_trigsel[20]
ltdc_up_li	Output	hpdma1_trigsel[110] gpdma1_trigsel[110]

43.3.3 LTDC reset and clocks

The LTDC uses the following clock domains:

- **AXI clock domain**

  This domain contains the LTDC AXI master interface for data transfer from the memories to the layer FIFO and the frame buffer configuration register.
- **APB clock domain**

  This domain contains the global configuration registers and the interrupt register.
- **Pixel clock domain**

  This domain contains the pixel data generation, the layer configuration register as well as the LTDC interface signal generator. The LCD_CLK output must be configured following the panel requirements.

Note: For normal operation, all clocks must be provided to allow clock domain crossing for read and write operations.

The following table lists the clock domains used to store the register values.

Table 382. Clock domain for each register

LTDC register	Clock domain
LTDC_SSCR	Pixel clock
LTDC_BPCR	AXI + pixel clock
LTDC_AWCR	
LTDC_TWCR	
LTDC_GCR	
LTDC_SRCR	APB clock
LTDC_GCCR	Pixel clock
LTDC_BCCR	
LTDC_IER	APB clock
LTDC_ISR	
LTDC_ICR	

Table 382. Clock domain for each register (continued)

LTDC register	Clock domain
LTDC_LIPCR	Pixel clock
LTDC_CPSR	
LTDC_CDSR	
LTDC_EDCR	
LTDC_IER2	APB clock
LTDC_ISR2	
LTDC_ICR2	
LTDC_LIPCR2	Pixel clock
LTDC_ECRCR	
LTDC_CCRCR	
LTDC_FUTR	Pixel clock
LTDC_LxC0R	APB clock
LTDC_LxC1R	
LTDC_LxRCR	
LTDC_LxCR	AXI + pixel clock
LTDC_LxWHPCR	Pixel clock
LTDC_LxWVPCR	
LTDC_LxCKCR	
LTDC_LxPFCR	
LTDC_LxCACR	
LTDC_LxDCCR	
LTDC_LxBFCR	
LTDC_LxBLCR	AXI clock
LTDC_LxPCR	AXI + pixel clock
LTDC_LxCFBAR	
LTDC_LxCFBLR	
LTDC_LxCFBLNR	
LTDC_L1AFBA0R	
LTDC_L1AFBA1R	
LTDC_L1AFBLR	
LTDC_L1AFBLNR	

Table 382. Clock domain for each register (continued)

LTDC register	Clock domain
LTDC_LxCLUTWR	Pixel clock
LTDC_LxCYR0R	
LTDC_LxCYR1R	
LTDC_LxFPF0R	
LTDC_LxFPR1R	

While accessing LTDC registers, the APB bus is stalled during the access for a given time period detailed in the following table.

Table 383. LTDC register access and updated durations

Access type	AXI register clock domain	APB register clock domain	Pixel register clock domain
Read	7 x ltdc_pclk + 5 x ltdc_aclk	7 x ltdc_pclk	7 x ltdc_pclk + 5 x ltdc_ker_clk
Write	6 x ltdc_pclk + 5 x ltdc_aclk	6 x ltdc_pclk	6 x ltdc_pclk + 5 x ltdc_ker_clk

The LTDC can be reset by setting the corresponding bit in the RCC, that resets the three clock domains.

43.4 LTDC configuration parameters

The LTDC configuration parameters are sequenced following the pixel flow as shown in *Figure 444*, starting at the input (AXI master that reads pixels) and ending at the pixel output.

Common controls (such as interrupts) are listed at the end.

43.4.1 AXI master

The AXI master busses perform the accesses to the memory and are driven by the internal logic. The LTDC internal per-layer FIFOs are fairly large to cope with reasonably long external memory latencies.

The LDTC generates long burst (128-byte) for higher external memory efficiency.
For more granularity, it is possible to decrease the max burst size to 64 bytes, but it is not recommended.

Parameter in LTDC_LxBLCR

BL: burst length, in 64-bit words, with a default 0 for max size

Default configuration

The default AXI burst length is set at its maximum value after reset, with BL = 0.

43.4.2 Input layer definition and cropping

The LTDC supports two layers. Each layer is defined by:

- Its base-address in the memory

  If the buffer is YUV semi-planar or full-planar, there are respectively two and three base-addresses. The first buffer is called main and stores the luminance data, while the other buffers are called auxiliary and store the chrominance data.
- A dimension in the memory
  - Width defined as its pitch (or address jump or stride): amount of bytes between the start of one line and the start of the next line
  - Height that does not need to be defined
- A dimension of the pixel rectangle that is cropped and read from the memory
  - Top-left start position: set by the base-address at exactly the byte where the top-left pixel is located
  - Width, number of bytes per line to read

    - If too few bytes versus required, a FIFO underrun interrupt is generated(FUWIF).

    - If too many bytes versus required, the useless data are discarded.
  - Height, number of lines per frame to read

Note: The pitch and number of bytes per line can be different if a sub-part of a frame is cropped to be displayed. If the full frame is displayed, the pitch and number of lines are the same.

Parameters

- LEN in LTDC_LxCR: activates this layer.
- CFBADD in LTDC_LxCFBAR: main base address, for ARGB or Y buffer
- CFBLL in LTDC_LxCFBLR: main number of bytes per line
- CFBP in LTDC_LxCFBLR: pitch between two consecutive lines of the main buffer
- CFBLNBR in LTDC_LxCFBLNR: main number of lines per frame

Specifically for multi-planar layers, the following additional registers have to be configured (only required for YUV420 semi-planar and full-planar pixel formats):

- AFBADD0 in LTDC_L1AFBA0R: auxiliary address 0, for UV if semi-planar or for V if full-planar
- AFBADD1 in LTDC_L1AFBA1R: auxiliary address 1, for U if full-planar
- AFBLL in LTDC_L1AFBLR: auxiliary 0 and 1, number of bytes per line
- AFBP in LTDC_L1AFBLR: pitch for the auxiliary 0 and 1 buffers
- AFBLNBR in LTDC_L1AFBLNR: auxiliary 0 and 1 number of lines per frame

Configuration examples

- A frame buffer having the format RGB565 (2 bytes per pixel) and a width of 256 pixels (total number of bytes per line is 256 * 2 = 512), where pitch = line length requires a value of 0x02000207 to be written into this register.
- A frame buffer having the format RGB888 (3 bytes per pixel) and a width of 320 pixels (total number of bytes per line is 320 * 3 = 960), where pitch = line length requires a value of 0x03C003C7 to be written into this register.

- A frame buffer having the format CbCr (2 bytes per pixel) and a width of 256 pixels (total number of bytes per line is 256 * 2 = 512), where pitch = line length requires a value of 0x02000207 to be written into this register.
- A frame buffer having the format Cb (1 byte per pixel) and a width of 320 pixels (total number of bytes per line is 320 * 1 = 320), where pitch = line length requires a value of 0x01400147 to be written into this register.

43.4.3 Input pixel format

The LTDC reads the pixel data from memory buffers and maps it onto the internal LTDC format that is based on the definition of the input pixel format, defined per layer.

The pixel formats are grouped as follows:

- **Predefined ARGB formats:**

  ARGB8888, ABGR8888, RGBA8888, BGRA8888, RGB565, BGR565, and RGB888pack
- **Flexible ARGB format:**

  Allows a configurable width and location of each A,R,G, and B component and is used to extend the above list to any multi-component pixel format except to coplanar ARGB.
- **YUV formats:**

  Used to read YUV with a 422 coplanar, 420 semi-planar, or 420 full-planar format, and to convert the YUV color into RGB.

  While the layer 1 supports all YUV formats, the layer 2 only supports the 422 coplanar format.

Note: The LTDC hardware pipeline uses 8 bits per component: when reading and mapping a pixel format, with less than 8 bits per component. That component is extended to 8 bits by stuffing the LSB, that are recopied from its MSB. This applies mostly on RGB565.

For instance, a component on 5 bits (named: 43210) results in the following 8 bits mapped into the pipe: 43210432. A 3-bit input (named 210) results in 21021021.

Parameter when using the predefined ARGB formats

- PF in LTDC_LxPFCR: 0 to 6 to select one of the seven predefined pixel formats

Parameters when using the flexible ARGB formats

- PR in LTDC_LxPFCR: seven to activate the flexible pixel format type
- PSIZE in LTDC_LxFPF1R: defines the amount of bytes per pixel (typically 1,2,3, or 4)
- APOS/RPOS in LTDC_LxFPF0R: defines the Alpha/Red location inside a 32-bit word
- BPOS/GPOS in LTDC_LxFPF1R: defines the Blue/Green location inside a 32-bit word
- ALEN/RLEN in LTDC_LxFPF0R: defines the amount of bits for Alpha/Red components
- BLEN/GLEN in LTDC_LxFPF1R: defines the amount of bits for Blue/Green components

Examples:

An RGB565 pixel format is defined as follows:

– Alpha: APOS = 0, ALEN = 0
– Red: RPOS= 11, RLEN = 5
– Green: GPOS = 5, GLEN = 6
– Blue: BPOS = 0, BLEN = 5

The BGRA8888 32 bpp pixel format is set as follows (default reset values of LTDC_LxFPF0R and LTDC_LxFPF1R):

– Alpha: APOS = 0, ALEN = 8
– Red: RPOS= 8, RLEN = 8
– Green: GPOS = 16, GLEN = 8
– Blue: BPOS = 24, BLEN = 8

Parameters when using the YUV formats

The YUV pixel format is defined and detailed in *Section 43.4.4: YUV planar* and supersedes any setting here.

The following table describes the memory mapping of the predefined ARGB pixel format.

Table 384. Pixel data mapping versus color format

ARGB8888 (PF = 000)			
@+3 A_x[7:0]	@+2 R_x[7:0]	@+1 G_x[7:0]	@ B_x[7:0]
@+7 A_{x+1}[7:0]	@+6 R_{x+1}[7:0]	@+5 G_{x+1}[7:0]	@+4 B_{x+1}[7:0]
ABGR8888 (PF = 001)			
@+3 A_x[7:0]	@+2 B_x[7:0]	@+1 G_x[7:0]	@ R_x[7:0]
@+7 A_{x+1}[7:0]	@+6 B_{x+1}[7:0]	@+5 G_{x+1}[7:0]	@+4 R_{x+1}[7:0]
RGBA8888 (PF = 010)			
@+3 R_x[7:0]	@+2 G_x[7:0]	@+1 B_x[7:0]	@ A_x[7:0]
@+7 R_{x+1}[7:0]	@+6 G_{x+1}[7:0]	@+5 B_{x+1}[7:0]	@+4 A_{x+1}[7:0]
BGRA8888 (PF = 011)			
@+3 B_x[7:0]	@+2 G_x[7:0]	@+1 R_x[7:0]	@ A_x[7:0]
@+7 B_{x+1}[7:0]	@+6 G_{x+1}[7:0]	@+5 R_{x+1}[7:0]	@+4 A_{x+1}[7:0]
RGB565 (PF = 100)			
@+3 R_{x+1}[4:0] G_{x+1}[5:3]	@+2 G_{x+1}[2:0] B_{x+1}[4:0]	@+1 R_x[4:0] G_x[5:3]	@ G_x[2:0] B_x[4:0]

Table 384. Pixel data mapping versus color format (continued)

@+7 R_{x+3}[4:0] G_{x+3}[5:3]	@+6 G_{x+3}[2:0] B_{x+3}[4:0]	@+5 R_{x+2}[4:0] G_{x+2}[5:3]	@+4 G_{x+2}[2:0] B_{x+2}[4:0]
BGR565 (PF = 101)			
@+3 B_{x+1}[4:0] G_{x+1}[5:3]	@+2 G_{x+1}[2:0] R_{x+1}[4:0]	@+1 B_x[4:0] G_x[5:3]	@ G_x[2:0] R_x[4:0]
@+7 B_{x+3}[4:0] G_{x+3}[5:3]	@+6 G_{x+3}[2:0] R_{x+3}[4:0]	@+5 B_{x+2}[4:0] G_{x+2}[5:3]	@+4 G_{x+2}[2:0] R_{x+2}[4:0]
RGB888 (PF = 110)			
@+3 B_{x+1}[7:0]	@+2 R_x[7:0]	@+1 G_x[7:0]	@ B_x[7:0]
@+7 G_{x+2}[7:0]	@+6 B_{x+2}[7:0]	@+5 R_{x+1}[7:0]	@+4 G_{x+1}[7:0]
The flexible format (PF = 111), not represented here, allows any above mapping.			

43.4.4 YUV planar

This operator handles the YUV planar formats, reading data from potentially several distinct buffers (Y versus Cb versus Cr) into a single pixel information. The feature is available for layer 1 and layer 2 (coplanar only for the layer 2).

The LTDC supports the following planar formats:

- YUV422 coplanar (one buffer)
- YUV420 semi-planar (two buffers)
- YUV420 full-planar (three buffers)

Following its configuration, the LTDC reads the relevant luminance and chrominance buffers, upscales the chrominance components required to reach YUV444 pixels, and converts the color from YUV to RGB as described in *Section 43.4.5*.

The LTDC allows swapping the memory locations of the Y and Cb/Cr data and of the Cb and Cr data. It also allows inverting the endianness of the Y data. This extends the amount of supported pixel format variants.

Parameters

Pixel format parameters are stored in LTDC_LxPCR:

- YCEN: activates the YUV mode (both the planar extraction and color conversion).
- YCM: selects the input scheme (422 for coplanar, 420 for semi-planar, 420 for full-planar).
- YF: defines if either Y or Cb/Cr is provided first (from LSB).
- CBF: defines if either Cb or Cr is provided first (from LSB).
- OF: defines if either odd or even Y is provided first (from LSB).

When using a semi-planar or full-planar buffer, some additional base-address must be configured for auxiliary (chrominance) buffers, as well as the pitch of these buffers (see *Section 43.4.2*).

Configuration examples

Settings of OF, CBF, YF bits for a YCbCr 422 pixel format:

- OF = 0, CBF = 0, YF = 0 define the buffer word to containWord[31:0] = Y1-Cb-Y0-Cr.
- OF = 1, CBF = 0, YF = 0 define the buffer word to contain Word[31:0] = Y0-Cb-Y1-Cr.
- OF = 0, CBF = 1, YF = 0 define the buffer word to contain Word[31:0] = Y0-Cr-Y1-Cb.
- OF = 0, CBF = 0, YF = 1 define the buffer word to contain Word[31:0] = Cb-Y1-Cr-Y0.
- OF = 1, CBF = 1, YF = 1 define the buffer word to contain Word[31:0] = Cr-Y0-Cb-Y1.

  These settings are also valid for YCbCr 420 pixel formats.

43.4.5 YUV-to-RGB color conversion

This operator handles the YUV-to-RGB color conversion of the YUV planar pixels. The color conversion is activated automatically, simultaneously with the YUV planar handling.

The color conversion is based on a semi-flexible 3x3 matrix multiplication:

- It is restricted to YUV-to-RGB conversion due to some frozen coefficients (see next).
- It handles the most used YUV references (ITU-R BT.601 and ITU-R BT.709) thanks to some configurable coefficients.
- It supports the full versus reduced dynamic of the input pixel (luminance defined either in 0 to 255, or in 16 to 235), by extending a potential reduced-range input to a full range.

Parameters

- YCEN in LTDC_LxPCR: activates the color conversion, same bit as for planar.
- YREN in LTDC_LxPCR: if active, extends a reduced range to a full range.
- CR2R in LTDC_LxCYR0R: matrix coefficient from Cr to Red
- CB2B in LTDC_LxCYR0R: matrix coefficient from Cb to Blue
- CR2G in LTDC_LxCYR0R: matrix coefficient from Cr to Green
- CB2G in LTDC_LxCYR0R: matrix coefficient from Cb to Green

The full matrix computation, to convert from YCbCr to RGB, is defined as the following:

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} CR2R & 256 & 0 \\ -CR2G & 256 & -CB2G \\ 0 & 256 & CB2B \end{pmatrix} \times \begin{pmatrix} Cr - 128 \\ Y \\ Cb - 128 \end{pmatrix}$$

The CR2R, CB2B, CR2G, and CB2G coefficients are defined on 10 bits unsigned: as the operator works with 8-bit components, the coefficients are thus equivalent, in fixed-point, to 2-bit integer and 8-bit decimals.

The CR2G and CB2G coefficients are configured as unsigned positive, but are set negative in the hardware computation, as shown in the above matrix where a “-” is mentioned.

When the luminance scaling is active (YREN = 1, thus where the YCbCr input is assumed in the range of 16 to 235/240, and RGB in 0 to 255), then the Y input value is first scaled in hardware to Y’ = 255/219 * (Y - 16). That first result is then inserted as input to the above matrix multiplication.

Configuration examples

To read an ITU-R BT.601 layer and to convert its YCbCr (assuming a default reduced dynamic in 16 to 235/240) to RGB (with full dynamic in 0 to 255), the recommended coefficients are:

- YCEN = 1: to activate the conversion
- YREN = 1: to select the reduced dynamic range
- CR2R = 409: specific coefficient set for ITU-R BT.601
- CR2G = 208
- CB2G = 100
- CB2B = 516

To read an ITU-R BT.709 layer and to convert its YCbCr (assuming a default reduced dynamic in 16 to 235/240) to RGB (with full dynamic in 0 to 255), the recommended values to configure coefficients are:

- YCEN = 1: to activate the conversion
- YREN = 1: to select the reduced dynamic range
- CR2R = 459: specific following coefficient set is for ITU-R BT.709
- CR2G = 136
- CB2G = 55
- CB2B = 541

43.4.6 Horizontal or vertical mirroring

The LTDC can mirror horizontally or vertically its input layers, thus to read them from right to left, and/or from bottom to top.

It applies on both coplanar (ARGB, YUV422) and multi-planar (YUV420) pixel formats.

Parameters for horizontal mirroring

To mirror a layer horizontally, the LTDC must be configured to extract the pixels from right to left (backward), and to set the layer base-address at the right-end of the line.

The configuration bitfields are the following:

- HMEN in LTDC_LxCR: activates the horizontal mirroring.
- CFBADD in LTDC_LxCFBAR: base + (width - 1) (increase for horizontal mirror)
- AFBADD0 in LTDC_L1AFBA0R: base + (width - 1)
- AFBADD1 in LTDC_L1AFBA1R: base + (width - 1)

Parameters for vertical mirroring

To mirror a layer vertically, the LTDC must be configured to start reading this layer from the bottom, and line per line, to come upward. The layer base-address must be set to the start of the bottom line, and a negative line-per-line pitch must be provided.

The configuration bitfields for the base-address are (all to be set at bottom line start):

- CFBADD in LTDC_LxCFBAR: main base-address
- AFBADD0 in LTDC_L1AFBA0R: auxiliary address 0
- AFBADD1 in LTDC_L1AFBA1R: auxiliary address 1
- CFBADD in LTDC_LxCFBAR: base + (lines - 1) * pitch (increase for vertical mirror)

- AFBADD0 in LTDC_L1AFBA0R: base + (lines - 1) * pitch
- AFBADD1 in LTDC_L1AFBA1R: base + (lines - 1) * pitch

The configuration bitfields for the pitch are (all to be set with a negative pitch):

- CFBP in LTDC_LxCFBLR: -pitch (negative line pitch for vertical mirror)
- AFBP in LTDC_L1AFBLR: -pitch

43.4.7 Default layer color

Every layer has a default color in the format ARGB8888, that is used outside the defined layer window or when a layer is disabled. This default color can be configured as active or absent.

Parameters

The default layer color parameters are:

- DCBEN in LTDC_LxCR: default color activation
- DCALPHA in LTDC_LxDCCR: default Alpha transparency setting
- DCRED in LTDC_LxDCCR: default Red component
- DCGREEN in LTDC_LxDCCR: default Green component
- DCBLUE in LTDC_LxDCCR: default Blue component

Default configuration

By default, and after reset, the default layer color is inactive (DCBEN = 0 and DCALPHA = 0.

Note: The blending of a layer is always performed even when a layer is disabled. To avoid displaying the default color when a layer is disabled, the default Alpha must be set to null (DCALPHA = 0), or the whole default color must be disabled (with DCBEN).

43.4.8 Color look-up table (CLUT)

The LTDC supports frame buffers with each pixel defined as an index to a color table, allowing a display with a set of 256 colors, but stored in memory with only 8 bpp.

The CLUT defines 256 colors and must be indexed with a maximum 8-bit pixel format (typically L8 or AL44). Such format can be extracted using the flexible RGB approach described in *Section 43.4.3*.

The CLUT table is differentiated for each layer. It must be initialized with one RGB888 value per color index, with the color index corresponding to the CLUT address:

- For L8, the CLUT must be loaded with 256 colors.
- For AL44, the CLUT must be only loaded with 16 colors. Each color address is defined by replicating the 4-bit L channel to 8 bits as follows:
  - Color L4 = 0: looks out its RGB888 from index = 0x00.
  - Color L4 = 1: looks out its RGB888 from index = 0x11.
  - Color L4 = 2: looks out its RGB888 from index = 0x22.
  - ...
  - Color L4 = 15: looks out its RGB888 from index = 0xFF.
- For ARGB, the Blue component is used as the CLUT index.

Parameters

The RGB888 values and color indices are configured through the following bitfields in LTDC_LxCLUTWR:

- CLUTADD defines the color index to update.
- RED defines the 8-bit Red component of that color index.
- GREEN defines the 8-bit Green component of that color index.
- BLUE defines the 8-bit Blue component of that color index.

43.4.9 Transparency color keying

Pixels can be defined as transparent using a color key. If the pixel RGB components equals the defined RGB color key, that pixel is considered transparent and its Alpha channel is forced null.

A pixel forced to transparent is transparent when proceeding to the later blending phase, and shows the pixel of the underlying layer.

The color key is defined as an RGB888 value, that is compared with the flowing pixel after some processing in the input format operator (where LSBs are stuffed by replication) and CLUT processing. The constant color keying parameters must take these previous processing into account.

Parameters

The color keying parameters are the following:

- CKEN in LTDC_LxCR enables the color keying.
- CKRED in LTDC_LxCKCR defines the 8-bit Red color key.
- CKGREEN in LTDC_LxCKCR defines the 8-bit Green color key.
- CKBLUE in LTDC_LxCKCR defines the 8-bit Blue color key.

Configuration example

A mid-yellow color (50 % Red + 50 % Green) is used as the transparent color key. The color keying parameters take it into account as follows:

- ARGB8888 input: The mid-yellow input is 0x808000. Set LTDC_LxCKCR = 0x808000.
- RGB565 input: The mid-yellow input is 0x8400. Set LTDC_LxCKCR = 0x848200 (due to the replication of input MSB onto the missing LSB).
- CLUT input for L8, AL88 or AL44: Set one of the palette entry to the mid-yellow color 0x808000 and set LTDC_LxCKCR = 0x808000.

43.4.10 Display composition - windowing

When performing the display composition, every layer can be cropped and positioned into the active display area. The window position and size are configured by defining its top-left and bottom-right corners.

Note: The locations are defined versus the internal timing generator that must take into account the potential synchronization header, the back porch size and the chosen active pixel area (as shown in Figure 445).

The mapping between the input layer and the visible window is performed as follows:

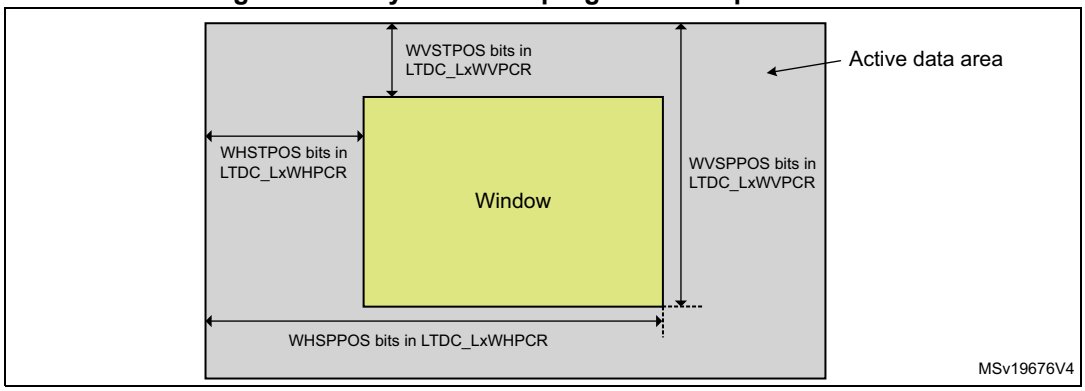
- The most top-left pixels of the input layer are mapped to the top-left of the visible window of this layer.
- If the visible window is smaller (less wide, less high) than the input layer, then the exceeding input pixels are automatically discarded from the pipe.

Parameters:

The following parameters are defined versus the top-left of the synchronization:

- WHSTPOS in LTDC_LxWHPCR: first (left) visible pixel of the layer
- WHSPPOS in LTDC_LxWHPCR: last (right) visible pixel of the layer
- WVSTPOS in LTDC_LxWVPCR: first (top) visible line of the layer
- WVSPPOS in LTDC_LxWVPCR: last (bottom) visible line of the layer

Figure 445. Layer window programmable parameters



Configuration examples

- LTDC_BPCR is configured to 0x000E0005 (AHBP[11:0] is 0xE), and LTDC_AWCR is configured to 0x028E01E5 (AAW[11:0] is 0x28E). To configure the horizontal position of a window size of 630x460, with horizontal start offset of 5 pixels in the active data area:
  - Layer window first pixel: WHSTPOS[11:0] must be programmed to 0x14 (0xE + 1 + 0x5).
  - Layer window last pixel: WHSPPOS[11:0] must be programmed to 0x28A.
- LTDC_BPCR is configured to 0x000E0005 (AVBP[11:0] is 0x5) and LTDC_AWCR is configured to 0x028E01E5 (AAH[11:0] is 0x1E5). To configure the vertical position of a window size of 630x460, with vertical start offset of eight lines in the active data area:
  - Layer window, first line: WVSTPOS[11:0] must be programmed to 0xE (0x5 + 1 + 0x8).
  - Layer window last line: WVSTPOS[11:0] must be programmed to 0x1DA.

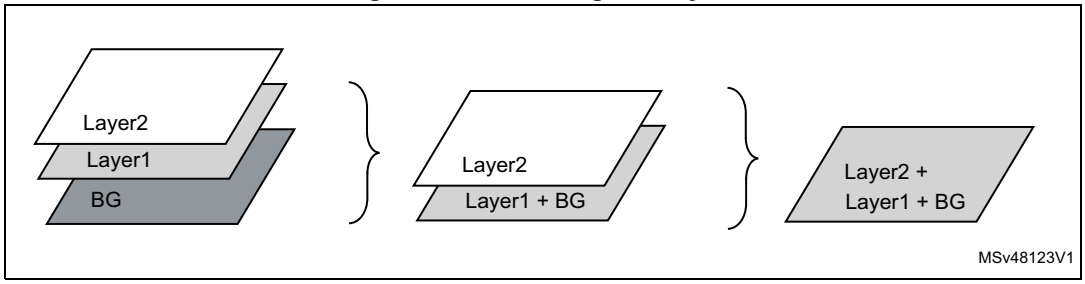
43.4.11 Display composition - blending

The Alpha blending stage is used to blend together the 2 input layers, on top of a background color.

It is possible to select the sequence of the layers (foreground, middle, and background), to apply a global multiplicative Alpha per-layer, and to define an underlying background color.

The blending computation is performed from back to front, sequentially, as shown the following figure.

Figure 446. Blending two layers



The general formula for a 2-to-1 blending is CR = (BF1 x C1) + (BF2 x C2) where:

- CR: resulting blended color
- BF1: blending-factor of current layer
- C1: layer-color of the current layer
- BF2: blend-factor of underlying pixels
- C2: layer-color of the underlying pixels

Both blending factor, BF1 and BF2, can be defined as one of the following:

- Pure per-pixel Alpha value
- Mix of per-pixel Alpha and a global per-layer Alpha (the mix is multiplicative)

Parameters

- CONSTA in LTDC_LxCACR: global multiplicative Alpha value of the current layer
- BOR in LTDC_LxBFCR: blending order of the current layer
- BF1 in LTDC_LxBFCR: blending factor for the current layer
- BF2 in LTDC_LxBFCR: blending factor for the underlying layer

The background color is defined by the following bitfields in LTDC_BCCR:

- BCRED: background Red component
- BCGREEN: background Green component
- BCBLUE: background Blue component

Note: For secure purposes, in case of an inconsistent configuration, the blending order falls back to a fixed order, with the layer 2 on the foreground, and layer 1 in background, followed by the underlying background color.

The inconsistent fallback is setup to protect a potential secure layer: If a secure software configures the secure layer2 to be on the foreground, and a malware tries to place that another layer on the foreground, the inconsistency is detected, and the secure layer is automatically forced on the foreground, guaranteeing that it is visible.

Configuration example

Only layer 1 is enabled, BF1 configured to constant alpha. BF2 configured to 1 - constant alpha. The constant alpha programmed in LTDC_LxCACR is 240 (0xF0). Thus, the constant alpha value is 240 / 255 = 0.94. C: current layer color is 128. Cs: background color is 48. Layer 1 is blended with the background color.
BC = constant alpha x C + (1 - constant alpha) x Cs = 0.94 x 128 + (1 - 0.94) x 48 = 123.

Default configuration

All active layers can be made visible and blended by keeping the following configuration values after reset:

- CONSTA: keep as reset CONSTA = 255.
- BOR: keep as reset BOR (layer) = LayerID-1.
- BF1: keep as reset BF1 = 6 for default mixed Alpha.
- BF12: keep as reset BF2 = 7 for default mixed 1-Alpha.
- Background color: keep as reset (black), although any color is ok.

43.4.12 Gamma correction

The gamma correction is a non-linear operator that adapts the pixel to the non-linear display, using a compensating pre-distortion. The R, G, B components can be independently adapted, using a configurable non-linear transformation.

The non-linear transformation is implemented in the LTDC as eight independent linearly-interpolated segments: the desired non-linear transformation must thus be mapped onto the eight segments, and the shape of the eight segments configured in LTDC registers.

Parameters

- GAMEN in LTDC_GCR: activation of the gamma operator
- ADDR in LTDC_GCCR: table configuration, index (among 256) to write-in
- COMP in LTDC_GCCR: table configuration, 8-bit component to write-in
- REN in LTDC_GCCR: table configuration, active to write into the Red table
- GEN in LTDC_GCCR: table configuration, active to write into the Green table
- BEN in LTDC_GCCR: table configuration, active to write into the Blue table

Note: As the gamma correction works with eight interpolated segments. The only table indices to write-in are located at index: 0, 32, 64, 96,...,224, 255. The value to provide for index 255 contains the value for the pseudo-index 256.

Example configuration

The gamma correction is recommended to be set to pre-compensate the pixels with a gamma exponent of 2.2: Red, Green, and Blue components are compensated the same way by the Gamma. The configuration of Red, Green, and Blue tables is then usually the same.

Default configuration

The gamma correction is disabled with the default configuration after reset, with GAMEN = 0.

43.4.13 YUV output conversion

The default RGB888 24 bpp output can be converted to a YUV422 16 bpp output (coplanar), in order to reduce the bitrate or the interface width, if necessary.

The operator can convert the RGB to YUV with either ITU-R BT.601 versus BT.709 reference, and filter down the chrominance, to output the pixels as a YUV422 interface.

When the conversion is enabled, the Y is output on the LCD_G (green bits) and the Cb/Cr are output on the LCD_R (red bits).

Parameters in LTDC_EDCR

- OCYEN: activates the output conversion to YUV422.
- OCSEL: selects between a color conversion to ITU-R BT.601 versus to BT.709.
- OCYCO: defines whether Cb or Cr is output first.

The configuration must follow what is expected by the external display panel.

Default configuration

The YUV output conversion is disabled with the default configuration after reset, with OCYEN = 0, allowing a default RGB888 to be output.

43.4.14 Dithering

The dithering operator allows a clean conversion of the RGB888 output into a reduced pixel format, like RGB565 or RGB666, and so avoiding the usual artifact of flat area that occurs when no dithering is available.

The dithering is based on a LFSR (polynomial generating random) and adds pseudo-random noise on the LSB of each of the displayed component (bit 2 in case of an RGB666), and makes that the average of the local pixels represents the information content of the full 8-component value.

Once the LTDC is enabled, the LFSR starts running with the first active pixel and it is kept running even during blanking periods and when dithering is switched off. If the LTDC is disabled, the LFSR is reset.

Parameters in LTDC_GCR

- DEN: dithering enable
- DRW: bit-id of LSB of the displayed Red component (among 8 bits)
- DGW: bit-id of LSB of the displayed Green component (among 8 bits)
- DBW: bit-id of LSB of the displayed Blue component (among 8 bits)

Example

To dither from RGB888 to RGB565 (assuming a RGB565 display panel):

- The LTDC outputs the Red component on pins LCD_R[7:0].
- The display panel probes the five MSB bits LCD_R[7:3].
- The red bit to dither, using its two following LSB as decimals, is the bit 3.
- DRW must be configured to 3.
- DGW = 2 and DBW = 3.

Similarly, a dithering to RGB666 requires DRW = DGW = DBW = 2.

Default configuration

The dithering is disabled with the default configuration after reset, with DEN = 0, allowing a default RGB888 to be output.

43.4.15 CRC hashing

The CRC operator allows a 16-bit signature to be extracted for each frame, by hashing the R,G,B components of all the frame pixels.

The computed CRC can be read in LTDC_CCRCR.

An expected CRC value can be configured: if the computed CRC does not match the expected CRC, the CRC interrupt is triggered. It allows a system test, letting the LTDC run for hours, and be warned only if a failed frame occurs.

Parameters

- CRCEN in LTDC_GCR: activates the CRC computation
- ECRC in LTDC_ECRCR: expected 16-bit CRC value
- CCRC in LTDC_CCRCR: computed 16-bit CRC value

Default configuration

The CRC hashing is disabled with the default configuration after reset, with CRCEN = 0.

43.4.16 Display timing

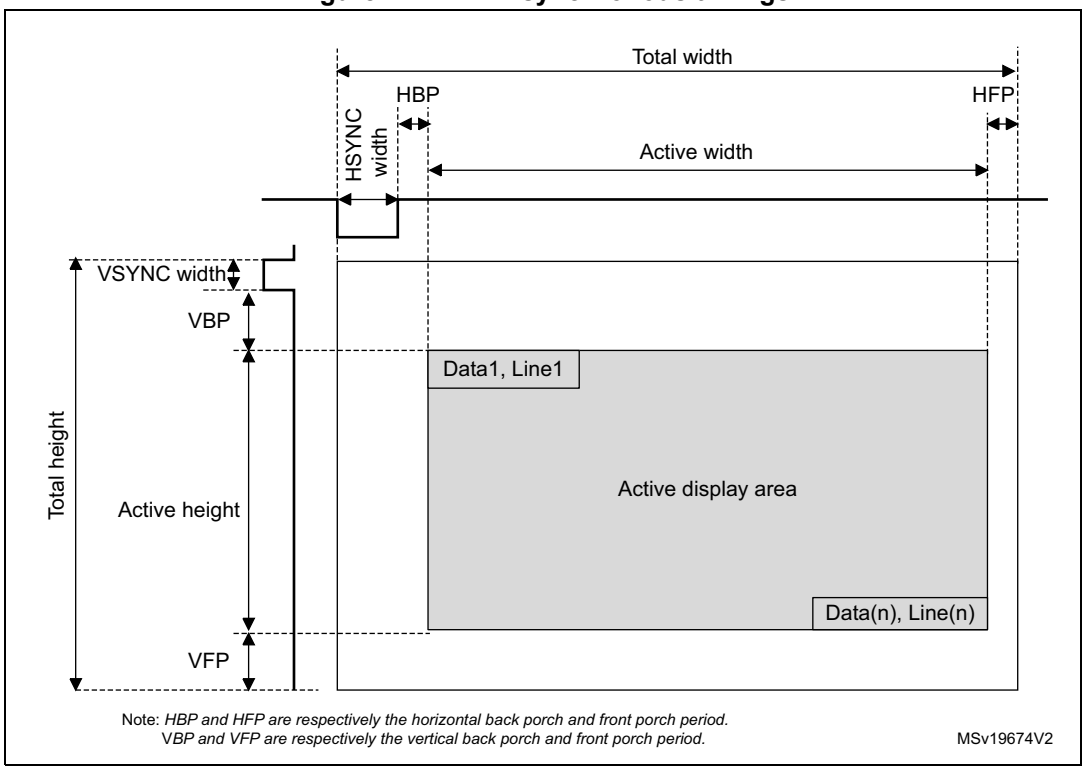
This section describes the configuration of the display timings, so that the LTDC matches what is expected by the external display panel.

The configuration defines the horizontal and vertical duration of each of the following items:

- Synchronization period (HSYNC and VSYNC)
- Back porch (HBP, VBP)
- Active display (pixels that are visible on the screen)
- Front porch (HFP, VFP)

These timings are described in the following figure.

Figure 447. LTDC synchronous timings



As a result of these timings, the LTDC samples in pixels during the active display duration, and generates the output synchronization signals (HSYNC, VSYNC), the pixel data, and the data-enable signal.

Parameters

All the timings configured next are absolute timings, computed from the display start position, and not the relative increment (or width or height) of each area.

All these bitfields indicate the last pixel of each area, and hence are subtracted 1.

- **HSYNC and VSYNC width** (horizontal and vertical synchronization width) in LTDC_SSCR:
  - HSW = HSYNC width - 1
  - VSW = VSYNC width - 1
- **HBP and VBP** (last pixel/line of the horizontal and vertical synchronization back porch) in LTDC_BPCR:
  - AHBP = HSYNC width + HBP - 1
  - AVBP = VSYNC width + VBP - 1
- **Active width and active height** (last pixel/line of the active display area) in LTDC_AWCR:
  - AAW = HSYNC width + HBP + active width - 1
  - AAW = VSYNC width + VBP + active height - 1

- **Total display width** (last pixel/line of the total display timings) in LTDC_TWCR:
  - TOTALW = HSYNC width + HBP + active width + HFP - 1
  - TOTALH = VSYNC height + VBP + active height + VFP - 1

Note: When the LTDC is enabled, timings generated start at the position (X,Y) = (0,0), and generate immediately the first horizontal synchronization pixel in the vertical synchronization, followed by the back porch, then the active display area, and finally the front porch.

When the LTDC is disabled, the timing generator block is reset to X = total width - 1, Y = total height - 1. It holds the last pixel before the vertical synchronization phase and the FIFO are flushed. Therefore, only blanking data is output continuously.

Example of synchronous timing configuration

Assume the following display panel timings (to be extracted from panel datasheet):

- Horizontal and vertical synchronization width: 0xA pixels, 0x2 lines
- Horizontal and vertical back porch: 0x14 pixels, 0x2 lines
- Active width and active height (320x240): 0x140 pixels, 0x0F0 lines
- Horizontal and vertical front porch: 0xA pixels 0x4 lines

Bitfield values to configure become:

- HSW[11:0] = 0x9 (horizontal synchro -1) in LTDC_SSCR.
- VSH[11:0] = 0x1 (vertical synchro -1) in LTDC_SSCR.
- AHBP[11:0] = 0x1D (horizontal synchro + back porch -1) in LTDC_BPCR.
- AVBP[11:0] = 0x3 (vertical synchro + back porch -1) in LTDC_BPCR.
- AAW[11:0] = 0x15D (horizontal synchro + back porch + width -1) in LTDC_AWCR.
- AAH[11:0] = 0xF3 (vertical synchro + back porch + height -1) in LTDC_AWCR.
- TOTALW[11:0] = 0x167 (horizontal synchro + back porch + width + front porch -1) in LTDC_TWCR.
- TOTALH[11:0] = 0xF7 (vertical synchro + back porch + width + front porch -1) in LTDC_TWCR.

Default configuration

There is no default configuration to propose. As display timings depend on the specific display panel and its own required timings, all these parameters must be extracted from the display panel specification.

Note: The configuration of all the above is mandatory to have the LTDC driving correctly the external display panel.

43.4.17 Output interface polarity

The polarity of the control signals of the output interface includes the clock polarity, the vertical/horizontal synchronization, and the data-enable signal.

Parameters in LTDC_GCR

- PCPOL: polarity of the output clock
- VSPOL: polarity of the vertical synchronization
- HSPOL: polarity of the vertical synchronization

- DEPOL: polarity of the data enable

Default configuration

There is not default configuration, as the configuration of the polarity depends on the display panel specifications.

The following configuration can be used to have all items working with a positive polarity:

- PCPOL = 1 (rising-edge)
- VSPOL = 1 (active high)
- HSPOL = 1 (active high)
- DEPOL = 1 (active high)

43.4.18 Shadow registers

When activated, the LTDC is expected to display continuously data, frame after frame.

The user may want to modify some settings: for example, add a new layer, modify the layer window location, or update the displayed content by updating the pointer to the buffer that is displayed.

The LTDC provides shadow registers for most of its configuration registers to allow transitions to occur cleanly synchronized during the short vertical synchronization, when nothing is displayed: while the LTDC is busy working with a copy of its configuration registers to control the ongoing display (the active set), a new version of the registers can be progressively updated by software (the shadow set). When ready, and at the exact VSync event, the shadow set is copied into the active set, ready for the next frame.

Once the shadow set is fully updated, there are two ways to transfer it in the active set:

- Immediate copy: the copy occurs immediately.
- Delayed to next VSync: the copy occurs at the next VSync.

Note: In case of delayed copy, shadow registers must not be modified again before the reload has been done.

Reading from shadow registers returns the current active value. The new written value can only be read after the reload has taken place.

The impossibility to read a temporary value in a shadow register makes impossible to perform a read-modify-write to set/reset a single bit inside a shadow register.

A register reload interrupt can be generated if enabled in LTDC_IER or LTDC_IER2.

Shadowed registers

Non-layer registers

The non-layer registers (not prefixed with LTDC_Lx) are all non-shadowed. They contain static configurations dependent mostly on the attached display panel. They do not require update during an ongoing display.

Layer registers

The layer registers are prefixed with LTDC_Lx. Most are shadowed to support a clean frame-to-frame update. The following are not shadowed, either because their shadowing

must be too costly or because they are not expected to require frequent or synchronized updates:

- LTDC_LxRCR reload control
- LTDC_LxCKCR color key (assumed static)
- LTDC_LxCLUTWR CLUT (indirect access, too costly to shadow)
- LTDC_LxCYR0R/R1R YUV-to-RGB color conversion (assumed static)
- LTDC_LxFPF0R/LTDC_LxFPF1R flexible pixel format (assumed static)

Parameters

The centralized reload of the shadow register is controlled with bitfields in LTDC_SRCR:

- IMR: writing 1 triggers an immediate copy.
- VBR: writing 1 triggers a copy delayed at the next blanking.

A per-layer reload control is available with bitfields in LTDC_LxRCR:

- GRMSK delegates the per-layer reload to the central reload (above).
- IMR: writing 1 triggers an immediate copy for this layer.
- VBR: writing 1 triggers a copy delayed at the next blanking.

Default configuration

For the initial LTDC configuration, when the display is inactive, the IMR mode must be used to copy the initial configuration to the active set.

When the display is active, it is recommended to use VBR to update any parameters: the parameter transition occurs then from frame-to-frame, and not inside an ongoing frame. Applying an immediate change of the LTDC configuration within a frame results in an unpredictable behavior.

43.4.19 General control

The final and mandatory step, after all the above configurations, is to trigger the display. The LTDC has two working mode:

- Continuous mode: the LTDC continuously sends frames to the display.
- Single-frame mode: the LTDC sends a single frame after a software-based trigger (generated by a software access to a register)

Parameters in LTDC_GCR

- LTDCEN: general LTDC activation
- SFEN: moves from the continuous to the single-frame mode.
- SFSWTR: software trigger, when in single-frame mode

43.4.20 Hardware trigger generation

The LTDC can generate an hardware trigger on a line event to synchronize with other peripherals.

As these hardware events reuse the interrupt circuitry, the enable bit must be set in order to properly generate the hardware event. The interrupt masking must thus be done at NVIC level.

Table 385. LTDC hardware triggers

Event line	Event	Event flag	Enable control bit	Event clear method
ltdc_lo_li	Lower-layer line interrupt	LIF (LTDC_ISR)	LIE (LTDC_IER)	write 1 in CLIF (LTDC_ICR)
ltdc_up_li	Upper-layer line interrupt	LIF (LTDC_ISR2)	LIE (LTDC_IER2)	write 1 in CLIF (LTDC_ICR2)

43.4.21 Provision for a secure layer

The LTDC allows a secure layer support, with the addition of a wrapper around the LTDC that secures the access to its registers (register-per-register), and secures the access of its AXI master to the layer framebuffers.

The layer 2 can be secured thanks to the measures described in the following sub-sections.

Configuration registers

The granularity between secure versus nonsecure items is at register level:

- A secure register contains only to-be-secured bitfields.
- Nonsecure bitfields are all in open-access registers.

There is no merge of secure and nonsecure bitfields in a same register. It allows all registers to be segregated between secure ones (with an access only by secure software) and open ones (with an access by any nonsecure software).

Interruptions

The interruption logic has been duplicated, with one secure interrupt logic (register and pin) reserved for secure software, and another open interrupt logic (second register set and second output pin) for nonsecure software.

Blending

The blending order (foreground to background) is defined in per-layer register, so that the blending order of the secure layer2 can be securely defined as foreground. If some nonsecure software defines layer 1 in foreground, the integrated hardware consistency check reverses the blending order to a fixed one, forcing the secure layer to be in the foreground.

CRC

The CRC configuration registers stated next can also be defined as secure, to secure the extracted signature of the displayed frame. The CRC that computes a signature of the displayed frame has its configuration located in specific registers that can be defined as secure, too.

- LTDC_ECRCR
- LTDC_CCRCR

43.5 LTDC interrupts

The LTDC provides various maskable interrupts, that are grouped into two global interrupt pins: one working interrupt pin, and one error interrupt pin.

These maskable interrupts and two global output pins are duplicated for the secure software usage, resulting in a duplicated set of registers (for masking) and a duplicated set of output pin. Hence a total of four interrupt output pins: global working, global error, secure global, secure global error.

The following working interrupts on which to synchronize software are ORed together as a global working interrupt:

- Line interrupt: generated when a configured line number is reached. Its position is Configured in LTDC_LIPCR.
- Register reload interrupt: generated when a shadow-register reload is performed during the vertical blanking period.

The following error interrupts warn about some internal processing errors, and are ORed together as a global error interrupt:

- FIFO underrun warning interrupt: generated when a pixel is requested from an empty layer FIFO.
- FIFO underrun error interrupt: generated when more pixels are requested from an empty layer FIFO.
- Transfer-error interrupt: generated when an AXI bus error occurs during data transfer.
- CRC-error interrupt: generated when the CRC computed and expected are different.

The LTDC also provides an observation mean, to retrieve X and Y positions of the currently displayed pixel.

Parameters

The following nonsecure registers are used to handle the interrupts:

- LTDC_LIPCR: defines the line-number on which the line-number interrupt occurs.
- LTDC_IER demasking bit: setting each bit enables the corresponding interrupt.
- LTDC_ISR: event bit where to read if an event has occurred
- LTDC_ICR: clear bit where to write to clean an interrupt.
- LTDC_CPSR: location with X and Y of the current displayed pixel
- LTDC_FUTR: trigger from which a FIFO underrun warning becomes a FIFO underrun error, and is the amount of missed words of 64 bits

The duplicated secure registers are used to handle the interrupts (with same respective content as the nonsecure ones): LTDC_LIPCR2, LTDC_IER2, LTDC_ISR2, LTDC_ICR2.

The following table lists the bitfields of the available interrupts, and shows the four pins on which these bitfields are directed:

- Nonsecure registers drive ltdc_lo_xx interrupts. Secure registers drive ltdc_up_xx interrupts.
- Interrupts triggered by an error are suffixed with a xx_err_it. Interrupts triggered during a normal LTDC handling are suffixed with an _it only.

Table 386. LTDC interrupt requests

Registers	Interrupt event	Event bit	Enable control bit	Interrupt signal
Nonsecure registers (IER, ISR, ICR)	Line number	LIF	LIE	ltdc_lo_li
	Register reload	RRIF	RRIE	ltdc_lo_li
	FIFO underrun warning	FUWIF	FUWIE	ltdc_lo_li
	FIFO underrun error	FUIF	FUIE	ltdc_lo_err_it
	Bus transfer error	TERRIF	TERRIE	ltdc_lo_err_it
	CRC error	CRCIF	CRCIE	ltdc_lo_err_it
Secure registers (IER2, ISR2, ICR2)	Line number	LIF	LIE	ltdc_up_li
	Register reload	RRIF	RRIE	ltdc_up_li
	FIFO underrun warning	FUWIF	FUWIE	ltdc_up_li
	FIFO underrun error	FUIF	FUIE	ltdc_up_err_it
	Bus transfer error	TERRIF	TERRIE	ltdc_up_err_it
	CRC error	CRCIF	CRCIE	ltdc_up_err_it

43.6 LTDC programming procedure

The LTDC programming procedure consists to activate the clocks, to follow all the operator sequence (as listed in *Section 43.4*) from the beginning (AXI master) to the end (output interface polarity), and to configure the needed operators as described.

The minimum configuration that must be performed is the following:

1. Enable the LTDC clock in the RCC register:
   - APB slave
   - AXI master
   - Pixel clock that fits display panel requirements (resolution, blanking, frame rate)
2. Define the buffer of at least one input layer (see *Section 43.4.2*, with for example, layer base-address, size, or buffer pitch).
3. Define this layer pixel format (see *Section 43.4.3*). ARGB8888 can simply be used, depending on the stored buffer.
4. Define where this layer is inserted in the final display (see *Section 43.4.10*): typically the top-left and bottom-right corners of the display.
5. Define how this layer is blended in the final display (see *Section 43.4.11*).
6. Define timings of the display panel to drive (see *Section 43.4.16*).

7. Define polarity of the output signals (see *Section 43.4.17*).
8. Copy all above settings, once, using the immediate global reload (see *Section 43.4.18*).
9. Start the display by activating the main enable (see *Section 43.4.18*).

Note: Most layer registers are shadowed. Once a register is written, it must not be modified again before the reload has been done. A new write to the same register overrides the previous configuration if not yet reloaded. It is also not possible to read the temporary value of a shadowed register, which makes impossible to perform read-modify-write to set/reset a single bit inside a register.

43.7 LTDC registers

43.7.1 LTDC synchronization size configuration register (LTDC_SSCR)

Address offset: 0x008

Reset value: 0x0000 0000

This register defines the number of horizontal synchronization pixels minus 1, and the number of vertical synchronization lines minus 1. Refer to *Figure 447* and *Section 43.4* for a configuration example.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	HSW[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	VSH[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **HSW[11:0]**: Horizontal synchronization width (in units of pixel clock period)
This bitfield defines the number of horizontal synchronization pixel minus 1.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **VSH[11:0]**: Vertical synchronization height (in units of horizontal scan line)
This bitfield defines the vertical synchronization height minus 1. It represents the number of horizontal synchronization lines.

43.7.2 LTDC back porch configuration register (LTDC_BPCR)

Address offset: 0x00C

Reset value: 0x0000 0000

This register defines the accumulated number of horizontal synchronization and back porch pixels minus 1 (HSYNC width + HBP - 1), and the accumulated number of vertical

synchronization and back porch lines minus 1 (VSYNC height + VBP - 1).
Refer to *Figure 447* and *Section 43.4* for a configuration example.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	AHBP[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	AVBP[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **AHBP[11:0]**: Accumulated horizontal back porch (in units of pixel clock period)
This bitfield defines the accumulated horizontal back porch width that includes the horizontal synchronization and horizontal back porch pixels minus 1.
The horizontal back porch is the period between horizontal synchronization going inactive, and the start of the active display part of the next scan line.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **AVBP[11:0]:** Accumulated Vertical back porch (in units of horizontal scan line)
This bitfield defines the accumulated vertical back porch width that includes the vertical synchronization and vertical back porch lines minus 1.
The vertical back porch is the number of horizontal scan lines at a start of frame to the start of the first active scan line of the next frame.

43.7.3 LTDC active width configuration register (LTDC_AWCR)

Address offset: 0x010

Reset value: 0x0000 0000

This register defines the accumulated number of horizontal synchronization, back porch and active pixels minus 1 (HSYNC width + HBP + active width - 1), and the accumulated number of vertical synchronization, back porch lines and active lines minus 1 (VSYNC height + BVBP + active height - 1). Refer to *Figure 447* and *Section 43.4* for a configuration example.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	AAW[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	AAH[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **AAW[11:0]**: Accumulated active width (in units of pixel clock period)
This bitfield defines the accumulated active width that includes the horizontal synchronization, horizontal back porch and active pixels minus 1. The active width is the number of pixels in active display area of the panel scan line.
Refer to device datasheet for the maximum active width supported following the maximum pixel clock.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **AAH[11:0]**: Accumulated active height (in units of horizontal scan line)
This bitfield defines the accumulated height that includes the vertical synchronization, vertical back porch and the active height lines minus 1. The active height is the number of active lines in the panel.
Refer to device datasheet for the maximum active height supported following the maximum pixel clock.

43.7.4 LTDC total width configuration register (LTDC_TWCR)

Address offset: 0x014

Reset value: 0x0000 0000

This register defines the accumulated number of horizontal synchronization, back porch, active and front porch pixels minus 1 (HSYNC width + HBP + active width + HFP - 1), and the accumulated number of vertical synchronization, back porch lines, active and front lines minus 1 (VSYNC height + BVBP + active height + VFP - 1). Refer to *Figure 447* and *Section 43.4* for a configuration example.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TOTALW[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TOTALH[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **TOTALW[11:0]**: Total width (in units of pixel clock period)
This bitfield defines the accumulated total width that includes the horizontal synchronization, horizontal back porch, active width and horizontal front porch pixels minus 1.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **TOTALH[11:0]**: Total height (in units of horizontal scan line)
This bitfield defines the accumulated height that includes the vertical synchronization, vertical back porch, the active height and vertical front porch height lines minus 1.

43.7.5 LTDC global control register (LTDC_GCR)

Address offset: 0x018

Reset value: 0x0000 2220

This register defines the LTDC global configuration.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HSPOL	VSPOL	DEPOL	PCPOL	Res.	Res.	SFSWTR	SFEN	Res.	Res.	Res.	Res.	CRCEN	Res.	Res.	DEN
rw	rw	rw	rw			w	rw					rw			rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DRW[2:0]			Res.	DGW[2:0]			Res.	DBW[2:0]			Res.	Res.	GAMEN	LTDCEN
	r	r	r		r	r	r		r	r	r			rw	rw

Bit 31 **HSPOL**: Horizontal synchronization polarity
This bit is set and cleared by software.
0: Horizontal synchronization polarity is active low.
1: Horizontal synchronization polarity is active high.

Bit 30 **VSPOL**: Vertical synchronization polarity
This bit is set and cleared by software.
0: Vertical synchronization is active low.
1: Vertical synchronization is active high.

Bit 29 **DEPOL**: Blanking (no data/pixel) polarity
This bit is set and cleared by software.
0: Blanking (no data/pixel) polarity is active low.
1: Blanking (no data/pixel) polarity is active high.

Bit 28 **PCPOL**: Pixel clock polarity
This bit is set and cleared by software.
0: The pixel and sync data are generated at the rising-edge of the output LCD_CLK clock.
1: The pixel and sync data are generated at the falling-edge of the output LCD_CLK clock.

Bits 27:26 Reserved, must be kept at reset value.

Bit 25 **SFSWTR**: Single-frame mode software trigger
This bit is set by software and cleared by hardware.
0: No action
1: Triggers one frame

Bit 24 **SFEN**: Single-frame mode enable
This bit is set and cleared by software.
0: Single-frame disabled: a trigger (on SFSWTR) generates a continuous flow.
1: Single-frame enabled: a trigger (on SFSWTR) generates a single frame.

Bits 23:20 Reserved, must be kept at reset value.

Bit 19 **CRCEN**: CRC enable
This bit is set and cleared by software.
0: CRC disabled
1: CRC enabled

Bits 18:17 Reserved, must be kept at reset value.

Bit 16 **DEN**: Dither enable
This bit is set and cleared by software.
0: Dither disabled
1: Dither enabled

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **DRW[2:0]**: Dither red width
This bitfield returns the dither red bits.

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **DGW[2:0]**: Dither green width
This bitfield returns the dither green bits.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **DBW[2:0]**: Dither blue width
This bitfield returns the dither Blue bits.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **GAMEN**: Gamma correction enable
This bit is set and cleared by software.
0: Gamma correction disabled (pixels bypass the gamma operator)
1: Gamma correction enabled

Bit 0 **LTDCEN**: LTDC global enable
This bit is set and cleared by software.
0: LTDC disabled
1: LTDC enabled

43.7.6 LTDC shadow reload configuration register (LTDC_SRCR)

Address offset: 0x024

Reset value: 0x0000 0000

This register allows the reload, either immediately or during the vertical blanking period, of shadow register values to the active registers. Shadow registers are all layer x ones, except LTDC_L1CLUTWR and LTDC_L2CLUTWR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBR	IMR
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **VBR**: Vertical blanking reload request
This bit is set by software and cleared only by hardware after reload (it cannot be cleared through register write once it is set).
0: No effect
1: The shadow registers are reloaded during the vertical blanking period (at the beginning of the first line after the active display area).

Bit 0 **IMR**: Immediate reload trigger
This bit is set by software and cleared only by hardware after reload.
0: No effect
1: The shadow registers are reloaded immediately.

43.7.7 LTDC gamma correction configuration register (LTDC_GCCR)

Address offset: 0x028

Reset value: 0x0000 0000

This register performs indirect access to the gamma correction R,G,B tables.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REN	GEN	BEN
													w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP[7:0]								ADDR[7:0]							
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **REN**: Write trigger to the red table
0: No action
1: COMP is written at ADDR in the red table.

Bit 17 **GEN**: Write trigger to the green table
0: No action
1: COMP is written at ADDR in the green table.

Bit 16 **BEN**: Write trigger to the blue table
0: No action
1: COMP is written at ADDR in the blue table.

Bits 15:8 **COMP[7:0]**: Color component to be written, in either (or all) the R,G,B tables

Bits 7:0 **ADDR[7:0]**: Address of the R,G,B table where the COMP component is written
When LTDC_GC1R.GCT=2, the gamma is implemented with eight interpolated segment. In that case, the valid ADDR addresses are: 0, 32, 64, 96, 128, 160, 192, 224, 255.
Note: For ADDR = 255, the gamma interpolation hardware considers that the address is 256.

43.7.8 LTDC background color configuration register (LTDC_BCCR)

Address offset: 0x02C

Reset value: 0x0000 0000

This register defines the background color (RGB888).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BCRED[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BCGREEN[7:0]								BCBLUE[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **BCRED[7:0]**: Background color red value
This bitfield configures the background red value.

Bits 15:8 **BCGREEN[7:0]**: Background color green value
This bitfield configures the background green value.

Bits 7:0 **BCBLUE[7:0]**: Background color blue value
This bitfield configures the background blue value.

43.7.9 LTDC interrupt enable register (LTDC_IER)

Address offset: 0x034

Reset value: 0x0000 0000

This register determines which status flags generate an interrupt request, by setting the corresponding bit to 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCIE	FUIE	Res.	Res.	RRIE	TERRIE	FUWIE	LIE
								rw	rw			rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **CRCIE**: CRC error interrupt enable
This bit is set and cleared by software.
0: CRC error disabled
1: CRC error interrupt enabled

Bit 6 **FUIE**: FIFO underrun interrupt enable
This bit is set and cleared by software.
0: FIFO underrun interrupt disabled
1: FIFO underrun Interrupt enabled

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **RRIE**: Register reload interrupt enable
This bit is set and cleared by software.
0: Register reload interrupt disabled
1: Register reload interrupt enabled

Bit 2 **TERRIE**: Transfer error interrupt enable
This bit is set and cleared by software.
0: Transfer error interrupt disabled
1: Transfer error interrupt enabled

Bit 1 **FUWIE**: FIFO underrun warning interrupt enable
This bit is set and cleared by software.
0: FIFO underrun interrupt disabled
1: FIFO underrun Interrupt enabled

Bit 0 **LIE**: Line interrupt enable
This bit is set and cleared by software.
0: line interrupt disabled
1: line interrupt enabled

43.7.10 LTDC interrupt status register (LTDC_ISR)

Address offset: 0x038

Reset value: 0x0000 0000

This register returns the interrupt status flag.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCIF	FUIF	Res.	Res.	RRIF	TERRIF	FUWIF	LIF
								r	r			r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **CRCIF**: CRC error interrupt flag
0: No CRC error interrupt generated
1: CRC error interrupt generated when a bus error occurs

Bit 6 **FUIF**: FIFO underrun interrupt flag
0: No FIFO underrun interrupt generated
1: FIFO underrun interrupt generated, if one of the layer FIFOs is empty and many pixel data are read from the FIFO

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **RRIF**: Register reload interrupt flag
0: No register reload interrupt generated
1: Register reload interrupt generated when a vertical blanking reload occurs (and the first line after the active area is reached)

Bit 2 **TERRIF**: Transfer error interrupt flag
0: No transfer error interrupt generated
1: Transfer error interrupt generated when a bus error occurs

Bit 1 **FUWIF**: FIFO underrun warning interrupt flag
0: No FIFO underrun warning interrupt generated
1: FIFO underrun warning interrupt generated, if one of the layer FIFO is empty and pixel data is read from the FIFO

Bit 0 **LIF**: Line interrupt flag
0: No line interrupt generated
1: Line interrupt generated when a programmed line is reached

43.7.11 LTDC interrupt clear register (LTDC_ICR)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCRCIF	CFUIF	Res.	Res.	CRRIF	CTERRIF	CFUWIF	CLIF
								w	w			w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **CCRCIF**: CRC error interrupt flag clear
0: No effect
1: Clear CRCIF flag in LTDC_ISR.

Bit 6 **CFUIF**: FIFO underrun interrupt flag clear
0: No effect
1: Clear FUIF flag in LTDC_ISR.

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **CRRIF**: Register reload interrupt flag clear
0: No effect
1: Clear RRIF flag in LTDC_ISR.

Bit 2 **CTERRIF**: Transfer error interrupt flag clear
0: No effect
1: Clear TERRIF flag in LTDC_ISR.

Bit 1 **CFUWIF**: FIFO underrun warning interrupt flag clear
0: No effect
1: Clear FUWIF flag in LTDC_ISR.

Bit 0 **CLIF**: Line-interrupt flag clear
0: No effect
1: Clear LIF flag in LTDC_ISR.

43.7.12 LTDC line interrupt position configuration register (LTDC_LIPCR)

Address offset: 0x040

Reset value: 0x0000 0000

This register defines the line interrupt position. The line value to be programmed depends on timing parameters (see *Figure 447*).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	LIPOS[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **LIPOS[11:0]**: Line interrupt position
This bitfield configures the line interrupt position.

43.7.13 LTDC current position status register (LTDC_CPSR)

Address offset: 0x044

Reset value: 0x0000 0000

This register returns the X,Y position of the currently displayed pixel.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	CXPOS[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	CYPOS[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **CXPOS[11:0]**: Current X position

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **CYPOS[11:0]**: Current Y position

43.7.14 LTDC current display status register (LTDC_CDSR)

Address offset: 0x048

Reset value: 0x0000 0003

This register returns the status of the current display phase that is controlled by HSYNC, VSYNC, and horizontal/vertical data-enable signals.

Example: if the current display phase is the vertical synchronization, VSYNCS is active high. If the current display phase is the horizontal synchronization, HSYNCS is active high.

The returned status does not depend on the configured polarity in LTDC_GCR: it returns the current active display phase.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSYNCS	VSYNCS	HDES	VDES
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **HSYNCS**: Horizontal synchronization display status
0: Active low
1: Active high

Bit 2 **VSYNCS**: Vertical synchronization display status
0: Active low
1: Active high

Bit 1 **HDES**: Horizontal data enable display status
0: Active low
1: Active high

Bit 0 **VDES**: Vertical data enable display status
0: Active low
1: Active high

43.7.15 LTDC external display control register (LTDC_EDCR)

Address offset: 0x060

Reset value: 0x0000 0000

This register controls the external display interface.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	OCYCO	OCYSEL	OCYEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
				rw	rw	rw									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **OCYCO**: Output conversion to YCbCr 422
This bit defines the chrominance order (whether Cb or Cr is output first).
0: Cb is output first (Y0Cb, then Y1Cr, Y2Cb and so on).
1: Cr is output first (Y0Cr, then Y1Cb, Y2Cr and so on).

Bit 26 **OCYSEL**: Output conversion to YCbCr 422
This bit selects the set of CCIR hard-wired coefficients (ITU-R BT.601 or 709).
0: Use ITU-R BT.601 set (for typically SDTV analog-like displays).
1: Use ITU-R BT.709 set (for typically HDTV digital-like displays).

Bit 25 **OCYEN**: Output conversion to YCbCr 422 enable
0: Conversion disabled
1: Conversion enabled

Bits 24:0 Reserved, must be kept at reset value.

43.7.16 LTDC interrupt enable register 2 (LTDC_IER2)

Address offset: 0x064

Reset value: 0x0000 0000

This register determines which status flags generate an interrupt request, by setting the corresponding bit to 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCIE	FUIE	Res.	Res.	RRIE	TERRIE	FUWIE	LIE
								rw	rw			rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **CRCIE**: CRC error interrupt enable
This bit is set and cleared by software.
0: CRC error disabled
1: CRC error interrupt enabled

Bit 6 **FUIE**: FIFO underrun interrupt enable
This bit is set and cleared by software.
0: FIFO underrun interrupt disabled
1: FIFO underrun Interrupt enabled

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **RRIE**: Register reload interrupt enable
This bit is set and cleared by software.
0: Register reload interrupt disabled
1: Register reload interrupt enabled

Bit 2 **TERRIE**: Transfer error interrupt enable
This bit is set and cleared by software.
0: Transfer error interrupt disabled
1: Transfer error interrupt enabled

Bit 1 **FUWIE**: FIFO underrun warning interrupt enable
This bit is set and cleared by software.
0: FIFO underrun interrupt disabled
1: FIFO underrun Interrupt enabled

Bit 0 **LIE**: Line interrupt enable
This bit is set and cleared by software.
0: Line interrupt disabled
1: Line interrupt enabled

43.7.17 LTDC interrupt status register 2 (LTDC_ISR2)

Address offset: 0x068

Reset value: 0x0000 0000

This register returns the interrupt status flag.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCIF	FUIF	Res.	Res.	RRIF	TERRIF	FUWIF	LIF
								r	r			r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **CRCIF**: CRC Error interrupt flag
0: No CRC error interrupt generated
1: CRC error interrupt generated when a bus error occurs

Bit 6 **FUIF**: FIFO underrun interrupt flag
0: No FIFO underrun interrupt generated.
1: FIFO underrun interrupt generated, if one of the layer FIFO is empty and many pixel data are read from the FIFO

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **RRIF**: Register reload interrupt flag
0: No register reload interrupt generated
1: Register reload interrupt generated when a vertical blanking reload occurs (and the first line after the active area is reached)

Bit 2 **TERRIF**: Transfer error interrupt flag
0: No transfer error interrupt generated
1: Transfer error interrupt generated when a bus error occurs

Bit 1 **FUWIF**: FIFO underrun warning interrupt flag
0: No FIFO underrun warning interrupt generated.
1: FIFO underrun warning interrupt generated, if one of the layer FIFO is empty and pixel data is read from the FIFO

Bit 0 **LIF**: Line interrupt flag
0: No line interrupt generated
1: Line interrupt generated when a programmed line is reached

43.7.18 LTDC interrupt clear register 2 (LTDC_ICR2)

Address offset: 0x06C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCRCIF	CFUIF	Res.	Res.	CRRIF	CTERRIF	CFUWIF	CLIF
								w	w			w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **CCRCIF**: Clear the CRC error interrupt flag
0: No effect
1: Clear the CRCIF flag in LTDC_ISR2.

Bit 6 **CFUIF**: Clear the FIFO underrun interrupt flag
0: No effect
1: Clear the FUIF flag in LTDC_ISR2.

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **CRRIF**: Clear register reload interrupt flag
0: No effect
1: Clear the RRIF flag in LTDC_ISR2.

Bit 2 **CTERRIF**: Clear the transfer error interrupt flag
0: No effect
1: Clear the TERRIF flag in LTDC_ISR2.

Bit 1 **CFUWIF**: Clear the FIFO underrun warning interrupt flag
0: No effect
1: Clear the FUWIF flag in LTDC_ISR2.

Bit 0 **CLIF**: Clear the line interrupt flag
0: No effect
1: Clear the LIF flag in LTDC_ISR2.

43.7.19 LTDC line interrupt position configuration register 2 (LTDC_LIPCR2)

Address offset: 0x070

Reset value: 0x0000 0000

This register defines the line interrupt position. The line value to be programmed depends on timing parameters (see *Figure 447*).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	LIPOS[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **LIPOS[11:0]**: Line interrupt position

43.7.20 LTDC expected CRC register (LTDC_ECRCR)

Address offset: 0x078

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ECRC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ECRC[15:0]**: Expected CRC of frame

43.7.21 LTDC computed CRC register (LTDC_CCRCR)

Address offset: 0x07C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CCRC[15:0]**: Computed CRC of frame

43.7.22 LTDC FIFO underrun threshold register (LTDC_FUTR)

Address offset: 0x090

Reset value: 0x0000 0010

A default value for THRE[15:0] is 0x80, that represents 128 words of 64 bits, thus 256 pixels 32bpp or 1024 luminance of a YUV420 pixel.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
THRE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **THRE[15:0]**: Threshold to trigger a FIFO underrun interrupt (per FIFO word, 64 bits)
Threshold above which a FIFO underrun warning becomes a FIFO underrun error.

43.7.23 LTDC layer x configuration 0 register (LTDC_LxC0R)

Address offset: 0x100 + 0x100 * (x - 1), (x = 1 to 2)

Reset value: 0xFF50 A075

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARGB8888	ABGR8888	RGBA8888	BGRA8888	RGB565	BGR565	RGB888	FF	F11PC	F1PC	F11C	F1C	F11P	F1P	F10	F11
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F21PC	F2PC	F21C	F2C	F21P	F2P	F20	F21	CKRA	CLUTA	WINA	DCP	APA	CFBPA	CFBDA	CKTA
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **ARGB8888**: Pixel format, ability for argb8888

Bit 30 **ABGR8888**: Pixel format, ability for abgr8888

Bit 29 **RGBA8888**: Pixel format, ability for rgba8888

Bit 28 **BGRA888**: Pixel format, ability for bgra8888

Bit 27 **RGB565**: Pixel format, ability for rgb565

Bit 26 **BGR565**: Pixel format, ability for bgr565

Bit 25 **RGB888**: Pixel format, ability for rgb888

Bit 24 **FF**: Flexible pixel format, ability

Bit 23 **F11PC**: Blending factor 1, ability for 1.0 - (pixel_alpha * constant_alpha)

Bit 22 **F1PC**: Blending factor 1, ability for pixel_alpha * constant_alpha

Bit 21 **F11C**: Blending factor 1, ability for 1.0 - constant_alpha

Bit 20 **F1C**: Blending factor 1, ability for constant_alpha

Bit 19 **F11P**: Blending factor 1, ability for 1.0 - pixel_alpha

Bit 18 **F1P**: Blending factor 1, ability for pixel_alpha

Bit 17 **F10**: Blending factor 1,ability for 0.0

Bit 16 **F11**: Blending factor 1, ability for 1.0

Bit 15 **F21PC**: Blending factor 2, ability for 1.0 - (pixel_alpha * constant_alpha)

Bit 14 **F2PC**: Blending factor 2, ability for pixel_alpha * constant_alpha

Bit 13 **F21C**: Blending factor 2, ability for 1.0 - constant_alpha

Bit 12 **F2C**: Blending factor 2, ability for constant_alpha

Bit 11 **F21P**: Blending factor 2, ability for 1.0 - pixel_alpha

Bit 10 **F2P**: Blending factor 2, ability for pixel_alpha

Bit 9 **F20**: Blending factor 2, ability for 0.0

Bit 8 **F21**: Blending factor 2, ability for 1.0

Bit 7 **CKRA**: Color key replace ability

Bit 6 **CLUTA**: CLUT ability

Bit 5 **WINA**: Windowing ability

Bit 4 **DCP**: Default color programmability

Bit 3 **APA**: Alpha plane ability

Bit 2 **CFBPA**: Color frame buffer pitch ability

Bit 1 **CFBDA**: Color frame buffer duplication ability

Bit 0 **CKTA**: Color key transparency ability

43.7.24 LTDC layer x configuration 1 register (LTDC_LxC1R)

Address offset: 0x104 + 0x100 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0007, 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SCA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	YFPA	YSPA	YIA
													r	r	r

Bit 31 **SCA**: Scaling ability for this layer
- 0: Scaling not available
- 1: Scaling available

Bits 30:3 Reserved, must be kept at reset value.

Bit 2 **YFPA**: YCbCr 420 full-planar ability for this layer
- 0: Full planar not available
- 1: Full planar available

Bit 1 **YSPA**: YCbCr 420 semi-planar ability for this layer
- 0: Semi-planar not available
- 1: Semi-planar available

Bit 0 **YIA**: YCbCr 422 interleaved ability for this layer
- 0: Interleaved not available
- 1: Interleaved available

43.7.25 LTDC layer x reload control register (LTDC_LxRCR)

Address offset: 0x108 + 0x100 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GRMSK	VBR	IMR
													rw	rw	w

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **GRMSK**: Shadow reload control, global (centralized) reload masked
This bit is set and cleared by software.
0: Global reload masked for this layer (control from LTDC_SRCR disabled)
1: Global reload active for this layer (control from LTDC_SRCR enabled)

Bit 1 **VBR**: Vertical blanking reload request
This bit is set by software and cleared only by hardware after reload (it cannot be cleared through register write once it is set).
0: No effect
1: Shadow registers are reloaded during the vertical blanking period (at the beginning of the first line after the active display area).

Bit 0 **IMR**: Immediate reload trigger
This bit is set by software and cleared only by hardware after reload.
0: No effect
1: Shadow registers are reloaded immediately.

43.7.26 LTDC layer x control register (LTDC_LxCR)

Address offset: 0x10C + 0x100 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	DCBEN	HMEN	Res.	Res.	Res.	CLUTEN	Res.	Res.	CKEN	LEN
						rw	rw				rw			rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **DCBEN**: Default color blending enable
This bit is set and cleared by software.
0: Blending disabled
1: Blending enabled

Bit 8 **HMEN**: Horizontal mirroring enable
This bit is set and cleared by software.
0: Mirror disabled
1: Mirror enabled (if so, the color frame buffer start address has to be set to the last byte of the first line, so for instance: if line is 100 pixels, 24 bpp, then address is set to 299)

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **CLUTEN**: Color look-up table enable
This bit is set and cleared by software.
0: Color look-up table disabled
1: Color look-up table enabled
The CLUT is only meaningful for L8, AL44 and AL88 pixel format.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **CKEN**: Color keying enable
This bit is set and cleared by software.
0: Color keying disabled
1: Color keying enabled: if RGB matches, then the ARGB are set to 0.

Bit 0 **LEN**: Layer enable
This bit is used to enable/disable the presence of this whole layer. it is set and cleared by software.
0: Layer disabled
1: Layer enabled

43.7.27 LTDC layer x window horizontal position configuration register (LTDC_LxWHPCR)

Address offset: 0x110 + 0x100 * (x -1), (x = 1 to 2)

Reset value: 0x0000 0000

This register defines the horizontal position (first and last pixel) of the window of layer 1/2.

The first visible pixel of a line is the programmed value of AHBP[11:0] + 1 in LTDC_BPCR.

The last visible pixel of a line is the programmed value of AAW[11:0] in LTDC_AWCR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	WHSPPOS[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	WHSTPOS[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **WHSPPOS[11:0]**: Window horizontal stop position
This bitfield configures the last visible pixel of a line of the layer window.
WHSPPOS[11:0] must be ≥ AHBP[11:0] bits + 1 (programmed in LTDC_BPCR).

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **WHSTPOS[11:0]**: Window horizontal start position
This bitfield configures the first visible pixel of a line of the layer window.
WHSTPOS[11:0] must be ≤ AAW[11:0] bits (programmed in LTDC_AWCR).

43.7.28 LTDC layer x window vertical position configuration register (LTDC_LxWVPCR)

Address offset: 0x114 + 0x100 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

This register defines the vertical position (first and last line) of the layer 1 or 2 window.

The first visible line of a frame is the programmed value of AVBP[11:0] + 1 in LTDC_BPCR.

The last visible line of a frame is the programmed value of AAH[11:0] in LTDC_AWCR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	WVSPPOS[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	WVSTPOS[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **WVSPPOS[11:0]**: Window vertical stop position
This bitfield configures the last visible line of the layer window.
WVSPPOS[11:0] must be ≥ AVBP[11:0] + 1 (programmed in LTDC_BPCR).

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **WVSTPOS[11:0]**: Window vertical start position
This bitfield configures the first visible line of the layer window.
WVSTPOS[11:0] must be ≤ AAH[11:0] (programmed in LTDC_AWCR).

43.7.29 LTDC layer x color keying configuration register (LTDC_LxCKCR)

Address offset: 0x118 + 0x100 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

This register defines the color key value (RGB), that is used by the color keying.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CKRED[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CKGREEN[7:0]								CKBLUE[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **CKRED[7:0]**: Color key red value

Bits 15:8 **CKGREEN[7:0]**: Color key green value

Bits 7:0 **CKBLUE[7:0]**: Color key blue value

43.7.30 LTDC layer x pixel format configuration register (LTDC_LxPFCR)

Address offset: 0x11C + 0x100 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

This register defines the pixel format that is used for the stored data in the frame buffer of a layer. The pixel data is read from the frame buffer and then transformed to the internal format 8888 (ARGB). When components are provided with less than 8 bits per component, they get expanded to 8 bits by replication of their MSB onto their LSB.

When the Alpha value is not defined in the input pixel format, Alpha is assumed for farther needs to be equal to 255 (opaque layer).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PF[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **PF[2:0]**: Pixel format

This bitfield configures the pixel format.
000: ARGB8888 (32 bpp)
001: ABGR8888 (32 bpp)
010: RGBA8888 (32 bpp)
011: BGRA8888 (32 bpp)
100: RGB565 (16 bpp, A = 255)
101: BGR565 (16 bpp, A = 255)
110: RGB888 (24 bpp packed, A = 255)
111: Flexible pixel format selected (see *Section 43.7.46* and *Section 43.7.47*)

43.7.31 LTDC layer x constant alpha configuration register (LTDC_LxCACR)

Address offset: 0x120 + 0x100 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 00FF

This register defines the constant alpha value (divided by 255 by hardware), that is used in the alpha blending (see LTDC_LxBFCR).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CONSTA[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **CONSTA[7:0]**: Constant alpha

This bitfield configures the constant alpha used for blending. The constant alpha is divided by 255 by hardware.

Example: if the programmed constant alpha is 0xFF, the floating alpha value is 255 / 255 = 1.

43.7.32 LTDC layer x default color configuration register (LTDC_LxDCCR)

Address offset: 0x124 + 0x100 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

This register defines the default color of a layer in the format ARGB. The default color is used outside the defined layer window or when a layer is disabled. The reset value of 0x00000000 defines a transparent black color.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DCALPHA[7:0]								DCRED[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DCGREEN[7:0]								DCBLUE[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **DCALPHA[7:0]**: Default color alpha

This bitfield configures the default alpha value.

Bits 23:16 **DCRED[7:0]**: Default color red

This bitfield configures the default red value.

Bits 15:8 **DCGREEN[7:0]**: Default color green

This bitfield configures the default green value.

Bits 7:0 **DCBLUE[7:0]**: Default color blue

This bitfield configures the default blue value.

43.7.33 LTDC layer x blending factors configuration register (LTDC_LxBFCR)

Address offset: 0x128 + 0x100 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0607, 0x0001 0607,

This register defines the blending order, and the factors BF1 and BF2. The constant alpha value is the programmed value in LTDC_LxCACR, divided by 255 by hardware.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOR
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	BF1[2:0]			Res.	Res.	Res.	Res.	Res.	BF2[2:0]		
					rw	rw	rw						rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **BOR**: Default order
This bit defines the blending order of the layer. If two layers have the same BOR value, the layer with the highest index is on the foreground.
0: layer set in background
1: layer set in foreground

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:8 **BF1[2:0]**: Default factor 1
This bitfield selects the blending factor F1.
100: Constant alpha
110: Pixel alpha x constant alpha
Others: Reserved

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 **BF2[2:0]**: Blending factor 2
This bitfield selects the blending factor F2.
101: 1 - constant alpha
111: 1 - (pixel alpha x constant alpha)
Others: Reserved

43.7.34 LTDC layer x burst length configuration register (LTDC_LxBLCR)

Address offset: 0x12C + 0x100 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

This register defines the length of the bursts requested by layer x.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BL[4:0]				
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **BL[4:0]**: Burst length
0x00: maximum burst length (16 words 64 bits, thus 128 Bytes)
0x01: 1 word (of 64 bits) per burst
0x10: 16 words (of 64 bits) per burst
Others: Reserved

43.7.35 LTDC layer x planar configuration register (LTDC_LxPCR)

Address offset: 0x130 + 0x100 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

This register defines, for layer x, the configuration when accessing planar or semi-planar buffers.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	YREN	OF	CBF	YF	YCM[1:0]		YCEN	Res.	Res.	Res.
						rw	rw	rw	rw	rw	rw	rw			

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **YREN**: Y rescale enable for the color dynamic range
When enabled, incoming Y values in range 16 to 235, are re-scaled to range 0 to 255.
0: Rescaling disabled (input component thus assumed provided in 0 to 255)
1: Rescaling enabled (input component thus assumed provided in 16 to 235).

Bit 8 **OF**: Odd pixel first
This bit defines if the byte 0 of a word (in LSB) contains the odd pixel.
0: Odd pixel disabled (thus even pixel on byte 0)
1: Odd pixel enabled (thus odd pixel on byte 0)

Bit 7 **CBF**: Cb component first
This bit defines if the byte 0 and 1 of a word (in LSB) contains the Cb component. The setting impacts only for interleaved and semi-planar modes, as it has no meaning in full-planar mode.
0: Cb disabled (thus Cr component is on byte 0 and 1)
1: Cb enabled (thus Cb component is on byte 0 and 1)

Bit 6 **YF**: Y component first
This bit defines if the byte 0 of a word (in LSB) contains the Y component.
0: Y component disabled (thus Cr or Cb component is on byte 0)
1: Y component enabled (thus Y component is on byte 0)

Bits 5:4 **YCM[1:0]**: YCbCr conversion mode
This bitfield defines the type of input that is considered and converted to a YCbCr 444.
00: Interleaved 422 (Cb and Cr component are replicated horizontally for pixels P and P+1)
01: Semi-Planar 420: (Cb and Cr component are replicated horizontally and vertically.The layer main configuration defines the access to the Y buffer, and auxiliary registers define the access to the Cb and Cr buffers).
10: Full-Planar 420: (Cb and Cr component are replicated horizontally and vertically. The layer main configuration defines the access to the Y buffer, and auxiliary registers define the access to the Cb and Cr buffers).
11: Reserved

Bit 3 **YCEN**: YCbCr-to-RGB conversion enable
0: Conversion disabled
1: YCbCr conversion enabled, using the YCM setting above

Bits 2:0 Reserved, must be kept at reset value.

43.7.36 LTDC layer x color frame buffer address register (LTDC_LxCFBAR)

Address offset: 0x134 + 0x100 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

This register defines the color frame buffer start address, which has to point to the address where the pixel data of the top-left pixel of a layer is stored in the frame buffer.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CFBADD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CFBADD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CFBADD[31:0]**: Color frame buffer start address

This bitfield defines the color frame buffer start address.

43.7.37 LTDC layer x color frame buffer length register (LTDC_LxCFBLR)

Address offset: 0x138 + 0x100 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

This register defines the color frame buffer line length and pitch.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	CFBP[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CFBLL[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value. Upon reading, this bit returns the value of the bit 30.

Bits 30:16 **CFBP[14:0]**: Color frame buffer pitch in bytes

This bitfield defines the pitch that is the increment from the start of one line of pixels to the start of the next line in bytes.

Negative values (with MSB bit = 1) are allowed, to read the buffer from bottom to top, and thus to flip is vertically. When vertically flipped, as the address register must provide the address of the first line to be read, the address register must point to the start of the bottom line of the buffer.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:0 **CFBLL[13:0]**: Color frame buffer line length

This bitfield defines the length of one line of pixels in bytes + 7.

The line length is computed as follows: active high width * number of bytes per pixel + 7.

43.7.38 LTDC layer x color frame buffer line number register (LTDC_LxCFBLNR)

Address offset: 0x13C + 0x100 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

This register defines the number of lines in the color frame buffer. The number of lines and line-length settings define how much data are fetched per frame for every layer. If it is configured to less bytes than required, a FIFO underrun interrupt is generated if enabled. The start address and pitch settings define the correct start of every line in memory.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	CFBLNBR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **CFBLNBR[11:0]**: Frame buffer line number

This bitfield defines the number of lines in the frame buffer that corresponds to the active high width.

43.7.39 LTDC layer 1 auxiliary frame buffer address 0 register (LTDC_L1AFBA0R)

Address offset: 0x140

Reset value: 0x0000 0000

This register defines the start address for the auxiliary 0 frame buffer. It has to point to the address where the pixel data of the top left pixel of a layer is stored in the frame buffer. The auxiliary 0 and 1 frame buffers are for YCbCr semi-planar or full-planar pixel formats: the color frame buffer is used to store the Y components, while the auxiliary buffers 0 and 1 are used to store the other one (for semi-planar) or two (for full-planar) planes of components.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFBADD0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFBADD0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AFBADD0[31:0]**: Frame buffer start address

This bitfield defines the color frame buffer start address.

43.7.40 LTDC layer 1 auxiliary frame buffer address 1 register (LTDC_L1AFBA1R)

Address offset: 0x144

Reset value: 0x0000 0000

This register defines the start address for the auxiliary 1 frame buffer. It has to point to the address where the pixel data of the top left pixel of a layer is stored in the frame buffer (the auxiliary 0 and 1 frame buffers are for YCbCr semi-planar or full-planar pixel formats).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFBADD1[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AFBADD1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AFBADD1[31:0]**: Auxiliary frame buffer start address
This bitfield defines the color frame buffer start address.

43.7.41 LTDC layer 1 auxiliary frame buffer length register (LTDC_L1AFBLR)

Address offset: 0x148

Reset value: 0x0000 0000

This register defines the auxiliary 0 and 1 frame buffer line length and pitch
(the auxiliary0 and 1 frame buffers are for YCbCr semi-planar or full-planar pixel formats).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	AFBP[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	AFBLL[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value. Upon reading, this bit returns the value of the bit 30.

Bits 30:16 **AFBP[14:0]**: Auxiliary frame buffer pitch in bytes
This bitfield defines the pitch that is the increment from the start of one line of pixels to the start of the next line in bytes.
Negative values (with MSB bit = 1) are allowed, to read the buffer from bottom to top, and thus to flip is vertically. When vertically flipped, as the address register must provide the address of the first line to be read, the address register must point to the start of the bottom line of the buffer.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:0 **AFBLL[13:0]**: Auxiliary frame buffer line length
This bitfield defines the length of one line of pixels in bytes + 7.
The line length is computed as follows:
active high width * number of bytes per pixel + 7.

43.7.42 LTDC layer 1 auxiliary frame buffer line number register (LTDC_L1AFBLNR)

Address offset: 0x14C

Reset value: 0x0000 0000

This register defines the number of lines in the auxiliary 0 and 1 frame buffers (the auxiliary 0 and 1 frame buffers are for YCbCr semi-planar or full-planar pixel formats). The number of lines and line length settings define how much data is fetched per frame for every layer. If it is configured to less bytes than required, a FIFO underrun interrupt is generated if enabled. The start address and pitch settings define the correct start of every line in memory.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	AFBLNBR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **AFBLNBR[11:0]**: Auxiliary frame buffer line number

This bitfield defines the number of lines in the auxiliary frame buffer that corresponds to the active high width.

43.7.43 LTDC layer x CLUT write register (LTDC_LxCLUTWR)

Address offset: 0x150 + 0x100 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

This register defines the CLUT address and the RGB value. The CLUT write register must only be configured during blanking period or if the layer is disabled. The CLUT can be enabled or disabled in LTDC_LxCR. The CLUT is only meaningful for L8, AL44, and AL88 pixel format.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CLUTADD[7:0]								RED[7:0]							
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GREEN[7:0]								BLUE[7:0]							
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:24 **CLUTADD[7:0]**: CLUT address

This bitfield configures the CLUT address (color position within the CLUT) of each RGB value.

Bits 23:16 **RED[7:0]**: Red value

This bitfield configures the red value.

Bits 15:8 **GREEN[7:0]**: Green value

This bitfield configures the green value.

Bits 7:0 **BLUE[7:0]**: Blue value
This bitfield configures the blue value.

43.7.44 LTDC layer x conversion YCbCr RGB 0 register (LTDC_LxCYR0R)

Address offset: 0x16C + 0x100 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

This register defines the coefficients to convert from YCbCr to RGB.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	CB2B[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CR2R[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **CB2B[9:0]**: Cb-to-Blue coefficient, with bits 9:8 as positive integer and 7:0 as decimals.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:0 **CR2R[9:0]**: Cr-to-Red coefficient, with bits 9:8 as positive integer and 7:0 as decimals.

43.7.45 LTDC layer x conversion YCbCr RGB 1 register (LTDC_LxCYR1R)

Address offset: 0x170 + 0x100 * (x - 1), (x = 1 to 2)

Reset value: 0x0000 0000

This register defines the coefficients to convert from YCbCr to RGB.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	CB2G[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CR2G[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **CB2G[9:0]**: Cb-to-Green coefficient, with bits 9:8 as positive integer and 7:0 as decimals.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:0 **CR2G[9:0]**: Cr-to-Green coefficient, with bits 9:8 as positive integer and 7:0 as decimals.

43.7.46 LTDC layer x flexible pixel format 0 register (LTDC_LxFPF0R)

Address offset: 0x174 + 0x100 * (x - 1), (x = 1 to 2)

Reset value: 0x0002 1100

This register is one of the registers that define the flexible color format, by specifying the location and width of all A, R, G, B components in a memory word. Components A and R are defined in this specific register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RLEN[3:2]	
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RLEN[1:0]		RPOS[4:0]					ALEN[3:0]				APOS[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:14 **RLEN[3:0]**: Width of the red component (in bits)

Bits 13:9 **RPOS[4:0]**: Location of the red component inside the pixel memory word (in bits)

Bits 8:5 **ALEN[3:0]**: Width of the alpha component (in bits)

Bits 4:0 **APOS[4:0]**: Location of the alpha component inside the pixel memory word (in bits)

43.7.47 LTDC layer x flexible pixel format 1 register (LTDC_LxFPF1R)

Address offset: 0x178 + 0x100 * (x - 1), (x = 1 to 2)

Reset value: 0x0012 3110

This register is one of the registers that define the flexible color format, by specifying the location and width of all A, R, G, B components in a memory word. Components G and B are defined in this specific register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSIZE[2:0]			BLEN[3:2]	
											rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BLEN[1:0]		BPOS[4:0]					GLEN[3:0]				GPOS[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:18 **PSIZE[2:0]**: Pixel size (in bytes)

Bits 17:14 **BLEN[3:0]**: Width of the blue component (in bits)

Bits 13:9 **BPOS[4:0]**: Location of the blue component inside the pixel memory word (in bits)

Bits 8:5 **GLEN[3:0]**: Width of the green component (in bits)

Bits 4:0 **GPOS[4:0]**: Location of the green component inside the pixel memory word (in bits)

43.7.48 LTDC register map

Table 387. LTDC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x008	LTDC_SSCR	Res.	Res.	Res.	Res.	Res.	HSW[11:0]											Res.	Res.	Res.	Res.	VSH[11:0]											
	Reset value						0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x00C	LTDC_BPCR	Res.	Res.	Res.	Res.	AHBP[11:0]												Res.	Res.	Res.	Res.	AVBP[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x010	LTDC_AWCR	Res.	Res.	Res.	Res.	AAW[11:0]												Res.	Res.	Res.	Res.	AAH[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x014	LTDC_TWCR	Res.	Res.	Res.	Res.	TOTALW[11:0]												Res.	Res.	Res.	Res.	TOTALH[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x018	LTDC_GCR	HSPOL	VSPOL	DEPOL	PCPOL	Res.	Res.	SFSWTR	SFEN	Res.	Res.	Res.	Res.	CRCEN	Res.	Res.	DEN	Res.	DRW[2:0]			Res.	DGW[2:0]			Res.	DBW[2:0]			Res.	Res.	GAMEN	LTDCEN
	Reset value	0	0	0	0			0	0					0			0		0	1	0		0	1	0		0	1	0			0	0
0x024	LTDC_SRCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBR	IMR
	Reset value																															0	0
0x028	LTDC_GCCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REN	GEN	BEN	COMP[7:0]								ADDR[7:0]							
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	LTDC_BCCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BCRED[7:0]								BCGREEN[7:0]								BCBLUE[7:0]							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	Reserved	Reserved																															
0x034	LTDC_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCIE	FUIE	Res.	Res.	RRIE	TERRIE	FUWIE	LIE
	Reset value																									0	0			0	0	0	0
0x038	LTDC_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCIF	FUIF	Res.	Res.	RRIF	TERRIF	FUWIF	LIF
	Reset value																									0	0			0	0	0	0
0x03C	LTDC_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCRCIF	CFUIF	Res.	Res.	CRRIF	CTERRIF	CFUWIF	CLIF
	Reset value																									0	0			0	0	0	0
0x040	LTDC_LIPCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LIPOS[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x044	LTDC_CPSR	Res.	Res.	Res.	Res.	CXPOS[11:0]												Res.	Res.	Res.	Res.	CYPOS[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x048	LTDC_CDSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSYNCS	VSYNCS	HDES	VDES
	Reset value																													0	0	1	1
0x04C-0x05C	Reserved	Reserved																															

Table 387. LTDC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x060	LTDC_EDCR	Res.	Res.	Res.	Res.	OCYCO	OCYSEL	OCYEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value					0	0	0																									
0x064	LTDC_IE2R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCIE	FUIE	Res.	Res.	RRIE	TERRIE	FUWIE	LIE
	Reset value																									0	0			0	0	0	0
0x068	LTDC_ISR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CRCIF	FUIF	Res.	Res.	RRIF	TERRIF	FUWIF	LIF
	Reset value																									0	0			0	0	0	0
0x06C	LTDC_ICR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCRCIF	CFUIF	Res.	Res.	CRRIF	CTERRIF	CFUWIF	CLIF
	Reset value																									0	0			0	0	0	0
0x070	LTDC_LIPCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LIPOS[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x074	Reserved	Reserved																															
0x078	LTDC_ECRCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ECRC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07C	LTDC_CCRCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCRC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x080-0x08C	Reserved	Reserved																															
0x090	LTDC_FUTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	THRE[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x094-0x0FC	Reserved	Reserved																															
0x100	LTDC_L1C0R	ARGB8888	ABGR8888	RGBA8888	BGRA888	RGB565	BGR565	RGB888	FF	F11PC	F1PC	F11C	F1C	F11P	F1P	F10	F11	F21PC	F2PC	F21C	F2C	F21P	F2P	F20	F21	CKRA	CLUTA	WINA	DCP	APA	CFBPA	CFBDA	CKTA
	Reset value	1	1	1	1	1	1	1	1	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	1	1	1	0	1	0	1
0x104	LTDC_L1C1R	SCA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	YFPA	YSP	YIA
	Reset value	0																													1	1	1
0x108	LTDC_L1RCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GRMSK	VBR	IMR
	Reset value																														1	0	0
0x10C	LTDC_L1CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCBEN	HMEN	Res.	Res.	Res.	CLUTEN	HPDEN	VPDEN	CKEN	LEN
	Reset value																							0	0				0	0	0	0	0
0x110	LTDC_L1WHPCR	Res.	Res.	Res.	Res.	WHSPPOS[11:0]												Res.	Res.	Res.	Res.	Res.	WHSTPOS[11:0]										
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x114	LTDC_L1WVPCR	Res.	Res.	Res.	Res.	WVSPPOS[11:0]												Res.	Res.	Res.	Res.	WVSTPOS[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0

Table 387. LTDC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x118	LTDC_L1CKCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CKRED[7:0]								CKGREEN[7:0]								CKBLUE[7:0]							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x11C	LTDC_L1PFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PF[2:0]		
	Reset value																														0	0	0
0x120	LTDC_L1CACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CONSTA[7:0]							
	Reset value																									1	1	1	1	1	1	1	1
0x124	LTDC_L1DCCR	DCALPHA[7:0]								DCRED[7:0]								DCGREEN[7:0]								DCBLUE[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x128	LTDC_L1BFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOR	Res.	Res.	Res.	Res.	Res.	BF1[2:0]			Res.	Res.	Res.	Res.	Res.	BF2[2:0]		
	Reset value																0						1	1	0						1	1	1
0x12C	LTDC_L1BLCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BL[4:0]				
	Reset value																												0	0	0	0	0
0x130	LTDC_L1PCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	YREN	OF	CBF	YF	YCM [1:0]		YCEN	Res.	Res.	Res.
	Reset value																							0	0	0	0	0	0	0			
0x134	LTDC_L1CFBAR	CFBADD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x138	LTDC_L1CFBLR	Res.	CFBP[14:0]															Res.	Res.	CFBLL[13:0]													
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x13C	LTDC_L1CFBLNR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFBLNBR[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x140	LTDC_L1AFBA0R	AFBADD0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x144	LTDC_L1AFBA1R	AFBADD1[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x148	LTDC_L1AFBLR	Res.	AFBP[14:0]															Res.	Res.	AFBLL[13:0]													
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14C	LTDC_L1AFBLNR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AFBLNBR[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x150	LTDC_ L1CLUTWR	CLUTADD[7:0]								RED[7:0]								GREEN[7:0]								BLUE[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x154-0x168	Reserved	Reserved																															
0x16C	LTDC_L1CYR0R	Res.	Res.	Res.	Res.	Res.	Res.	CB2B[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	CR2R[9:0]									
	Reset value							0	0	0	0	0	0	0	0	0	0							0	0	0	0	0	0	0	0	0	0
0x170	LTDC_L1CYR1R	Res.	Res.	Res.	Res.	Res.	Res.	CB2G[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	CR2G[9:0]									
	Reset value							0	0	0	0	0	0	0	0	0	0							0	0	0	0	0	0	0	0	0	0
0x174	LTDC_L1FPF0R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RLEN[3:0]				RPOS[4:0]						ALEN[3:0]				APOS[4:0]			
	Reset value															1	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0
0x178	LTDC_L1FPF1R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSIZE [2:0]			BLEN[3:0]				BPOS[4:0]						GLEN[3:0]				GPOS[4:0]			
	Reset value												1	0	0	1	0	0	0	1	1	0	0	0	1	0	0	0	1	0	0	0	0

Table 387. LTDC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x17C-0x1FC	Reserved	Reserved																															
0x200	LTDC_L2C0R	ARGB8888	ABGR8888	RGBA8888	BGRA888	RGB565	BGR565	RGB888	FF	F11PC	F1PC	F11C	F1C	F11P	F1P	F10	F11	F21PC	F2PC	F21C	F2C	F21P	F2P	F20	F21	CKRA	CLUTA	WINA	DCP	APA	CFBPA	CFBDA	CKTA
	Reset value	1	1	1	1	1	1	1	1	0	1	0	1	0	0	0	0	1	0	1	0	0	0	0	0	0	1	1	1	0	1	0	1
0x204	LTDC_L2C1R	SCA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	YFPA	YSP	YIA
	Reset value	0																													0	0	1
0x208	LTDC_L2RCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GRMSK	VBR	IMR
	Reset value																														1	0	0
0x20C	LTDC_L2CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCBEN	HMEN	Res.	Res.	Res.	CLUTEN	HPDEN	VPDEN	CKEN	LEN
	Reset value																							0	0				0	0	0	0	0
0x210	LTDC_L2WHPCR	Res.	Res.	Res.	Res.	WHSPPOS[11:0]												Res.	Res.	Res.	Res.	Res.	WHSTPOS[11:0]										
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0						0	0	0	0	0	0	0	0	0	0	0
0x214	LTDC_L2WVPCR	Res.	Res.	Res.	Res.	WVSPPOS[11:0]												Res.	Res.	Res.	Res.	WVSTPOS[11:0]											
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x218	LTDC_L2CKCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CKRED[7:0]								CKGREEN[7:0]								CKBLUE[7:0]							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x21C	LTDC_L2PFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PF[2:0]		
	Reset value																														0	0	0
0x220	LTDC_L2CACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CONSTA[7:0]							
	Reset value																									1	1	1	1	1	1	1	1
0x224	LTDC_L2DCCR	DCALPHA[7:0]								DCRED[7:0]								DCGREEN[7:0]								DCBLUE[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x228	LTDC_L2BFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BOR	Res.	Res.	Res.	Res.	Res.	BF1[2:0]			Res.	Res.	Res.	Res.	Res.	BF2[2:0]		
	Reset value																0						1	1	0						1	1	1
0x22C	LTDC_L2BLCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BL[4:0]				
	Reset value																												0	0	0	0	0
0x230	LTDC_L2PCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	YREN	OF	CBF	YF	YCM[1:0]		YCEN	Res.	Res.	Res.
	Reset value																							0	0	0	0	0	0	0			
0x234	LTDC_L2CFBAR	CFBADD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x238	LTDC_L2CFBLR	Res.	CFBP[14:0]															Res.	Res.	CFBLL[13:0]													
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x23C	LTDC_L2CFBLNR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFBLNBR[11:0]												Res.
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	
0x240-0x24C	Reserved	Reserved																															

Table 387. LTDC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x250	LTDC_ L2CLUTWR	CLUTADD[7:0]								RED[7:0]								GREEN[7:0]								BLUE[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x254-0x268	Reserved	Reserved																															
0x26C	LTDC_L2CYR0R	Res.	Res.	Res.	Res.	Res.	Res.	CB2B[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	CR2R[9:0]									
	Reset value							0	0	0	0	0	0	0	0	0	0							0	0	0	0	0	0	0	0	0	0
0x270	LTDC_L2CYR1R	Res.	Res.	Res.	Res.	Res.	Res.	CB2G[9:0]										Res.	Res.	Res.	Res.	Res.	Res.	CR2G[9:0]									
	Reset value							0	0	0	0	0	0	0	0	0	0							0	0	0	0	0	0	0	0	0	0
0x274	LTDC_L2FPF0R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RLEN[3:0]				RPOS[4:0]						ALEN[3:0]				APOS[4:0]			
	Reset value															1	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0
0x278	LTDC_L2FPF1R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSIZE [2:0]			BLEN[3:0]				BPOS[4:0]						GLEN[3:0]				GPOS[4:0]			
	Reset value												1	0	0	1	0	0	0	1	1	0	0	0	1	0	0	0	1	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

44 Neo-Chrom graphic processor (GPU2D)

44.1 GPU2D introduction

GPU2D is a dedicated graphics processing unit accelerating numerous 2.5D graphics applications such as graphical user interface (GUI), menu display or animations.

The GPU2D peripheral works together with an optimized software stack designed for state of the art graphic rendering.

44.2 GPU2D main features

Main features

- Multithreaded fragment (pixel) processing core with a VLIW (very long instruction word) instruction set
- Fixed-point functional units
- Command-list-based DMAs to minimize CPU overhead
- Two 64-bit AXI master interfaces for texture and frame buffer access
- Dedicated 64-bit AXI master interface for command list
- 32-bit AHB slave interface for register bank access
- Up to four general-purpose flags for system-level synchronization
- Texture decompression unit with TSC™4 and TSC™6/TSC™6a support

2D drawing features

- Pixel/line drawing
- Filled rectangles
- Triangles, quadrilateral drawing
- Anti-aliasing 8xMSAA (multisample anti-aliasing)

Image transformations

- 3D perspective correct projections
- Texture mapping with bilinear filtering or point sampling

Blit support

- Rotation, mirroring, stretching (independently on x and y axis)
- Source and/or destination color keying
- Pixel format conversions

Text rendering support

- A1, A2, A4, and A8 bitmap anti-aliased
- Subsampled anti-aliased

Color formats

- ABGR8888, ARGB8888, BGRA8888, RGBA8888
- xBGR8888, xRGB8888, BGRx8888, RGBx8888, RGB888, BGR888
- BGR565, RGB565
- RGB322, BGR322
- TSC4, TSC6, TSC6A
- L1, L2, L4, L8 (grayscale)
- A1, A2, A4, A8

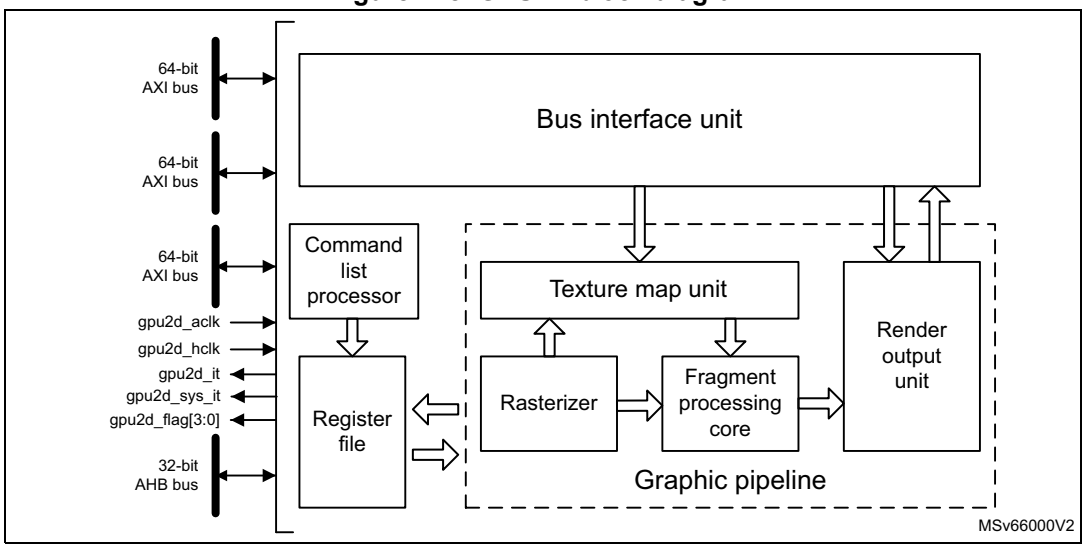
Full alpha blending with hardware blender

- Programmable blending modes
- Source/destination color keying

44.3 GPU2D general description

44.3.1 GPU2D block diagram

Figure 448. GPU2D block diagram



44.3.2 GPU2D pins and internal signals

The internal signals of the GPU2D peripheral are given in the following table.

Table 388. GPU2D internal input/output signals

Internal signal name	Signal type	Description
gpu2d_aclk	Input	GPU2D AXI clock
gpu2d_hclk	Input	GPU2D AHB clock
gpu2d_it	Output	GPU2D interrupt request (GPU2D)

Table 388. GPU2D internal input/output signals (continued)

Internal signal name	Signal type	Description
gpu2d_sys_it	Output	GPU2D system interrupt request (GPU2D_SYS)
gpu2d_flag[3:0]	Output	GPU2D general purpose flags

Table 389. GPU2D trigger connections

Trigger name	Direction	Trigger source/destination
gpu2d_flag[0]	Output	hpdma_trigsel[21]
		gpdma_trigsel[21]
gpu2d_flag[1]	Output	hpdma_trigsel[22
		gpdma_trigsel[22]
gpu2d_flag[2]	Output	hpdma_trigsel[23]
		gpdma_trigsel[23]
gpu2d_flag[3]	Output	hpdma_trigsel[24]
		gpdma_trigsel[24]

45 Video encoder (VENC)

45.1 VENC introduction

The video encoder (VENC) provides a hardware acceleration to encode a 1080p15 video stream in H264 (= MPEG4_Part10/AVC).

The VENC also provides hardware acceleration to encode still images (JPEG) of up to 300 Mpixel/s.

The VENC implementation embeds a large 128-Kbyte video RAM (VENCRAM).

When the VENC is disabled, the VENCRAM is unused for video purposes, and is accessible by the system as a contiguous extension of the system SRAM.

45.2 VENC main features

The VENC supports the following features:

- Video encode:
  - codecs: H264 (MPEG4_Part10/AVC, baseline/Main/High up to 5.2)
  - performance: 1080p15 for H264
- Still-image encode:
  - codecs: JPEG (baseline interleaved)
  - performance: 300 Mpixel/s for JPEG
- VENCRAM:
  - size: 128 Kbytes
  - access: can be statically assigned to the system by SYSCFG settings

The VENC also provides the following features:

- Security via RIF:
  - The VENC is protected by a default slave and master control (RISUP and RIMU).
  - By default, the VENC works in non-protected mode.
  - The VENC can be set in protected mode, for instance to handle DRM tasks with a secure datapath. In such cases, the drivers must run in a protected process.
- Synchronization from an upstream peripheral:

  Thanks to this feature, pixels can be streamed from an upstream peripheral (such as the camera) directly to the VENC, without writing to a full frame buffer, nor requiring external bandwidth.
- Encode with multiple codecs on a same system:

  The VENC can encode using different codecs on a same system: the VENC hardware processes the encode tasks sequentially, interleaved at frame level. Performances are shared across all the tasks.

45.3 VENC implementation

The VENC with a 400 MHz core clock supports up to 1080p15 video encode, or 300 Mpixel/s JPEG encode.

Since the VENC encoder is multi-instance, the input image can be changed at each encoding. The video format and resolution can thus be different for each instance.

The method for sharing the resource between several instances must be managed by software. This is to be done with locking mechanism, so that exclusive access to the VENC hardware is ensured for one instance at a time. The number of streams to be encoded this way is limited only by the external memory availability.

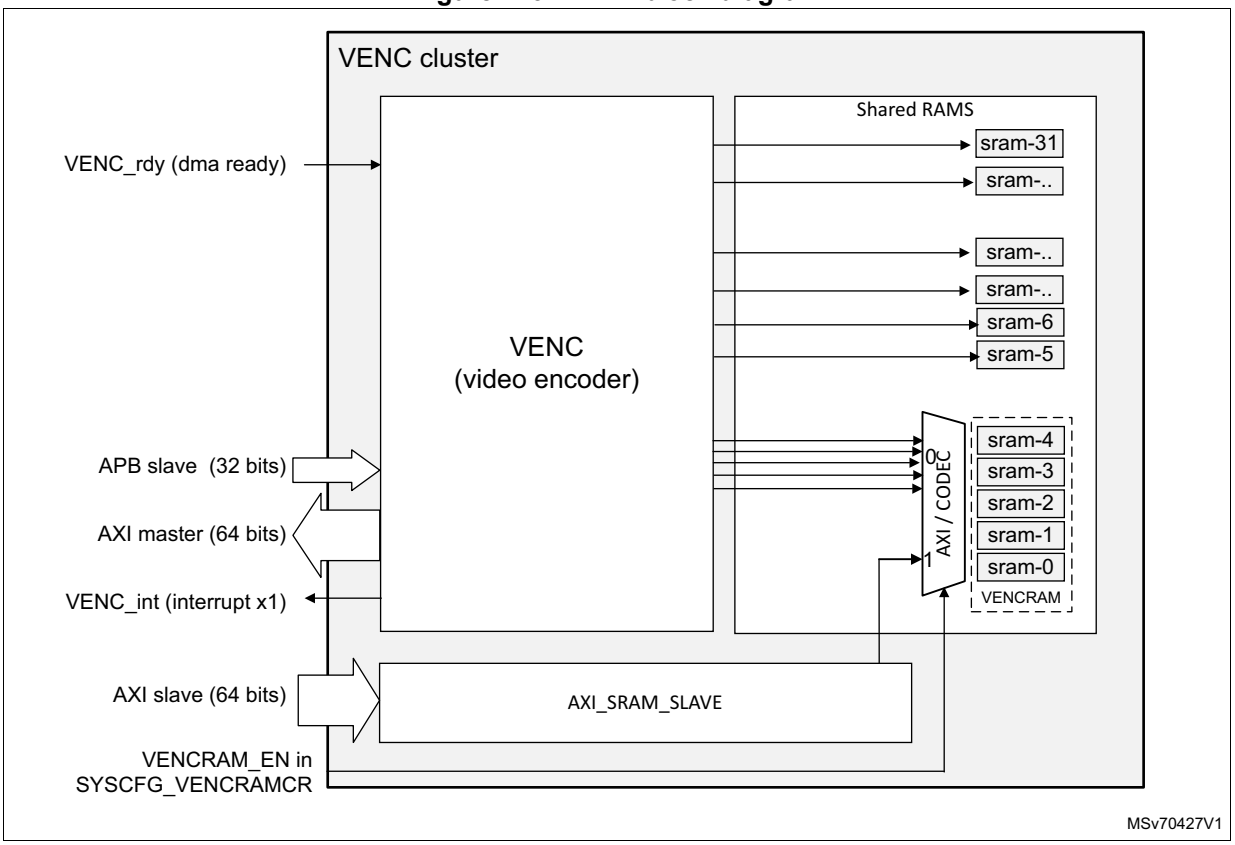
Note: Any use case with two (or more) simultaneous video encoding sessions must be tuned within the bandwidth limit of the system (especially memories accesses). Typical tunings are on pixel format, frame rate, and resolution.

45.4 VENC functional description

45.4.1 VENC block diagram

The following block diagram shows the VENC, the VENCRAM, and its interfaces.

Figure 449. VENC block diagram



45.4.2 VENC pins and internal signals

The VENC implements the following group of signals:

- APB Slave 32-bit bus: to configure the VENC and start its tasks.
- AXI Master 64-bit bus: for the VENC to fetch and store its needed data

A 1080p15 video encode generates a bandwidth of 250 Mbyte/s to the buffer of its reference frame. An additional 90 Mbyte/s can be used to load the frame to encode.

- interrupts returned to the CPU interrupt controller
- venc_rdy: synchronization signals with an external peripheral like the DCMIPP

45.4.3 VENC reset and clocks

The VENC receives the following external signals:

- rstn_venc: to reset the whole VENC
- ck_icn_s_venc: to clock the APB Slave to configure the VENC
- ck_icn_m_venc: to clock the AXI Master controlled by the VENC

45.4.4 VENC pre-processor

The VENC includes a pre-processor implemented at the input of the encoding pipe. This pre-processor allows on-the-fly processing of a frame that is read to be encoded, at no bandwidth cost.

The VENC pre-processor supports the following features:

- xRGB-to-YUV420 conversion: for direct encoding of an xRGB frame, for instance to store compressed the display output (that is usually in xRGB format)
- Cropping: for direct encoding of a specific region of interest
- Rotation90/270: for compensation of a rotated camera

Note: Any other feature is not supported.

The VENC pre-processor supports the following input pixel formats:

- YCbCr 4:2:0 planar (3 buffers)
- YCbCr 4:2:0 semi-planar (2 buffers)
- YCrCb 4:2:0 semi-planar (2 buffers)
- YCbYCr 4:2:2 interleaved (1 buffer)
- CbYCrY 4:2:2 interleaved (1 buffer)
- RGB444, BGR444 (16 bpp)
- RGB555, BGR555 (16 bpp)
- RGB565, BGR565 (16 bpp)
- RGB888, BGR888 (32 bpp)

45.4.5 VENC H264 encoder

The VENC supports the following features in H264 mode:

- Performance: 1080p15
- Spec version: 12ISO/IEC 14496-10 / ITU-T Rec. H.264 (03/2010)
  - Baseline profile: levels 1 – 5.2
  - Main profile: levels 1 - 5.2
  - High profile: levels 1 – 5.2
- Profile: Baseline/Main/High, levels 1-52
- MVC supported
- Motion search: horizontal: ±126 pixels, vertical: ±30 pixels

- Output format: for H.264: byte unit stream or NAL unit stream
- Image size: 96x96 to 4080x4080 (by steps of 4 pixels)
- Bit rate: 10 Kbit/s to 40 Mbit/s
- Coding output: such as best matching motion vector, quality estimation (SAD)
- Bandwidth: optimized by buffering the luma search area in the SRAM
- Region of interest: two user definable rectangular areas with separate QP settings
- Region of intra: one user definable rectangular area coded as intra
- Cyclic intra refresh supported
- H264 ROI map: can define a different quantization delta for each macro-block, according to a table defined in the external DDR. A total of three quantization delta values can be selected in one picture.

Note: SVC is not supported.

45.4.6 VENC JPEG encoder

The VENC supports the following feature in JPEG mode:

- Performance: 300 Mpixel/s at 400 MHz
- Spec version: JPEG ISO/IEC 10918-1, ITU-T T.81
- Baseline process
- Huffman coding interleaved
- YUV420 supported internally
- Output format: JFIF file format 1.02, non-progressive JPEG 4:2:0
- Image size: 96x32 to 8176x8176 (by step of 4 pixels)
- Thumbnail insertion: RGB 8-bit, RGB 24-bit, JPEG compressed thumbnails
- User-defined quantization table: luma and chroma quantization tables
- Partial encoding: slice-by-slice according to the user defined restart interval
- ROI implemented by crop feature in pre-processor

The VENC shares its performance between video and JPEG encode: one task runs exclusively versus the other on a time-sharing basis.

45.4.7 VENC synchronization

The VENC can be synchronized, if needed, onto an external peripheral: it is used to encode a full frame, per chunk of one/more line of macro-blocks.

Initialization

The software allocates a double buffer, with each one line of macro-blocks (16 pixel high). This double buffer is used in a ping-pong way: when the external peripheral writes into one buffer, the VENC reads from the other buffer, and encodes it.

Before one frame

1. The software configures and triggers the VENC for a new frame.
2. The VENC waits for venc_rdy to become high (set by the source peripheral).
3. The VENC starts to encode the current ping-pong buffer.
4. The VENC finishes its encoding: it swaps its ping-pong buffer, and waits to a next venc_rdy signal high.

Note: The VENC is thus fully slaved onto the external peripheral, and solely driven by the venc_rdy trigger from the source signal:
- The encoding of one line of macroblock must be guaranteed faster than the reception of the next line of macroblock.
- The external peripheral triggers the VENC each time it has finished writing one line of the macroblock.

45.4.8 VENC security

The VENC security is based on the RIF, with the following protections:

- protection of VENC register map (AHB slave) with a RISUP (see *Table 20: RISUP indexes*, the index for the VENC is 97).
- protection of VENC access to memory (AXI master) with a RIMU (index = 12)

The VENC RIMU by default inherits from its RISUP settings. More information can be found in *Section 6: Resource isolation framework security controller (RIFSC)*.

The security configuration is, by default, non-protected:

- It allows the default OS drivers to handle the VENC.
- AXI transactions are not protected, although they can be tagged with a CID specific for pixel buffers.
- The VENC can work in protected mode to allow secure data path (SDP) support with both the input flow and output bitstream being stored in protected buffers:
- The VENC AXI transaction becomes protected.
- The VENC APB configuration space becomes protected.
- The low-level software must work in protected mode to have access to the VENC registers.

45.4.9 VENC power-on sequence

The VENC power-on sequence is similar for all supported codecs, H264 or JPEG.

The RCC provides the needed master, slave, and core clocks. The software makes sure the VENC can access its VENCRAM, and lets the driver configure the VENC to start encoding.

The corresponding set up includes the following steps:

1. Set up security settings (see VENC index in *Table 20: RISUP indexes*).
   – The VENC is kept by default as non-protected.
2. Allocate the VENCRAM to the VENC:
   – Clear VENCRAM_EN to 0 in SYSCFG_VENCRAMCR.
3. Set up the clock and reset (see *Section 14: Reset and clock control (RCC)*):
   – VENC master: ck_icn_m_venc to activate in the RCC (400 MHz typical)

– VENC slave: ck_icn_p_venc to activate in the RCC (200 MHz typical)

45.4.10 VENCRAM power-on sequence for AXI access

This section describes settings to access the VENCRAM with the AXI-Slave. The RCC provides needed slave clocks, and makes sure the AXI Slave can access its VENCRAM, so that the system can start to use the SRAM.

The steps to set up this path are as follows:

1. Set up the security settings (see VENCRAM index in *Table 20: RISUP indexes*).
   – The VENCRAM can be set as protected or not-protected, depending on the system needs.
2. Allocate the VENCRAM to the CPU (the VENC no longer works correctly).
   – Set VENCRAM to 1 in SYSCFG_VDERAMCR.
3. Setup the clock and reset (see *Section 14: Reset and clock control (RCC)*):
   – VENCRAM AXI: ck_icn_s_vencram to activate in the RCC (400 MHz typical).

45.5 VENC low power modes

The VENC has no voltage power modes.

45.6 VENC interrupts

The VENC can trigger a global interrupt (venc_gbl_int) that is referenced in the CPU interrupt handler as VENC.

45.7 VENC registers

45.7.1 VENC ID register (VENC_SWREG0)

Address offset: 0x000

Reset value: 0x6E65 5000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Identification register (all format mode)

45.7.2 VENC interrupt register (VENC_SWREG1)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Interrupt register (all format mode)

45.7.3 VENC bus interface configuration register (VENC_SWREG2)

Address offset: 0x008

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Bus interface configuration register (all format mode)

45.7.4 VENC device configuration register (VENC_SWREG3)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Device configuration register (all format mode)

45.7.5 VENC base address for output stream data register (VENC_SWREG5)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Base address for output stream data (all format mode)

45.7.6 VENC base address for output control data register (VENC_SWREG6)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Base address for output control data (all format mode)

45.7.7 VENC base address for reference luma register (VENC_SWREG7)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Base address for reference luma (all format mode)

45.7.8 VENC base address for reference chroma register (VENC_SWREG8)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Base address for reference chroma (all format mode)

45.7.9 VENC base address for reconstructed luma register (VENC_SWREG9)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Base address for reconstructed luma (all format mode)

45.7.10 VENC base address for reconstructed chroma register (VENC_SWREG10)

Address offset: 0x028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Base address for reconstructed chroma (all format mode)

45.7.11 VENC base address for input picture luma register (VENC_SWREG11)

Address offset: 0x02C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Base address for input picture luma (all format mode)

45.7.12 VENC base address for input picture cb register (VENC_SWREG12)

Address offset: 0x030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Base address for input picture cb (all format mode)

45.7.13 VENC base address for input picture cr register (VENC_SWREG13)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Base address for input picture cr (all format mode)

45.7.14 VENC encoder control register 0 (VENC_SWREG14)

Address offset: 0x038

Reset value: 0x0000 0000

This register is used to control data such as picture information or encoding mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Encoder control register 0 (such as picture information or encoding mode) (all format mode)

45.7.15 VENC encoder control register 1 (VENC_SWREG15)

Address offset: 0x03C

Reset value: 0x0000 0000

This register is used to control data such as preprocessing control, crop, rotate, or input format.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Encoder control register 1 (such as preprocessing control, crop, rotate, input format) (all format mode)

45.7.16 VENC encoder control register 2 (VENC_SWREG16)

Address offset: 0x040

Reset value: 0x0000 0000

This register controls the base address for second reference luma (H264 control).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Base address for second reference luma (H264 control) (all format mode)

45.7.17 VENC encoder control register 3 (VENC_SWREG17)

Address offset: 0x044

Reset value: 0x0000 0000

This register controls the base address for second reference chroma (H264 control).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Base address for second reference chroma (H264 control) (all format mode)

45.7.18 VENC encoder control register 4 (VENC_SWREG18)

Address offset: 0x048

Reset value: 0x0000 0000

This register controls the deblock filter mode (H264 control).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Encoder control register 4 (deblock filter mode, H264 control) (all format mode)

45.7.19 VENC encoder control register 5 (VENC_SWREG19)

Address offset: 0x04C

Reset value: 0x0000 0000

This register controls data such as input format or motion vector.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Encoder control register 5 (input format, motion vector etc) (all format mode)

45.7.20 VENC encoder control register 6 (VENC_SWREG20)

Address offset: 0x050

Reset value: 0x0000 0000

This register controls JPEG.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Control of data JPEG (all format mode)

45.7.21 VENC encoder control register 7 (VENC_SWREG21)

Address offset: 0x054

Reset value: 0x0000 0000

This register controls data such as H264.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**:Control of H264 (all format mode)

45.7.22 VENC stream header remainder MSB bits register (VENC_SWREG22)

Address offset: 0x058

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Stream header remainder bits MSB (MSB aligned) (all format mode)

45.7.23 VENC stream header remainder LSB bits register (VENC_SWREG23)

Address offset: 0x05C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Stream header remainder bits LSB (MSB aligned) (all format mode)

45.7.24 VENC stream buffer limit/output stream size register (VENC_SWREG24)

Address offset: 0x060

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Stream buffer limit (64-bit addresses)/output stream size (bits) (all format mode)

45.7.25 VENC encoder control register 8 (VENC_SWREG25)

Address offset: 0x064

Reset value: 0x0000 0000

This register controls the MAD control and QP sum output.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Control of MAD control and QP sum output (all format mode)

45.7.26 VENC intra-slice bitmap register (VENC_SWREG26)

Address offset: 0x068

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: intra-slice bitmap for probability updates (all format mode)

45.7.27 VENC encoder control register 9 (VENC_SWREG27)

Address offset: 0x06C

Reset value: 0x0000 0000

This register controls H264 QP.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**:Control of H264 QP (all format mode)

45.7.28 VENC encoder control register 10 (VENC_SWREG28)

Address offset: 0x070

Reset value: 0x0000 0000

This register controls H264 checkpoint 1-2 .

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: H264 checkpoint 1-2 (all format mode)

45.7.29 VENC encoder control register 11 (VENC_SWREG29)

Address offset: 0x074

Reset value: 0x0000 0000

This register controls H264 checkpoint 3-4 .

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: H.264 Checkpoint 3 -4 (all format mode)

45.7.30 VENC encoder control register 12 (VENC_SWREG30)

Address offset: 0x078

Reset value: 0x0000 0000

This register controls H264 checkpoint 5-6 .

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: H.264 checkpoint 5 -6 (all format mode)

45.7.31 VENC encoder control register 13 (VENC_SWREG31)

Address offset: 0x07C

Reset value: 0x0000 0000

This register controls H264 checkpoint 7-8.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: H.264 checkpoint 7 -8 (all format mode)

45.7.32 VENC encoder control register 14 (VENC_SWREG32)

Address offset: 0x080

Reset value: 0x0000 0000

This register controls H264 checkpoint 8-10.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: H.264 Checkpoint 8 -10 / Encoder control register 14 (all format mode)

45.7.33 VENC encoder control register 15 (VENC_SWREG33)

Address offset: 0x084

Reset value: 0x0000 0000

This register controls H264 checkpoint 1-2.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: H.264 Checkpoint word error 1-2 (all format mode)

45.7.34 VENC encoder control register 16 (VENC_SWREG34)

Address offset: 0x088

Reset value: 0x0000 0000

This register controls H264 checkpoint 3-4 and the second reference frame.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: H.264 checkpoint word error 3-4 and the second reference frame control (all format mode)

45.7.35 VENC H.264 checkpoint word error 5-6/encoder control register 17 (VENC_SWREG35)

Address offset: 0x08C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: H.264 Checkpoint word error 5-6 / Encoder control register 17 (all format mode)

45.7.36 VENC H.264 checkpoint delta QP 1-8/encoder control register 18 (VENC_SWREG36)

Address offset: 0x090

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: H.264 Checkpoint delta QP 1-8 / Encoder control register 18 (all format mode)

45.7.37 VENC encoder control register 19, stream start offset (VENC_SWREG37)

Address offset: 0x094

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Encoder control register 19 (all format mode)

45.7.38 VENC macroblock count output register (VENC_SWREG38)

Address offset: 0x098

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Macroblock count output (all format mode)

45.7.39 VENC base address for next pic luminance register (VENC_SWREG39)

Address offset: 0x09C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Base address for next pic luminance (all format mode)

45.7.40 VENC stabilization mode control register (VENC_SWREG40)

Address offset: 0x0A0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Stabilization mode control (all format mode)

45.7.41 VENC stabilization motion sum div8 output register (VENC_SWREG41)

Address offset: 0x0A4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Stabilization motion sum div8 output (all format mode)

45.7.42 VENC stabilization GMV output, matrix 1, up-left position output register (VENC_SWREG42)

Address offset: 0x0A8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Stabilization GMV output, matrix 1 (up-left position) output (all format mode)

45.7.43 VENC stabilization GMV output, matrix 2, up position output register (VENC_SWREG43)

Address offset: 0x0AC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Stabilization GMV output, matrix 2 (up position) output (all format mode)

45.7.44 VENC stabilization matrix 3, up-right position output register (VENC_SWREG44)

Address offset: 0x0B0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Stabilization matrix 3 (up-right position) output (all format mode)

45.7.45 VENC stabilization matrix 4, left position output register (VENC_SWREG45)

Address offset: 0x0B4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Stabilization matrix 4 (left position) output (all format mode)

45.7.46 VENC stabilization matrix 5, GMV position output register (VENC_SWREG46)

Address offset: 0x0B8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Stabilization matrix 5 (GMV position) output (all format mode)

45.7.47 VENC stabilization matrix 6, right position output register (VENC_SWREG47)

Address offset: 0x0BC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Stabilization matrix 6 (right position) output (all format mode)

45.7.48 VENC stabilization matrix 7, down-left position output register (VENC_SWREG48)

Address offset: 0x0C0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Stabilization matrix 7 (down-left position) output (all format mode)

45.7.49 VENC stabilization matrix 8, down position output register (VENC_SWREG49)

Address offset: 0x0C4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Stabilization matrix 8 (down position) output (all format mode)

45.7.50 VENC stabilization matrix 9, down-right position output register (VENC_SWREG50)

Address offset: 0x0C8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Stabilization matrix 9 (down-right position) output (all format mode)

45.7.51 VENC base address for cabac context tables H264 register (VENC_SWREG51)

Address offset: 0x0CC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Base address for cabac context tables (H264) or probability tables (all format mode)

45.7.52 VENC base address for MV output writing register (VENC_SWREG52)

Address offset: 0x0D0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Base address for MV output writing (all format mode)

45.7.53 VENC RGB to YUV conversion coefficient A - B register (VENC_SWREG53)

Address offset: 0x0D4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: RGB to YUV conversion coefficient A - B (all format mode)

45.7.54 VENC RGB to YUV conversion coefficient C - E register (VENC_SWREG54)

Address offset: 0x0D8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: RGB to YUV conversion coefficient C - E (all format mode)

45.7.55 VENC RGB to YUV conversion coefficient F, RGB mask MSB bit position register (VENC_SWREG55)

Address offset: 0x0DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: RGB to YUV conversion coefficient F, RGB mask MSB bit position (all format mode)

45.7.56 VENC intra area register (VENC_SWREG56)

Address offset: 0x0E0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: intra area (all format mode)

45.7.57 VENC CIR intra mb position register (VENC_SWREG57)

Address offset: 0x0E4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: CIR intra mb position (all format mode)

45.7.58 VENC intra slice bitmap for slices 0..31/base address for 1st DCT partition register (VENC_SWREG58)

Address offset: 0x0E8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: intra slice bitmap for slices 0..31 / Base address for 1st DCT partition (all format mode)

45.7.59 VENC intra slice bitmap for slices 32..63/base address for 2nd DCT partition register (VENC_SWREG59)

Address offset: 0x0EC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: intra slice bitmap for slices 32..63 / Base address for 2nd DCT partition (all format mode)

45.7.60 VENC 1st ROI area register (VENC_SWREG60)

Address offset: 0x0F0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: 1st ROI area (all format mode)

45.7.61 VENC 2nd ROI area register (VENC_SWREG61)

Address offset: 0x0F4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: 2nd ROI area (all format mode)

45.7.62 VENC ROI area delta QP, MV register (VENC_SWREG62)

Address offset: 0x0F8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: ROI area delta QP, MV (all format mode)

45.7.63 VENC synthesis configuration register encoder 0 register (VENC_SWREG63)

Address offset: 0x0FC

Reset value: 0x1A62 2780

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SWREG_FIELD[31:0]**: Synthesis configuration register encoder 0 (read only) (all format mode)

45.7.64 VENC JPEG luma quantization 1/intra 16x16 mode 0-1 penalty register (VENC_SWREG64)

Address offset: 0x100

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: JPEG luma quantization 1 / intra 16x16 mode 0-1 penalty (all format mode)

45.7.65 VENC JPEG luma quantization 2/intra 16x16 mode 2-3 penalty register (VENC_SWREG65)

Address offset: 0x104

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: JPEG luma quantization 2 / intra 16x16 mode 2-3 penalty (all format mode)

45.7.66 VENC JPEG luma quantization 3/intra 4x4 mode 0-1 penalty register (VENC_SWREG66)

Address offset: 0x108

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: JPEG luma quantization 3 / intra 4x4 mode 0-1 penalty (all format mode)

45.7.67 VENC JPEG luma quantization 4/intra 4x4 mode 2-3 penalty register (VENC_SWREG67)

Address offset: 0x10C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: JPEG luma quantization 4 / intra 4x4 mode 2-3 penalty (all format mode)

45.7.68 VENC JPEG luma quantization 5/intra 4x4 mode 4-5 penalty register (VENC_SWREG68)

Address offset: 0x110

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: JPEG luma quantization 5 / intra 4x4 mode 4-5 penalty (all format mode)

45.7.69 VENC JPEG luma quantization 6/intra 4x4 mode 6-7 penalty register (VENC_SWREG69)

Address offset: 0x114

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: JPEG luma quantization 6 / intra 4x4 mode 6-7 penalty (all format mode)

45.7.70 VENC JPEG luma quantization 7/intra 4x4 mode 8-9 penalty register (VENC_SWREG70)

Address offset: 0x118

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: JPEG luma quantization 7 / intra 4x4 mode 8-9 penalty (all format mode)

45.7.71 VENC JPEG luma quantization 8/base address for segmentation map register (VENC_SWREG71)

Address offset: 0x11C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: JPEG luma quantization 8 / Base address for segmentation map (all format mode)

45.7.72 VENC JPEG luma quantization 9/segment1 parameter register (VENC_SWREG72)

Address offset: 0x120

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: JPEG luma quantization 9 / segment1 parameter (all format mode)

45.7.73 VENC JPEG luma quantization 10/segment1 parameter register (VENC_SWREG73)

Address offset: 0x124

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: JPEG luma quantization 10 / segment1 parameter (all format mode)

45.7.74 VENC JPEG luma quantization 11/segment1 parameter register (VENC_SWREG74)

Address offset: 0x128

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: JPEG luma quantization 11 / segment1 parameter (all format mode)

45.7.75 VENC JPEG luma quantization 12/segment1 parameter register (VENC_SWREG75)

Address offset: 0x12C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: JPEG luma quantization 12 / segment1 parameter (all format mode)

45.7.76 VENC JPEG luma quantization 13/segment1 parameter register (VENC_SWREG76)

Address offset: 0x130

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: JPEG luma quantization 13 / segment1 parameter (all format mode)

45.7.77 VENC JPEG luma quantization 14/segment1 parameter register (VENC_SWREG77)

Address offset: 0x134

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: JPEG luma quantization 14 / segment1 parameter (all format mode)

45.7.78 VENC JPEG luma quantization 15/segment1 parameter register (VENC_SWREG78)

Address offset: 0x138

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: JPEG luma quantization 15 / segment1 parameter (all format mode)

45.7.79 VENC JPEG luma quantization 16/segment2 parameter register (VENC_SWREG79)

Address offset: 0x13C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: JPEG luma quantization 16 / segment2 parameter (all format mode)

45.7.80 VENC JPEG chroma quantization 1/segment2 parameter register (VENC_SWREG80)

Address offset: 0x140

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: JPEG chroma quantization 1 / segment2 parameter (all format mode)

45.7.81 VENC JPEG chroma quantization 2/segment2 parameter register (VENC_SWREG81)

Address offset: 0x144

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: JPEG chroma quantization 2 / segment2 parameter (all format mode)

45.7.82 VENC JPEG chroma quantization 3/segment2 parameter register (VENC_SWREG82)

Address offset: 0x148

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: JPEG chroma quantization 3 / segment2 parameter (all format mode)

45.7.83 VENC JPEG chroma quantization 4/segment2 parameter register (VENC_SWREG83)

Address offset: 0x14C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: JPEG chroma quantization 4 / segment2 parameter (all format mode)

45.7.84 VENC JPEG chroma quantization 5/segment2 parameter register (VENC_SWREG84)

Address offset: 0x150

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: JPEG chroma quantization 5 / segment2 parameter (all format mode)

45.7.85 VENC JPEG chroma quantization 6/segment2 parameter register (VENC_SWREG85)

Address offset: 0x154

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: JPEG chroma quantization 6 / segment2 parameter (all format mode)

45.7.86 VENC JPEG chroma quantization 7/segment2 parameter register (VENC_SWREG86)

Address offset: 0x158

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: JPEG chroma quantization 7 / segment2 parameter (all format mode)

45.7.87 VENC JPEG chroma quantization 8/segment2 parameter register (VENC_SWREG87)

Address offset: 0x15C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: JPEG chroma quantization 8 / segment2 parameter (all format mode)

45.7.88 VENC JPEG chroma quantization 9/segment3 parameter register (VENC_SWREG88)

Address offset: 0x160

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: JPEG chroma quantization 9 / segment3 parameter (all format mode)

45.7.89 VENC JPEG chroma quantization 10/segment3 parameter register (VENC_SWREG89)

Address offset: 0x164

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: JPEG chroma quantization 10 / segment3 parameter (all format mode)

45.7.90 VENC JPEG chroma quantization 11/segment3 parameter register (VENC_SWREG90)

Address offset: 0x168

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: JPEG chroma quantization 11 / segment3 parameter (all format mode)

45.7.91 VENC JPEG chroma quantization 12/segment3 parameter register (VENC_SWREG91)

Address offset: 0x16C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: JPEG chroma quantization 12 / segment3 parameter (all format mode)

45.7.92 VENC JPEG chroma quantization 13/segment3 parameter register (VENC_SWREG92)

Address offset: 0x170

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: JPEG chroma quantization 13 / segment3 parameter (all format mode)

45.7.93 VENC JPEG chroma quantization 14/segment3 parameter register (VENC_SWREG93)

Address offset: 0x174

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: JPEG chroma quantization 14 / segment3 parameter (all format mode)

45.7.94 VENC JPEG chroma quantization 15/segment3 parameter register (VENC_SWREG94)

Address offset: 0x178

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: JPEG chroma quantization 15 / segment3 parameter (all format mode)

45.7.95 VENC JPEG chroma quantization 16/segment3 parameter register (VENC_SWREG95)

Address offset: 0x17C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: JPEG chroma quantization 16 / segment3 parameter (all format mode)

45.7.96 VENC DMV 4p/1p penalty values 0-3 register (VENC_SWREG96)

Address offset: 0x180

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **SWREG_FIELD[31:0]**: DMV 4p/1p penalty values 0-3 (all format mode)

45.7.97 VENC DMV 4p/1p penalty values 4-7 register (VENC_SWREG97)

Address offset: 0x184

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **SWREG_FIELD[31:0]**: DMV 4p/1p penalty values 4-7 (all format mode)

45.7.98 VENC DMV 4p/1p penalty values register (VENC_SWREGx)

Address offset: 0x188 + 0x4 * (x - 98), (x = 98 to 126)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **SWREG_FIELD[31:0]**: DMV 4p/1p penalty values (all format mode)

45.7.99 VENC DMV 4p/1p penalty values 124-127 register (VENC_SWREG127)

Address offset: 0x1FC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **SWREG_FIELD[31:0]**: DMV 4p/1p penalty values 124-127 (all format mode)

45.7.100 VENC DMV qpel penalty values 0-3 register (VENC_SWREG128)

Address offset: 0x200

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **SWREG_FIELD[31:0]**: DMV qpel penalty values 0-3 (all format mode)

45.7.101 VENC DMV qpel penalty values 4-7 register (VENC_SWREG129)

Address offset: 0x204

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **SWREG_FIELD[31:0]**: DMV qpel penalty values 4-7 (all format mode)

45.7.102 VENC DMV qpel penalty values register (VENC_SWREGx)

Address offset: 0x208 + 0x4 * (x - 130), (x = 130 to 158)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **SWREG_FIELD[31:0]**: DMV qpel penalty values (all format mode)

45.7.103 VENC DMV qpel penalty values 124-127 register (VENC_SWREG159)

Address offset: 0x27C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **SWREG_FIELD[31:0]**: DMV qpel penalty values 124-127 (all format mode)

45.7.104 VENC base address for output of down-scaled encoder image in YUYV 4:2:2 format register (VENC_SWREG231)

Address offset: 0x39C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Base address for output of down-scaled encoder image in YUYV 4:2:2 format (all format mode)

45.7.105 VENC scaling control register (VENC_SWREGx)

Address offset: 0x3A0 + 0x4 * (x - 232), (x = 232 to 233)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Scaling control (all format mode)

45.7.106 VENC squared error output calculated for 13x13 pixels per macroblock register (VENC_SWREG236)

Address offset: 0x3B0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Squared error output calculated for 13x13 pixels per macroblock (all format mode)

45.7.107 VENC MAD 2 control and output register (VENC_SWREG237)

Address offset: 0x3B4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: MAD 2 control and output (all format mode)

45.7.108 VENC MAD 3 control and output register (VENC_SWREG238)

Address offset: 0x3B8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: MAD 3 control and output (all format mode)

45.7.109 VENC segment 1: intra 16x16 mode 0-2 penalty register (VENC_SWREG256)

Address offset: 0x400

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 1: intra 16x16 mode 0-2 penalty (all format mode)

45.7.110 VENC segment 1: intra 16x16 mode 3, intra 4x4 0-1 penalty register (VENC_SWREG257)

Address offset: 0x404

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 1: intra 16x16 mode 3 and intra 4x4 0-1 penalty (all format mode)

45.7.111 VENC segment 1: intra 4x4 mode 2-4 penalty register (VENC_SWREG258)

Address offset: 0x408

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 1: intra 4x4 mode 2-4 penalty (all format mode)

45.7.112 VENC segment 1: intra 4x4 mode 5-7 penalty register (VENC_SWREG259)

Address offset: 0x40C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 1: intra 4x4 mode 5-7 penalty (all format mode)

45.7.113 VENC segment 1: intra 4x4 mode 8-9 penalty, previous mode favor for H.264 register (VENC_SWREG260)

Address offset: 0x410

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 1: intra 4x4 mode 8-9 penalty, previous mode favor for H.264 (all format mode)

45.7.114 VENC segment 1: bit cost of inter type, intra 16x16 mode favor register (VENC_SWREG261)

Address offset: 0x414

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 1: Bit cost of inter type, intra 16x16 mode favor (all format mode)

45.7.115 VENC segment 1: inter MB mode favor, skip mode penalty, penalty value for 2nd reference frame register (VENC_SWREG262)

Address offset: 0x418

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 1: inter MB mode favor, skip mode penalty, penalty value for second reference frame (all format mode)

45.7.116 VENC segment 1: penalty value register (VENC_SWREGx)

Address offset: 0x41C + 0x4 * (x - 263), (x = 263 to 264)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 1: penalty value (all format mode)

45.7.117 VENC segment 1: deadzone rate multiplier for plane 0-1 register (VENC_SWREG265)

Address offset: 0x424

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 1: Deadzone rate multiplier for plane 0-1 (all format mode)

45.7.118 VENC segment 1: deadzone rate multiplier for plane 2-3 register (VENC_SWREG266)

Address offset: 0x428

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 1: Deadzone rate multiplier for plane 2-3 (all format mode)

45.7.119 VENC segment 1: deadzone rate for macroblock skip token 0-1, dmv penalty coefficient register (VENC_SWREG267)

Address offset: 0x42C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 1: Deadzone rate for macroblock skip token 0-1, dmv penalty coefficient (all format mode)

45.7.120 VENC segment 2: intra 16x16 mode 0-2 penalty register (VENC_SWREG268)

Address offset: 0x430

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 2: intra 16x16 mode 0-2 penalty (all format mode)

45.7.121 VENC segment 2: intra 16x16 mode 3 penalty, intra 4x4 mode 0-1 penalty register (VENC_SWREG269)

Address offset: 0x434

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 2: intra 16x16 mode 3 penalty, intra 4x4 mode 0-1 penalty (all format mode)

45.7.122 VENC segment 2: intra 4x4 mode 2-4 penalty register (VENC_SWREG270)

Address offset: 0x438

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 2: intra 4x4 mode 2-4 penalty (all format mode)

45.7.123 VENC segment 2: intra 4x4 mode 5-7 penalty register (VENC_SWREG271)

Address offset: 0x43C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 2: intra 4x4 mode 5-7 penalty (all format mode)

45.7.124 VENC segment 2: intra 4x4 mode 8-9 penalty, intra 4x4 previous mode favor for H.264 register (VENC_SWREG272)

Address offset: 0x440

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 2: intra 4x4 mode 8-9 penalty, intra 4x4 previous mode favor for H.264 (all format mode)

45.7.125 VENC segment 2: bit cost of inter type, intra 16x16 mode favor register (VENC_SWREG273)

Address offset: 0x444

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 2: Bit cost of inter type, intra 16x16 mode favor (all format mode)

45.7.126 VENC segment 2: inter MB mode favor, skip mode penalty, penalty value register (VENC_SWREG274)

Address offset: 0x448

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 2: inter MB mode favor, skip mode penalty, panelty value (all format mode)

45.7.127 VENC segment 2: penalty value register (VENC_SWREGx)

Address offset: 0x44C + 0x4 * (x - 275), (x = 275 to 276)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 2: penalty value (all format mode)

45.7.128 VENC segment 2: deadzone rate multiplier for plane 0-1 register (VENC_SWREG277)

Address offset: 0x454

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 2: Deadzone rate multiplier for plane 0-1 (all format mode)

45.7.129 VENC segment 2: deadzone rate multiplier for plane 2-3 register (VENC_SWREG278)

Address offset: 0x458

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 2: Deadzone rate multiplier for plane 2-3 (all format mode)

45.7.130 VENC segment 2: deadzone rate for macroblock skip token 0-1, dmv penalty coefficient register (VENC_SWREG279)

Address offset: 0x45C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 2: Deadzone rate for macroblock skip token 0-1, dmv penalty coefficient (all format mode)

45.7.131 VENC segment 3: intra 16x16 mode 0-2 penalty register (VENC_SWREG280)

Address offset: 0x460

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 3: intra 16x16 mode 0-2 penalty (all format mode)

45.7.132 VENC segment 3: intra 16x16 mode 3 penalty, intra 4x4 mode 0-1 penalty register (VENC_SWREG281)

Address offset: 0x464

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 3: intra 16x16 mode 3 penalty, intra 4x4 mode 0-1 penalty (all format mode)

45.7.133 VENC segment 3: intra 4x4 mode 2-4 penalty register (VENC_SWREG282)

Address offset: 0x468

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 3: intra 4x4 mode 2-4 penalty (all format mode)

45.7.134 VENC segment 3: intra 4x4 mode 5-7 penalty register (VENC_SWREG283)

Address offset: 0x46C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 3: intra 4x4 mode 5-7 penalty (all format mode)

45.7.135 VENC segment 3: intra 4x4 mode 8-9 penalty, intra 4x4 previous mode favor for H.264 register (VENC_SWREG284)

Address offset: 0x470

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 3: intra 4x4 mode 8-9 penalty, intra 4x4 previous mode favor for H.264 (all format mode)

45.7.136 VENC segment 3: bit cost of inter type, intra 16x16 mode favor register (VENC_SWREG285)

Address offset: 0x474

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 3: Bit cost of inter type, intra 16x16 mode favor (all format mode)

45.7.137 VENC segment 3: inter MB mode favor in intra/inter selection, inter MB mode favor, penalty value for second reference frame register (VENC_SWREG286)

Address offset: 0x478

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 3: inter MB mode favor in intra/inter selection, inter MB mode favor, penalty value for second reference frame (all format mode)

45.7.138 VENC segment 3: penalty value register (VENC_SWREGx)

Address offset: 0x47C + 0x4 * (x - 287), (x = 287 to 288)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 3: penalty value (all format mode)

45.7.139 VENC segment 3: deadzone rate multiplier for plane 0-1 register (VENC_SWREG289)

Address offset: 0x484

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 3: Deadzone rate multiplier for plane 0-1 (all format mode)

45.7.140 VENC segment 3: deadzone rate multiplier for plane 2-3 register (VENC_SWREG290)

Address offset: 0x488

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 3: Deadzone rate multiplier for plane 2-3 (all format mode)

45.7.141 VENC segment 3: deadzone rate for macroblock skip token 0-1, dmv penalty coefficient register (VENC_SWREG291)

Address offset: 0x48C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 3: Deadzone rate for macroblock skip token 0-1, dmv penalty coefficient (all format mode)

45.7.142 VENC Mb boost register (VENC_SWREG294)

Address offset: 0x498

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Mb boost (all format mode)

45.7.143 VENC variance control, Pskop conding mode register (VENC_SWREG295)

Address offset: 0x49C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Variance control, Pskop conding mode (all format mode)

45.7.144 VENC synthesis configuration register encoder 1 read only register (VENC_SWREG296)

Address offset: 0x4A0

Reset value: 0x0680 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SWREG_FIELD[31:0]**: Synthesis configuration register encoder 1 (read only) (all format mode)

45.7.145 VENC MBRC control register (VENC_SWREG297)

Address offset: 0x4A4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: MBRC control (all format mode)

45.7.146 VENC segment 4: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG298)

Address offset: 0x4A8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 4: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame (all format mode)

45.7.147 VENC segment 4: skip mode penalty, inter MB mode favor register (VENC_SWREG299)

Address offset: 0x4AC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 4: skip mode penalty, inter MB mode favor (all format mode)

45.7.148 VENC segment 4: penalty value register (VENC_SWREGx)

Address offset: 0x4B0 + 0x4 * (x - 300), (x = 300 to 301)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 4: penalty value (all format mode)

45.7.149 VENC segment 5: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG302)

Address offset: 0x4B8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 5: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame (all format mode)

45.7.150 VENC segment 5: skip mode penalty, inter MB mode favor register (VENC_SWREG303)

Address offset: 0x4BC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 5: skip mode penalty, inter MB mode favor (all format mode)

45.7.151 VENC segment 5: penalty value register (VENC_SWREGx)

Address offset: 0x4C0 + 0x4 * (x - 304), (x = 304 to 305)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 5: penalty value (all format mode)

45.7.152 VENC segment 6: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG306)

Address offset: 0x4C8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 6: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame (all format mode)

45.7.153 VENC segment 6: skip mode penalty, inter MB mode favor register (VENC_SWREG307)

Address offset: 0x4CC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 6: skip mode penalty, inter MB mode favor (all format mode)

45.7.154 VENC segment 6: penalty value register (VENC_SWREGx)

Address offset: 0x4D0 + 0x4 * (x - 308), (x = 308 to 309)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 6: penalty value (all format mode)

45.7.155 VENC segment 7: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG310)

Address offset: 0x4D8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 7: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame (all format mode)

45.7.156 VENC segment 7: skip mode penalty, inter MB mode favor register (VENC_SWREG311)

Address offset: 0x4DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 7: skip mode penalty, inter MB mode favor (all format mode)

45.7.157 VENC segment 7: penalty value register (VENC_SWREGx)

Address offset: 0x4E0 + 0x4 * (x - 312), (x = 312 to 313)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 7: penalty value (all format mode)

45.7.158 VENC segment 8: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG314)

Address offset: 0x4E8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 8: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame (all format mode)

45.7.159 VENC segment 8: skip mode penalty, inter MB mode favor register (VENC_SWREG315)

Address offset: 0x4EC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 8: skip mode penalty, inter MB mode favor (all format mode)

45.7.160 VENC segment 8: penalty value register (VENC_SWREGx)

Address offset: 0x4F0 + 0x4 * (x - 316), (x = 316 to 317)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 8: penalty value (all format mode)

45.7.161 VENC segment 9: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG318)

Address offset: 0x4F8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 9: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame (all format mode)

45.7.162 VENC segment 9: skip mode penalty, inter MB mode favor register (VENC_SWREG319)

Address offset: 0x4FC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 9: skip mode penalty, inter MB mode favor (all format mode)

45.7.163 VENC segment 9: penalty value register (VENC_SWREGx)

Address offset: 0x500 + 0x4 * (x - 320), (x = 320 to 321)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 9: penalty value (all format mode)

45.7.164 VENC segment 10: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG322)

Address offset: 0x508

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 10: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame (all format mode)

45.7.165 VENC segment 10: skip mode penalty, inter MB mode favor register (VENC_SWREG323)

Address offset: 0x50C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 10: skip mode penalty, inter MB mode favor (all format mode)

45.7.166 VENC segment 10: penalty value register (VENC_SWREGx)

Address offset: 0x510 + 0x4 * (x - 324), (x = 324 to 325)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 10: penalty value (all format mode)

45.7.167 VENC segment 11: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG326)

Address offset: 0x518

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 11: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame (all format mode)

45.7.168 VENC segment 11: skip mode penalty, inter MB mode favor register (VENC_SWREG327)

Address offset: 0x51C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 11: skip mode penalty, inter MB mode favor (all format mode)

45.7.169 VENC segment 11: penalty value register (VENC_SWREGx)

Address offset: 0x520 + 0x4 * (x - 328), (x = 328 to 329)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 11: penalty value (all format mode)

45.7.170 VENC segment 12: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG330)

Address offset: 0x528

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 12: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame (all format mode)

45.7.171 VENC segment 12: skip mode penalty, inter MB mode favor register (VENC_SWREG331)

Address offset: 0x52C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 12: skip mode penalty, inter MB mode favor (all format mode)

45.7.172 VENC segment 12: penalty value register (VENC_SWREGx)

Address offset: 0x530 + 0x4 * (x - 332), (x = 332 to 333)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 12: penalty value (all format mode)

45.7.173 VENC segment 13: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG334)

Address offset: 0x538

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 13: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame (all format mode)

45.7.174 VENC segment 13: skip mode penalty, inter MB mode favor register (VENC_SWREG335)

Address offset: 0x53C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 13: skip mode penalty, inter MB mode favor (all format mode)

45.7.175 VENC segment 13: penalty value register (VENC_SWREGx)

Address offset: 0x540 + 0x4 * (x - 336), (x = 336 to 337)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 13: penalty value (all format mode)

45.7.176 VENC segment 14: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG338)

Address offset: 0x548

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 14: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame (all format mode)

45.7.177 VENC segment 14: skip mode penalty, inter MB mode favor register (VENC_SWREG339)

Address offset: 0x54C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 14: skip mode penalty, inter MB mode favor (all format mode)

45.7.178 VENC segment 14: penalty value register (VENC_SWREGx)

Address offset: 0x550 + 0x4 * (x - 340), (x = 340 to 341)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 14: penalty value (all format mode)

45.7.179 VENC segment 15: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG342)

Address offset: 0x558

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 15: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame (all format mode)

45.7.180 VENC segment 15: skip mode penalty, inter MB mode favor register (VENC_SWREG343)

Address offset: 0x55C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 15: skip mode penalty, inter MB mode favor (all format mode)

45.7.181 VENC segment 15: penalty value register (VENC_SWREGx)

Address offset: 0x560 + 0x4 * (x - 344), (x = 344 to 345)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 15: penalty value (all format mode)

45.7.182 VENC segment 16: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG346)

Address offset: 0x568

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 16: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame (all format mode)

45.7.183 VENC segment 16: skip mode penalty, inter MB mode favor register (VENC_SWREG347)

Address offset: 0x56C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 16: skip mode penalty, inter MB mode favor (all format mode)

45.7.184 VENC segment 16: penalty value register (VENC_SWREGx)

Address offset: 0x570 + 0x4 * (x - 348), (x = 348 to 349)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 16: penalty value (all format mode)

45.7.185 VENC segment 17: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG350)

Address offset: 0x578

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 17: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame (all format mode)

45.7.186 VENC segment 17: skip mode penalty, inter MB mode favor register (VENC_SWREG351)

Address offset: 0x57C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 17: skip mode penalty, inter MB mode favor (all format mode)

45.7.187 VENC segment 17: penalty value register (VENC_SWREGx)

Address offset: 0x580 + 0x4 * (x - 352), (x = 352 to 353)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 17: penalty value (all format mode)

45.7.188 VENC segment 18: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG354)

Address offset: 0x588

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 18: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame (all format mode)

45.7.189 VENC segment 18: skip mode penalty, inter MB mode favor register (VENC_SWREG355)

Address offset: 0x58C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 18: skip mode penalty, inter MB mode favor (all format mode)

45.7.190 VENC segment 18: penalty value register (VENC_SWREGx)

Address offset: 0x590 + 0x4 * (x - 356), (x = 356 to 357)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 18: penalty value (all format mode)

45.7.191 VENC segment 19: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG358)

Address offset: 0x598

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 19: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame (all format mode)

45.7.192 VENC segment 19: skip mode penalty, inter MB mode favor register (VENC_SWREG359)

Address offset: 0x59C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 19: skip mode penalty, inter MB mode favor (all format mode)

45.7.193 VENC segment 19: penalty value register (VENC_SWREGx)

Address offset: 0x5A0 + 0x4 * (x - 360), (x = 360 to 361)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 19: penalty value (all format mode)

45.7.194 VENC segment 20: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG362)

Address offset: 0x5A8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 20: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame (all format mode)

45.7.195 VENC segment 20: skip mode penalty, inter MB mode favor register (VENC_SWREG363)

Address offset: 0x5AC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 20: skip mode penalty, inter MB mode favor (all format mode)

45.7.196 VENC segment 20: penalty value register (VENC_SWREGx)

Address offset: 0x5B0 + 0x4 * (x - 364), (x = 364 to 365)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 20: penalty value (all format mode)

45.7.197 VENC segment 21: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG366)

Address offset: 0x5B8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 21: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame (all format mode)

45.7.198 VENC segment 21: skip mode penalty, inter MB mode favor register (VENC_SWREG367)

Address offset: 0x5BC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 21: skip mode penalty, inter MB mode favor (all format mode)

45.7.199 VENC segment 21: penalty value register (VENC_SWREGx)

Address offset: 0x5C0 + 0x4 * (x - 368), (x = 368 to 369)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 21: penalty value (all format mode)

45.7.200 VENC segment 22: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG370)

Address offset: 0x5C8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 22: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame (all format mode)

45.7.201 VENC segment 22: skip mode penalty, inter MB mode favor register (VENC_SWREG371)

Address offset: 0x5CC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 22: skip mode penalty, inter MB mode favor (all format mode)

45.7.202 VENC segment 22: penalty value register (VENC_SWREGx)

Address offset: 0x5D0 + 0x4 * (x - 372), (x = 372 to 373)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 22: penalty value (all format mode)

45.7.203 VENC segment 23: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG374)

Address offset: 0x5D8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 23: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame (all format mode)

45.7.204 VENC segment 23: skip mode penalty, inter MB mode favor register (VENC_SWREG375)

Address offset: 0x5DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 23: skip mode penalty, inter MB mode favor (all format mode)

45.7.205 VENC segment 23: penalty value register (VENC_SWREGx)

Address offset: 0x5E0 + 0x4 * (x - 376), (x = 376 to 377)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 23: penalty value (all format mode)

45.7.206 VENC segment 24: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG378)

Address offset: 0x5E8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 24: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame (all format mode)

45.7.207 VENC segment 24: skip mode penalty, inter MB mode favor register (VENC_SWREG379)

Address offset: 0x5EC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 24: skip mode penalty, inter MB mode favor (all format mode)

45.7.208 VENC segment 24: penalty value register (VENC_SWREGx)

Address offset: 0x5F0 + 0x4 * (x - 380), (x = 380 to 381)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 24: penalty value (all format mode)

45.7.209 VENC segment 25: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG382)

Address offset: 0x5F8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 25: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame (all format mode)

45.7.210 VENC segment 25: skip mode penalty, inter MB mode favor register (VENC_SWREG383)

Address offset: 0x5FC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 25: skip mode penalty, inter MB mode favor (all format mode)

45.7.211 VENC segment 25: penalty value register (VENC_SWREGx)

Address offset: 0x600 + 0x4 * (x - 384), (x = 384 to 385)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 25: penalty value (all format mode)

45.7.212 VENC segment 26: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG386)

Address offset: 0x608

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 26: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame (all format mode)

45.7.213 VENC segment 26: skip mode penalty, inter MB mode favor register (VENC_SWREG387)

Address offset: 0x60C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 26: skip mode penalty, inter MB mode favor (all format mode)

45.7.214 VENC segment 26: penalty value register (VENC_SWREGx)

Address offset: 0x610 + 0x4 * (x - 388), (x = 388 to 389)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 26: penalty value (all format mode)

45.7.215 VENC segment 27: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG390)

Address offset: 0x618

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 27: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame (all format mode)

45.7.216 VENC segment 27: skip mode penalty, inter MB mode favor register (VENC_SWREG391)

Address offset: 0x61C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 27: skip mode penalty, inter MB mode favor (all format mode)

45.7.217 VENC segment 27: penalty value register (VENC_SWREGx)

Address offset: 0x620 + 0x4 * (x - 392), (x = 392 to 393)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 27: penalty value (all format mode)

45.7.218 VENC segment 28: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG394)

Address offset: 0x628

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 28: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame (all format mode)

45.7.219 VENC segment 28: skip mode penalty, inter MB mode favor register (VENC_SWREG395)

Address offset: 0x62C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 28: skip mode penalty, inter MB mode favor (all format mode)

45.7.220 VENC segment 28: penalty value register (VENC_SWREGx)

Address offset: 0x630 + 0x4 * (x - 396), (x = 396 to 397)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 28: penalty value (all format mode)

45.7.221 VENC segment 29: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG398)

Address offset: 0x638

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 29: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame (all format mode)

45.7.222 VENC segment 29: skip mode penalty, inter MB mode favor register (VENC_SWREG399)

Address offset: 0x63C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 29: skip mode penalty, inter MB mode favor (all format mode)

45.7.223 VENC segment 29: penalty value register (VENC_SWREGx)

Address offset: 0x640 + 0x4 * (x - 400), (x = 400 to 401)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 29: penalty value (all format mode)

45.7.224 VENC segment 30: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG402)

Address offset: 0x648

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 30: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame (all format mode)

45.7.225 VENC segment 30: skip mode penalty, inter MB mode favor register (VENC_SWREG403)

Address offset: 0x64C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 30: skip mode penalty, inter MB mode favor (all format mode)

45.7.226 VENC segment 30: penalty value register (VENC_SWREGx)

Address offset: 0x650 + 0x4 * (x - 404), (x = 404 to 405)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 30: penalty value (all format mode)

45.7.227 VENC segment 31: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame register (VENC_SWREG406)

Address offset: 0x658

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 31: intra 4x4 previous mode favor, intra 16x16 mode favor, penalty value for second reference frame (all format mode)

45.7.228 VENC segment 31: skip mode penalty, inter MB mode favor register (VENC_SWREG407)

Address offset: 0x65C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 31: skip mode penalty, inter MB mode favor (all format mode)

45.7.229 VENC segment 31: penalty value register (VENC_SWREGx)

Address offset: 0x660 + 0x4 * (x - 408), (x = 408 to 409)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: segment 31: penalty value (all format mode)

45.7.230 VENC MBRC control, QP, offset, enable register (VENC_SWREG410)

Address offset: 0x668

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: MBRC control (QP, offset, enable) (all format mode)

45.7.231 VENC gain of MB QP delta. 8.8 format register (VENC_SWREG411)

Address offset: 0x66C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: gain of MB QPdelta. 8.8 format (all format mode)

45.7.232 VENC average of MB complexity register (VENC_SWREG412)

Address offset: 0x670

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: average of MB complexity (all format mode)

45.7.233 VENC reference compression control register (VENC_SWREG413)

Address offset: 0x674

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: average of MB complexity (all format mode)

45.7.234 VENC base address for reference luma register (VENC_SWREG414)

Address offset: 0x678

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: average of MB complexity (all format mode)

45.7.235 VENC base address for reference chroma register (VENC_SWREG415)

Address offset: 0x67C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: average of MB complexity (all format mode)

45.7.236 VENC base address for reconstructed luma register (VENC_SWREG416)

Address offset: 0x680

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: average of MB complexity (all format mode)

45.7.237 VENC base address for reconstructed chroma register (VENC_SWREG417)

Address offset: 0x684

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: average of MB complexity (all format mode)

45.7.238 VENC base address for second reference luma register (VENC_SWREG418)

Address offset: 0x688

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: average of MB complexity (all format mode)

45.7.239 VENC base address for second reference chroma register (VENC_SWREG419)

Address offset: 0x68C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: average of MB complexity (all format mode)

45.7.240 VENC limit of chroma RFC buffer register (VENC_SWREG420)

Address offset: 0x690

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: average of MB complexity (all format mode)

45.7.241 VENC reorder control register (VENC_SWREG421)

Address offset: 0x694

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Reorder control (all format mode)

45.7.242 VENC AXI read ID register (VENC_SWREG422)

Address offset: 0x698

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: AXI Read ID (all format mode)

45.7.243 VENC base address MSB for reference luma compression table register (VENC_SWREG423)

Address offset: 0x69C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: AXI Read ID (all format mode)

45.7.244 VENC base address MSB for reference chroma compression table register (VENC_SWREG424)

Address offset: 0x6A0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: AXI Read ID (all format mode)

45.7.245 VENC base address MSB for reconstructed luma compression table register (VENC_SWREG425)

Address offset: 0x6A4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: AXI Read ID (all format mode)

45.7.246 VENC base address for reconstructed chroma compression table register (VENC_SWREG426)

Address offset: 0x6A8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Base address for reconstructed chroma compression table (all format mode)

45.7.247 VENC base address MSB for second reference luma compression table register (VENC_SWREG427)

Address offset: 0x6AC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Base address MSB for second reference luma compression table (all format mode)

45.7.248 VENC base address MSB for second reference chroma compression table register (VENC_SWREG428)

Address offset: 0x6B0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Base address MSB for second reference chroma compression table (all format mode)

45.7.249 VENC high 32 bits of base address for output stream data register (VENC_SWREG429)

Address offset: 0x6B4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: high 32 bits of Base address for output stream data (all format mode)

45.7.250 VENC high 32 bits of base address for output control data register (VENC_SWREG430)

Address offset: 0x6B8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: high 32 bits of Base address for output control data (all format mode)

45.7.251 VENC high 32 bits of base address for reference luma register (VENC_SWREG431)

Address offset: 0x6BC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: high 32 bits of Base address for reference luma (all format mode)

45.7.252 VENC high 32 bits of base address for reference chroma register (VENC_SWREG432)

Address offset: 0x6C0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: high 32 bits of Base address for reference chroma (all format mode)

45.7.253 VENC high 32 bits of base address for reconstructed luma register (VENC_SWREG433)

Address offset: 0x6C4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: high 32 bits of Base address for reconstructed luma (all format mode)

45.7.254 VENC high 32 bits of base address for reconstructed chroma register (VENC_SWREG434)

Address offset: 0x6C8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: high 32 bits of Base address for reconstructed chroma (all format mode)

45.7.255 VENC high 32 bits of base address for input picture luma register (VENC_SWREG435)

Address offset: 0x6CC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: high 32 bits of Base address for input picture luma (all format mode)

45.7.256 VENC high 32 bits of base address for input picture cb register (VENC_SWREG436)

Address offset: 0x6D0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: high 32 bits of Base address for input picture cb (all format mode)

45.7.257 VENC high 32 bits of base address for input picture cr register (VENC_SWREG437)

Address offset: 0x6D4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: high 32 bits of Base address for input picture cr (all format mode)

45.7.258 VENC high 32 bits of base address for second reference luma register (VENC_SWREG438)

Address offset: 0x6D8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: high 32 bits of Base address for second reference luma (all format mode)

45.7.259 VENC high 32 bits of base address for second reference chroma register (VENC_SWREG439)

Address offset: 0x6DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: high 32 bits of Base address for second reference chroma (all format mode)

45.7.260 VENC high 32 bits of H264 secondary ref pic base register (VENC_SWREGx)

Address offset: 0x6E0 + 0x4 * (x - 440), (x = 440 to 441)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: high 32 bits of H264 secondary ref pic base (all format mode)

45.7.261 VENC high 32 bits of base address for next pic luminance register (VENC_SWREG442)

Address offset: 0x6E8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: high 32 bits of Base address for next pic luminance (all format mode)

45.7.262 VENC high 32 bits of base address for cabac context tables H264 register (VENC_SWREG443)

Address offset: 0x6EC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: high 32 bits of Base address for cabac context tables (H264) or probability tables (all format mode)

45.7.263 VENC high 32 bits of base address for MV output writing register (VENC_SWREG444)

Address offset: 0x6F0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: high 32 bits of Base address for MV output writing (all format mode)

45.7.264 VENC high 32 bits of base address for output of down-scaled encoder image in YUYV 4:2:2 format register (VENC_SWREG449)

Address offset: 0x704

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: high 32 bits of Base address for output of down-scaled encoder image in YUYV 4:2:2 format (all format mode)

45.7.265 VENC low-latency control register (VENC_SWREG497)

Address offset: 0x7C4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Low latency control (all format mode)

45.7.266 VENC encoder line buffer offset register (VENC_SWREG498)

Address offset: 0x7C8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWREG_FIELD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWREG_FIELD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SWREG_FIELD[31:0]**: Low latency control (all format mode)

45.7.267 VENC register map

Table 390. VENC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	VENC_SWREG0	SWREG_FIELD[31:0]																															
	Reset value	0	1	1	0	1	1	1	0	0	1	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	VENC_SWREG1	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	VENC_SWREG2	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0x00C	VENC_SWREG3	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	Reserved	Reserved																															
0x014	VENC_SWREG5	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	VENC_SWREG6	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	VENC_SWREG7	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	VENC_SWREG8	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	VENC_SWREG9	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	VENC_SWREG10	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	VENC_SWREG11	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	VENC_SWREG12	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	VENC_SWREG13	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 390. VENC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x038	VENC_SWREG14	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	VENC_SWREG15	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x040	VENC_SWREG16	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	VENC_SWREG17	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048	VENC_SWREG18	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	VENC_SWREG19	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	VENC_SWREG20	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054	VENC_SWREG21	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x058	VENC_SWREG22	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x05C	VENC_SWREG23	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x060	VENC_SWREG24	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x064	VENC_SWREG25	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x068	VENC_SWREG26	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x06C	VENC_SWREG27	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x070	VENC_SWREG28	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x074	VENC_SWREG29	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x078	VENC_SWREG30	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07C	VENC_SWREG31	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x080	VENC_SWREG32	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084	VENC_SWREG33	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x088	VENC_SWREG34	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08C	VENC_SWREG35	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x090	VENC_SWREG36	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 390. VENC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x094	VENC_SWREG37	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x098	VENC_SWREG38	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x09C	VENC_SWREG39	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A0	VENC_SWREG40	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A4	VENC_SWREG41	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A8	VENC_SWREG42	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0AC	VENC_SWREG43	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B0	VENC_SWREG44	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B4	VENC_SWREG45	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B8	VENC_SWREG46	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0BC	VENC_SWREG47	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C0	VENC_SWREG48	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C4	VENC_SWREG49	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C8	VENC_SWREG50	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0CC	VENC_SWREG51	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0D0	VENC_SWREG52	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0D4	VENC_SWREG53	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0D8	VENC_SWREG54	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0DC	VENC_SWREG55	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0E0	VENC_SWREG56	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0E4	VENC_SWREG57	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0E8	VENC_SWREG58	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0EC	VENC_SWREG59	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 390. VENC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0F0	VENC_SWREG60	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0F4	VENC_SWREG61	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0F8	VENC_SWREG62	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0FC	VENC_SWREG63	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	1	1	0	1	0	0	1	1	0	0	0	1	0	0	0	1	0	0	1	1	1	1	0	0	0	0	0	0	0
0x100	VENC_SWREG64	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x104	VENC_SWREG65	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x108	VENC_SWREG66	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10C	VENC_SWREG67	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x110	VENC_SWREG68	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x114	VENC_SWREG69	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x118	VENC_SWREG70	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x11C	VENC_SWREG71	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x120	VENC_SWREG72	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x124	VENC_SWREG73	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x128	VENC_SWREG74	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x12C	VENC_SWREG75	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x130	VENC_SWREG76	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x134	VENC_SWREG77	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x138	VENC_SWREG78	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x13C	VENC_SWREG79	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x140	VENC_SWREG80	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x144	VENC_SWREG81	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x148	VENC_SWREG82	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 390. VENC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x14C	VENC_SWREG83	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x150	VENC_SWREG84	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x154	VENC_SWREG85	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x158	VENC_SWREG86	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x15C	VENC_SWREG87	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x160	VENC_SWREG88	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x164	VENC_SWREG89	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x168	VENC_SWREG90	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x16C	VENC_SWREG91	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x170	VENC_SWREG92	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x174	VENC_SWREG93	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x178	VENC_SWREG94	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x17C	VENC_SWREG95	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x180	VENC_SWREG96	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x184	VENC_SWREG97	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x188 + 0x4*(x-98), x=98 to 126	VENC_SWREGx	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1FC	VENC_SWREG127	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x200	VENC_SWREG128	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x204	VENC_SWREG129	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x208 + 0x4*(x-130), x=130 to 158	VENC_SWREGx	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x27C	VENC_SWREG159	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x280 - 0x398	Reserved	Reserved																															
0x39C	VENC_SWREG231	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 390. VENC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x3A0	VENC_SWREG232	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3A4	VENC_SWREG233	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3A8 - 0x3AC	Reserved	Reserved																															
0x3B0	VENC_SWREG236	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3B4	VENC_SWREG237	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3B8	VENC_SWREG238	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3BC - 0x3FC	Reserved	Reserved																															
0x400	VENC_SWREG256	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x404	VENC_SWREG257	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x408	VENC_SWREG258	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40C	VENC_SWREG259	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x410	VENC_SWREG260	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x414	VENC_SWREG261	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x418	VENC_SWREG262	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x41C	VENC_SWREG263	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x420	VENC_SWREG264	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x424	VENC_SWREG265	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x428	VENC_SWREG266	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x42C	VENC_SWREG267	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x430	VENC_SWREG268	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x434	VENC_SWREG269	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x438	VENC_SWREG270	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x43C	VENC_SWREG271	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 390. VENC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x440	VENC_SWREG272	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x444	VENC_SWREG273	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x448	VENC_SWREG274	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44C	VENC_SWREG275	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x450	VENC_SWREG276	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x454	VENC_SWREG277	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x458	VENC_SWREG278	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x45C	VENC_SWREG279	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x460	VENC_SWREG280	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x464	VENC_SWREG281	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x468	VENC_SWREG282	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x46C	VENC_SWREG283	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x470	VENC_SWREG284	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x474	VENC_SWREG285	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x478	VENC_SWREG286	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x47C	VENC_SWREG287	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x480	VENC_SWREG288	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x484	VENC_SWREG289	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x488	VENC_SWREG290	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48C	VENC_SWREG291	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x490 - 0x494	Reserved	Reserved																															
0x498	VENC_SWREG294	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x49C	VENC_SWREG295	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 390. VENC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x4A0	VENC_SWREG296	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4A4	VENC_SWREG297	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4A8	VENC_SWREG298	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4AC	VENC_SWREG299	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4B0	VENC_SWREG300	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4B4	VENC_SWREG301	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4B8	VENC_SWREG302	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4BC	VENC_SWREG303	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C0	VENC_SWREG304	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C4	VENC_SWREG305	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C8	VENC_SWREG306	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4CC	VENC_SWREG307	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4D0	VENC_SWREG308	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4D4	VENC_SWREG309	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4D8	VENC_SWREG310	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4DC	VENC_SWREG311	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4E0	VENC_SWREG312	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4E4	VENC_SWREG313	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4E8	VENC_SWREG314	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4EC	VENC_SWREG315	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4F0	VENC_SWREG316	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4F4	VENC_SWREG317	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4F8	VENC_SWREG318	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 390. VENC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x4FC	VENC_SWREG319	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x500	VENC_SWREG320	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x504	VENC_SWREG321	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x508	VENC_SWREG322	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50C	VENC_SWREG323	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x510	VENC_SWREG324	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x514	VENC_SWREG325	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x518	VENC_SWREG326	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x51C	VENC_SWREG327	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x520	VENC_SWREG328	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x524	VENC_SWREG329	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x528	VENC_SWREG330	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x52C	VENC_SWREG331	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x530	VENC_SWREG332	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x534	VENC_SWREG333	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x538	VENC_SWREG334	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x53C	VENC_SWREG335	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x540	VENC_SWREG336	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x544	VENC_SWREG337	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x548	VENC_SWREG338	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x54C	VENC_SWREG339	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x550	VENC_SWREG340	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x554	VENC_SWREG341	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 390. VENC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x558	VENC_SWREG342	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x55C	VENC_SWREG343	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x560	VENC_SWREG344	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x564	VENC_SWREG345	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x568	VENC_SWREG346	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x56C	VENC_SWREG347	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x570	VENC_SWREG348	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x574	VENC_SWREG349	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x578	VENC_SWREG350	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x57C	VENC_SWREG351	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x580	VENC_SWREG352	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x584	VENC_SWREG353	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x588	VENC_SWREG354	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x58C	VENC_SWREG355	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x590	VENC_SWREG356	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x594	VENC_SWREG357	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x598	VENC_SWREG358	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x59C	VENC_SWREG359	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5A0	VENC_SWREG360	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5A4	VENC_SWREG361	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5A8	VENC_SWREG362	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5AC	VENC_SWREG363	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5B0	VENC_SWREG364	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 390. VENC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x5B4	VENC_SWREG365	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5B8	VENC_SWREG366	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5BC	VENC_SWREG367	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5C0	VENC_SWREG368	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5C4	VENC_SWREG369	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5C8	VENC_SWREG370	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5CC	VENC_SWREG371	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5D0	VENC_SWREG372	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5D4	VENC_SWREG373	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5D8	VENC_SWREG374	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5DC	VENC_SWREG375	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5E0	VENC_SWREG376	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5E4	VENC_SWREG377	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5E8	VENC_SWREG378	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5EC	VENC_SWREG379	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5F0	VENC_SWREG380	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5F4	VENC_SWREG381	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5F8	VENC_SWREG382	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x5FC	VENC_SWREG383	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x600	VENC_SWREG384	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x604	VENC_SWREG385	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x608	VENC_SWREG386	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x60C	VENC_SWREG387	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 390. VENC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x610	VENC_SWREG388	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x614	VENC_SWREG389	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x618	VENC_SWREG390	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x61C	VENC_SWREG391	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x620	VENC_SWREG392	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x624	VENC_SWREG393	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x628	VENC_SWREG394	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x62C	VENC_SWREG395	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x630	VENC_SWREG396	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x634	VENC_SWREG397	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x638	VENC_SWREG398	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x63C	VENC_SWREG399	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x640	VENC_SWREG400	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x644	VENC_SWREG401	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x648	VENC_SWREG402	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x64C	VENC_SWREG403	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x650	VENC_SWREG404	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x654	VENC_SWREG405	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x658	VENC_SWREG406	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x65C	VENC_SWREG407	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x660	VENC_SWREG408	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x664	VENC_SWREG409	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x668	VENC_SWREG410	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 390. VENC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x66C	VENC_SWREG411	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x670	VENC_SWREG412	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x674	VENC_SWREG413	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x678	VENC_SWREG414	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x67C	VENC_SWREG415	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x680	VENC_SWREG416	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x684	VENC_SWREG417	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x688	VENC_SWREG418	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x68C	VENC_SWREG419	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x690	VENC_SWREG420	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x694	VENC_SWREG421	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x698	VENC_SWREG422	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x69C	VENC_SWREG423	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x6A0	VENC_SWREG424	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x6A4	VENC_SWREG425	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x6A8	VENC_SWREG426	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x6AC	VENC_SWREG427	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x6B0	VENC_SWREG428	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x6B4	VENC_SWREG429	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x6B8	VENC_SWREG430	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x6BC	VENC_SWREG431	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x6C0	VENC_SWREG432	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x6C4	VENC_SWREG433	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 390. VENC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x6C8	VENC_SWREG434	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x6CC	VENC_SWREG435	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x6D0	VENC_SWREG436	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x6D4	VENC_SWREG437	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x6D8	VENC_SWREG438	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x6DC	VENC_SWREG439	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x6E0	VENC_SWREG440	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x6E4	VENC_SWREG441	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x6E8	VENC_SWREG442	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x6EC	VENC_SWREG443	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x6F0	VENC_SWREG444	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x6F4 - 0x700	Reserved	Reserved																															
0x704	VENC_SWREG449	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x708 - 0x7C0	Reserved	Reserved																															
0x7C4	VENC_SWREG497	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x7C8	VENC_SWREG498	SWREG_FIELD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3* for the register boundary addresses.

46 JPEG codec (JPEG)

46.1 JPEG introduction

The hardware 8-bit JPEG codec encodes an uncompressed image data stream or decodes JPEG-compressed image data stream. It also fully manages JPEG headers.

46.2 JPEG codec main features

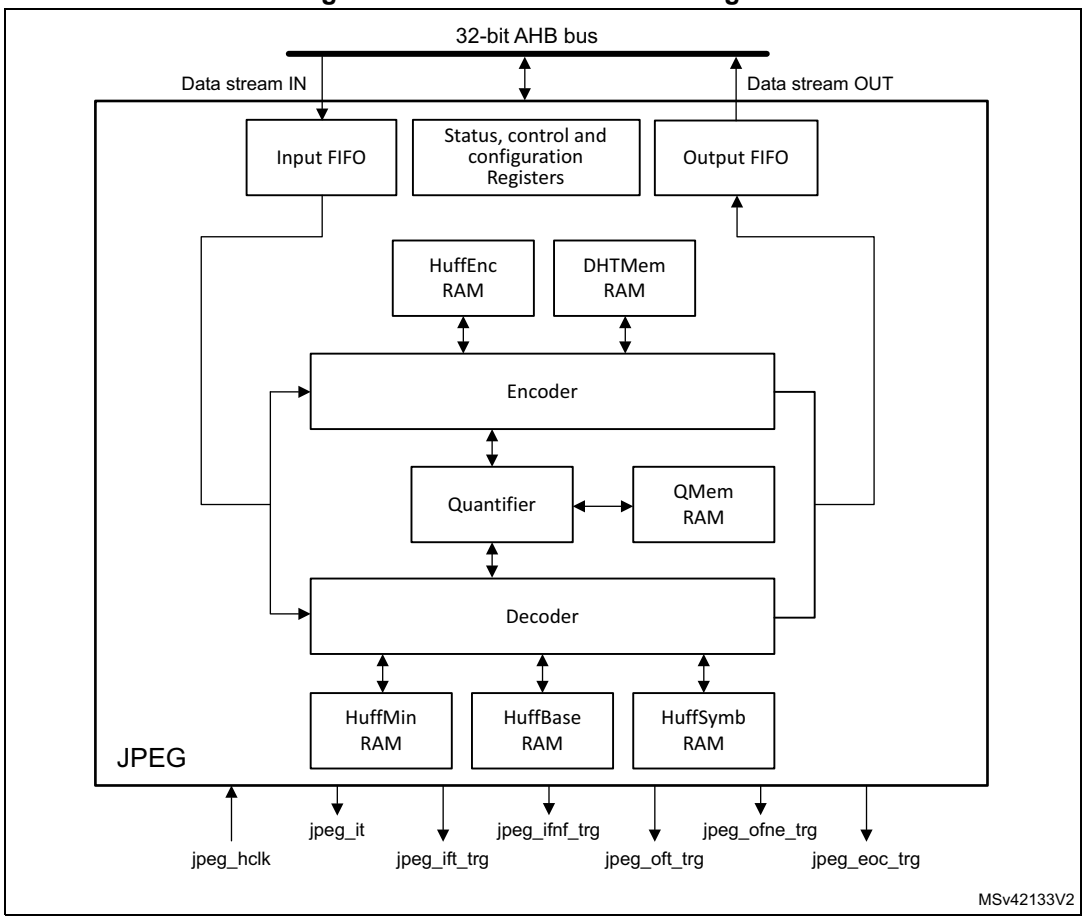
- High-speed fully synchronous operation
- Configurable as encoder or decoder
- Single-clock-per-pixel encode/decode
- RGB, YCbCr, YCMK, and BW (gray scale) image color space support
- 8-bit depth per image component at encode/decode
- JPEG header generator/parser with enable/disable
- Four programmable quantization tables
- Single-clock Huffman coding and decoding
- Fully programmable Huffman tables (two AC and two DC)
- Fully programmable minimum coded unit (MCU)
- Concurrent input and output data stream interfaces

46.3 JPEG codec block functional description

46.3.1 General description

The block diagram of the JPEG codec is shown in *Figure 450*.

Figure 450. JPEG codec block diagram



46.3.2 JPEG internal signals

Table 391 lists the JPEG internal signals.

Table 391. JPEG internal signals

Signal name	Signal type	Description
jpeg_hclk	Digital input	JPEG kernel and register interface clock
jpeg_it	Digital output	JPEG global interrupt
jpeg_ift_trg	Digital output	JPEG input FIFO threshold for DMA trigger
jpeg_ifnf_trg	Digital output	JPEG input FIFO not full for DMA trigger
jpeg_oft_trg	Digital output	JPEG output FIFO threshold for DMA trigger

Table 391. JPEG internal signals (continued)

Signal name	Signal type	Description
jpeg_ofne_trg	Digital output	JPEG output FIFO not empty for DMA trigger
jpeg_eoc_trg	Digital output	JPEG end of conversion for DMA trigger

Table 392. JPEG trigger connections

Trigger name	Direction	Trigger source/destination
jpeg_eoc_trg	Output	hpdma_trigsel[15] gpdma_trigsel[15]
jpeg_ifnf_trg	Output	hpdma_trigsel[16] gpdma_trigsel[16]
jpeg_ift_trg	Output	hpdma_trigsel[17] gpdma_trigsel[17]
jpeg_ofne_trg	Output	hpdma_trigsel[18] gpdma_trigsel[18]
jpeg_oft_trg	Output	hpdma_trigsel[19] gpdma_trigsel[19]

46.3.3 JPEG decoding procedure

The JPEG codec can decode a JPEG stream as defined in the *ISO/IEC 10918-1* specification.

It can optionally parse the JPEG header and accordingly update the JPEG codec registers, the quantization tables, and the Huffman tables.

The JPEG codec is configured in decode mode setting the DE bit (decode enable) of the JPEG_CONFR1 register.

The JPEG decode starts by setting the START bit of the JPEG_CONFR0 register.

The JPEG codec requests data for its input FIFO through generating one of:

- DMA request
- DMA trigger
- Interrupts

DMA generation for input FIFO

A DMA request is generated when the 32-byte input FIFO becomes at least half-empty, that is, when there is free room for writing 16 bytes of data.

The DMA request generation is independent of the START bit of the JPEG_CONFR0 register. If the input FIFO can accept 16 bytes and the DMA for the input FIFO is enabled (setting the IDMAEN bit of the JPEG_CR register), a DMA request is generated regardless of the state of the JPEG codec kernel.

A burst transfer is launched by the DMA to write 16 bytes of data.

Writes are ignored if the input FIFO is full.

At the end of the decoding process, extra bytes may remain in the input FIFO and/or a DMA request may be pending. The FIFO can be flushed by setting the IFF bit (input FIFO flush) of the JPEG_CR register.

Prior to flushing the FIFO, the DMA for the input FIFO must be disabled to prevent unwanted DMA request upon flushing the FIFO.

The consequence of not flushing the FIFO at the end of the decoding process is that any remaining data is taken into the next JPEG decoding.

DMA requests are no more generated once the EOCF flag of the JPEG_SR register is set.

Interrupt or DMA trigger generation for input FIFO

Input FIFO can be managed using interrupts or DMA triggers through two flags according to the FIFO state:

- Input FIFO not full flag: a 32-bit value can be written in.
- Input FIFO threshold flag: 8 words (32 bytes) can be written in.

The interrupt or DMA trigger generation is independent of the START bit of the JPEG_CONFR0 register. The input FIFO flags are generated regardless of the state of the JPEG codec kernel.

Writes are ignored if the input FIFO is full.

At the end of the decoding process, extra bytes may remain in the input FIFO and/or an interrupt request / DMA trigger may be pending. The FIFO can be flushed by setting the IFF bit (Input FIFO Flush) of the JPEG_CR register.

Prior to flushing the FIFO:

- The interrupts for the input FIFO must be disabled to prevent an unwanted interrupt request upon flushing the FIFO.
- The DMA channel must be stopped to prevent an unwanted DMA trigger.

The consequence of not flushing the FIFO at the end of the decoding process is that any remaining data is taken into the next JPEG decoding.

Header parsing

The header parsing can be activated by setting the HDR bit of the JPEG_CONFR1 register.

The JPEG header parser supports all markers relevant to the JPEG baseline algorithm indicated in *Annex B* of the *ISO/IEC 10918-1*.

When parsing a supported marker, the JPEG header parser extracts the required parameters and stores them in shadow registers. At the end of the parsing the JPEG codec registers are updated.

If a DQT marker segment is located, quantization data associated with it is written into the quantization table memory.

If a DHT marker segment is located, the Huffman table data associated with it is converted into three different table formats (HuffMin, HuffBase, and HuffSymb) and stored in their respective memories.

Once the parsing operation is completed, the HPDF (header parsing done flag) bit of the JPEG_SR register is set. An interrupt is generated if the EHPIE (end of header parsing interrupt enable) bit of the JPEG_CR register is set.

JPEG decoding

Once the JPEG header is parsed or the JPEG codec registers and memories are properly programmed, the incoming data stream is decoded and the resulting MCUs are sent to the output FIFO.

When decoding two images successively, the START bit of the JPEG_CONFR0 register must be set again (even if already 1) after the header processing of the second image is completed.

DMA generation for output FIFO

A DMA request is generated when the 32-byte output FIFO becomes at least half-full, that is, when there are at least 16 bytes of data.

A burst transfer is launched by the DMA to read 16 bytes of data.

Reads return 0 if the output FIFO is empty.

Once the decoding process is done, no extra bytes must remain in the output FIFO and no DMA request must be pending as the JPEG decoding generates blocks of 64 bytes.

In case of abort of the JPEG codec operations by resetting the START bit of the JPEG_CONFR0 register, the output FIFO can be flushed by setting the OFF bit (input FIFO flush) of the JPEG_CR register.

Prior to flushing the FIFO, the DMA for the output FIFO must be disabled to prevent an unwanted DMA request upon flushing the FIFO.

Interrupt or DMA trigger generation for output FIFO

The output FIFO can be managed using interrupts or DMA triggers through two flags according to the FIFO state:

- Output FIFO not empty flag: a 32-bit value can be read out.
- Output FIFO threshold flag: 8 words (32 bytes) can be read out.

Reads return 0 if the output FIFO is empty.

In case of abort of the JPEG codec operations by resetting the START bit of the JPEG_CONFR0 register, the output FIFO can be flushed. If the FIFO needs to be flushed, it must be done by software setting the FF bit (FIFO flush) of the JPEG_CR register.

Prior to flushing the FIFO:

- The interrupts for the output FIFO must be disabled to prevent unwanted interrupt request upon flushing the FIFO.
- The DMA channel must be stopped to prevent an unwanted DMA trigger.

The output FIFO must be flushed at the end of processing before any JPEG configuration change.

46.3.4 JPEG encoding procedure

The JPEG codec can encode a JPEG stream as defined in the *ISO/IEC 10918-1* specification.

It can optionally generate the JPEG header.

The JPEG codec is configured in encode mode resetting the DE bit (decode enable) of the JPEG_CONFR1 register.

The configuration used for encoding the JPEG must be loaded in the JPEG codec:

- JPEG codec configuration registers
- Quantization tables
- Huffman tables

The JPEG codec is started setting the START bit of the JPEG_CONFR0 register.

Once the JPEG codec has been started, it requests data for its input FIFO generating one of:

- DMA request
- DMA trigger
- Interrupts

DMA generation for input FIFO

A DMA request is generated when the 32-byte input FIFO becomes at least half-empty, that is, when there is free room for writing 16 bytes of data.

The DMA request generation is independent of the START bit of the JPEG_CONFR0 register. If the input FIFO can accept 16 bytes and the DMA for the input FIFO is enabled (setting the IDMAEN bit of the JPEG_CR register), a DMA request is generated regardless of the state of the JPEG codec kernel.

A burst transfer is launched by the DMA to write 16 bytes of data.

Writes are ignored if the input FIFO is full.

At the end of the encoding process, extra bytes may remain in the input FIFO and/or a DMA request may be pending. The FIFO can be flushed by setting the IFF bit (input FIFO flush) of the JPEG_CR register.

Prior to flushing the FIFO, the DMA for the input FIFO must be disabled to prevent unwanted DMA request upon flushing the FIFO.

The consequence of not flushing the FIFO at the end of the encoding process is that any remaining data is taken into the next JPEG encoding.

The DMA requests are no more generated once the EOCF flag of the JPEG_SR register is set.

Interrupt or DMA trigger generation for input FIFO

Input FIFO can be managed using interrupts or DMA triggers through two flags according to the FIFO state:

- Input FIFO not full flag: a 32-bit value can be written in.
- Input FIFO threshold flag: 8 words (32 bytes) can be written in.

The interrupt or DMA trigger generation is independent of the START bit of the JPEG_CONFR0 register. The input FIFO flags are generated regardless of the state of the JPEG codec kernel.

Writes are ignored if the input FIFO is full.

At the end of the encoding process, extra bytes may remain in the input FIFO and/or an interrupt request / DMA trigger may be pending. The FIFO can be flushed by setting the IFF bit (input FIFO flush) of the JPEG_CR register.

Prior to flushing the FIFO:

- The interrupts for the input FIFO must be disabled to prevent an unwanted interrupt request upon flushing the FIFO.
- The DMA channel must be stopped to prevent an unwanted DMA trigger.

The consequence of not flushing the FIFO at the end of the encoding process is that any remaining data is taken into the next JPEG encoding.

JPEG encoding

Once the JPEG header is generated, the incoming MCUs are encoded and the resulting data stream sent to the output FIFO.

DMA generation for output FIFO

A DMA request is generated when the 32-byte output FIFO becomes at least half-full, that is, when there are at least 16 bytes of data.

A burst transfer is launched by the DMA to read 16 bytes of data.

Read returns 0 if the output FIFO is empty.

At the end of the encoding process, the last bytes may remain in the output FIFO as the stream padding may not be on 16 bytes.

These additional bytes must be managed by the CPU using the output FIFO not empty flag.

In case of abort of the JPEG codec operations by resetting the START bit of the JPEG_CONFR0 register, the output FIFO can be flushed. The FIFO can be flushed by setting the OFF bit (output FIFO flush) of the JPEG_CR register.

Prior to flushing the FIFO, the DMA for the input FIFO must be disabled to prevent an unwanted DMA request upon flushing the FIFO.

Interrupt or DMA trigger generation for output FIFO

Output FIFO can be managed using interrupts or DMA triggers through two flags according to the FIFO state:

- Output FIFO not empty flag: a 32-bit value can be read out.
- Output FIFO threshold flag: 8 words (32 bytes) can be read out.

Reads return 0 if the output FIFO is empty.

In case of abort of the JPEG codec operations by resetting the START bit of the JPEG_CONFR0 register, the output FIFO can be flushed. The FIFO can be flushed by setting the FF bit (FIFO flush) of the JPEG_CR register.

Prior to flushing the FIFO:

- The interrupts for the output FIFO must be disabled to prevent an unwanted interrupt request upon flushing the FIFO.
- The DMA channel must be stopped to prevent an unwanted DMA trigger.

The output FIFO must be flushed at the end of processing before any JPEG configuration change.

The EOCF bit (end of conversion flag) of the JPEG_SR register can only be cleared when the output FIFO is empty.

Clearing either of the HDR bit (header processing) of the JPEG_CONFR1 register and the JCEN bit (JPEG codec enable) of the JPEG_CR register is allowed only when the EOCF bit of the JPEG_SR register is cleared.

46.4 JPEG codec interrupts

An interrupt can be produced on the following events:

- input FIFO threshold reached
- input FIFO not full
- output FIFO threshold reached
- output FIFO not empty
- end of conversion
- header parsing done

Separate interrupt enable bits are available for flexibility.

Table 393. JPEG codec interrupt requests

Interrupt event	Event flag	Enable Control bit
Input FIFO threshold reached	IFTF	IFTIE
Input FIFO not full	IFNFF	IFNFIE
Output FIFO threshold reached	OFTF	OFTIE
Output FIFO not empty	OFNEF	OFNEIE
End of conversion	EOCF	EOCIE
Header parsing done	HPDF	HPDIE

46.5 JPEG codec registers

46.5.1 JPEG codec control register (JPEG_CONFR0)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	START
															w

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **START**: Start
This bit start or stop the encoding or decoding process.
0: Stop/abort
1: Start
Note: Reads always return 0.

46.5.2 JPEG codec configuration register 1 (JPEG_CONFR1)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
YSIZE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDR	NS[1:0]		COLSPACE[1:0]		DE	Res.	NF[1:0]	
							rw	rw	rw	rw	rw	rw		rw	rw

Bits 31:16 **YSIZE[15:0]**: Y Size
This field defines the number of lines in source image.

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **HDR**: Header processing
This bit enables the header processing (generation/parsing).
0: Disable
1: Enable

Bits 7:6 **NS[1:0]**: Number of components for scan
This field defines the number of components minus 1 for scan header marker segment.

Bits 5:4 **COLSPACE[1:0]**: Color space
This filed defines the number of quantization tables minus 1 to insert in the output stream.
00: Grayscale (1 quantization table)
01: YUV (2 quantization tables)
10: RGB (3 quantization tables)
11: CMYK (4 quantization tables)

Bit 3 **DE**: Codec operation as coder or decoder
This bit selects the code or decode process
0: Code
1: Decode

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **NF[1:0]**: Number of color components
This field defines the number of color components minus 1.
00: Grayscale (1 color component)
01: - (2 color components)
10: YUV or RGB (3 color components)
11: CMYK (4 color components)

46.5.3 JPEG codec configuration register 2 (JPEG_CONFR2)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	NMCU[25:16]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NMCU[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:0 **NMCU[25:0]**: Number of MCUs

For encoding: this field defines the number of MCU units minus 1 to encode.

For decoding: this field indicates the number of complete MCU units minus 1 to be decoded (this field is updated after the JPEG header parsing). If the decoded image size has not a X or Y size multiple of 8 or 16 (depending on the sub-sampling process), the resulting incomplete or empty MCU must be added to this value to get the total number of MCUs generated.

46.5.4 JPEG codec configuration register 3 (JPEG_CONFR3)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
XSIZE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:16 **XSIZE[15:0]**: X size

This field defines the number of pixels per line.

Bits 15:0 Reserved, must be kept at reset value.

46.5.5 JPEG codec configuration register x (JPEG_CONFRx)

Address offset: 0x000 + 0x4 * x, (x = 4 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HSF[3:0]				VSF[3:0]				NB[3:0]				QT[1:0]		HA	HD
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **HSF[3:0]**: Horizontal sampling factor
Horizontal sampling factor for component {x-4}.

Bits 11:8 **VSF[3:0]**: Vertical sampling factor
Vertical sampling factor for component {x-4}.

Bits 7:4 **NB[3:0]**: Number of blocks
Number of data units minus 1 that belong to a particular color in the MCU.

Bits 3:2 **QT[1:0]**: Quantization table
Selects quantization table used for component {x-4}.
00: Quantization table 0
01: Quantization table 1
10: Quantization table 2
11: Quantization table 3

Bit 1 **HA**: Huffman AC
Selects the Huffman table for encoding AC coefficients.
0: Huffman AC table 0
1: Huffman AC table 1

Bit 0 **HD**: Huffman DC
Selects the Huffman table for encoding DC coefficients.
0: Huffman DC table 0
1: Huffman DC table 1

46.5.6 JPEG control register (JPEG_CR)

Address offset: 0x030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	OFF	IFF	ODMAEN	IDMAEN	Res.	Res.	Res.	Res.	HPDIE	EOCIE	OFNEIE	OFTIE	IFNFIE	IFTIE	JCEN
	w	w	rw	rw					rw	rw	rw	rw	rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **OFF**: Output FIFO flush
This bit flushes the output FIFO.
0: No effect
1: Output FIFO is flushed
Note: Reads always return 0.

Bit 13 **IFF**: Input FIFO flush
This bit flushes the input FIFO.
0: No effect
1: Input FIFO is flushed
Note: Reads always return 0.

Bit 12 **ODMAEN**: Output DMA enable
Enables DMA request generation for the output FIFO.
0: Disabled
1: Enabled

Bit 11 **IDMAEN**: Input DMA enable
Enables DMA request generation for the input FIFO.
0: Disabled
1: Enabled

Bits 10:7 Reserved, must be kept at reset value.

Bit 6 **HPDIE**: Header parsing done interrupt enable
This bit enables interrupt generation upon the completion of the header parsing operation.
0: Disabled
1: Enabled

Bit 5 **EOCIE**: End of conversion interrupt enable
This bit enables interrupt generation at the end of conversion.
0: Disabled
1: Enabled

Bit 4 **OFNEIE**: Output FIFO not empty interrupt enable
This bit enables interrupt generation when the output FIFO is not empty.
0: Disabled
1: Enabled

Bit 3 **OFTIE**: Output FIFO threshold interrupt enable
This bit enables interrupt generation when the output FIFO reaches a threshold.
0: Disabled
1: Enabled

Bit 2 **IFNFIE**: Input FIFO not full interrupt enable
This bit enables interrupt generation when the input FIFO is not empty.
0: Disabled
1: Enabled

Bit 1 **IFTIE**: Input FIFO threshold interrupt enable
This bit enables interrupt generation when the input FIFO reaches a threshold.
0: Disabled
1: Enabled

Bit 0 **JCEN**: JPEG core enable
This bit enables the JPEG codec core.
0: Disabled (internal registers are reset).
1: Enabled (internal registers are accessible).

46.5.7 JPEG status register (JPEG_SR)

Address offset: 0x034

Reset value: 0x0000 0006

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COF	HPDF	EOCF	OFNEF	OFTF	IFNFF	IFTF	Res.
								r	r	r	r	r	r	r	

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **COF**: Codec operation flag
This bit flags code/decode operation in progress.
0: Not in progress
1: In progress

Bit 6 **HPDF**: Header parsing done flag
In decode mode, this bit flags the completion of header parsing and updating internal registers.
0: Not completed
1: Completed

Bit 5 **EOCF**: End of conversion flag
This bit flags the completion of encode/decode process and data transfer to the output FIFO.
0: Not completed
1: Completed

Bit 4 **OFNEF**: Output FIFO not empty flag

This bit flags that data is available in the output FIFO. This flag must not be considered when using DMA.

0: Empty (data not available)

1: Not empty (data available)

Bit 3 **OFTF**: Output FIFO threshold flag

This bit flags that the amount of data in the output FIFO reaches or exceeds a threshold. This flag must not be considered when using DMA.

0: Below threshold

1: At or above threshold

Bit 2 **IFNFF**: Input FIFO not full flag

This bit flags that the input FIFO is not full (data can be written). This flag must not be considered when using DMA.

0: Full

1: Not full

Bit 1 **IFTF**: Input FIFO threshold flag

This bit flags that the amount of data in the input FIFO is below a threshold. This flag must not be considered when using DMA.

0: At or above threshold

1: Below threshold.

Bit 0 Reserved, must be kept at reset value.

46.5.8 JPEG clear flag register (JPEG_CFR)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHPDF	CEOCF	Res.	Res.	Res.	Res.	Res.
									rt_w1	rt_w1					

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CHPDF**: Clear header parsing done flag

Writing 1 clears the HPDF bit of the JPEG_SR register.

0: No effect

1: Clear

Bit 5 **CEOCF**: Clear end of conversion flag

Writing 1 clears the ECF bit of the JPEG_SR register.

0: No effect

1: Clear

Bits 4:0 Reserved, must be kept at reset value.

46.5.9 JPEG data input register (JPEG_DIR)

Address offset: 0x040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATAIN[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAIN[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **DATAIN[31:0]**: Data input FIFO
Input FIFO data register

46.5.10 JPEG data output register (JPEG_DOR)

Address offset: 0x044

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATAOUT[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAOUT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **DATAOUT[31:0]**: Data output FIFO
Output FIFO data register.

46.5.11 JPEG quantization memory x (JPEG_QMEMx_y)

Address offset: 0x050 + 0x40 * x + 0x4 * y, (x = 0 to 3; y = 0 to 15)

Reset value: 0xXXXX XXXX

Four quantization tables as specified by ISO documentation.

For decoding with header parsing, no quantization table programming is required, the coefficients are directly written in the quantization memories by the header parser.

For decoding without header parsing or for encoding, the quantization table must be written by software in zig zag order.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
QCOEF{4*y+3}[7:0]								QCOEF{4*y+2}[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
QCOEF{4*y+1}[7:0]								QCOEF{4*y}[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **QCOEF{4*y+3}[7:0]**: Quantization coefficient {4*y+3}
8-bit quantization coefficient.

Bits 23:16 **QCOEF{4*y+2}[7:0]**: Quantization coefficient {4*y+2}
8-bit quantization coefficient.

Bits 15:8 **QCOEF{4*y+1}[7:0]**: Quantization coefficient {4*y+1}
8-bit quantization coefficient.

Bits 7:0 **QCOEF{4*y}[7:0]**: Quantization coefficient {4*y}
8-bit quantization coefficient.

46.5.12 JPEG Huffman min (JPEG_HUFFMINx_y)

Address offset: 0x150 + 0x10 * x + 0x4 * y, (x = 0 to 3; y = 0 to 2)

Reset value: 0xXXXX XXXX

This memory stores the minimum Huffman values used internally by the JPEG decoder. The memory content is written by hardware during the header parsing.

- DATA0: Min AC0 value
- DATA1: Min DC0 value
- DATA2: Min AC1 value
- DATA3: Min DC1 value

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA{x}[{32*y+31}:{32*y+16}]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA{x}[{32*y+15}:{32*y}]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATA{x}[{32*y+31}:{32*y}]**: Minimum Huffman value
100-bit minimum Huffman value used internally by the JPEG decoder.

46.5.13 JPEG Huffman min x (JPEG_HUFFMINx_y)

Address offset: 0x150 + 0x10 * x + 0x4 * y, (x = 0 to 3; y = 3)

Reset value: 0xXXXX XXXX

This memory stores the minimum Huffman values used internally by the JPEG decoder. The memory content is written by hardware during the header parsing.

- DATA0: Min AC0 value
- DATA1: Min DC0 value
- DATA2: Min AC1 value
- DATA3: Min DC1 value

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATA{x}[99: 96]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **DATA{x}[99:96]**: Minimum Huffman value
100-bit minimum Huffman value used internally by the JPEG decoder.

46.5.14 JPEG Huffman base (JPEG_HUFFBASEx)

Address offset: 0x190 + 0x4*x, (x = 0 to 31)

Reset value: 0xXXXX XXXX

This memory stores the base Huffman values used internally by the JPEG decoder. The memory content is written by hardware during the header parsing:

- DATA0 to DATA15: Base AC0 value
- DATA16 to DATA31: Base DC0 value
- DATA32 to DATA47: Base AC1 value
- DATA48 to DATA63: Base DC1 value

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATA{2*x+1}[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATA{2*x}[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 24:16 **DATA{2*x+1}[8:0]**: Data {2*x+1}
Base Huffman value.

Bits 15:9 Reserved, must be kept at reset value.

Bits 8:0 **DATA{2*x}[8:0]**: Data {2*x}
Base Huffman value.

46.5.15 JPEG Huffman symbol (JPEG_HUFFSYMBx)

Address offset: 0x210 + 0x4 * x, (x = 0 to 83)

Reset value: 0xXXXX XXXX

This memory stores the Huffman symbols used internally by the JPEG decoder. The memory content is written by hardware during the header parsing:

- DATA0 to DATA161: AC0 symbols
- DATA162 to DATA173: DC0 and DC1 symbols
- DATA174 to DATA335: AC1 symbols

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA{4*x+3}[7:0]								DATA{4*x+2}[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA{4*x+1}[7:0]								DATA{4*x}[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **DATA{4*x+3}[7:0]**: Data {4*x+3}
Huffman symbol.

Bits 23:16 **DATA{4*x+2}[7:0]**: Data {4*x+2}
Huffman symbol.

Bits 15:8 **DATA{4*x+1}[7:0]**: Data {4*x+1}
Huffman symbol.

Bits 7:0 **DATA{4*x}[7:0]**: Data {4*x}
Huffman symbol.

46.5.16 JPEG DHT memory (JPEG_DHTMEMx)

Address offset: 0x360 + 0x4 * x, (x = 0 to 102)

Reset value: 0xXXXX XXXX

For encoding process with header generation, this memory stores the DHT marker segment AC and DC Huffman tables in the ISO/IEC specification format:

- DATA0 to DATA27: DC Huffman table0
- DATA28 to DATA205: AC Huffman table0
- DATA206 to DATA233: DC Huffman table1
- DATA234 to DATA411: AC Huffman table1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA{4*x+3}[7:0]								DATA{4*x+2}[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA{4*x+1}[7:0]								DATA{4*x}[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **DATA{4*x+3}[7:0]**: Huffman table data {4*x+3}
Huffman table data for DHT marker segment generation.

Bits 23:16 **DATA{4*x+2}[7:0]**: Huffman table data {4*x+2}
Huffman table data for DHT marker segment generation.

Bits 15:8 **DATA{4*x+1}[7:0]**: Huffman table data {4*x+1}
Huffman table data for DHT marker segment generation.

Bits 7:0 **DATA{4*x}[7:0]**: Huffman table data {4*x}
Huffman table data for DHT marker segment generation.

46.5.17 JPEG Huffman encoder ACx (JPEG_HUFFENC_ACx_y)

Address offset: 0x500 + 0x160 * x + 0x4 * y, (x = 0 to 1; y = 0 to 87)

Reset value: 0xXXXX XXXX

This memory defines the Huffman codes used during the encoding process of AC components.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	HLEN{2*y+1}[3:0]				HCODE{2*y+1}[7:0]							
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HLEN{2*y}[3:0]				HCODE{2*y}[7:0]							
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **HLEN{2*y+1}[3:0]**: Huffman length {2*y+1}
Number of bits in the Huffman code HCODE{2*y+1} minus 1.

Bits 23:16 **HCODE{2*y+1}[7:0]**: Huffman code {2*y+1}
8 least significant bits of the Huffman code.
If the Huffman code is less than 8 bits long, the unused bits must be 0.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **HLEN{2*y}[3:0]**: Huffman length {2*y}
Number of bits in the Huffman code HCODE{2*y} minus 1.

Bits 7:0 **HCODE{2*y}[7:0]**: Huffman code {2*y}
8 least significant bits of the Huffman code.
If the Huffman code is less than 8 bits long, the unused bits must be 0.

46.5.18 JPEG Huffman encoder DCx (JPEG_HUFFENC_DCx_y)

Address offset: 0x7C0 + 0x20*x + 0x4*y, (x = 0 to 1; y = 0 to 7)

Reset value: 0xXXXX XXXX

This memory defines the Huffman codes used during the encoding process of DC components.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	HLEN{2*y+1}[3:0]				HCODE{2*y+1}[7:0]							
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	HLEN{2*y}[3:0]				HCODE{2*y}[7:0]							
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **HLEN{2*y+1}[3:0]**: Huffman length {2*y+1}
Number of bits in the Huffman code HCODE{2*y+1} minus 1.

Bits 23:16 **HCODE{2*y+1}[7:0]**: Huffman code {2*y+1}
8 least significant bits of the Huffman code.
If the Huffman code is less than 8 bits long, the unused bits must be 0.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **HLEN{2*y}[3:0]**: Huffman length {2*y}
Number of bits in the Huffman code HCODE{2*y} minus 1.

Bits 7:0 **HCODE{2*y}[7:0]**: Huffman code {2*y}
8 least significant bits of the Huffman code.
If the Huffman code is less than 8 bits long, the unused bits must be 0.

46.5.19 JPEG codec register map

The following table summarizes the JPEG codec registers. Refer to the register boundary addresses table for the JPEG codec register base address.

Table 394. JPEG codec register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	JPEG_CONFR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	START
	Reset value																																0
0x004	JPEG_CONFR1	YSIZE[15:0]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDR	NS[1:0]		COLSPACE [1:0]		DE	Res.	NF[1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								0	0	0	0	0	0		0	0
0x008	JPEG_CONFR2	Res.	Res.	Res.	Res.	Res.	Res.	NMCU[25:0]																									
	Reset value							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	JPEG_CONFR3	XSIZE[15:0]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0																
0x010	JPEG_CONFR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSF[3:0]				VSF[3:0]				NB[3:0]				QT[1:0]		HA	HD
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	JPEG_CONFR5	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSF[3:0]				VSF[3:0]				NB[3:0]				QT[1:0]		HA	HD
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	JPEG_CONFR6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSF[3:0]				VSF[3:0]				NB[3:0]				QT[1:0]		HA	HD
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	JPEG_CONFR7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HSF[3:0]				VSF[3:0]				NB[3:0]				QT[1:0]		HA	HD
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020-0x02C	Reserved	Reserved																															
0x030	JPEG_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFF	IFF	ODMAEN	IDMAEN	Res.	Res.	Res.	Res.	HPDIE	EOCIE	OFNEIE	OFTIE	IFNFIE	IFTIE	JCEN
	Reset value																		0	0	0	0					0	0	0	0	0	0	0
0x034	JPEG_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COF	HPDF	EOCF	OFNEF	OFTF	IFNFF	IFTF	Res.
	Reset value																									0	0	0	0	0	1	1	
0x038	JPEG_CFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHPDF	CEOCF	Res.	Res.	Res.	Res.	Res.
	Reset value																										0	0					
0x040	JPEG_DIR	DATAIN[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	JPEG_DOR	DATAOUT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048-0x04C	Reserved	Reserved																															
0x050-0x08C	JPEG_QMEM0	QCOEF{4*y+3}[7:0]								QCOEF{4*y+2}[7:0]								QCOEF{4*y+1}[7:0]								QCOEF{4*y}[7:0]							
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

Table 394. JPEG codec register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x090-0x0CC	JPEG_QMEM1	QCOEF{4*y+3}[7:0]								QCOEF{4*y+2}[7:0]								QCOEF{4*y+1}[7:0]								QCOEF{4*y}[7:0]							
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x0D0-0x10F	JPEG_QMEM2	QCOEF{4*y+3}[7:0]								QCOEF{4*y+2}[7:0]								QCOEF{4*y+1}[7:0]								QCOEF{4*y}[7:0]							
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x110-0x14C	JPEG_QMEM3	QCOEF{4*y+3}[7:0]								QCOEF{4*y+2}[7:0]								QCOEF{4*y+1}[7:0]								QCOEF{4*y}[7:0]							
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x150-0x18C	JPEG_HUFFMIN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATA{x}[99:96]			
		DATA{x}[95:64]																															
		DATA{x}[63:32]																															
		DATA{x}[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x190-0x20C	JPEG_HUFFBASE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATA{2*x+1}[8:0									Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATA{2*x}[8:0								
	Reset value								X	X	X	X	X	X	X	X	X								X	X	X	X	X	X	X	X	X
0x210-0x35C	JPEG_HUFFSYMB	DATA{4*x+3}[7:0]								DATA{4*x+2}[7:0]								DATA{4*x+1}[7:0]								DATA{4*x}[7:0]							
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x360-0x4F8	JPEG_DHTMEM	DATA{4*x+3}[7:0]								DATA{4*x+2}[7:0]								DATA{4*x+1}[7:0]								DATA{4*x}[7:0]							
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x4FC	Reserved	Res.																															
0x500-0x65C	JPEG_HUFFENC_AC0	Res.	Res.	Res.	Res.	HLEN{2*y+1}[3:0]				HCODE{2*y+1}[7:0]								Res.	Res.	Res.	Res.	HLEN{2*y}[3:0]				HCODE{2*y}[7:0]							
	Reset value					X	X	X	X	X	X	X	X	X	X	X	X					X	X	X	X	X	X	X	X	X	X	X	X
0x660-0x7BC	JPEG_HUFFENC_AC1	Res.	Res.	Res.	Res.	HLEN{2*y+1}[3:0]				HCODE{2*y+1}[7:0]								Res.	Res.	Res.	Res.	HLEN{2*y}[3:0]				HCODE{2*y}[7:0]							
	Reset value					X	X	X	X	X	X	X	X	X	X	X	X					X	X	X	X	X	X	X	X	X	X	X	X
0x7C0-0x7DC	JPEG_HUFFENC_DC0	Res.	Res.	Res.	Res.	HLEN{2*y+1}[3:0]				HCODE{2*y+1}[7:0]								Res.	Res.	Res.	Res.	HLEN{2*y}[3:0]				HCODE{2*y}[7:0]							
	Reset value					X	X	X	X	X	X	X	X	X	X	X	X					X	X	X	X	X	X	X	X	X	X	X	X
0x7E0-0x7FC	JPEG_HUFFENC_DC1	Res.	Res.	Res.	Res.	HLEN{2*y+1}[3:0]				HCODE{2*y+1}[7:0]								Res.	Res.	Res.	Res.	HLEN{2*y}[3:0]				HCODE{2*y}[7:0]							
	Reset value					X	X	X	X	X	X	X	X	X	X	X	X					X	X	X	X	X	X	X	X	X	X	X	X

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

47 True random number generator (RNG)

47.1 RNG introduction

The RNG is a true random number generator that provides full entropy outputs to the application as 32-bit samples. It is composed of a live entropy source (analog) and an internal conditioning component.

The RNG is a NIST SP 800-90B compliant entropy source that can be used to construct a nondeterministic random bit generator (NDRBG).

The RNG true random number generator can be certified NIST SP800-90B. It can also be tested using the German BSI statistical tests of AIS-31 (T0 to T8).

47.2 RNG main features

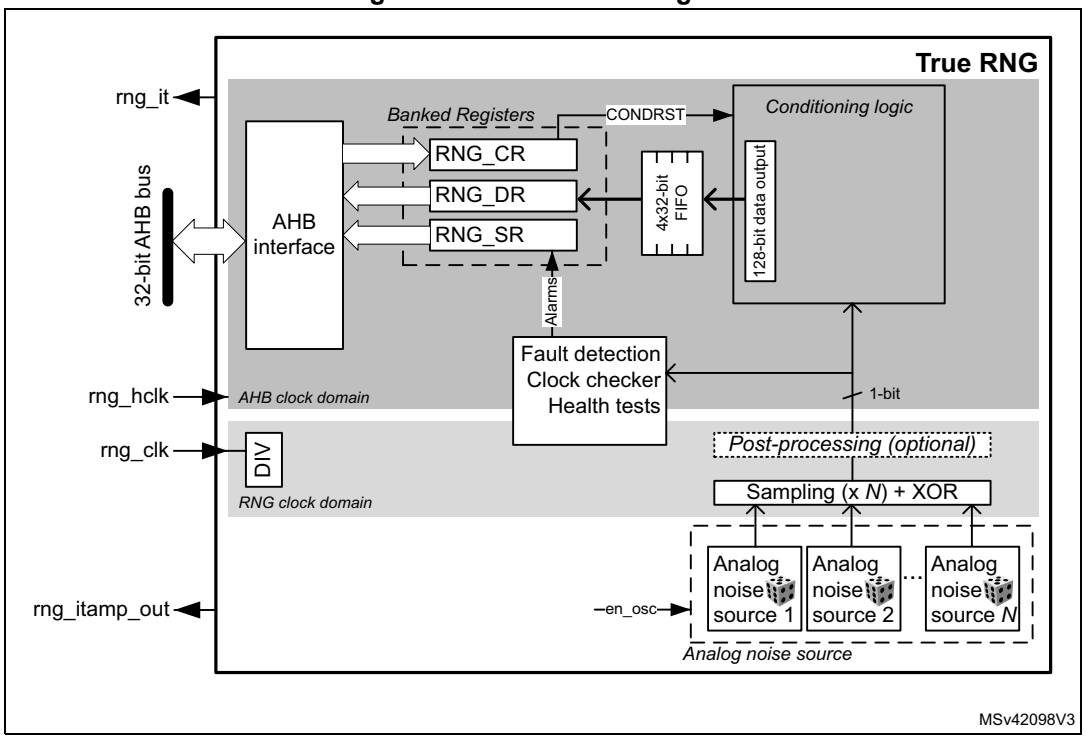
- The RNG delivers 32-bit true random numbers, produced by an analog entropy source conditioned by a NIST SP800-90B approved conditioning stage.
- It can be used as the entropy source to construct a nondeterministic random bit generator (NDRBG).
- In the NIST configuration, it produces four 32-bit random samples every 412 AHB clock cycles if $f_{AHB} < f_{threshold}$ (256 RNG clock cycles otherwise).
- It embeds startup and NIST SP800-90B approved continuous health tests (repetition count and adaptive proportion tests), associated with specific error management
- It can be disabled to reduce power consumption, or enabled with an automatic low power mode (default configuration).
- It has an AMBA® AHB slave peripheral, accessible through 32-bit word single accesses only (else an AHB bus error is generated, and the write accesses are ignored).

47.3 RNG functional description

47.3.1 RNG block diagram

Figure 451 shows the RNG block diagram.

Figure 451. RNG block diagram



47.3.2 RNG internal signals

Table 395 describes a list of useful-to-know internal signals available at the RNG level, not at the STM32 product level (on pads).

Table 395. RNG internal input/output signals

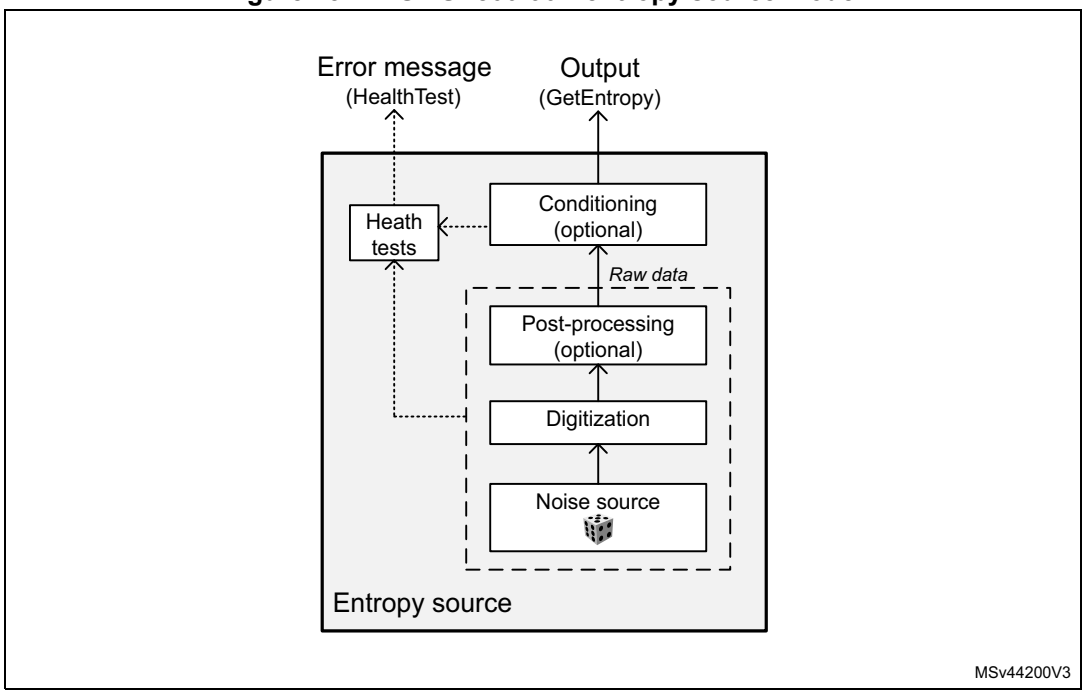
Signal name	Signal type	Description
rng_it	Digital output	RNG global interrupt request
rng_hclk	Digital input	AHB clock
rng_clk	Digital input	RNG dedicated clock, asynchronous to rng_hclk
rng_itamp_out	Digital output	RNG internal tamper event signal to TAMP (XORed), triggered when an unexpected hardware fault occurs. When this signal is triggered, RNG stops delivering random samples, requiring a reset and a new initialization to be usable again.

47.3.3 Random number generation

The true random number generator (RNG) delivers truly random data through its AHB interface at deterministic intervals.

Within its boundary RNG integrates all the required NIST components depicted on *Figure 452*. Those components are an analog noise source, a digitization stage, a conditioning algorithm, a health monitoring block and two interfaces that are used to interact with the entropy source: GetEntropy and HealthTest.

Figure 452. NIST SP800-90B entropy source model



The components pictured above are detailed hereafter.

Noise source

The noise source is the component that contains the non-deterministic, entropy-providing activity that is ultimately responsible for the uncertainty associated with the bitstring output by the entropy source. This noise source provides 1-bit samples. It is composed of:

- Multiple analog noise sources (x6), each based on three XORed free-running ring oscillator outputs. It is possible to disable those analog oscillators to save power, as described in *Section 47.3.8*. Multiple oscillators are also disabled for the configuration A (see *Table 398: RNG configurations*).
- The XORing of all the noise sources into a single analog output.
- A sampling stage of this output clocked by a dedicated clock input (rng_clk with integrated divider), delivering a 1-bit raw data output.

This noise source sampling is independent to the AHB interface clock frequency (rng_hclk), with a possibility for the software to decrease the sampling frequency by using the integrated divider.

Note: In Section 47.6 the recommended RNG clock frequencies and associated divider value are given.

Post processing

In the NIST configuration no post-processing is applied to the sampled noise source. In non-NIST configuration B (as defined in *Section 47.6.2*) a normalization debiasing is applied,

that is half of the bits are taken from the sampled noise source, half of the bits are taken from the inverted sampled noise source.

Conditioning

The conditioning component in the RNG is a deterministic function that increases the entropy rate of the resulting fixed-length bitstrings output (128-bit). The NIST SP800-90B target is full entropy on the output (128-bit).

The times required between two random number generations, and between the RNG initialization and availability of first sample are described in *Section 47.5*.

Output buffer

A data output buffer can store up to four 32-bit words that have been output from the conditioning component. When four words have been read from the output FIFO through the RNG_DR register, the content of the 128-bit conditioning output register is pushed into the output FIFO, and a new conditioning round is automatically started. Four new words are added to the conditioning output register after a number of clock cycles specified in *Section 47.5*.

Whenever a random number is available through the RNG_DR register, the DRDY flag changes from 0 to 1. This flag remains high until the output buffer becomes empty after reading four words from the RNG_DR register.

Note: When interrupts are enabled an interrupt is generated when this data ready flag transitions from 0 to 1. Interrupt is then cleared automatically by the RNG as explained above.

Health checks

This component ensures that the entire entropy source (with its noise source) starts then operates as expected, obtaining assurance that failures are caught quickly and with a high probability and reliability.

The RNG implements the following health check features in accordance with NIST SP800-90B. The described thresholds correspond to the value recommended for register RNG_HTCR (configuration A in *Section 47.6.2*).

1. Startup health tests, performed after reset and before the first use of the RNG as entropy source:
   - Repetition count test, flagging an error when the noise source has provided more than 42 consecutive bits at a constant value (0 or 1).
   - Adaptive proportion test running on a window of 1024 consecutive bits: the RNG verifies that the first bit on the outputs of the noise source is not repeated more than 711 times.
   - Known-answer tests, to verify the conditioning stage.
2. Continuous health tests, running indefinitely on the outputs of the noise source:
   - Repetition count test, similar to the one running in startup tests.
   - Adaptive proportion test, similar to the one running in startup tests.
3. Vendor specific continuous tests
   - Transition count test, flagging an error when the noise source has delivered more than 32 consecutive occurrences of 2-bit patterns (01 or 10).
   - Real-time “too slow” sampling clock detector, flagging an error when one RNG clock cycle (before divider) is smaller than AHB clock cycle divided by 32.
4. On-demand test of digitized noise source (raw data)
   - Supported by restarting the entropy source and rerunning the startup tests (see software reset sequence in *Section 47.3.4*). Other kinds of on-demand testing (software based) are *not supported*.

The CECS and SECS status bits in the RNG_SR register indicate when an error condition is detected, as detailed in *Section 47.3.7*.

Note: An interrupt can be generated when an error is detected.

Above the health test thresholds are modified by changing the value in the RNG_HTCR register. See Section 47.6 for details.

47.3.4 RNG initialization

The RNG simplified state machine is pictured on *Figure 453*.

After enabling the RNG (RNGEN = 1 in RNG_CR), the following chain of events occurs:

1. The analog noise source is enabled, and by default the RNG waits 16 cycles of RNG clock cycles (before divider) before starting to sample the analog output and filling the 128-bit conditioning shift register.
2. The conditioning hardware initializes, automatically triggering startup behavior test on the raw data samples and known-answer tests.
3. Start-up health tests are completed, with the conditioning stage being ready to use.
4. The conditioning stage internal input data buffer is filled again with 128-bit and a number of conditioning rounds defined by the RNG configuration (NIST or non-NIST) is performed. The output buffer is then filled with the post processing result.
5. The output buffer is refilled automatically according to the RNG usage.

The associated initialization time can be found in *Section 47.5*.

Figure 453. RNG initialization overview

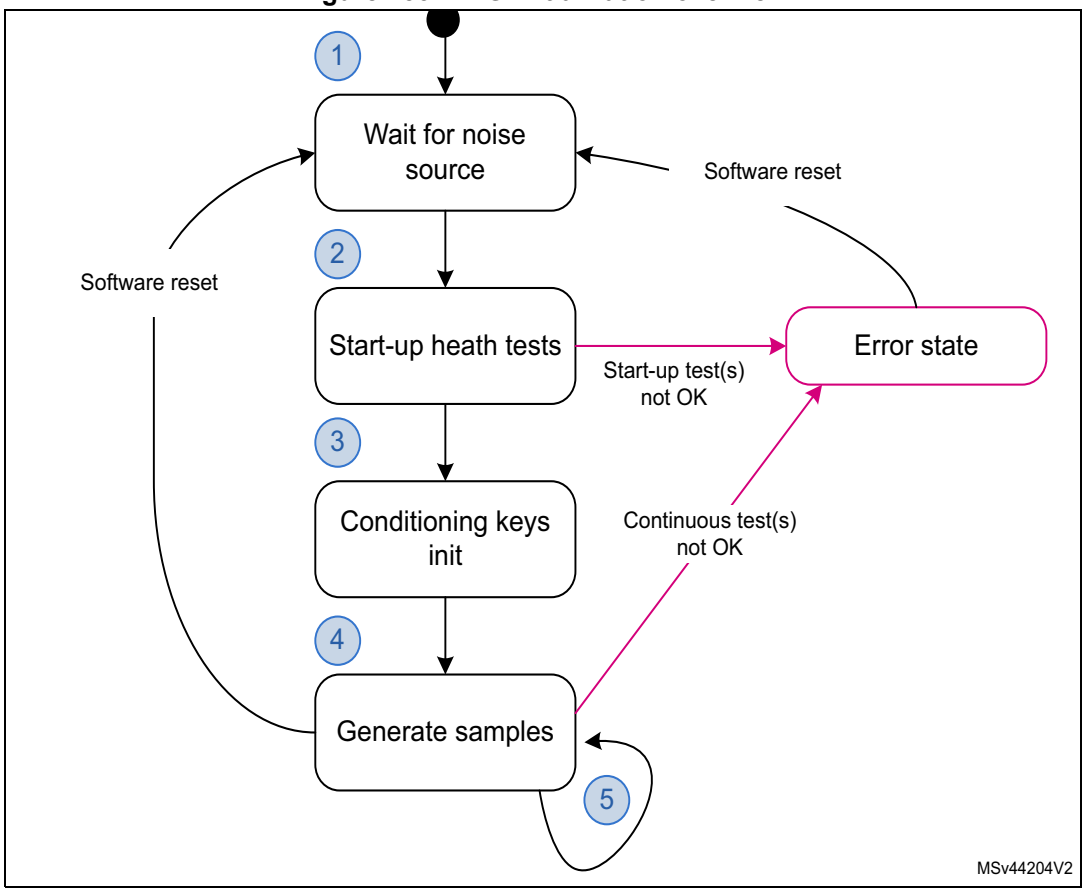


Figure 453 also highlights a possible software reset sequence, implemented by:

1. Writing bits RNGEN = 0 and CONDRST = 1 in the RNG_CR register with the same RNG configuration and a new CLKDIV if needed.
2. Then writing RNGEN = 1 and CONDRST = 0 in the RNG_CR register.
3. Wait for random number to be ready, after initialization completes.

Note: When the RNG peripheral is reset through RCC (hardware reset), the RNG configuration for optimal randomness is lost in the RNG registers. Software reset with CONFIGLOCK set preserves the RNG configuration.

47.3.5 RNG operation

Normal operation

To run the RNG using interrupts, the following steps are recommended:

1. Check *Section 47.6* to verify if a specific RNG configuration is required for the application.
   – In this case, write in the RNG_CR register the bit CONDRST = 1 together with the correct RNG configuration. Then perform a second write to the RNG_CR register

with the bit CONDRST = 0, the interrupt enable bit IE = 1 and the RNG enable bit RNGEN = 1.

– If it is not the case perform a write to the RNG_CR register with the interrupt enable bit IE = 1 and the RNG enable bit RNGEN = 1.

2. An interrupt is now generated when a random number is ready or when an error occurs. Therefore, at each interrupt, check that:
   – No error occurred. The SEIS and CEIS bits must be set to 0 in the RNG_SR register.
   – A random number is ready. The DRDY bit must be set to 1 in the RNG_SR register.
   – If the above two conditions are true the content of the RNG_DR register can be read up to four consecutive times. If valid data is available in the conditioning output buffer, four additional words can be read by the application (in this case the DRDY bit is still high). If one or both of the above conditions are false, the RNG_DR register must not be read. If an error occurred, the error recovery sequence described in *Section 47.3.7* must be used.

To run the RNG in polling mode following steps are recommended:

1. Check *Section 47.6* to verify if a specific RNG configuration is required for the application.
   – If it is the case write in the RNG_CR register the bit CONDRST = 1 together with the correction RNG configuration. Then perform a second write to the RNG_CR register with the bit CONDRST = 0 and the RNG enable bit RNGEN = 1.
   – If it is not the case only enable the RNG by setting the RNGEN bit to 1 in the RNG_CR register.
2. Read the RNG_SR register and check that:
   – No error occurred (the SEIS and CEIS bits must be set to 0)
   – A random number is ready (the DRDY bit must be set to 1)
3. If above conditions are true read the content of the RNG_DR register up to four consecutive times. If valid data is available in the conditioning output buffer four additional words can be read by the application (in this case the DRDY bit is still high). If one or both of the above conditions are false, the RNG_DR register must not be read. If an error occurred, the error recovery sequence described in *Section 47.3.7* must be used.

Note: When data is not ready (DRDY = 0) RNG_DR returns zero.
It is recommended to always verify that RNG_DR is different from zero. Because when it is the case a seed error occurred between RNG_SR polling and RND_DR output reading (rare event).

If the random number generation period is a concern to the application and if NIST compliance is not required it is possible to select a faster RNG configuration by using the RNG configuration “B”, described in *Section 47.6*. The gain in random number generation speed is summarized in *Section 47.5*.

Low-power operation

If the power consumption is a concern to the application, low-power strategies can be used, as described in *Section 47.3.8*.

Software post-processing

No specific software post-processing/conditioning is expected to meet the AIS-31 or NIST SP800-90B approvals.

Built-in health check functions are described in *Section 47.3.3: Random number generation*.

47.3.6 RNG clocking

The RNG runs on two different clocks: the AHB bus clock and a dedicated RNG clock.

The AHB clock is used to clock the AHB banked registers and conditioning component. The RNG clock, coupled with a programmable divider (see CLKDIV bitfield in the RNG_CR register) is used for noise source sampling. Recommended clock configurations are detailed in *Section 47.6: RNG entropy source validation*.

*Note: When the CED bit in the RNG_CR register is set to 0, the RNG clock frequency before the internal divider **must be higher** than the AHB clock frequency divided by 32, otherwise the clock checker always flags a clock error (CECS = 1 in the RNG_SR register).*

See *Section 47.3.1: RNG block diagram* for details (AHB and RNG clock domains).

47.3.7 Error management

In parallel to random number generation a health check block verifies the correct noise source behavior and the frequency of the RNG source clock as detailed in this section. Associated error state is also described.

Clock error detection

When the clock error detection is enabled (CED = 0) and if the RNG clock frequency is too low, the RNG sets to 1 both the CEIS and CECS bits to indicate that a clock error occurred. In this case, the application must check that the RNG clock is configured correctly (see *Section 47.3.6: RNG clocking*) and then it must clear the CEIS bit interrupt flag. The CECS bit is automatically cleared when the clocking condition is normal.

When CECS is set, values are available in the RNG_DR register, but it should not be used as described in *Section 47.3.5: RNG operation*.

Note: CEIS is set only when CECS is set to 1 by RNG.

Noise source error detection

When a noise source (or seed) error occurs, the RNG stops generating random numbers and sets to 1 both SEIS and SECS bits to indicate that a seed error occurred.

When SECS is set, values are available in the RNG_DR register, but it should not be used as described in *Section 47.3.5: RNG operation*.

When ARDIS is cleared (auto-reset enabled) clear the SEIS interrupt status bit in the RNG_SR register before drawing random numbers again.

When ARDIS is set (auto-reset disabled) the following sequence must be used to fully recover from a seed error:

1. Write CONDRST at 1 and at 0 in the RNG_CR register to reset the conditioning logic.
2. Wait for the CONDRST bit to be cleared by hardware. Then clear the SEIS interrupt status bit in the RNG_SR register.
3. If SECS is still set in RNG_SR it means a new seed error occurred (unlikely). In this case go back to step 1).

Note: After a seed error RNG restarts generating random numbers when SECS is cleared.

When the application sets the ARDIS bit in the RNG_CR register, the auto-reset is disabled. CONDRST must be used in step 1.

RNG tamper errors

When an unexpected error is found by the RNG an internal tamper event is triggered in the TAMP peripheral, and the RNG stops delivering random data.

When this event occurs, the secure application needs to reset the RNG peripheral either using the central reset management or the global SoC reset. Then a proper initialization of the RNG is required, again.

47.3.8 RNG low-power use

If power consumption is a concern, the RNG can be disabled as soon as the DRDY bit is set to 1 by setting the RNGEN bit to 0 in the RNG_CR register. As the post-processing logic and the output buffer remain operational while RNGEN = 0 following features are available to the software:

- If there are valid words in the output buffer four random numbers can still be read from the RNG_DR register.
- If there are valid bits in the conditioning output internal register four additional random numbers can be still be read from the RNG_DR register. If it is not the case RNG must be reenabled by the application until the expected new noise source bits threshold is reached (128-bit in NIST mode) and a complete conditioning round is done.
  Four new random words are then available only if the expected number of conditioning round is reached (two if NISTC = 0). The overall time can be found in *Section 47.5: RNG processing time on page 2296*.

When disabling the RNG the user deactivates all the analog seed generators, whose power consumption is given in the datasheet electrical characteristics section. The user also gates all the logic clocked by the RNG clock. Note that this strategy is adding latency before a random sample is available on the RNG_DR register, because of the RNG initialization time.

If the RNG block is disabled during initialization (that is well before the DRDY bit rises for the first time), the initialization sequence resumes from where it was stopped when RNGEN bit is set to 1, unless the application resets the conditioning logic using CONDRST bit in the RNG_CR register.

When the application wants to disable the RNG clock it is recommended to wait two RNG kernel clock cycles between clearing the RNGEN bit and disabling the RNG kernel clock using the RCC.

Also, when application needs to enter a power mode where RNG is de-activated, it is recommended to wait two RNG kernel clock cycles between clearing the RNGEN bit and entering the low power mode using the PWR.

In the two cases above, to avoid unexpected consumption when RNG analog oscillators stay active, application can set the bit 13 in RNG_CR register. Setting this bit adds some marginal power consumption while RNGEN bit is set (RNG activated).

Note: The power modes where RNG is deactivated (that is retained or not available) can be found in the PWR section.

47.4 RNG interrupts

In the RNG an interrupt can be produced on the following events:

- Data ready flag
- Seed error, see *Section 47.3.7: Error management*
- Clock error, see *Section 47.3.7: Error management*

Dedicated interrupt enable control bits are available as shown in *Table 396*.

Table 396. RNG interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method
RNG	Data ready flag	DRDY	IE	None (automatic)
	Seed error flag	SEIS	IE	Write CONDRST to 1 then to 0 unless ARDIS is cleared (see *Section 47.3.7*). Write 0 to SEIS.
	Clock error flag	CEIS	IE	Write 0 to CEIS

The user can enable or disable the above interrupt sources globally by using the general interrupt control bit IE in the RNG_CR register. The status of the individual interrupt sources can be read from the RNG_SR register.

Note: Interrupts are generated only when RNG is enabled.

47.5 RNG processing time

In recommended configuration A or C described in *Table 398*, the time between two sets of four 32-bit data is either:

- 203 x N AHB cycles if $f_{AHB} < f_{threshold}$ (conditioning stage is limiting), or
- 128 x N RNG cycles $f_{AHB} \geq f_{threshold}$ (noise source stage is limiting).

With $f_{threshold} = 1.6 \times f_{RNG}$, for instance 77 MHz if f_{RNG}= 48 MHz. Value N is defined in *Section 47.6: RNG entropy source validation*.

Note: When CLKDIV is different from zero, f_{RNG} must take into account the internal divider ratio.

If configuration B is selected the performance figures become:

- 203 AHB cycles if $f_{AHB} < f_{threshold}$ or
- 32 RNG cycles $f_{AHB} \geq f_{threshold}$

with $f_{threshold} = 6.5 \times f_{RNG}$.

Initialization time

More time is needed for the first set of random numbers after the device exits reset (see Section 1.3.4: RNG initialization). Table below gives details on how to compute the time spent in each initialization step.

Table 397. RNG initialization times

Initialization step	Configuration A or C, reset value	Configuraton B
Wait for noise source, then startup health tests	Max((wait_noise(1) + 11 + 1024 x CLKDIV) RNG_cycles, 13 x 203 AHB_cycles)	Max(16 + 1035 RNG_cycles, 13 x 203 AHB_cycles)
Conditioning keys initialization	Max((1+2xN) x 128 x CLKDIV) RNG_cycles + 203 AHB_cycles, (128 x CLKDIV) RNG_cycles + (2xN+2) x 203 AHB_cycles)	Max (3 x 32 RNG_cycles + 203 AHB_cycles, 32 RNG_cycles + 4 x 203 AHB_cycles)

1. 192 RNG_cycles (configuration A or C), 16 RNG_cycles (reset value)

As an example, if AHB clock= 200 MHz and RNG clock= 48 MHz then the initialization time is around 56 µs in the configuration C (CLKDIV=2, N=2) and 37 µs in the configuration B.

47.6 RNG entropy source validation

47.6.1 Introduction

To assess the amount of entropy available from the RNG, the peripheral has been tested using the German BSI AIS-31 statistical tests (T0 to T8), and NIST SP800-90B test suite.

47.6.2 Validation conditions

STMicroelectronics has tested the RNG true random number generator in the following conditions:

- RNG clock rng_clk = 64 MHz, with CED bit cleared in RNG_CR
- RNG configurations are described in *Table 398*. Note that only configuration A can be certified NIST SP800-90B. Refer to *Table 399* to select the best configuration for the application.
- Configuration C can be used when configuration A is not flagged as certified.

Table 398. RNG configurations

Configuration	RNG_CR bits					Loop number (N)	RNG_ HTCR register	RNG_ NSCR register
	NISTC bit	RNG_ CONFIG1 [7:0]	CLKDIV [3:0]	RNG_ CONFIG2 [2:0](1)	RNG_ CONFIG3 [3:0](2)			
A	Refer to *NIST compliant RNG configuration* table in AN4230 available from *www.st.com*. This application note also indicates if this configuration is part of an existing NIST SP800-90B Entropy Certificate listed on *https://csrc.nist.gov/projects/cryptographic-module-validation-program*.							

Table 398. RNG configurations (continued)

Configuration	RNG_CR bits					Loop number (N)	RNG_ HTCR register	RNG_ NSCR register
	NISTC bit	RNG_ CONFIG1 [7:0]	CLKDIV [3:0]	RNG_ CONFIG2 [2:0](1)	RNG_ CONFIG3 [3:0](2)			
B	1	0x18	0x0(3)	0x0	0x0	1	0x0000 AAC7(4)	default
C	0	0x0F		0x0	0xD	2		default

1. 0x1 value is recommended instead of 0x0 for RNG_CONFIG2[2:0], when RNG power consumption is critical. See the end of *Section 47.3.8* for details.
2. RNG_CONFIG3[1:0] defines the loop number N: 0x0 corresponds to N=1, 0x1 to N=2, 0x2 to N=3, 0x3 to N=4
3. The noise source sampling must be 32 MHz or less. Hence, if the RNG clock is different from 64 MHz, this value of CLKDIV must be adapted. See the CLKDIV bitfield description in *Section 47.7.1* for details.
4. This value can be fixed in the RNG driver (it does not depend upon the STM32 product).

Table 399. Configuration selection

Section criteria	Config A	Config B	Config C
Suitable to generate NIST compliant cryptographic keys	Yes	No	
Entropy(1)	Certified	Good	Very good
Speed(2)	Baseline	Faster	Baseline

1. For configurations B and C entropy is verified using AIS-31 test suite (T0 to T8).
2. When speed is not enough for application a NIST compliant DRBG can be used to increase throughput.

For details on data collection and the running of statistical test suites refer to AN4230 "*STM32 microcontrollers random number generation validation using NIST statistical test suite*" available from *www.st.com*.

47.7 RNG registers

The RNG is associated with a control register, a data register and a status register.

47.7.1 RNG control register (RNG_CR)

Address offset: 0x000

Reset value: 0x0080 0D00

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CONFIGLOCK	CONDRST	Res.	Res.	RNG_CONFIG1[7:0]								CLKDIV[3:0]			
rs	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RNG_CONFIG2[2:0]			NISTC	RNG_CONFIG3[3:0]				ARDIS	Res.	CED	Res.	IE	RNGEN	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw		rw		rw	rw		

Bit 31 **CONFIGLOCK**: RNG Config lock

0: Writes to the RNG_NSCR, RNG_HTCR and RNG_CR configuration bits [29:4] are allowed.

1: Writes to the RNG_NSCR, RNG_HTCR and RNG_CR configuration bits [29:4] are ignored until the next RNG reset.

Once set, this bit can only be cleared when RNG is reset (set once bit).

Bit 30 **CONDRST**: Conditioning soft reset

Write 1 and then write 0 to reset the conditioning logic, clear all the FIFOs and start a new RNG initialization process, with RNG_SR cleared. Registers RNG_CR, RNG_NSCR and RNG_HTCR are not changed by CONDRST.

This bit must be set to 1 in the same access that set any configuration bits [29:4]. In other words, when CONDRST bit is set to 1 correct configuration in bits [29:4] must also be written.

When CONDRST is set to 0 by the software, its value goes to 0 when the reset process is done. It takes about 2 AHB clock cycles + 2 RNG clock cycles.

Bits 29:28 Reserved, must be kept at reset value.

Bits 27:20 **RNG_CONFIG1[7:0]:** RNG configuration 1

Reserved to the RNG configuration (bitfield 1). Must be initialized using the recommended value documented in *Section 47.6: RNG entropy source validation*.

Writing any bit of RNG_CONFIG1 is taken into account only if the CONDRST bit is set to 1 in the same access, while CONFIGLOCK remains at 0. Writing to this bit is ignored if CONFIGLOCK = 1.

Bits 19:16 **CLKDIV[3:0]**: Clock divider factor

This value used to configure an internal programmable divider (from 1 to 16) acting on the incoming RNG clock. These bits can be written only when the core is disabled (RNGEN = 0).

0x0: internal RNG clock after divider is similar to incoming RNG clock.

0x1: two RNG clock cycles per internal RNG clock.

0x2: 2^2 (= 4) RNG clock cycles per internal RNG clock.

...

0xF: 2^{15} RNG clock cycles per internal clock (for example. an incoming 48 MHz RNG clock becomes a 1.5 kHz internal RNG clock)

Writing these bits is taken into account only if the CONDRST bit is set to 1 in the same access, while CONFIGLOCK remains at 0. Writing to this bit is ignored if CONFIGLOCK = 1.

Bits 15:13 **RNG_CONFIG2[2:0]:** RNG configuration 2

Reserved to the RNG configuration (bitfield 2). Bit 13 can be set when RNG power consumption is critical. See *Section 47.3.8: RNG low-power use*. Refer to the RNG_CONFIG1 bitfield for details.

Bit 12 **NISTC**: NIST custom

0: Hardware default values for NIST compliant RNG. In this configuration per 128-bit output two conditioning loops are performed and 256 bits of noise source are used.

1: Custom values for NIST compliant RNG. See *Section 47.6: RNG entropy source validation* for recommended configuration.

Writing this bit is taken into account only if CONDRST bit is set to 1 in the same access, while CONFIGLOCK remains at 0. Writing to this bit is ignored if CONFIGLOCK = 1.

Bits 11:8 **RNG_CONFIG3[3:0]:** RNG configuration 3

Reserved to the RNG configuration (bitfield 3). Refer to RNG_CONFIG1 bitfield for details.

If the NISTC bit is cleared in this register RNG_CONFIG3 bitfield values are ignored by RNG.

Bit 7 **ARDIS**: Auto reset disable

Set this bit to deactivate the auto-reset feature.

0: Auto-reset enabled

1: Auto-reset disabled

Keeping the auto-reset enabled (automatic clearance of the SECS bit) simplifies the management of noise source errors, as described in *Section 47.3.7: Error management*.

Writing this bit is taken into account only if CONDRST bit is set to 1 in the same access, while CONFIGLOCK remains at 0. Writing to this bit is ignored if CONFIGLOCK = 1.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CED:** Clock error detection

0: Clock error detection enabled

1: Clock error detection is disabled

The clock error detection cannot be enabled nor disabled on-the-fly when the RNG is enabled, that is to enable or disable CED, the RNG must be disabled.

Writing this bit is taken into account only if the CONDRST bit is set to 1 in the same access, while CONFIGLOCK remains at 0. Writing to this bit is ignored if CONFIGLOCK = 1.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **IE:** Interrupt enable

0: RNG interrupt is disabled

1: RNG interrupt is enabled. An interrupt is pending as soon as the DRDY, SEIS, or CEIS is set in the RNG_SR register.

Bit 2 **RNGEN:** True random number generator enable

0: True random number generator is disabled. Analog noise sources are powered off and logic clocked by the RNG clock is gated.

1: True random number generator is enabled.

Bits 1:0 Reserved, must be kept at reset value.

47.7.2 RNG status register (RNG_SR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEIS	CEIS	Res.	Res.	SECS	CECS	DRDY
									rc_w0	rc_w0			r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **SEIS:** Seed error interrupt status

This bit is set at the same time as SECS. It is cleared by writing 0 (unless CONDRST is used). Writing 1 has no effect.

0: No faulty sequence detected

1: At least one faulty sequence is detected. See SECS bit description for details.

An interrupt is pending if IE = 1 in the RNG_CR register.

Bit 5 **CEIS:** Clock error interrupt status

This bit is set at the same time as CECS. It is cleared by writing 0. Writing 1 has no effect.

0: The RNG clock is correct ($f_{RNGCLK} > f_{HCLK}/32$)

1: The RNG clock before the internal divider is detected too slow ($f_{RNGCLK} < f_{HCLK}/32$)

An interrupt is pending if IE = 1 in the RNG_CR register.

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **SECS:** Seed error current status

0: No faulty sequence has currently been detected. If the SEIS bit is set, this means that a faulty sequence was detected and the situation has been recovered.

1: At least one of the following faulty sequences has been detected:

- Runtime repetition count test failed (noise source has provided more than 24 consecutive bits at a constant value 0 or 1, or more than 32 consecutive occurrence of two bits patterns 01 or 10)
- Startup or continuous adaptive proportion test on noise source failed.
- Startup post-processing/conditioning sanity check failed.

Bit 1 **CECS:** Clock error current status

0: The RNG clock is correct ($f_{RNGCLK} > f_{HCLK}/32$). If the CEIS bit is set, this means that a slow clock was detected and the situation has been recovered.

1: The RNG clock is too slow ($f_{RNGCLK} < f_{HCLK}/32$).

Note: CECS bit is valid only if the CED bit in the RNG_CR register is set to 0.

Bit 0 **DRDY:** Data ready

0: The RNG_DR register is not yet valid, no random data is available.

1: The RNG_DR register contains valid random data.

Once the output buffer becomes empty (after reading the RNG_DR register), this bit returns to 0 until a new random value is generated.

Note: The DRDY bit can rise when the peripheral is disabled (RNGEN = 0 in the RNG_CR register).

If IE=1 in the RNG_CR register, an interrupt is generated when DRDY = 1.

47.7.3 RNG data register (RNG_DR)

Address offset: 0x008

Reset value: 0x0000 0000

The RNG_DR register is a read-only register that delivers a 32-bit random value when read. The content of this register is valid when the DRDY = 1 and the value is not 0x0, even if RNGEN = 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RNDATA[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RNDATA[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RNDATA[31:0]:** Random data

32-bit random data, which are valid when DRDY = 1. When DRDY = 0, the RNDATA value is zero.
When DRDY is set, it is recommended to always verify that RNG_DR is different from zero. The zero value means that a seed error occurred between RNG_SR polling and RND_DR output reading (a rare event).

47.7.4 RNG noise source control register (RNG_NSCR)

Address offset: 0x00C

Reset value: 0x0003 FFFF

Writing in RNG_NSCR is taken into account only if the CONDRST bit is set, and the CONFIGLOCK bit is cleared in RNG_CR. Writing to this register is ignored if CONFIGLOCK= 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN_OSC6[2:1]	
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN_OSC6[0]	EN_OSC5[2:0]			EN_OSC4[2:0]			EN_OSC3[2:0]			EN_OSC2[2:0]			EN_OSC1[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:15 **EN_OSC6[2:0]:**

When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 6. The bitfield has no effect otherwise.

Bits 14:12 **EN_OSC5[2:0]:**

When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 5. The bitfield has no effect otherwise.

Bits 11:9 **EN_OSC4[2:0]:**

When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 4. The bitfield has no effect otherwise.

Bits 8:6 **EN_OSC3[2:0]:**

When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 3. The bitfield has no effect otherwise.

Bits 5:3 **EN_OSC2[2:0]:**

When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 2. The bitfield has no effect otherwise.

Bits 2:0 **EN_OSC1[2:0]:**

When the RNG is enabled (RNGEN bit set), each bit of this bitfield enables one of the three inputs from the oscillator instance number 1. The bitfield has no effect otherwise.

47.7.5 RNG health test control register (RNG_HTCR)

Address offset: 0x010

Reset value: 0x0000 72AC

Writing in RNG_HTCR is taken into account only if the CONDRST bit is set, and the CONFIGLOCK bit is cleared in the RNG_CR. Writing to this register is ignored if CONFIGLOCK=1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HTCFG[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HTCFG[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **HTCFG[31:0]:** health test configuration

This configuration is used by RNG to configure the health tests. See *Section 47.6: RNG entropy source validation* for the recommended value.

Note: The RNG behavior, including the read to this register, is not guaranteed if a different value from the recommended value is written.

47.7.6 RNG register map

Table 400. RNG register map and reset map

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	RNG_CR	CONFIGLOCK	CONDRST	Res.	Res.	RNG_CONFIG1[7:0]								CLKDIV [3:0]				RNG_ CONFIG2 [2:0]			NISTC	RNG_ CONFIG3 [3:0]				ARDIS	Res.	CED	Res.	IE	RNGEN	Res.	Res.
	Reset value	0	0					0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0		0		0	0		
0x000	RNG_CR	CONFIGLOCK	CONDRST	Res.	Res.	Res.	Res.	RNG_CONFIG1[5:0]						CLKDIV [3:0]				RNG_ CONFIG2 [2:0]			NISTC	RNG_ CONFIG3 [3:0]				ARDIS	Res.	CED	Res.	IE	RNGEN	Res.	Res.
	Reset value	0	0					0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0		0		0	0		
0x004	RNG_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEIS	CEIS	Res.	Res.	SECS	CECS	DRDY
	Reset value																										0	0			0	0	0
0x008	RNG_DR	RNDATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	RNG_NSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN_OSC6[2:0]			EN_OSC5[2:0]			EN_OSC4[2:0]			EN_OSC3[2:0]			EN_OSC2[2:0]			EN_OSC1[2:0]		
	Reset value															1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x010	RNG_HTCR	HTCFG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	1	0	1	0	1	0	1	1	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

48 Secure AES coprocessor (SAES)

48.1 SAES introduction

The secure AES coprocessor (SAES) encrypts or decrypts data in compliance with the advanced encryption standard (AES) defined by NIST. It incorporates a protection against side-channel attacks (SCA), including differential power analysis (DPA), required for SESIP and PSA security assurance level 3 target.

SAES supports ECB, CBC, CTR, GCM, GMAC, and CCM chaining modes for key sizes of 128 or 256 bits, as well as special modes such as hardware secret key encryption/ decryption (wrapped-key mode) and key sharing with faster CRYP peripheral (shared-key mode).

SAES has the possibility to load by hardware STM32 hardware secret master keys (boot hardware key BHK and derived hardware unique key DHUK), usable but not readable by the application.

The peripheral supports DMA single transfers for incoming and outgoing data (two DMA channels are required). It is hardware-linked with the true random number generator (TRNG) and with the CRYP peripheral.

48.2 SAES main features

- Compliant with NIST FIPS publication 197 *"Advanced encryption standard (AES)"* (November 2001)
- Encryption and decryption with multiple chaining modes:
  - Electronic codebook (ECB) mode
  - Cipher block chaining (CBC) mode
  - Counter (CTR) mode
  - Galois counter mode (GCM)
  - Galois message authentication code (GMAC) mode
  - Counter with CBC-MAC (CCM) mode
- Protection against side-channel attacks (SCA), incl. differential power analysis (DPA), required for SESIP and PSA security assurance level 3 target
- 128-bit data block processing, supporting cipher key lengths of 128-bit and 256-bit
  - 480 or 680 clock cycle latency in ECB mode for processing one 128-bit block with, respectively, 128-bit or 256-bit key
- Hardware secret key encryption/ decryption (Wrapped-key mode)
- Using dedicated key bus, optional key sharing with faster CRYP peripheral (shared-key mode), controlled by SAES
- Integrated key scheduler to compute the last round key for ECB/CBC decryption
- 256-bit of write-only registers for storing cryptographic keys (eight 32-bit registers)
  - Optional 128-bit or 256-bit hardware loading of two hardware secret keys (BHK, DHUK) that can be XOR-ed together
- Security context enforcement for keys
- 128-bit of registers for storing initialization vectors (four 32-bit registers)

- 32-bit buffer for data input and output
- Automatic data flow control supporting two direct memory access (DMA) channels, one for incoming data, one for processed data. Only single transfers are supported.
- Data-swapping logic to support 1-, 8-, 16-, or 32-bit data
- AMBA AHB slave peripheral, accessible through 32-bit word single accesses only. Other access types generate an AHB error, and other than 32-bit writes may corrupt the register content.
- Possibility for software (in CPU mode only, not in DMA mode) to suspend a message if SAES needs to process another message with a higher priority, then resume the original message

48.3 SAES implementation

The devices have one SAES peripheral, implemented as per the following table. It shares the key with the CRYP peripheral. For comparison, the CRYP peripheral is also included in the table.

Table 401. CRYP versus SAES features

Modes or features(1)	CRYP	SAES
ECB, CBC chaining	X	X
CTR, CCM, GCM chaining	X	X
AES 128-bit ECB encryption in cycles	14	480
DHUK and BHK key selection	-	X
Resistance to side-channel attacks	-	X
Shared key between SAES and CRYP	X	
Key sizes in bits	128, 192, 256	128, 256

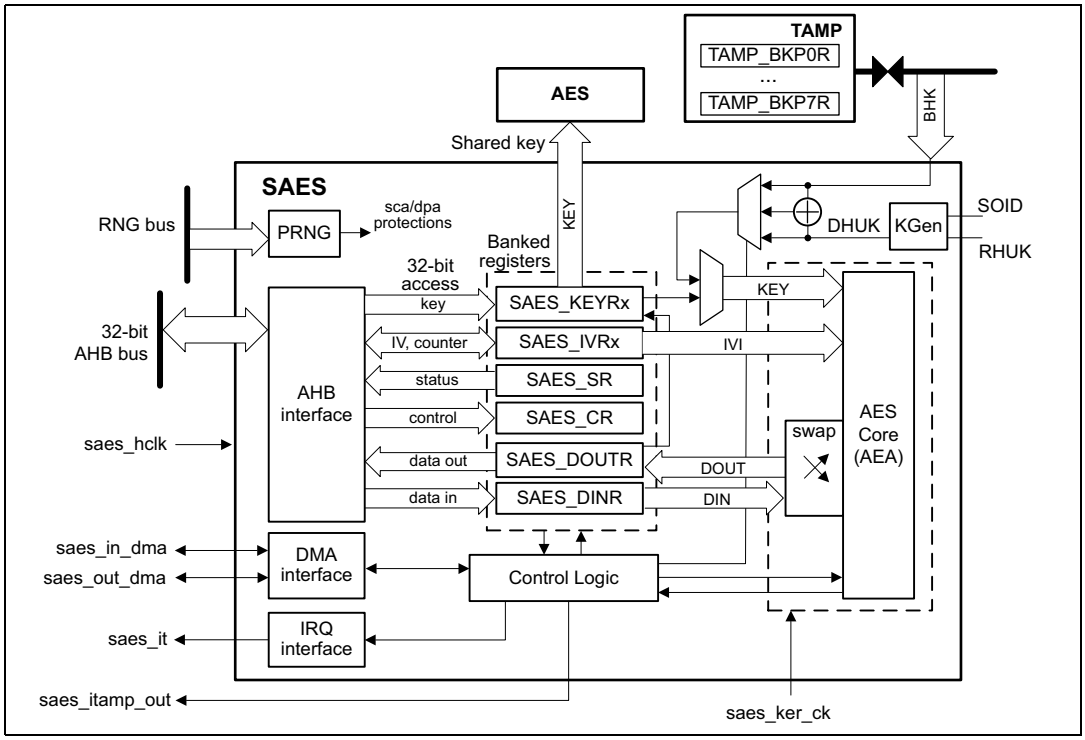
1. X = supported.

48.4 SAES functional description

48.4.1 SAES block diagram

Figure 454 shows the block diagram of SAES.

Figure 454. SAES block diagram



Note: AES represents the CRYP peripheral.

48.4.2 SAES internal signals

Table 402 describes the user relevant internal signals interfacing the SAES peripheral.

Table 402. SAES internal input/output signals

Signal name	Signal type	Description
saes_hclk	Input	AHB bus clock
saes_ker_ck	Input	SAES kernel clock.
saes_it	Output	SAES interrupt request
saes_in_dma	Input/Output	SAES incoming data DMA single request/acknowledge
saes_out_dma	Input/Output	SAES processed data DMA single request/acknowledge
saes_itamp_out	Output	Tamper event signal to TAMP (XOR-ed), triggered when an unexpected hardware fault occurs. When this signal is triggered, SAES automatically clears key registers. A reset is required for SAES to be usable again.
RHUK	Input	256-bit root hardware unique key (nonvolatile, unique per device and secret to software), used to internally compute the derived hardware unique key (DHUK)

Table 402. SAES internal input/output signals (continued)

Signal name	Signal type	Description
BHK(1)	Input	256-bit boot hardware key (BHK) stored in tamper-resistant secure backup registers and written by a secure code during boot. Once written, this key cannot be read nor written by any application until the next product reset.
SOID	Input	Static operating ID hardware input used during each DHUK computation. Its change causes the erasure of key registers and sets the KEIF flag.

1. Connected to a set of backup registers in TAMP peripheral that are written, then read/write locked, by the application software (see *Section 48.4.17* for details).

48.4.3 SAES reset and clocks

The SAES peripheral is clocked by the AHB bus clock. It has a dedicated reset bit and a dedicated kernel clock, controlled through the RCC.

The SAES reset in RCC is mandatory when a hardware tamper event occurs in SAES. Refer to *Managing tamper errors* in *Section 48.4.19* for details.

After clocking then releasing the reset of the SAES peripheral in the RCC, SAES automatically draws random numbers from the RNG, setting the BUSY bit of the SAES_SR register. Refer to the *RNG error interrupt flag (RNGEIF)* in *Section 48.4.19* for details.

This background task must be completed before the device enters a low-power mode.

48.4.4 SAES symmetric cipher implementation

The secure AES coprocessor (SAES) is a 32-bit AHB peripheral that encrypts or decrypts 16-byte blocks of data using the advanced encryption standard (AES). It also implements a set of approved AES symmetric key security functions summarized in *Table 403*. Those functions can be certified NIST PUB 140-3.

Table 403. SAES approved symmetric key functions

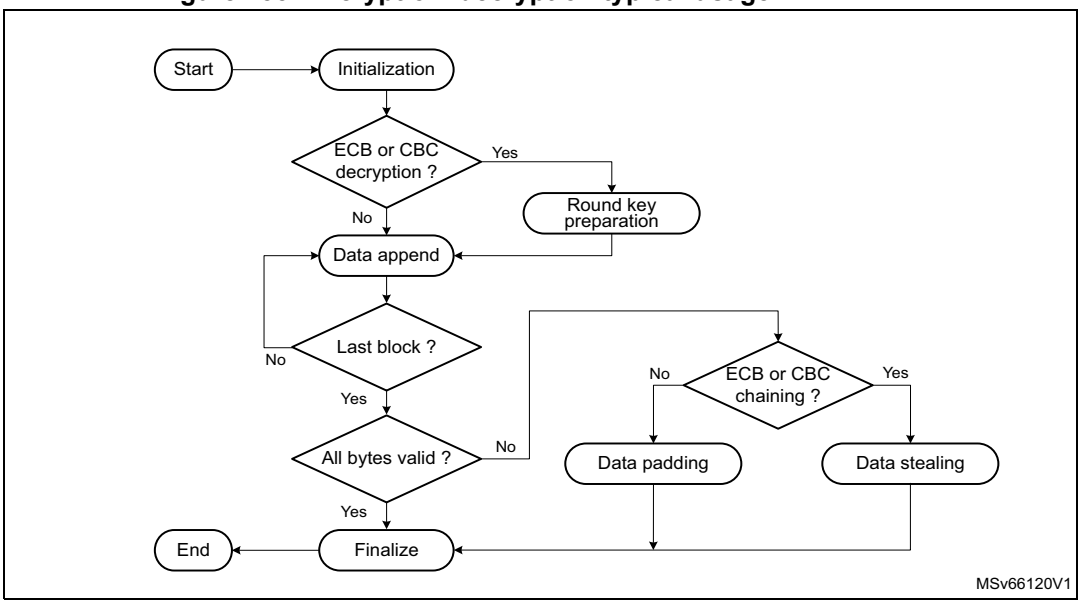
Operations	Algorithm	Specification	Key bit lengths	Chaining modes
Encryption, decryption	AES	FIPS PUB 197 NIST SP800-38A	128, 256	ECB, CBC, CTR
Authenticated encryption or decryption		NIST SP800-38C NIST SP800-38D		GCM, CCM
Cipher-based message authentication code		NIST SP800-38D		GMAC

SAES can be used directly by the CPU, or indirectly, using two DMA channels (one for the plaintext, one for the ciphertext).
It is possible to suspend then resume any SAES processing, following the sequence described in *Section 48.4.8*.

48.4.5 SAES encryption or decryption typical usage

The following figure shows a typical operation for encryption or decryption.

Figure 455. Encryption/ decryption typical usage



Initialization

The SAES peripheral is initialized according to the chaining mode. Refer to *Section 48.4.9: SAES basic chaining modes (ECB, CBC)* and *Section 48.4.10: SAES counter (CTR) mode* for details.

Data append

This section describes different ways of appending data for processing. For ECB or CBC chaining modes, refer to *Section 48.4.7: SAES ciphertext stealing and data padding* if the size of data to process is not a multiple of 16 bytes. The last block management in these cases is more complex than what is described in this section.

Appending data using the CPU in polling mode

This method uses flag polling to control the data append through the following sequence:

1. Enable the SAES peripheral when KEYVALID is set, by setting the EN bit of the SAES_CR register (if not already done).
2. Repeat the following sub-sequence until the payload is entirely processed:
   a) Write four input data words into the SAES_DINR register.
   b) Wait until the status flag CCF is set in the SAES_ISR register, then read the four data words from the SAES_DOUTR register.
   c) Clear the CCF flag, by setting the CCF bit of the SAES_ICR register.
   d) If the next processing block is the last block, pad (when applicable) the data with zeros to obtain a complete block, and specify the number of non-valid bytes (using

NPBLB[3:0]) in case of GCM payload encryption or CCM payload decryption (otherwise the tag computation is wrong).

3. As the data block just processed is the last block of the message, optionally discard the data that is not part of the message/payload, then disable the SAES peripheral by clearing EN.

Note: Up to three wait cycles are automatically inserted between two consecutive writes to the SAES_DINR register, to allow sending the key to the AES co-processor.

NPBLB[3:0] bitfield is not used in header phase of GCM, GMAC and CCM chaining modes.

Appending data using the CPU in interrupt mode

The method uses interrupt from the SAES peripheral to control the data append, through the following sequence:

1. Enable interrupts from SAES, by setting the CCFIE bit of the SAES_IER register.
2. Enable the SAES peripheral when KEYVALID is set, by setting EN (if not already done).
3. Write first four input data words into the SAES_DINR register.
4. Handle the data in the SAES interrupt service routine. Upon each interrupt:
   a) Read four output data words from the SAES_DOUTR register.
   b) Clear the CCF flag and thus the pending interrupt, by setting the CCF bit of the SAES_ICR register.
   c) If the next processing block is the last block of the message, pad (when applicable) the data with zeros to obtain a complete block, and specify the number of non-valid bytes (through NPBLB[3:0]) in case of GCM payload encryption or CCM payload decryption (otherwise the tag computation is wrong). Then proceed with point 4e).
   d) If the data block just processed is the last block of the message, optionally discard the data that are not part of the message/payload, then disable the SAES peripheral by clearing EN and quit the interrupt service routine.
   e) Write next four input data words into the SAES_DINR register and quit the interrupt service routine.

Note: SAES is tolerant of delays between consecutive read or write operations, which allows, for example, an interrupt from another peripheral to be served between two SAES computations.

The NPBLB[3:0] bitfield is not used in the header phase of GCM, GMAC, and CCM chaining modes.

Appending data using DMA

With this method, all the transfers and processing are managed by DMA and SAES. Proceed as follows:

1. If the last block of the message to process is shorter than 16 bytes, prepare the last four-word data block by padding the remainder of the block with zeros.
2. Configure the DMA controller so as to transfer the data to process from the memory to the SAES peripheral input and the processed data from the SAES peripheral output to the memory, as described in *Section 48.6: SAES DMA requests*. Configure the DMA controller so as to generate an interrupt on transfer completion. For GCM payload encryption or CCM payload decryption, the DMA transfer **must not** include the last four-word block if padded with zeros. The sequence described in *Appending data using the CPU in polling mode* must be used instead for this last block, because the

NPBLB[3:0] bitfield must be set up before processing the block, for SAES to compute a correct tag.

3. Enable the SAES peripheral when KEYVALID is set, by setting EN (if not already done).
4. Enable DMA requests, by setting DMAINEN and DMAOUTEN.
5. Upon DMA interrupt indicating the transfer completion, get the SAES-processed data from the memory.

When appending data using DMA, the suspend/resume operation as described in *Section 48.4.8* is not supported.

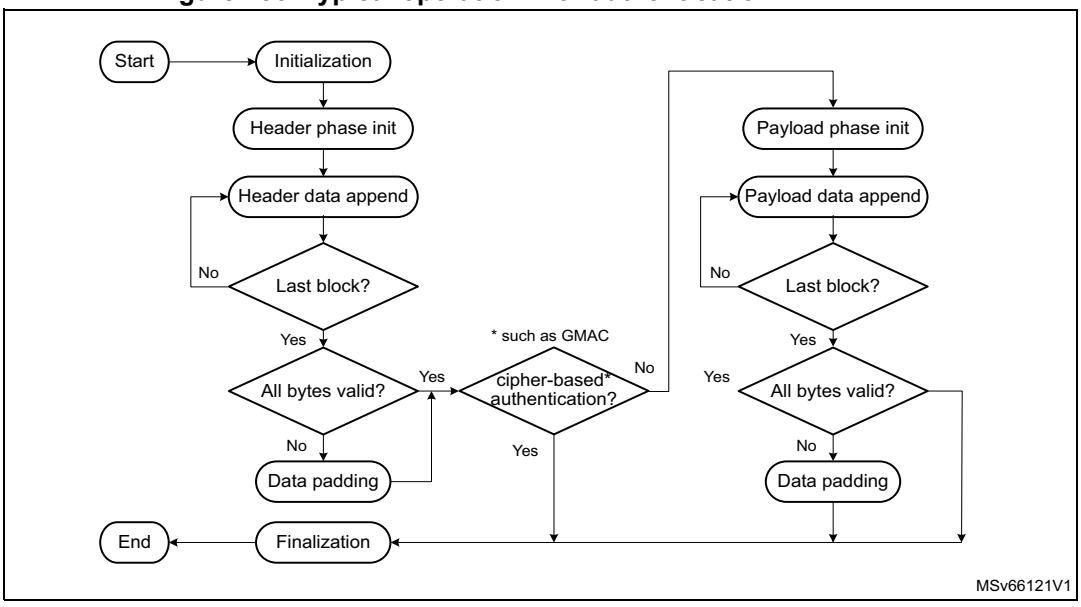
Note: The CCF flag has no use with this method because the reading of the SAES_DOUTR register is managed by DMA automatically, without any software action, at the end of the computation phase.

The NPBLB[3:0] bitfield is not used in the header phase of GCM, GMAC, and CCM chaining modes.

48.4.6 SAES authenticated encryption, decryption, and cipher-based message authentication

The following figure shows a typical operation for authenticated encryption or decryption, and for cipher-based message authentication.

Figure 456. Typical operation with authentication



Section 48.4.11: SAES Galois/counter mode (GCM) and *Section 48.4.13: SAES counter with CBC-MAC (CCM)* describe detailed sequences supported by SAES.

Cipher-based message authentication flow omits the payload phase, as shown in the figure. Detailed sequence supported by SAES is described in *Section 48.4.12: SAES Galois message authentication code (GMAC)*.

48.4.7 SAES ciphertext stealing and data padding

When using SAES in ECB or CBC modes to manage messages the size of which is not a multiple of the block size (16 bytes), the application must use ciphertext stealing techniques such as those described in NIST Special Publication 800-38A*, Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode*. Since SAES does not implement such techniques, the application must complete the *last block* of input data using data from the *second last* block.

Note: Ciphertext stealing techniques are not documented in this reference manual.

Similarly, in modes other than ECB or CBC, an incomplete input data block (that is, a block with input data shorter than 16 bytes) must be padded with zeros prior to encryption. That is, extra bits must be appended to the trailing end of the data string. After decryption, the extra bits must be discarded. Since SAES does not implement automatic data padding operation to the *last block*, the application must follow the recommendation given in this document to manage messages the size of which is not a multiple of 16 bytes.

48.4.8 SAES suspend and resume operations

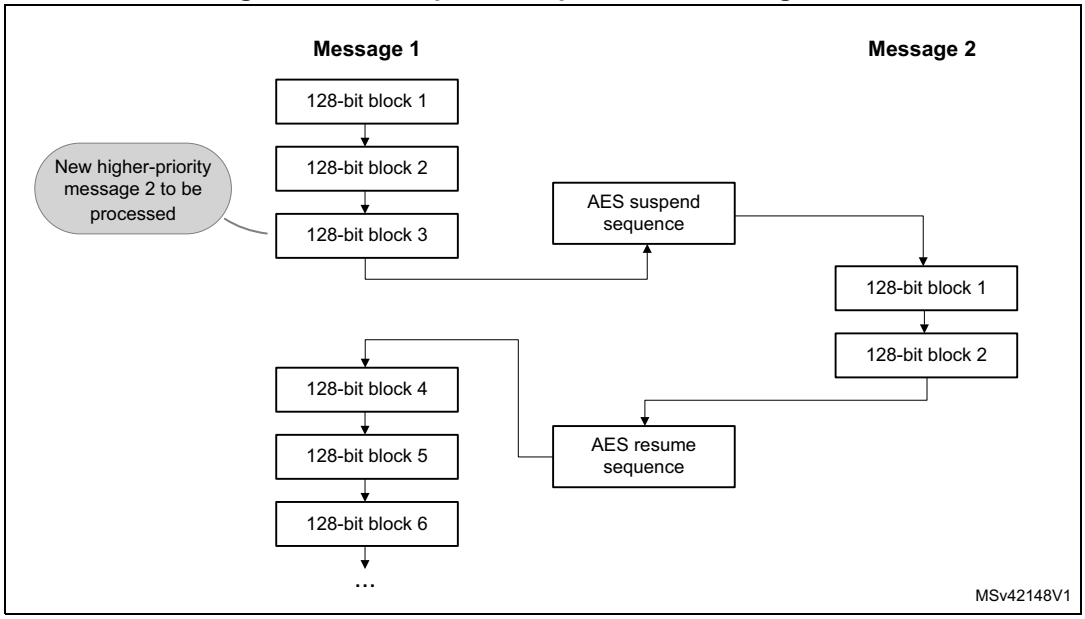
A message can be suspended to process another message with a higher priority. When the higher-priority message is sent, the suspended message can resume. This applies to both encryption and decryption mode.

Suspend and resume operations do not break the chaining operation. The message processing can resume as soon as SAES is enabled again, to receive a next data block.

The suspend and resume operations are only supported when SAES is used in CPU mode, not in DMA mode.

Figure 457 gives an example of suspend and resume operations: Message 1 is suspended in order to send a shorter and higher-priority Message 2.

Figure 457. Example of suspend mode management



A detailed description of suspend and resume operations is in the sections dedicated to each chaining mode.

48.4.9 SAES basic chaining modes (ECB, CBC)

ECB is the simplest mode of operation. There are no chaining operations, and no special initialization stage. The message is divided into blocks and each block is encrypted or decrypted separately. When decrypting in ECB, a special key scheduling is required before processing the first block.

Figure 458 and *Figure 459* describe the electronic codebook (ECB) chaining implementation in encryption and in decryption, respectively. To select ECB chaining mode, write CHMOD[2:0] with 0x0.

Figure 458. ECB encryption

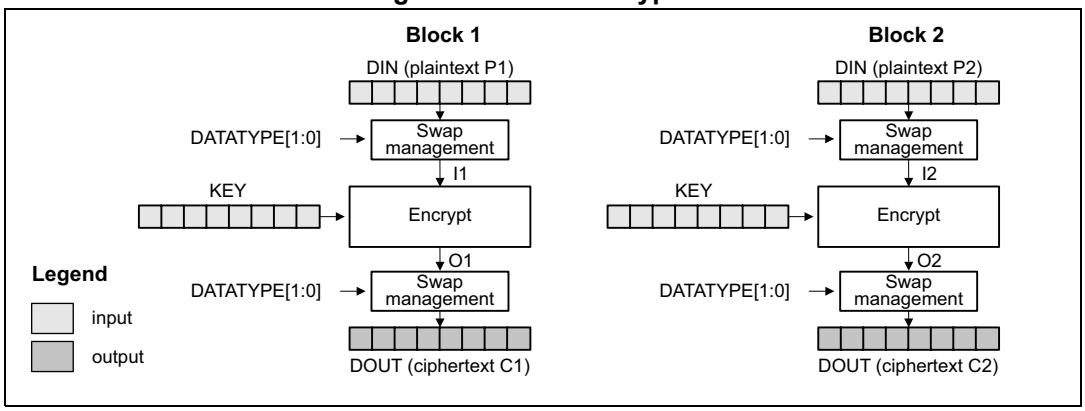
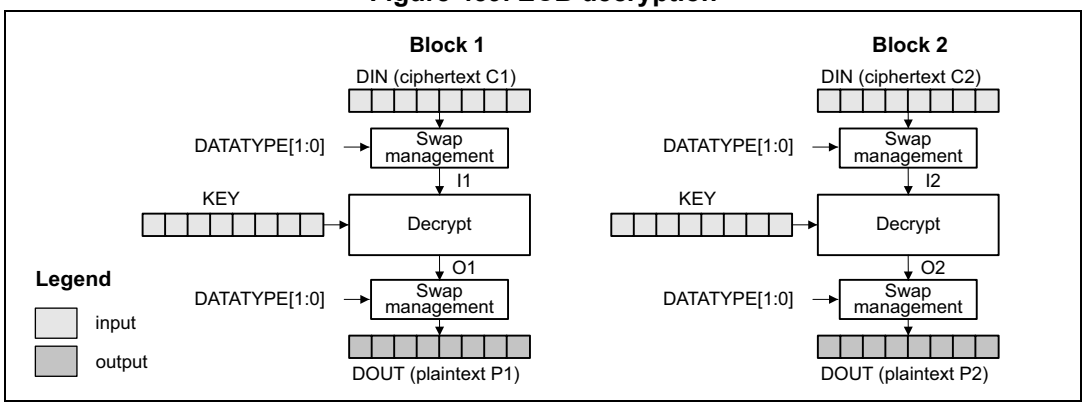


Figure 459. ECB decryption



In CBC encryption mode the output of each block chains with the input of the following block. To make each message unique, an initialization vector is used during the first block processing. When decrypting in CBC, a special key scheduling is required before processing the first block.

Figure 460 and *Figure 461* describe the cipher block chaining (CBC) implementation in encryption and in decryption, respectively. To select this chaining mode, write CHMOD[2:0] with 0x1.

Figure 460. CBC encryption

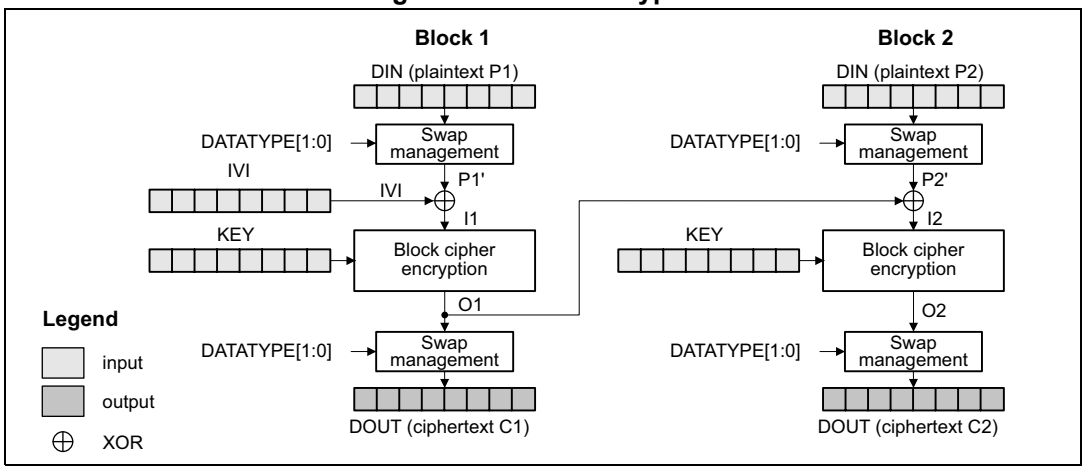
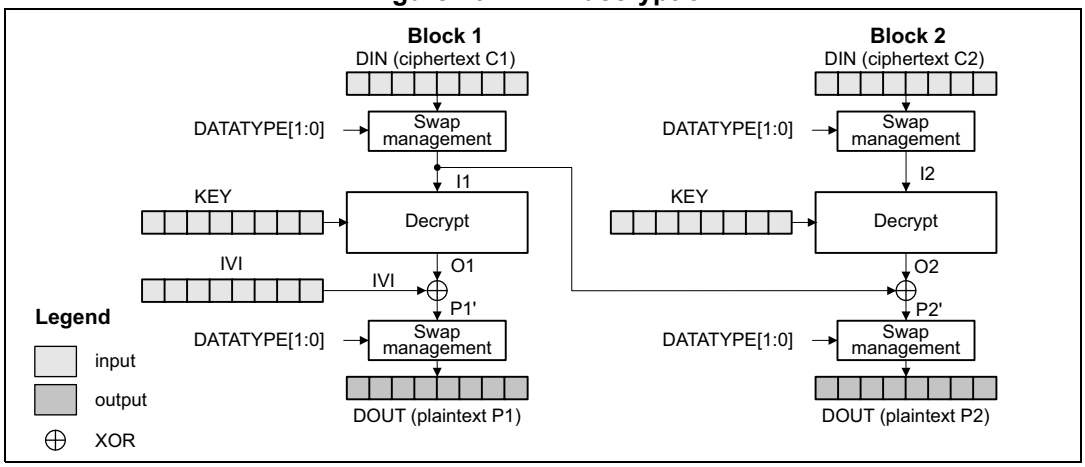


Figure 461. CBC decryption



For more details, refer to NIST Special Publication 800-38A, *Recommendation for Block Cipher Modes of Operation.*

ECB and CBC encryption process

This process is described in *Section 48.4.5*, with the following sequence of events:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register as follows:
   - Select ECB or CBC chaining mode (write CHMOD[2:0] with 0x0 or 0x1) in *encryption* mode (write MODE[1:0] with 0x0).
   - Configure the data type, through DATATYPE[1:0].
   - Configure the key size, through KEYSIZE. If the key must not be shared with a different security context (different secure attribute), the KEYPROT bit must also be set.
   - Select normal key mode by writing KMOD[1:0] with 0x0. For the other KMOD[1:0] values, refer to *Section 48.4.14* (wrapped keys) and *Section 48.4.15* (shared keys).
4. Write the initialization vector into the SAES_IVRx registers if CBC mode is selected in the previous step.
5. Write the key into the SAES_KEYRx registers. Alternatively, select a key source different from the key registers by writing KEYSEL[2:0] with a value different from 0x0. Refer to *Section 48.4.17: SAES key registers* for details.
6. Wait until KEYVALID is set (the key loading completed).
7. Enable the SAES peripheral, by setting EN.
8. Append cleartext data:
   a) If it is the second-last or the last block and the plaintext size of the message is not a multiple of 16 bytes, follow the guidance in *Section 48.4.7*.
   b) Append the cleartext block into SAES as described in *Section 48.4.5*, then read the SAES_DOUTR register four times to save the ciphertext block.
   c) Repeat the step *b)* until the third-last plaintext block is encrypted. For the last two blocks, follow the steps *a)* and *b)*.
9. Finalize the sequence: disable the SAES peripheral, by clearing EN.

ECB/CBC decryption process

This process is described in *Section 48.4.5*, with the following sequence of events:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register as follows:
   - Select the *key derivation* mode (write MODE[1:0] with 0x1). The CHMOD[2:0] bitfield is not significant during this operation.
   - Configure the data type, through DATATYPE[1:0].
   - Configure the key size, through KEYSIZE. If the key must not be shared with a different security context (different secure attribute), the KEYPROT bit must also be set.
   - Select normal key mode by writing KMOD[1:0] with 0x0. For the other KMOD[1:0] values, refer to *Section 48.4.14* (wrapped keys) and *Section 48.4.15* (shared keys).

4. Write the key into the SAES_KEYRx registers. Alternatively, select a key source different from the key registers by writing KEYSEL[2:0] with a value different from 0x0. Refer to *Section 48.4.17: SAES key registers* for details.
5. Wait until KEYVALID is set (the key loading completed).
6. Enable the SAES peripheral, by setting EN. The peripheral immediately starts an AES round for key preparation.
7. Wait until the CCF flag in the SAES_ISR register is set.
8. Clear the CCF flag, by setting the CCF bit of the SAES_ICR register. The decryption key is available in the AES core and SAES is disabled automatically.
9. Select ECB or CBC chaining mode (write CHMOD[2:0] with 0x0 or 0x1) in *decryption* mode (write MODE[1:0] with 0x2). Do not change other parameters.
10. Write the initialization vector into the SAES_IVRx registers if CBC mode is selected in the previous step.
11. Enable the SAES peripheral, by setting EN.
12. Append encrypted data:
    a) If it is the second-last or the last block and the ciphertext size of the message is not a multiple of 16 bytes, follow the guidance in *Section 48.4.7*.
    b) Append the ciphertext block into SAES as described in *Section 48.4.5*, then read the SAES_DOUTR register four times to save the cleartext block (MSB first).
    c) Repeat the step *b)* until the third-last ciphertext block is decrypted. For the last two blocks, follow the steps *a)* and *b)*.
13. Finalize the sequence: disable the SAES peripheral, by clearing EN.

Suspend/resume operations in ECB/CBC modes

The following sequences are valid for normal key mode (KMOD[1:0] at 0x0).

The suspend and resume operations are only supported when SAES is used in CPU mode, not in DMA mode.

To suspend the processing of a message, proceed as follows:

1. Wait until the CCF flag in the SAES_ISR register is set (computation completed).
2. Read four times the SAES_DOUTR register to save the last processed block.
3. Clear the CCF flag, by setting the CCF bit of the SAES_ICR register.
4. Save initialization vector registers (only required in CBC mode as the SAES_IVRx registers are altered during the data processing).
5. Disable the SAES peripheral, by clearing EN.
6. Save the SAES_CR register and clear the key registers if they are not needed, to process the higher-priority message.

To resume the processing of a message, proceed as follows:

1. Disable the SAES peripheral, by clearing EN.
2. Restore the SAES_CR register (with correct KEYSIZE) then restore the SAES_KEYRx registers. For KEYSEL[2:0] selecting a key source different from key registers, refer to *Section 48.4.17: SAES key registers* for details.
3. Prepare the decryption key, as described in *ECB/CBC decryption process* (only required for ECB or CBC decryption).

4. Restore the SAES_IVRx registers, using the saved configuration (only required in CBC mode).
5. Enable the SAES peripheral, by setting EN.

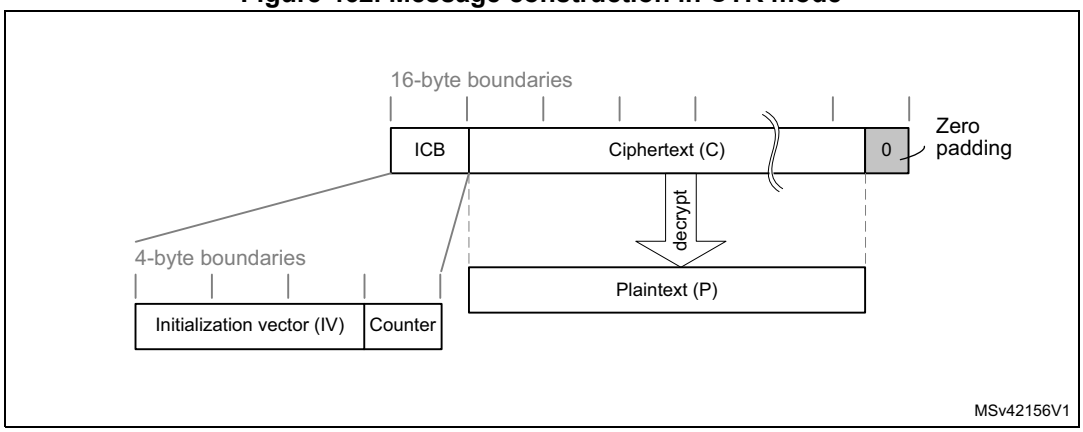
Note: It is not required to save the key registers as the application knows the original key.

48.4.10 SAES counter (CTR) mode

The CTR mode uses the AES core to generate a key stream. The keys are then XOR-ed with the plaintext to obtain the ciphertext. Unlike with ECB and CBC modes, no key scheduling is required for the CTR decryption since the AES core is always used in encryption mode.

A typical message construction in CTR mode is given in *Figure 462*.

Figure 462. Message construction in CTR mode



The structure of this message is:

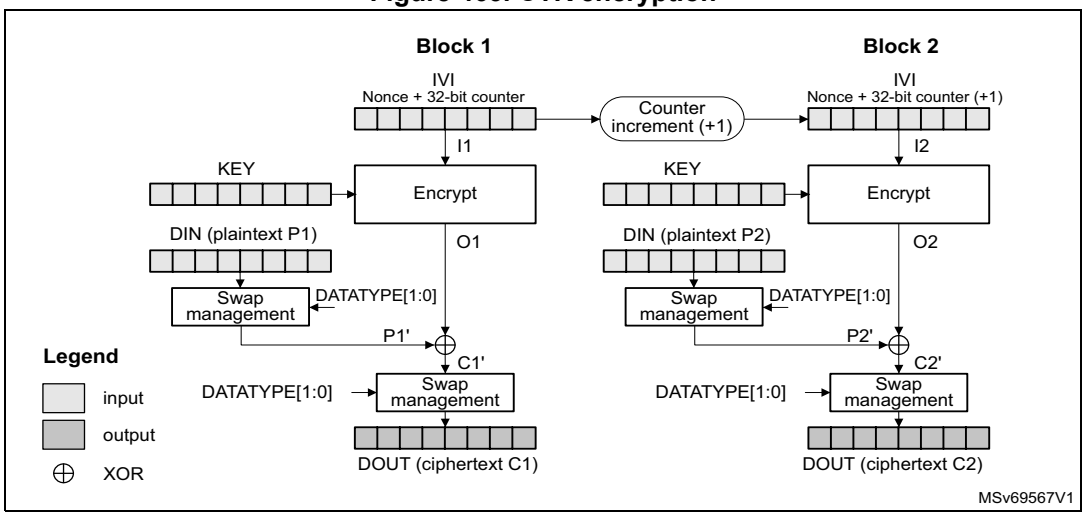
- A 16-byte initial counter block (ICB), composed of two distinct fields:
  - **Initialization vector** (IV): a 96-bit value that must be unique for each encryption cycle with a given key.
  - **Counter**: a 32-bit big-endian integer that is incremented each time a block processing is completed. The initial value of the counter must be set to 1.
- The plaintext P is encrypted as ciphertext C, with a known length. This length can be non-multiple of 16 bytes, in which case a plaintext padding is required.

For more details, refer to NIST *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation.*

CTR encryption and decryption

Figure 463 describes the counter (CTR) chaining implementation in the SAES peripheral (encryption). To select this chaining mode, write CHMOD[2:0] with 0x2.

Figure 463. CTR encryption



Initialization vectors in SAES must be initialized as shown in *Table 404*.

Table 404. Counter mode initialization vector definition

SAES_IVR3[31:0]	SAES_IVR2[31:0]	SAES_IVR1[31:0]	SAES_IVR0[31:0]
IVI[127:96]	IVI[95:64]	IVI[63:32]	IVI[31:0] 32-bit counter = 0x0001

CTR encryption and decryption process

This process is described in *Section 48.4.5*, with the following sequence of events:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register:
   - Select CTR chaining mode (write CHMOD[2:0] with 0x2) in encryption or decryption mode (write MODE[1:0] with 0x0 or 0x2).
   - Configure the data type, through DATATYPE[1:0].
   - Configure the key size, through KEYSIZE. If the key must not be shared with a different security context (different secure attribute), the KEYPROT bit must also be set.
   - Select normal key mode, by writing KMOD[1:0] with 0x0. For the other KMOD[1:0] values, refer to *Section 48.4.14* (wrapped keys) and *Section 48.4.15* (shared keys).
4. Write the initialization vector into the SAES_IVRx registers according to *Table 404*.
5. Write the key into the SAES_KEYRx registers. Alternatively, select a key source different from the key registers by writing KEYSEL[2:0] with a value different from 0x0. Refer to *Section 48.4.17: SAES key registers* for details.
6. Wait until KEYVALID is set (the key loading completed).

7. Enable the SAES peripheral, by setting EN.
8. Append data:
   a) If it is the last block and the plaintext (encryption) or ciphertext (decryption) size in the block is less than 16 bytes, pad the remainder of the block with zeros.
   b) Append the data block into SAES as described in *Section 48.4.5*, then read the SAES_DOUTR register four times to save the resulting block (MSB first).
   c) Repeat the step *b)* until the second-last block is processed. For the last block of plaintext (encryption only), follow the steps *a)* and *b)*. For the last block, discard the bits that are not part of the message when the last block is smaller than 16 bytes.
9. Finalize the sequence: disable the SAES peripheral, by clearing EN.

Suspend/resume operations in CTR mode

Like for the CBC mode, it is possible to interrupt a message to send a higher-priority message, then resume the interrupted message. Detailed CBC suspend and resume sequence is described in *Section 48.4.9: SAES basic chaining modes (ECB, CBC)*.

The suspend and resume operations are only supported when SAES is used in CPU mode, not in DMA mode.

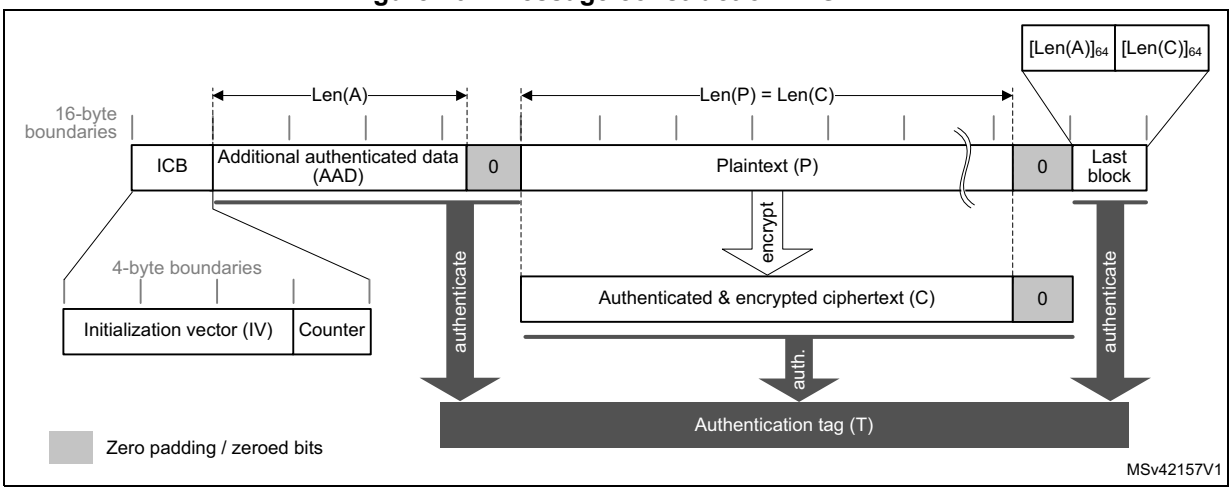
Note: Like for CBC mode, the IV registers must be reloaded during the resume operation.

48.4.11 SAES Galois/counter mode (GCM)

The AES Galois/counter mode (GCM) allows encrypting and authenticating a plaintext message into the corresponding ciphertext and tag (also known as message authentication code).

GCM mode is based on AES in counter mode for confidentiality. It uses a multiplier over a fixed finite field for computing the message authentication code. The following figure shows a typical message construction in GCM mode.

Figure 464. Message construction in GCM



The message has the following structure:

- **16-byte initial counter block (ICB)**, composed of two distinct fields:
  - **Initialization vector (IV):** a 96-bit value that must be unique for each encryption cycle with a given key. The GCM standard supports IVs with less than 96 bits, but in this case strict rules apply.
  - **Counter:** a 32-bit big-endian integer that is incremented each time a block processing is completed. According to NIST specification, the counter value is 0x2 when processing the first block of payload.
- **Authenticated header AAD** (also knows as additional authentication data) has a known length Len(A) that may be a non-multiple of 16 bytes, and must not exceed $2^{64} – 1$ bits. This part of the message is only authenticated, not encrypted.
- **Plaintext message P** is both authenticated and encrypted as ciphertext C, with a known length Len(P) that may be non-multiple of 16 bytes, and cannot exceed $2^{32} - 2$ 16-byte blocks.
- **Last block** contains the AAD header length (bits [32:63]) and the payload length (bits [96:127]) information, as shown in *Table 406*.

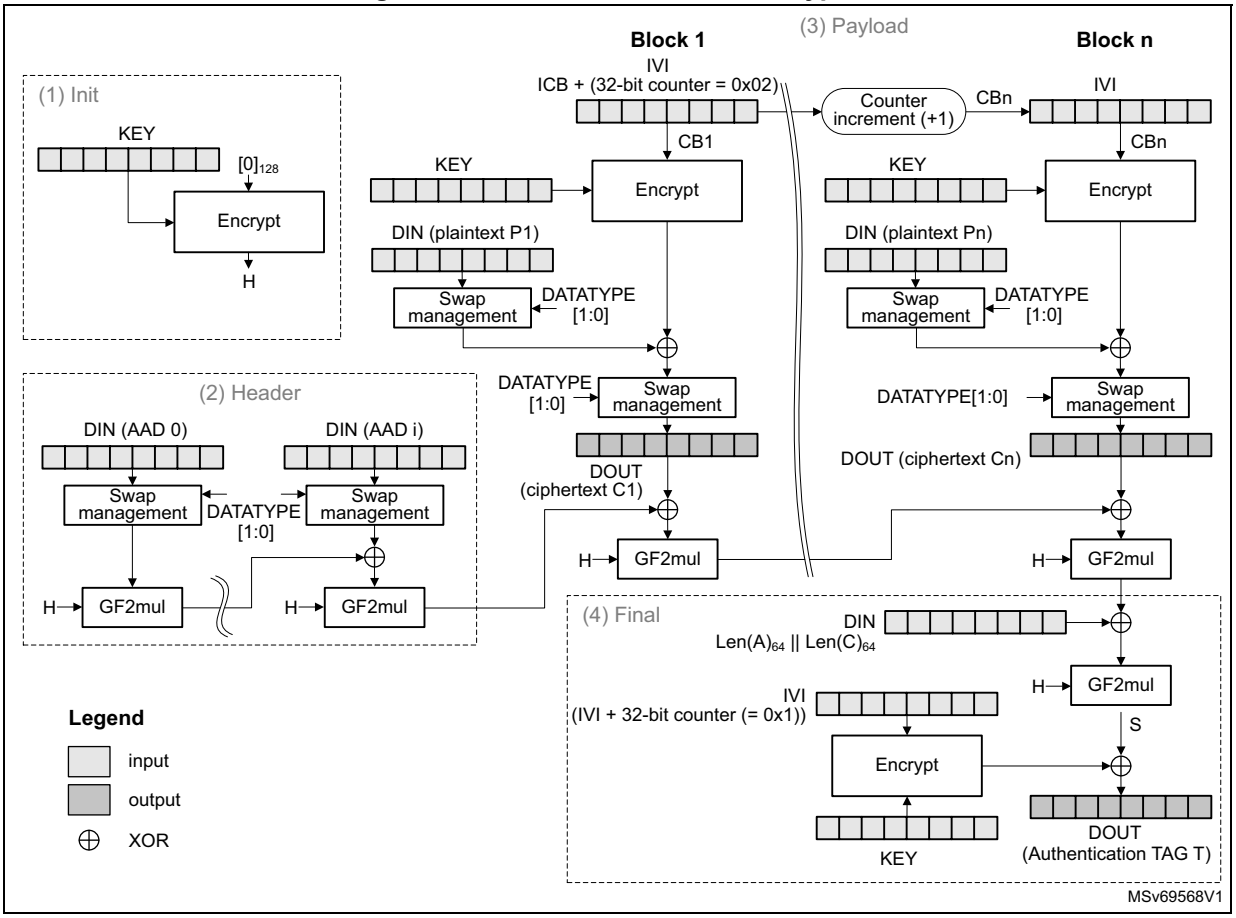
The GCM standard specifies that ciphertext C has the same bit length as the plaintext P.

When a part of the message (AAD or P) has a length that is a non-multiple of 16-bytes a special padding scheme is required.

For more details, refer to NIST Special Publication 800-38D*, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.*

Figure 465 describes the GCM chaining implementation in the SAES peripheral (encryption). To select this chaining mode, write CHMOD[2:0] with 0x3.

Figure 465. GCM authenticated encryption



The first counter block (CB1) is derived from the initial counter block ICB by the application software, as defined in *Table 405*.

Table 405. Initialization of IV registers in GCM mode

SAES_IVR3[31:0]	SAES_IVR2[31:0]	SAES_IVR1[31:0]	SAES_IVR0[31:0]
ICB[127:96]	ICB[95:64]	ICB[63:32]	ICB[31:0] 32-bit counter = 0x0002

The last block of a GCM message contains the AAD header length and the payload length information, as shown in *Table 406*.

Table 406. GCM last block definition

Word order to SAES_DINR	First word	Second word	Third word	Fourth word
Input data	AAD length[63:32]	AAD length[31:0]	Payload length[63:32]	Payload length[31:0]

GCM encryption and decryption process

This process is described in *Section 48.4.6*, with the following sequence of events:

GCM initialize

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register:
   - Select GCM chaining mode (write CHMOD[2:0] with 0x3) in encryption or decryption mode (write MODE[1:0] with 0x0 or 0x2). Do not write MODE[1:0] with 0x1.
   - Configure the data type, through DATATYPE[1:0]
   - Configure the key size, through KEYSIZE. If the key must not be shared with a different security context (such as secure, nonsecure, specific CPU), also set KEYPROT.
   - Select normal key mode, by writing KMOD[1:0] with 0x0. For the other KMOD[1:0] values, refer to *Section 48.4.14* (wrapped keys) and *Section 48.4.15* (shared keys).
   - Select the GCM initialization phase, by writing GCMPH[1:0] with 0x0.
4. Write the initialization vector in SAES_IVRx registers according to *Table 405*.
5. Write the key into the SAES_KEYRx registers. Alternatively, select a key source different from the key registers by writing KEYSEL[2:0] with a value different from 0x0. Refer to *Section 48.4.17: SAES key registers* for details.
6. Wait until KEYVALID is set (the key loading completed).
7. Set EN to start the calculation of the hash key. EN is automatically cleared when the calculation is completed.
8. Wait until the CCF flag is set in the SAES_ISR register, indicating that the GCM hash subkey (H) computation is completed.
9. Clear the CCF flag by setting the CCF bit of the SAES_ICR register.

GCM header phase

10. Initialize header phase:
    a) Select the GCM header phase, by writing 0x1 to GCMPH[1:0]. Do not change the other configurations written during GCM initialization.
    b) Enable the SAES peripheral, by setting EN.
11. Append header data:
    a) If it is the last block and the AAD in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into SAES as described in *Section 48.4.5*.
    c) Repeat the step *b)* until the second-last AAD data block is processed. For the last block, follow the steps *a)* and *b)*.

Note: This phase can be skipped if there is no AAD, that is, Len(A) = 0.

No data are read during header phase.

GCM payload phase

12. Initialize payload phase:
    a) Select the GCM payload phase, by writing GCMPH[1:0] with 0x2. Do not change the other configurations written during GCM initialization.
    b) If the header phase is skipped, enable the SAES peripheral by setting EN.
13. Append payload data:
    a) If it is the last block and the message in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into SAES as described in *Section 48.4.5*, then read the SAES_DOUTR register four times to save the resulting block
    c) Repeat the step *b)* until the second-last plaintext block is encrypted or until the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), follow the steps *a)* and *b)*. For the last block, discard the bits that are not part of the payload when the last block is smaller than 16 bytes.

Note: This phase can be skipped if there is no payload, that is, Len(C)=0 (see GMAC mode).

GCM finalization

14. Encryption only: wait until the BUSY flag in the SAES_SR register is cleared.
15. Select the GCM final phase, by writing GCMPH[1:0] with 0x3. Do not change the other configurations written during GCM initialization.
16. Write the final GCM block into the SAES_DINR register. It is the concatenated AAD bit and payload bit lengths, as shown in *Table 406*.
17. Wait until the CCF flag in the SAES_ISR register is set.
18. Get the GCM authentication tag, by reading the SAES_DOUTR register four times.
19. Clear the CCF flag, by setting the CCF bit of the SAES_ICR register.
20. Disable the SAES peripheral, by clearing EN. If it is an authenticated decryption, compare the generated tag with the expected tag passed with the message.

Note: In the final phase, data are written to SAES_DINR normally (no swapping), while swapping is applied to tag data read from SAES_DOUTR.

When transiting from the header or the payload phase to the final phase, the SAES peripheral must not be disabled, otherwise the result is wrong.

Suspend/resume operations in GCM mode

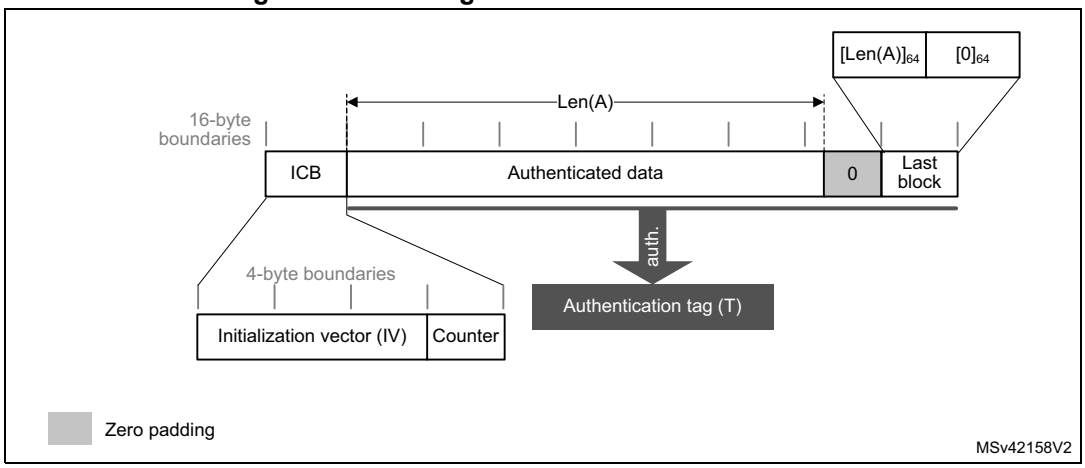
Suspend/resume operations are not supported in GCM mode.

48.4.12 SAES Galois message authentication code (GMAC)

The Galois message authentication code (GMAC) allows the authentication of a plaintext, generating the corresponding tag information (also known as message authentication code).

GMAC is similar to GCM, except that it is applied on a message composed only by plaintext authenticated data (that is, only header, no payload). The following figure shows typical message construction for GMAC.

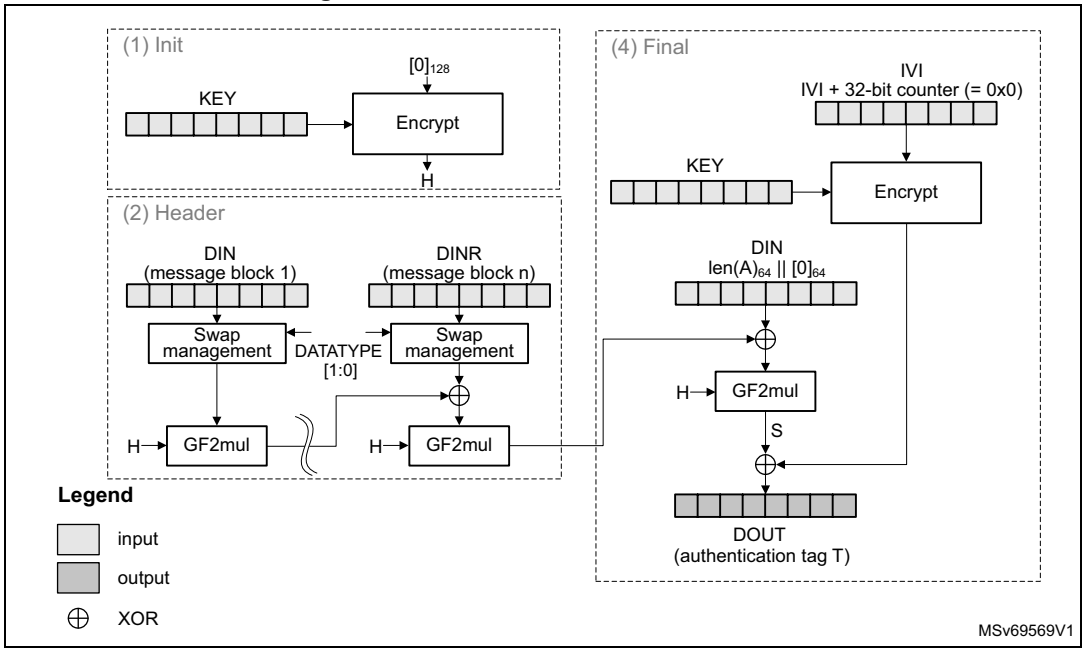
Figure 466. Message construction in GMAC mode



For more details, refer to NIST Special Publication 800-38D*, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.*

Figure 467 describes the GMAC chaining implementation in the SAES peripheral. To select this chaining mode, write CHMOD[2:0] with 0x3.

Figure 467. GMAC authentication mode



The GMAC algorithm corresponds to the GCM algorithm applied on a message that only contains a header. As a consequence, all steps and settings are the same as with the GCM, except that the payload phase is omitted.

Suspend/resume operations in GMAC

Suspend/resume operations are not supported in GMAC mode.

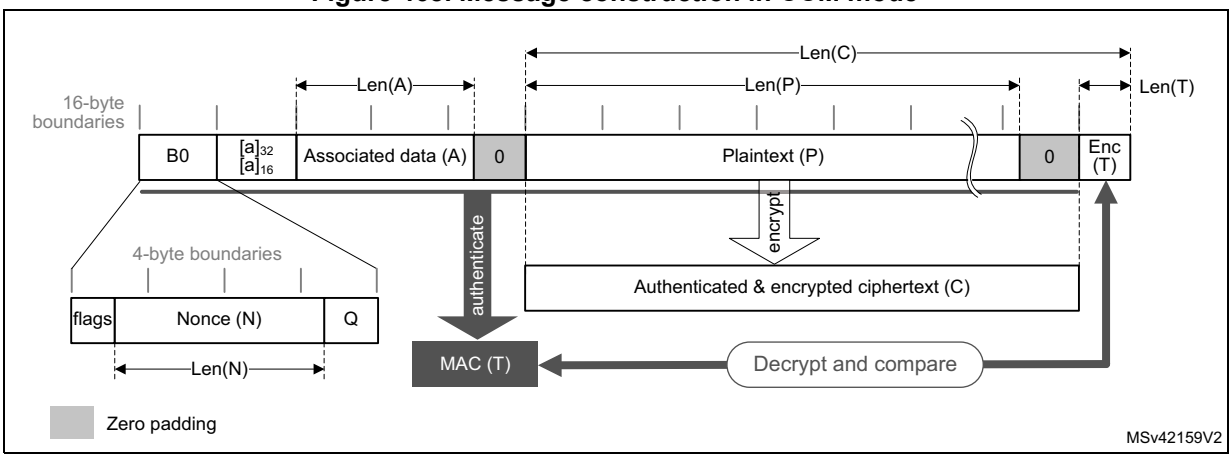
48.4.13 SAES counter with CBC-MAC (CCM)

The AES counter with cipher block chaining-message authentication code (CCM) algorithm allows encryption and authentication of plaintext, generating the corresponding ciphertext and tag (also known as message authentication code). To ensure confidentiality, the CCM algorithm is based on AES counter mode processing. It uses cipher block chaining technique to generate the message authentication code. This is commonly called CBC-MAC.

Note: NIST does not approve CBC-MAC as an authentication mode outside the context of the CCM specification.

The following figure shows typical message construction for CCM.

Figure 468. Message construction in CCM mode



The structure of the message is:

- **16-byte first authentication block (B0)**, composed of three distinct fields:
  - **Q:** a bit string representation of the octet length of P (Len(P))
  - **Nonce (N):** a single-use value (that is, a new nonce must be assigned to each new communication) of Len(N) size. The sum Len(N) + Len(P) must be equal to 15 bytes.
  - **Flags:** most significant octet containing four flags for control information, as specified by the standard. It contains two 3-bit strings to encode the values **t** (MAC length expressed in bytes) and **Q** (plaintext length such that Len(P) < 2^{8Q} bytes). The counter blocks range associated to **Q** is equal to 2^{8Q-4}, that is, if the maximum value of **Q** is 8, the counter blocks used in cipher must be on 60 bits.
- **16-byte blocks (B)** associated to the associated data (A).
  This part of the message is only authenticated, not encrypted. This section has a known length Len(A) that can be a non-multiple of 16 bytes (see *Figure 468*). The standard also states that, on MSB bits of the first message block (B1), the associated data length expressed in bytes (a) must be encoded as follows:
  - If $0 < a < 2^{16} - 2^8$, then it is encoded as $[a]_{16}$, that is, on two bytes.
  - If $2^{16} - 2^8 < a < 2^{32}$, then it is encoded as 0xff || 0xfe || $[a]_{32}$, that is, on six bytes.
  - If $2^{32} < a < 2^{64}$, then it is encoded as 0xff || 0xff || $[a]_{64}$, that is, on ten bytes.

- **16-byte blocks (B)** associated to the plaintext message P, which is both authenticated and encrypted as ciphertext C, with a known length Len(P). This length can be a non-multiple of 16 bytes (see *Figure 468*).
- **Encrypted MAC (T)** of length Len(T) appended to the ciphertext C of overall length Len(C).

When a part of the message (A or P) has a length that is a non-multiple of 16-bytes, a special padding scheme is required.

Note: CCM chaining mode can also be used with associated data only (that is, no payload).

As an example, the C.1 section in NIST Special Publication 800-38C gives the following values (hexadecimal numbers):

N: 10111213 141516 (Len(N) = 56 bits or 7 bytes)

A: 00010203 04050607 (Len(A) = 64 bits or 8 bytes)

P: 20212223 (Len(P) = 32 bits or 4 bytes)

T: 6084341B (Len(T) = 32 bits or t = 4)

B0: 4F101112 13141516 00000000 00000004
B1: 00080001 02030405 06070000 00000000
B2: 20212223 00000000 00000000 00000000

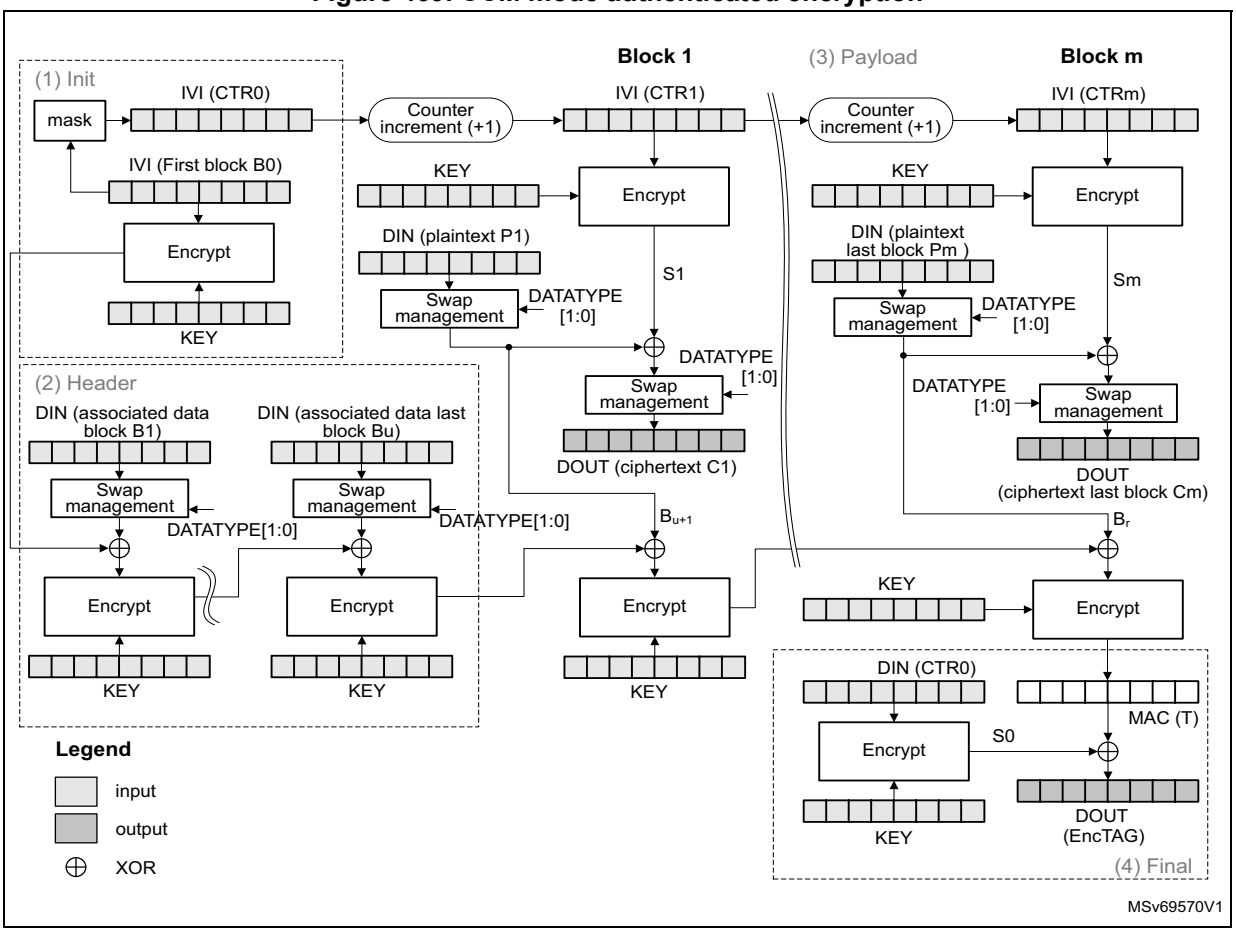
CTR0: 0710111213 141516 00000000 00000000

CTR1: 0710111213 141516 00000000 00000001

For more details, refer to NIST Special Publication 800-38C*, Recommendation for Block Cipher Modes of Operation - The CCM Mode for Authentication and Confidentiality*.

Figure 469 describes the CCM chaining implementation in the SAES peripheral (encryption). To select this chaining mode, write CHMOD[2:0] with 0x4.

Figure 469. CCM mode authenticated encryption



The first block of a CCM message (B0) must be prepared by the application as defined in *Table 407*.

Table 407. Initialization of IV registers in CCM mode

SAES_IVR3[31:0]	SAES_IVR2[31:0]	SAES_IVR1[31:0]	SAES_IVR0[31:0]
B0[127:96](1)	B0[95:64]	B0[63:32]	B0[31:0](2)

1. The five most significant bits are cleared (flag bits).
2. Q length bits are cleared, except for the bit 0 that is set.

SAES supports counters up to 64 bits, as specified by NIST.

CCM encryption and decryption process

This process is described in *Section 48.4.6*, with the following sequence of events:

CCM initialize

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register:
   - Select CCM chaining mode (write CHMOD[2:0] with 0x4) in encryption or decryption mode (write MODE[1:0] with 0x0 or 0x2). Do not write MODE[1:0] with 0x1.
   - Configure the data type, through DATATYPE[1:0]
   - Configure the key size, through KEYSIZE. If the key must not be shared with a different security context, also set the KEYPROT bit.
   - Select normal key mode, by writing KMOD[1:0] with 0x0. For the other KMOD[1:0] values, refer to *Section 48.4.14* (wrapped keys) and *Section 48.4.15* (shared keys).
   - Select the CCM initialization phase, by writing GCMPH[1:0] with 0x0.
4. Write the B0 data in SAES_IVRx registers according to *Table 407*.
5. Write the key into the SAES_KEYRx registers. Alternatively, select a key source different from the key registers by writing KEYSEL[2:0] with a value different from 0x0. Refer to *Section 48.4.17: SAES key registers* for details.
6. Wait until KEYVALID is set (the key loading completed).
7. Set EN to start the first mask calculation. The EN bit is automatically cleared when the calculation is completed.
8. Wait until the CCF flag in the SAES_ISR register is set.
9. Clear the CCF flag, by setting the CCF bit of the SAES_ICR register.

CCM header phase

10. Initialize header phase:
    a) Prepare the first block of the (B1) data associated with the message, in accordance with CCM chaining rules.
    b) Select the CCM header phase, by writing GCMPH[1:0] with 0x1. Do not change the other configurations written during the CCM initialization.
    c) Enable the SAES peripheral, by setting EN.
11. Append header data:
    a) If it is the last block and the associated data in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into SAES as described in *Section 48.4.5*.
    c) Repeat the step *b)* until the second-last associated data block is processed. For the last block, follow the steps *a)* and *b)*.

Note: This phase can be skipped if there is no associated data, that is, Len(A) = 0

No data are read during the header phase.

CCM payload phase

12. Initialize payload phase:
    a) Select the CCM payload phase, by writing GCMPH[1:0] with 0x2. Do not change the other configurations written during the CCM initialization.
    b) If the header phase is skipped, enable the SAES peripheral, by setting EN.
13. Append payload data:
    a) In encryption only, if it is the last block and the plaintext in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data block into SAES as described in *Section 48.4.5*, then read the SAES_DOUTR register four times to save the resulting block.
    c) Repeat the step *b)* until the second-last plaintext block is encrypted or until the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), follow the steps *a)* and *b)*. For the last block, discard the bits that are not part of the payload when the last block is smaller than 16 bytes.

Note: This phase can be skipped if there is no payload, that is, Len(P) = 0 or Len(C) = Len(T).

Remove $LSB_{Len(T)}(C)$ encrypted tag information when decrypting ciphertext C.

CCM finalization

14. Select the CCM final phase, by writing GCMPH[1:0] with 0x3. Do not change the other configurations written during the CCM initialization.
15. Wait until CCF flag in the SAES_ISR register is set.
16. Get the CCM authentication tag, by reading the SAES_DOUTR register four times.
17. Clear the CCF flag, by setting the CCF bit of the SAES_ICR register.
18. Disable the SAES peripheral, by clearing EN. If it is an authenticated decryption, compare the generated tag with the expected tag passed with the message. Mask the authentication tag output with tag length to obtain a valid tag.

Note: In the final phase, swapping is applied to tag data read from SAES_DOUTR register.

When transiting from the header or the payload phase to the final phase, the SAES peripheral must not be disabled, otherwise the result is wrong.

Suspend and resume operations in CCM mode

The suspend and resume operations are only supported when SAES is used in CPU mode, not in DMA mode.

To suspend the processing of a message in header or payload phase, proceed as follows:

1. Wait until the CCF flag of the SAES_ISR register is set (computation completed).
2. In the payload phase, read four times the SAES_DOUTR register to save the last-processed block.
3. Clear the CCF flag in the SAES_ISR register, by setting the CCF bit of the SAES_ICR register.
4. Save the SAES_SUSPRx registers in the memory.
5. Save the IV registers as they are altered during the data processing.
6. Disable the SAES peripheral, by clearing EN.
7. Save the current SAES_CR configuration in the memory. Key registers do not need to be saved as the original key value is known by the application.

To resume the processing of a message, proceed as follows:

1. Disable the SAES peripheral, by clearing EN.
2. Write the suspend register values, previously saved in the memory, back into their corresponding SAES_SUSPRx registers.
3. Restore SAES_IVRx registers using the saved configuration.
4. Restore the initial setting values in the SAES_CR and SAES_KEYRx registers. For KEYSEL[2:0] selecting a key source different from the key registers, refer to *Section 48.4.17: SAES key registers* for details.
5. Enable the SAES peripheral, by setting EN.

48.4.14 SAES operation with wrapped keys

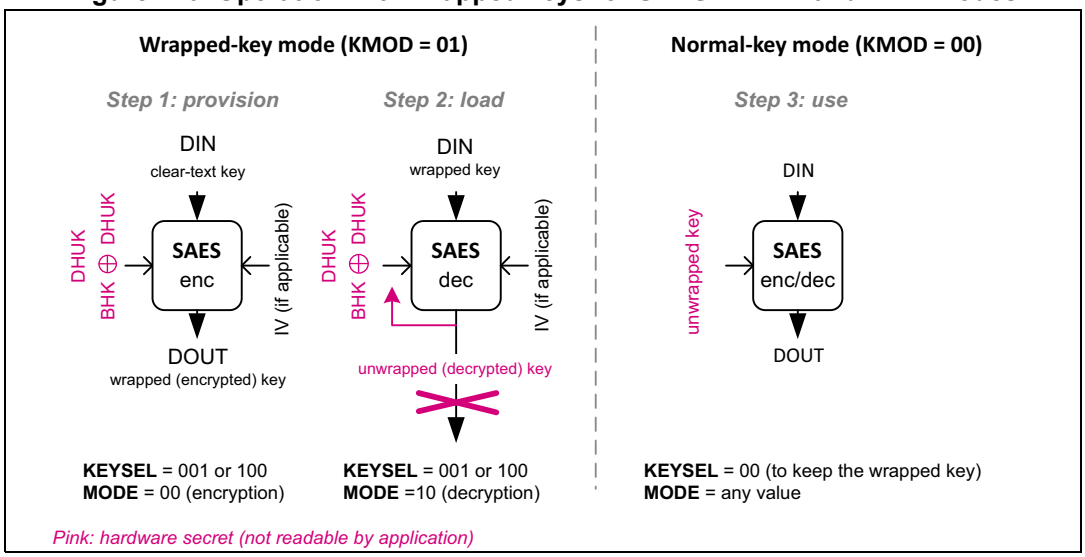
SAES peripheral can wrap (encrypt) and unwrap (decrypt) application keys using hardware-secret key DHUK, XOR-ed or not with application key BHK. With this feature, AES keys can be made usable by application software without being exposed in clear-text (unencrypted).

Wrapped key sequences are too small to be suspended/resumed. SAES cannot unwrap a key using an unwrapped key.

Operation with wrapped keys for SAES in ECB and CBC modes

Figure 470 summarizes how to wrap or unwrap keys for SAES in ECB and CBC modes. To protect the wrapped key, select DHUK by writing KEYSEL[2:0] with 0x1 or 0x4. Alternatively, select BHK by writing KEYSEL[2:0] with 0x2 if the corresponding registers are read/write-locked in the TAMP peripheral.

Figure 470. Operation with wrapped keys for SAES in ECB and CBC modes



Note: DHUK value depends on privilege, security, KMOD[1:0], KEYSEL[2:0], CHMOD[2:0], and KEYSIZE.

Key wrapping for SAES

The recommended sequence to wrap (that is, encrypt) a key is as follows:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register as follow:
   - Select ECB or CBC chaining mode (write CHMOD[2:0] with 0x0 or 0x1) in *encryption* mode (MODE[1:0] at 0x0)
   - Select 32-bit data type (DATATYPE[1:0] at 0x0)
   - Configure the key size with KEYSIZE. This information is used both for the encryption key and for the key to be encrypt.
   - Select wrapped key mode by writing KMOD[1:0] with 0x1
4. Write the initialization vector in SAES_IVRx registers if CBC mode has been selected in previous step.
5. Select the DHUK key source by writing KEYSEL[2:0] with 0x1 or 0x4. Optionally, select the BHK, by writing KEYSEL[2:0] with 0x2. Refer to *Section 48.4.17* for details on the use of KEYSEL[2:0].
6. Wait until KEYVALID is set (DHUK loading completed).
7. Enable the SAES peripheral, by setting EN.
8. Write the SAES_DINR register four times to input the key to encrypt (MSB first, see *Table 409*).
9. Wait until CCF flag is set in the SAES_ISR register.
10. Get the encrypted key (MSB first) by reading the SAES_DOUTR register four times. Then clear the CCF flag, by setting the CCF bit in SAES_ICR register.
11. Repeat steps *8.* to *10.* if KEYSIZE is set.
12. Disable the SAES peripheral, by clearing EN.

Note: Encryption in Wrapped-key mode is only supported when ECB or CBC is selected.

Key unwrapping for SAES

The recommended sequence to unwrap (or decrypt) a wrapped (encrypted) key using ECB/CBC is as follows:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register as follow:
   - Select the chaining mode used during the wrapping process (CHMOD[2:0] at 0x0 or 0x1) in *key derivation* mode (MODE[1:0] at 0x1)
   - Select 32-bit data type (DATATYPE[1:0] at 0x0)
   - Configure the key size used during the wrapping process, with KEYSIZE. This information is used both for the decryption key and for the key to decrypt.
   - Select wrapped key mode, by writing KMOD[1:0] with 0x1.
4. With KEYSEL[2:0], select the same key source as when the key was wrapped/encrypted.
5. Wait until KEYVALID is set (the key loading completed).
6. Set EN bit in SAES_CR to enable the peripheral
7. Wait until CCF flag is set in the SAES_ISR register.
8. Clear the CCF flag, by setting the CCF bit in SAES_ICR register. The decryption key is available in the AES core, and SAES is disabled automatically.
9. Select the *decryption* mode (MODE[1:0] at 0x2). Other parameters are unchanged.
10. Write the initialization vector in SAES_IVRx registers if CBC mode has been selected in previous step.
11. Enable the SAES peripheral, by setting EN.
12. Write the SAES_DINR register four times to input the key to decrypt (MSB first, see *Table 409*).
13. Wait until CCF flag is set in the SAES_ISR register. Then clear the CCF flag by setting the CCF bit in SAES_ICR register. Reading SAES_DOUTR returns zero and triggers a read error (RDERRF).
14. Repeat steps *12.* and *13.* if KEYSIZE is set.
15. Disable the SAES peripheral, by clearing EN.

At the end of this sequence, the decrypted wrapped key is immediately usable by the application for any AES operation (normal key mode, that is, with KMOD[1:0] = 0x0).

Decrypted wrapped key can be shared with an application running in a different security context (different security attribute) if KEYPROT bit was cleared during step 3.

Note: When KMOD[1:0] = 0x1 (wrapped key) and MODE[1:0] = 0x2 (decryption) a read access to SAES_DOUTR register triggers a read error (RDERRF).

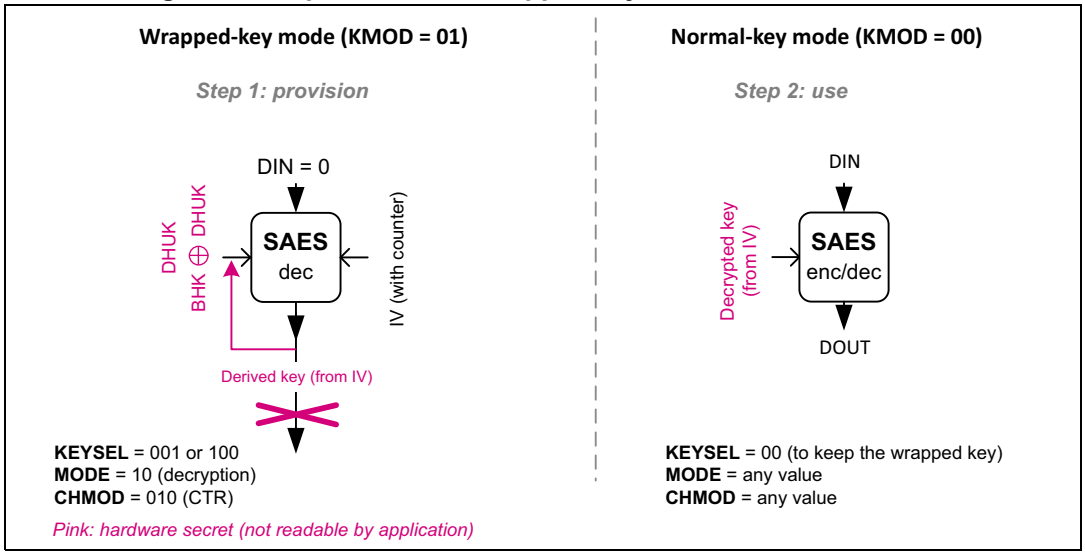
When KEYSEL[2:0] is at 0x1 (DHUK) or 0x4 (DHUK XOR BHK), the application software must use the same privilege, security, KMOD[1:0], CHMOD[2:0] and KEYSIZE context for encryption and decryption. Otherwise, the result is incorrect.

Operation with wrapped keys for SAES in CTR mode

Figure 471 summarizes how to unwrap keys for SAES in CTR mode. To protect the derived key, select DHUK by writing KEYSEL[2:0] with 0x1 or 0x4. Alternatively, select BHK by

writing KEYSEL[2:0] with 0x2 if the corresponding registers are read/write-locked in the TAMP peripheral.

Figure 471. Operation with wrapped keys for SAES in CTR mode



Note: DHUK value depends on privilege, security, KMOD[1:0], KEYSEL[2:0], CHMOD[2:0], and KEYSIZE.

The recommended sequence for SAES wrapped key mode using CTR is as follows:

1. Disable the SAES peripheral, by clearing EN.
2. Wait until BUSY is cleared (no RNG random number fetch in progress).
3. Initialize the SAES_CR register as follow:
   - Select the CTR chaining mode (CHMOD[2:0] at 0x2) in decryption mode (MODE[1:0] at 0x2). Other MODE[1:0] values are not supported.
   - Select 32-bit data type (DATATYPE[1:0] at 0x0)
   - Configure the key size with KEYSIZE. It is used for encryption key and for the key to share.
   - Select wrapped key mode, by writing KMOD[1:0] with 0x1.
4. Write the initialization vector in SAES_IVRx registers, keeping the two least significant bits of SAES_IVR0 at zero.
5. Select the DHUK key source by writing KEYSEL[2:0] with 0x1 or 0x4. Optionally, select the BHK, by writing KEYSEL[2:0] with 0x2. Refer to *Section 48.4.17* for details on the use of KEYSEL[2:0].
6. Wait until KEYVALID is set (the key loading completed).
7. Enable the SAES peripheral, by setting EN.
8. Wait until CCF flag is set in the SAES_ISR register.
9. Clear the CCF flag, by setting the CCF bit in SAES_ICR register. The derived hardware secret key is available in SAES_KEYRx registers.
10. Repeat steps *8.* and *9.* if KEYSIZE is set.
11. Disable the SAES peripheral, by clearing EN.

At the end of this sequence, the hardware secret key derived from the public data in the SAES_IVRx registers is then immediately usable by the application for any AES operation (normal key mode, that is, with KMOD[1:0] = 0x0).

Note: The configuration KMOD[1:0] at 0x1 (wrapped key), CHMOD[2:0] at 0x2 (CTR chaining), and MODE at 0x0 (encryption) disables the peripheral, by automatically clearing the EN bit of the SAES_*CR register.*

48.4.15 SAES operation with shared keys

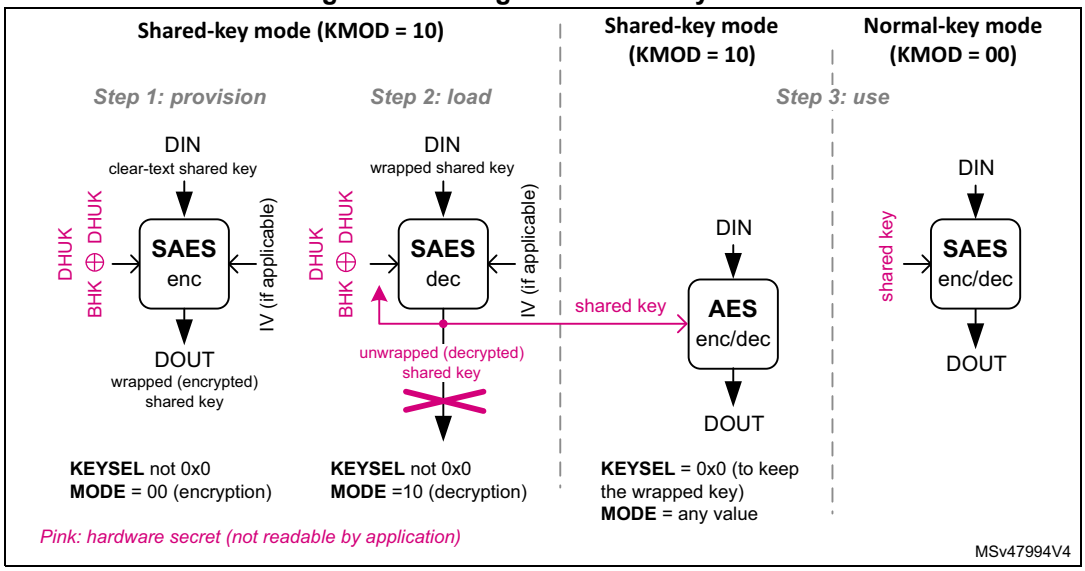
SAES peripheral can share application keys wrapped with hardware-secret key DHUK, XOR-ed or not with application key BHK. With this feature, the application software can make the AES keys available to the CRYP peripheral, without exposing them in clear-text (unencrypted).

Shared key sequences are too small to be suspended/resumed. SAES cannot unwrap a shared key using an unwrapped key.

Note: When a key stored in SAES is shared with CRYP, the protection given by KEYPROT bit is lost. The protection is detailed in Section 48.4.17: SAES key registers.

Figure 472 summarizes how to wrap or unwrap keys to share with *CRYP* peripheral. To protect the shared key, DHUK must be selected, by writing KEYSEL[2:0] with 0x1 or 0x4. Alternatively, select BHK by writing KEYSEL[2:0] with 0x2 if the corresponding registers are read/write-locked in the TAMP peripheral.

Figure 472. Usage of Shared-key mode



Note: DHUK value depends on privilege, security, KMOD[1:0], KSHAREID, KEYSEL[2:0], CHMOD[2:0], and KEYSIZE.

In the step 3, AES represents the CRYP peripheral.

Key wrapping for CRYP peripheral

Before SAES can share a key with the CRYP peripheral, the key must be encrypted (wrapped) once. The encryption sequence of a shared key is the same as for a wrapped

key, with KMOD[1:0] at 0x2 (shared key) and KSHAREID[1:0] kept at 0x0 in the step 3 in *Figure 472*. See *Key wrapping for SAES* for details.

Note: Encryption in Shared-key mode is only supported when ECB or CBC is selected.

Key unwrapping for CRYP peripheral (shared key)

Each time SAES needs to share a key with the CRYP peripheral, shared encrypted key must be decrypted (unwrapped) in SAES, then loaded by CRYP. The overall sequence is described next.

Sequence in the SAES peripheral

The decryption sequence of a shared key is the same as for a wrapped key, with KMOD[1:0] at 0x2 (shared key) and KSHAREID[1:0] kept at 0x0 in the step 3 in *Figure 472*. See *Key unwrapping for SAES* for details.

In shared key mode when decryption mode is selected (MODE[1:0] at 0x2), a read access to the SAES_DOUTR register triggers a read error (RDERRF).

Note: Instead of being shared, a decrypted shared key can be used directly in SAES as the KEYSEL[2:0] bitfield is automatically cleared. In this case, KMOD[1:0] must be written with 0x0 (normal key mode).

Sequence in the CRYP peripheral

Once the shared key is decrypted in SAES key registers, it can be shared with the CRYP peripheral, while SAES peripheral remains in key sharing state, that is, with KMOD[1:0] at 0x2 and KEYVALID set. The sequence in the CRYP key share target peripheral is described in *AES key sharing with secure AES co-processor* of the corresponding section in this document. It can be run multiple times (for example, to manage a suspend/resume situation) as long as SAES is unused and duly remains in key sharing state.

Note: When KMOD[1:0] is at 0x2 and BUSY set in the CRYP peripheral, and KEYSIZE value of CRYP and SAES differs, the key sharing fails and the KEIF flag is raised in both peripherals.

When KEYSEL[2:0] is at 0x1 (DHUK) or 0x4 (DHUK XOR BHK), the application software must use the same privilege, security, KMOD[1:0] / KSHAREID[1:0], CHMOD[2:0], and KEYSIZE context for encryption and decryption. Otherwise, the result is incorrect.

48.4.16 SAES data registers and data swapping

Data input and output

A 16-byte data block enters the SAES peripheral with four successive 32-bit word writes into the SAES_DINR register (bitfield DIN[31:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

A 16-byte data block is retrieved from the SAES peripheral with four successive 32-bit word reads of the SAES_DOUTR register (bitfield DOUT[31:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

The four 32-bit words of a 16-byte data block must be stored in the memory consecutively and in big-endian order, that is, with the most significant word on the lowest address. See *Table 408* “no swapping” option for details.

Data swapping

The SAES peripheral can be configured to perform a bit-, a byte-, a half-word-, or no swapping on the input data word in the SAES_DINR register, before loading it to the AES processing core, and on the data output from the AES processing core, before sending it to the SAES_DOUTR register. The choice depends on the type of data. For example, a byte swapping is used for an ASCII text stream.

The data swap type is selected through DATATYPE[1:0]. The selection applies to both SAES input and output.

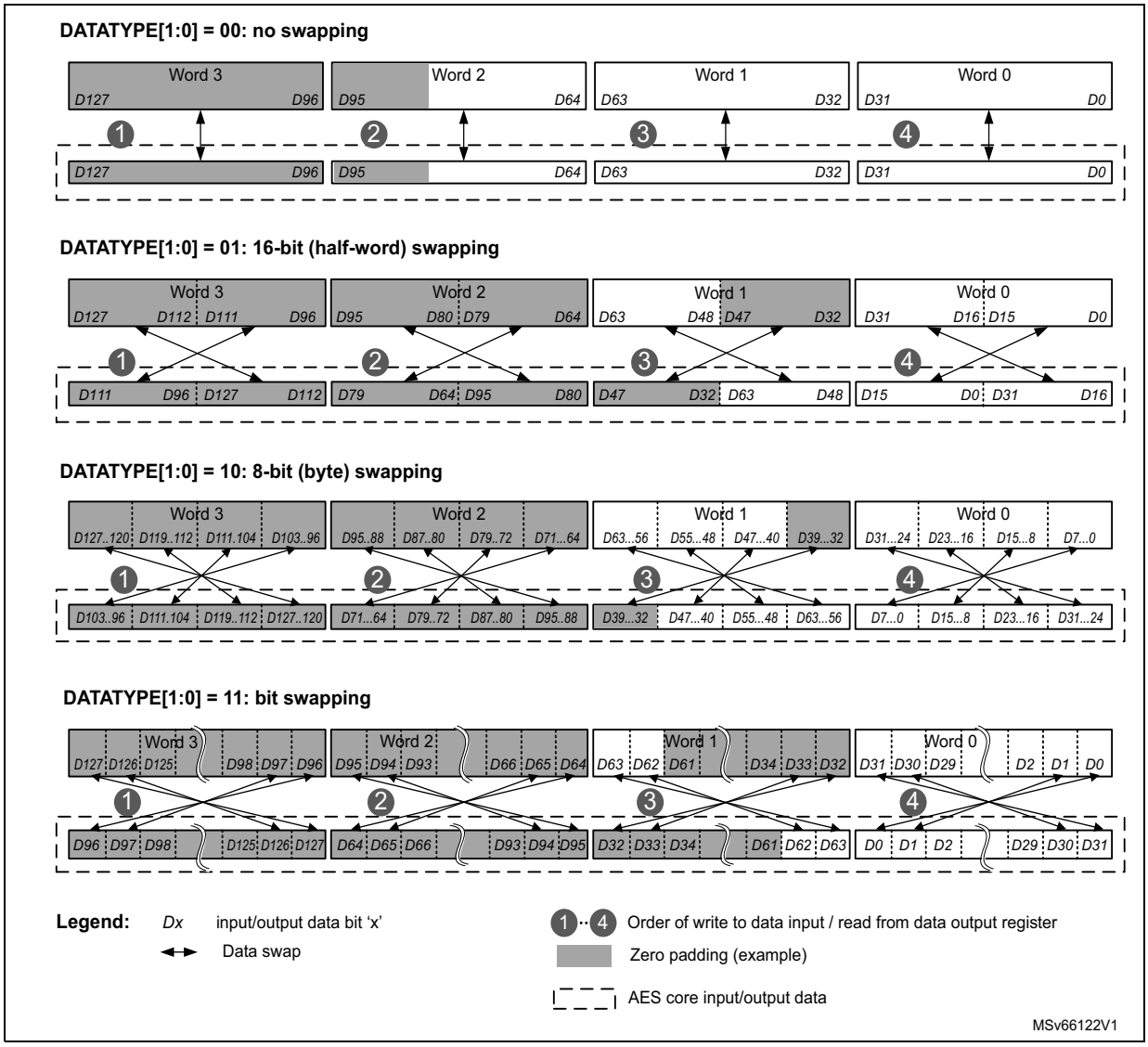
Note: The data in SAES key registers (SAES_KEYRx) and initialization vector registers (SAES_IVRx) are not sensitive to the swap mode selection.

The SAES data swapping feature is summarized in *Table 408* and *Figure 473*.

Table 408. AES data swapping example

DATATYPE[1:0]	Swapping performed	Data block
		System memory data (big-endian)
0x0	No swapping	Block[127..64]: 0x04EEF672 2E04CE96 Block[63..0]: 0x4E6F7720 69732074
		Address @, word[127..96]: 0x04EEF672 Address @ + 0x4, word[95..64]: 0x2E04CE96 Address @ + 0x8, word[63..32]: 0x4E6F7720 Address @ + 0xC, word[31..0]: 0x69732074
0x1	Half-word (16-bit) swapping	Block[63..0]: 0x**4E6F** 7720 **6973** 2074
		Address @, word[63..32]: 0x7720 **4E6F** Address @ + 0x4, word[31..0]: 0x2074 **6973**
0x2	Byte (8-bit) swapping	Block[63..0]: 0x**4E** 6F **77** 20 **69** 73 **20** 74
		Address @, word[63..32]: 0x20**77** 6F**4E** Address @ + 0x4, word[31..0]: 0x74**20** 73**69**
0x3	Bit swapping	Block[63..32]: 0x4E6F7720 0100 1110 0110 1111 0111 0111 0010 0000 Block[31..0]: 0x69732074 0110 1001 0111 0011 0010 0000 0111 0100
		Address @, word[63..32]: 0x04EE F672 0000 0100 1110 1110 1111 0110 0111 0010 Address @ + 0x4, word[31..0]: 0x2E04 CE96 0010 1110 0000 0100 1100 1110 1001 0110

Figure 473. 128-bit block construction according to the data type



Data padding

Figure 473 also gives an example of memory data block padding with zeros such that the zeroed bits after the data swap form a contiguous zone at the MSB end of the AES core input buffer. The example shows the padding of an input data block containing:

- 84 message bits, with DATATYPE[1:0] = 0x0
- 48 message bits, with DATATYPE[1:0] = 0x1
- 56 message bits, with DATATYPE[1:0] = 0x2
- 34 message bits, with DATATYPE[1:0] = 0x3

48.4.17 SAES key registers

The eight SAES_KEYRx write-only registers store the encryption or decryption key information, as shown on *Table 409*. Reads are not allowed for security reason.

Note: In memory and in SAES *key registers, keys are stored in little-endian format, with most significant byte on the highest address.*

Table 409. Key endianness in SAES_KEYRx registers (128/256-bit keys)

SAES_KEYR7 [31:0]	SAES_KEYR6 [31:0]	SAES_KEYR5 [31:0]	SAES_KEYR4 [31:0]	SAES_KEYR3 [31:0]	SAES_KEYR2 [31:0]	SAES_KEYR1 [31:0]	SAES_KEYR0 [31:0]
-	-	-	-	KEY[127:96]	KEY[95:64]	KEY[63:32]	KEY[31:0]
KEY[255:224]	KEY[223:192]	KEY[191:160]	KEY[159:128]	KEY[127:96]	KEY[95:64]	KEY[63:32]	KEY[31:0]
TAMP_BKP7R [31:0]	TAMP_BKP6R [31:0]	TAMP_BKP5R [31:0]	TAMP_BKP4R [31:0]	TAMP_BKP3R [31:0]	TAMP_BKP2R [31:0]	TAMP_BKP1R [31:0]	TAMP_BKP0R [31:0]

The key registers are not affected by the data swapping feature controlled by the DATATYPE[1:0] bitfield.

Write operations to the SAES_KEYRx registers are ignored when SAES peripheral is enabled (EN bit set) and KEYSEL[2:0] is different from zero. The application must check this before modifying key registers.

The entire key must be written before starting an AES computation.

In normal key mode (KMOD[1:0] at 0x0), with KEYSEL[2:0] at 0x0, the key registers must always be written in either ascending or descending order. The write sequence becomes:

- SAES_KEYRx (x = 0 to 3 or x=3 to 0) for KEYSIZE cleared
- SAES_KEYRx (x = 0 to 7 or x=7 to 0) for KEYSIZE set

Note: KEYSIZE must be written before the key.

As soon as the first key register is written, the KEYVALID flag is cleared. Once the key registers writing sequence is completed, KEYVALID is set and EN becomes writable. If an error occurs, KEYVALID is cleared and KEIF set (see *Section 48.4.19*).

Key selection

With KEYSEL[2:0] at 0x0, the application must write the key in the SAES_KEYRx registers.

With KEYSEL[2:0] at 0x1, a derived hardware unique key (DHUK), computed inside SAES from a nonvolatile and secret root hardware unique key, is loaded directly into key registers, based on KEYSIZE information. Thanks to the key derivation function, the computed DHUK depends on the SAES usage (privilege, security, KMOD[1:0] / KSHAREID[1:0], CHMOD[2:0], CPU, and KEYSIZE context).

With KEYSEL[2:0] at 0x2, the boot hardware key (BHK), stored in tamper-resistant secure backup registers, is entirely transferred into key registers upon a secure application performing a single read of all TAMP_BKPxR registers (x = 0 to 3 for KEYSIZE cleared, x = 0 to 7 for KEYSIZE set) in ascending order. Refer to *Table 409*.

With KEYSEL[2:0] at 0x4, the XOR combination of DHUK and BHK is entirely transferred into key registers upon a secure application performing a single read of all TAMP_BKPxR

registers (x = 0 to 3 for KEYSIZE cleared, x = 0 to 7 for KEYSIZE set) in ascending order. Refer to *Table 409*.

Repeated writing of KEYSEL[2:0] with the same non-zero value only triggers the loading of DHUK or BHK if KEYVALID is cleared. The recommended method to clear KEYVALID is to set IPRST. Such method is required for example when switching from ECB decryption to ECB encryption, selecting the same BHK (KEYSEL[2:0] at 0x2).

For all KEYSEL[2:0] values, initiating the key-loading sequence sets the BUSY flag and clears the KEYVALID flag. Once the amount of bits defined by KEYSIZE is transfered to the SAES_KEYRx registers, BUSY is cleared, KEYVALID set and the EN bit becomes writable. If an error occurs, BUSY and KEYVALID are cleared and KEIF set (see *Section 48.4.19*).

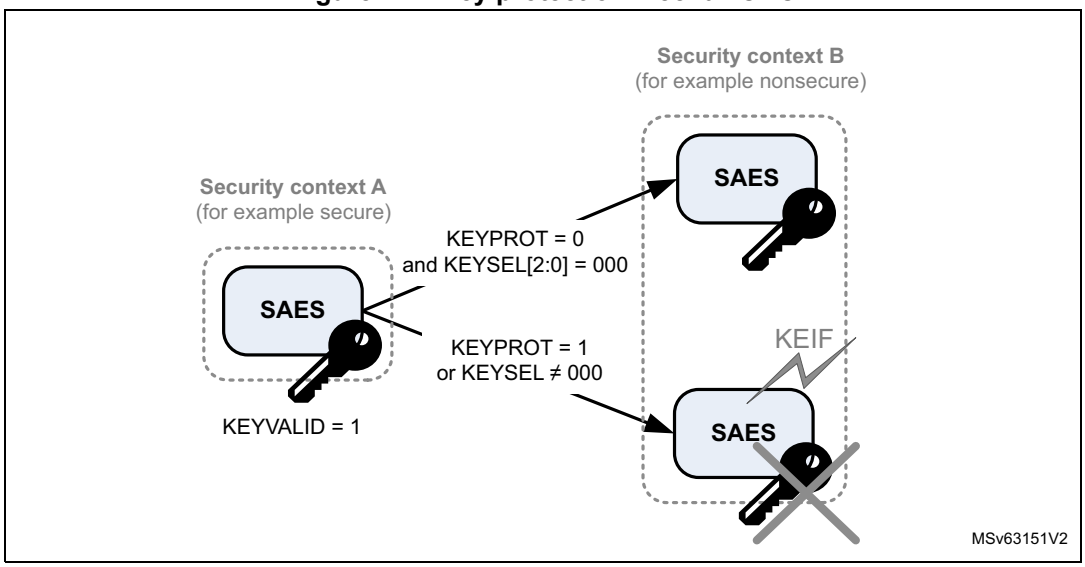
Note: DHUK, BHK and their XOR combination are not readable by any software (even secure).

Key protection

As depicted in *Figure 474*, when an application sets the KEYPROT bit before writing a key in SAES_KEYRx, any application executing in a different security context (that is, different security attribute) triggers a KEIF error flag upon access to any SAES register when KEYVALID is set.

Note: KEYSEL[2:0] values different from zero (normal key) automatically protect the key registers.

Figure 474. Key protection mechanisms



48.4.18 SAES initialization vector registers

The four SAES_IVRx registers store the initialization vector (IV) information, as shown in *Table 410*. They can only be written if the SAES peripheral is disabled (EN cleared).

Note: In memory and in SAES IV registers, initialization vectors are stored in little-endian format, with most significant byte on the highest address.

Table 410. IVI bitfield spread over SAES_IVRx registers

SAES_IVR3[31:0]	SAES_IVR2[31:0]	SAES_IVR1[31:0]	SAES_IVR0[31:0]
IVI[127:96]	IVI[95:64]	IVI[63:32]	IVI[31:0]

Initialization vector information depends on the chaining mode selected. When used, SAES_IVRx registers are updated upon each AES computation cycle (useful for managing suspend mode).

The initialization vector registers are not affected by the data swapping feature controlled through DATATYPE[1:0].

48.4.19 SAES error management

The SAES peripheral manages the errors described in this section.

Read error flag (RDERRF)

Unexpected read attempt of the SAES_DOUTR register returns zero, setting the RDERRF flag and the RWEIF flag. RDERRF is triggered during the computation phase or during the input phase.

Note: Unless otherwise indicated, SAES is not disabled when RDERRF rises and it continues processing.

An interrupt is generated if the RWEIE bit is set. For more details, refer to *Section 48.5: SAES interrupts*.

The RDERRF and RWEIF flags are cleared by setting the RWEIF bit of the SAES_ICR register.

Write error flag (WRERRF)

Unexpected write attempt of the SAES_DINR register is ignored, setting the WRERRF and the RWEIF flags. WRERRF is triggered during the computation phase or during the output phase.

Note: Unless otherwise indicated, SAES is not disabled when WRERRF rises and it continues processing.

An interrupt is generated if the RWEIE bit is set. For more details, refer to *Section 48.5: SAES interrupts*.

The WRERRF and RWEIF flags are cleared by setting the RWEIF bit of the SAES_ICR register.

Key error interrupt flag (KEIF)

There are multiple sources of errors that set the KEIF flag of the SAES_ISR register and clear the KEYVALID bit of the SAES_SR register:

- **Key protection error:** while KEYVALID is set, then if KEYPROT is set or KEYSEL[2:0] is different from zero, this error is triggered when an application executing in a security context different from the one used to load the key (that is, different security attribute) accesses SAES.

- **Key writing sequence error:** triggered upon detecting an incorrect sequence of writing key registers. See *Section 48.4.17: SAES key registers* for details.
- **Key sharing size mismatch error:** triggered when KMOD[1:0] is at 0x2 and KEYSIZE in CRYP peripheral does not match KEYSIZE in SAES peripheral.
- **Key sharing error:** triggered upon failing transfer of SAES shared key to CRYP peripheral. See *Section 48.4.15: SAES operation with shared keys* for details.
- **Hardware secret key loading error:** triggered upon failing load of DHUK or BHK into SAES. KEYSEL[2:0] at 0x1 (DHUK), 0x2 (BHK) or 0x4 (DHUK XOR BHK) is not functional.

The KEIF flag is cleared with corresponding bit of the SAES_ICR register. An interrupt is generated if the KEIE bit of the SAES_IER register is set. For more details, refer to *Section 48.5: SAES interrupts*.

Upon a key selection error, clearing the KEIF flag automatically restarts the key selection process. Persisting problems (for example, RHUK load failing) may require a power-on reset.

Upon a key sharing error, reset both CRYP and SAES peripherals through the IPRST bit of their corresponding control register, then restart the key sharing sequence.

Note: For any key error, clear KEIF flag prior to disabling and re-configuring SAES.

RNG error interrupt flag (RNGEIF)

SAES fetches random numbers from the RNG peripheral automatically after an IP reset triggered in the RCC. SAES cannot be used when RNGEIF is set.

An error detected while fetching a random number from RNG peripheral (due to, for example, bad entropy) sets the RNGEIF flag of the SAES_ISR register. The flag is cleared by setting the corresponding bit of the SAES_ICR register. An interrupt is generated if the RNGEIE bit of the SAES_IER register is set. For more details, refer to *Section 48.5: SAES interrupts*.

Upon an RNG error:

- Make sure that the RNG peripheral is properly configured.
- Verify that the RNG peripheral AHB clock is enabled and no noise source (or seed) error is pending in this peripheral.
- Clear RNGEIF or reset the peripheral by setting IPRST. The clearance of the BUSY flag then indicates the completion of the random number fetch from RNG.

Note: To avoid RNGEIF errors, it is recommended to activate the RNG AHB clock each time SAES AHB clock is activated.

Managing tamper errors

An unexpected error triggers an SAES internal tamper event in the TAMP peripheral, and stops any SAES co-processor processing.

To resume normal operation, reset the SAES peripheral through RCC or global reset.

48.5 SAES interrupts

There are multiple individual maskable interrupt sources generated by the SAES peripheral to signal the following events:

- computation completed (CCF)
- read error (RDERRF)
- write error (WRERRF)
- key error (KEIF)
- RNG error (RNGEIF)

See *Section 48.4.19: SAES error management* for details on SAES errors.

These sources are combined into a common interrupt signal from the SAES peripheral that connects to the Cortex® CPU interrupt controller. Application can enable or disable SAES interrupt sources individually by setting/clearing the corresponding enable bit of the SAES_IER register.

The status of the individual maskable interrupt sources can be read from the SAES_ISR register. They are cleared by setting the corresponding bit of the SAES_ICR register.

Table 411 gives a summary of the available features.

Table 411. SAES interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable bit	Interrupt clear method
SAES	computation completed flag	CCF	CCFIE	set CCF(1)
	read error flag	RDERRF(2)	RWEIE	set RWEIF(1)
	write error flag	WRERRF(2)		
	key error flag	KEIF	KEIE	set KEIF(1)
	RNG error flag	RNGEIF	RNGEIE	set RNGEIF(1)

1. Bit of the SAES_ICR register.
2. Flag of the SAES_SR register, mirrored by the flag RWEIF of the SAES_ISR register.

48.6 SAES DMA requests

The SAES peripheral provides an interface to connect to the DMA (direct memory access) controller. The DMA operation is controlled through the DMAINEN and DMAOUTEN bits of the SAES_CR register. When key derivation is selected (MODE[1:0] is at 0x1), setting those bits has no effect.

SAES only supports single DMA requests.

Suspend and resume operations are not supported in DMA mode.

Detailed usage of DMA with SAES can be found in *Appending data using DMA* subsection of *Section 48.4.5: SAES encryption or decryption typical usage*.

Data input using DMA

Setting DMAINEN enables DMA writing into SAES. SAES then initiates, during the input phase, a set of single DMA requests for each 16-byte data block to write to the SAES_DINR register (quadruple 32-bit word, MSB first).

Note: According to the algorithm and the mode selected, special padding / ciphertext stealing might be required (see Section 48.4.7).

Data output using DMA

Setting DMAOUTEN enables DMA reading from SAES. SAES then initiates, during the output phase, a set of single DMA requests for each 16-byte data block to read from the SAES_DOUTR register (quadruple 32-bit word, MSB first).

After the output phase, at the end of processing of a 16-byte data block, SAES switches automatically to a new input phase for the next data block, if any.

In DMA mode, the CCF flag has no use because the reading of the SAES_DOUTR register is managed by DMA automatically at the end of the computation phase. The CCF flag must only be cleared when transiting back to managing the data transfers by software.

Note: According to the message size, extra bytes might need to be discarded by application in the last block.

Stopping DMA transfers

All DMA request signals are de-asserted when SAES is disabled (EN cleared) or the DMA enable bit (DMAINEN for input data, DMAOUTEN for output data) is cleared.

48.7 SAES processing latency

The following tables provide the 16-byte data block processing latency per operating mode.

Table 412. Processing latency for ECB, CBC and CTR

Key size	Mode of operation	Chaining algorithm	Clock cycles(1)
128-bit	Encryption or decryption(2)	ECB, CBC, CTR	**480**
	Key preparation	-	**145**
256-bit	Encryption or decryption(2)	ECB, CBC, CTR	**680**
	Key preparation	-	**230**

1. SAES kernel clock
2. Excluding key preparation time (ECB and CBC only).

Table 413. Processing latency for GCM and CCM (in SAES kernel clock cycles)

Key size	Mode of operation	Chaining algorithm	Initialization phase	Header phase(1)	Payload phase(1)	Final phase(1)
128-bit	Mode 1: Encryption/ Mode 3: Decryption	GCM	490	72(2)	480(3)	490
		CCM	490	490	800	490
256-bit	Mode 1: Encryption/ Mode 3: Decryption	GCM	650	72(2)	690(3)	650
		CCM	650	680	1350	650

1. Data insertion can include wait states forced by SAES on the AHB bus (maximum 3 cycles, typical 1 cycle).
2. SAES AHB clock cycles instead of kernel clock cycle (Galois multiplier only).
3. As a worst case in encryption mode, add extra 72 AHB clock cycles for the last block computation.

48.8 SAES registers

The registers are accessible through 32-bit word single accesses only. Other access types generate an AHB error, and other than 32-bit writes may corrupt the register content.

48.8.1 SAES control register (SAES_CR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IPRST	KEYSEL[2:0]			KSHAREID[1:0]		KMOD[1:0]		NPBLB[3:0]				KEYPROT	KEYSIZE	Res.	CHMOD[2]
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	GCMPH[1:0]		DMAOUTEN	DMAINEN	Res.	Res.	Res.	Res.	CHMOD[1:0]		MODE[1:0]		DATATYPE[1:0]		EN
	rw	rw	rw	rw					rw	rw	rw	rw	rw	rw	rw

Bit 31 **IPRST**: SAES peripheral software reset

Setting the bit resets the SAES peripheral, putting all registers to their default values, except the IPRST bit itself. Hence, any key-relative data are lost. For this reason, it is recommended to set the bit before handing over the SAES to a less secure application.

The bit must be kept low while writing any configuration registers.

Bits 30:28 **KEYSEL[2:0]**: Key selection

The bitfield defines the source of the key information to use in the AES cryptographic core.

0x0: Software key, loaded in key registers SAES_KEYx

0x1: Derived hardware unique key (DHUK)

0x2: Boot hardware key (BHK)

0x4: XOR of DHUK and BHK

Others: Reserved (if used, unfreeze SAES with IPRST)

When KEYSEL[2:0] is different from zero, selected key value is available in key registers when BUSY bit is cleared and KEYVALID is set in the SAES_SR register. Otherwise, the key error flag KEIF is set. Repeated writing of KEYSEL[2:0] with the same non-zero value only triggers the loading of DHUK or BHK when KEYVALID is cleared.

When the application software changes the key selection by writing the KEYSEL[2:0] bitfield, the key registers are immediately erased and the KEYVALID flag cleared.

At the end of the decryption process, if KMOD[1:0] is other than zero, KEYSEL[2:0] is cleared.

With the bitfield value other than zero and KEYVALID set, the application cannot transfer the ownership of SAES with a loaded key to an application running in another security context (such as secure, nonsecure). More specifically, when security of an access to any register does not match the information recorded by SAES, the KEIF flag is set.

Attempts to write the bitfield are ignored when the BUSY flag of SAES_SR register is set, as well as when the EN bit of the SAES_CR register is set before the write access and it is not cleared by that write access.

Bits 27:26 **KSHAREID[1:0]**: Key share identification

This bitfield defines, at the end of a decryption process with KMOD[1:0] at 0x2 (shared key), which target can read the SAES key registers using a dedicated hardware bus.

0x0: CRYP peripheral

Others: Reserved

Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bits 25:24 **KMOD[1:0]**: Key mode selection

The bitfield defines how the SAES key can be used by the application. KEYSIZE must be correctly initialized when setting KMOD[1:0] different from zero.

0x0: Normal key mode. Key registers are freely usable and no specific use or protection applies to SAES_DINR and SAES_DOUTR registers.

0x1: Wrapped key for SAES mode. Key loaded in key registers can only be used to encrypt or decrypt AES keys. Hence, when a decryption is selected, read-as-zero SAES_DOUTR register is automatically loaded into SAES key registers after a successful decryption process.

0x2: Shared key mode. After a successful decryption process (unwrapping), SAES key registers are shared with the peripheral described in KSHAREID[1:0] bitfield. This sharing is valid only while KMOD[1:0] at 0x2 and KEYVALID=1. When a decryption is selected, read-as-zero SAES_DOUTR register is automatically loaded into SAES key registers after a successful decryption process.

Others: Reserved

With KMOD[1:0] other than zero, any attempt to configure the SAES peripheral for use by an application belonging to a different security context (such as secure or nonsecure) results in automatic key erasure and setting of the KEIF flag.

Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bits 23:20 **NPBLB[3:0]:** Number of padding bytes in last block

This padding information must be filled by software before processing the last block of GCM payload encryption or CCM payload decryption, otherwise authentication tag computation is incorrect.

0x0: All bytes are valid (no padding)

0x1: Padding for the last LSB byte

...

0xF: Padding for the 15 LSB bytes of last block.

Bit 19 **KEYPROT**: Key protection

When set, hardware-based key protection is enabled.

0: When KEYVALID is set and KEYSEL[2:0] = 0 application can transfer the ownership of the SAES, with its loaded key, to an application running in another security context (such as nonsecure, secure).

1: When KEYVALID is set, key error flag (KEIF) is set when an access to any registers is detected, this access having a security context (for example, secure, nonsecure) that does not match the one of the application that loaded the key.

Attempts to write the bit are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bit 18 **KEYSIZE:** Key size selection

This bitfield defines the key length in bits of the key used by SAES.

0: 128-bit

1: 256-bit

When KMOD[1:0] is at 0x1 or 0x2, KEYSIZE also defines the length of the key to encrypt or decrypt.

Attempts to write the bit are ignored when BUSY is set, as well as when the EN is set before the write access and it is not cleared by that write access.

Bit 17 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bits 14:13 **GCMPH[1:0]:** GCM or CCM phase selection

This bitfield selects the phase, applicable only with GCM, GMAC or CCM chaining modes.
0x0: Initialization phase
0x1: Header phase
0x2: Payload phase
0x3: Final phase

Bit 12 **DMAOUTEN**: DMA output enable

This bit enables automatic generation of DMA requests during the data phase, for outgoing data transfers from SAES via DMA.
0: Disable
1: Enable
Setting this bit is ignored when MODE[1:0] is at 0x1 (key derivation).

Bit 11 **DMAINEN**: DMA input enable

This bit enables automatic generation of DMA requests during the data phase, for incoming data transfers to SAES via DMA.
0: Disable
1: Enable
Setting this bit is ignored when MODE[1:0] is at 0x1 (key derivation).

Bits 10:7 Reserved, must be kept at reset value.

Bits 16, 6:5 **CHMOD[2:0]**: Chaining mode

This bitfield selects the AES chaining mode:
0x0: Electronic codebook (ECB)
0x1: Cipher-block chaining (CBC)
0x2: Counter mode (CTR)
0x3: Galois counter mode (GCM) and Galois message authentication code (GMAC)
0x4: Counter with CBC-MAC (CCM)
others: Reserved
Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bits 4:3 **MODE[1:0]**: Operating mode

This bitfield selects the SAES operating mode:
0x0: Encryption
0x1: Key derivation (or key preparation), for ECB/CBC decryption only
0x2: Decryption
0x3: Reserved
Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bits 2:1 **DATATYPE[1:0]**: Data type

This bitfield defines the format of data written in the SAES_DINR register or read from the SAES_DOUTR register, through selecting the mode of data swapping. This swapping is defined in *Section 48.4.16: SAES data registers and data swapping*.
0x0: No swapping (32-bit data).
0x1: Half-word swapping (16-bit data)
0x2: Byte swapping (8-bit data)
0x3: Bit-level swapping
Attempts to write the bitfield are ignored when BUSY is set, as well as when EN is set before the write access and it is not cleared by that write access.

Bit 0 **EN**: Enable

This bit enables/disables the SAES peripheral.

0: Disable

1: Enable

At any moment, clearing then setting the bit re-initializes the SAES peripheral. When KMOD[1:0] is different from 0x0, using IPRST bit is recommended instead.

This bit is automatically cleared by hardware upon the completion of the key preparation (MODE[1:0] at 0x1) and upon the completion of GCM/GMAC/CCM initialization phase.

The bit cannot be set as long as KEYVALID is cleared, or when SAES is in one of the following configurations:

– KMOD[1:0] at 0x1 (wrap), CHMOD[2:0] at 0x3 (GCM)

– KMOD[1:0] at 0x1 (wrap), CHMOD[2:0] at 0x2 (CTR), MODE[1:0] at 0x0 (encryption).

48.8.2 SAES status register (SAES_SR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEYVALID	Res.	Res.	Res.	BUSY	WRERRF	RDERRF	Res.
								r				r	r	r	

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **KEYVALID**: Key valid flag

This bit is set by hardware when the key of size defined by KEYSIZE is loaded in SAES_KEYRx key registers.

0: Key not valid

1: Key valid

The EN bit can only be set when KEYVALID is set.

In normal mode when KEYSEL[2:0] is at zero, the key must be written in the key registers in the correct sequence, otherwise the KEIF flag is set and KEYVALID remains cleared.

When KEYSEL[2:0] is different from zero, the BUSY flag is automatically set by SAES. When the key is loaded successfully, BUSY is cleared and KEYVALID set. Upon an error, KEIF is set, BUSY cleared and KEYVALID remains cleared.

If set, KEIF must be cleared through the SAES_ICR register, otherwise KEYVALID cannot be set. See the KEIF flag description for more details.

For further information on key loading, refer to *Section 48.4.17: SAES key registers*.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **BUSY:** Busy

This flag indicates whether SAES is idle or busy.

0: Idle

1: Busy

SAES is flagged as idle when disabled (when EN is low) or when the last processing is completed.

SAES is flagged as busy when processing a block data, preparing a key (ECB or CBC decryption only), fetching random number from the RNG, or transferring a shared key to the target peripheral.

When GCM encryption payload phase is selected, this flag must be at zero before suspending current process to manage a higher-priority message. BUSY must also be cleared before selecting the GCM final phase.

Bit 2 **WRERRF**: Write error flag

This bit is set when an unexpected write to the SAES_DINR register occurred. When set WRERRF bit has no impact on the SAES operations.

0: No error

1: Unexpected write to SAES_DINR register occurred during computation or data output phase.

The flag setting generates an interrupt if the RWEIE bit of the SAES_IER register is set.

The flag is cleared by setting the RWEIF bit of the SAES_ICR register.

Bit 1 **RDERRF**: Read error flag

This bit is set when an unexpected read to the SAES_DOUTR register occurred. When set RDERRF bit has no impact on the SAES operations.

0: No error

1: Unexpected read to SAES_DOUTR register occurred during computation or data input phase.

The flag setting generates an interrupt if the RWEIE bit of the SAES_IER register is set.

The flag is cleared by setting the RWEIF bit of the SAES_ICR register.

Bit 0 Reserved, must be kept at reset value.

48.8.3 SAES data input register (SAES_DINR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DIN[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIN[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **DIN[31:0]**: Data input

A four-fold sequential write to this bitfield during the Input phase results in writing a complete 16-bytes block of input data to the SAES peripheral. From the first to the fourth write, the corresponding data weights are [127:96], [95:64], [63:32], and [31:0]. Upon each write, the data from the 32-bit input buffer are handled by the data swap block according to the DATATYPE[1:0] bitfield, then written into the AES core 16-bytes input buffer.

Reads return zero.

48.8.4 SAES data output register (SAES_DOUTR)

Address offset: 0x00C

Reset value: 0x0000 0000

Read when KMOD[1:0] is at 0x1 or 0x2 while MODE[1:0] is at 0x2 and EN is set triggers a read error.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DOUT[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DOUT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **DOUT[31:0]**: Data output

This read-only bitfield fetches a 32-bit output buffer. A four-fold sequential read of this bitfield, upon the computation completion (CCF flag set), virtually reads a complete 16-byte block of output data from the SAES peripheral. Before reaching the output buffer, the data produced by the AES core are handled by the data swap block according to the DATATYPE[1:0] bitfield.

Data weights from the first to the fourth read operation are: [127:96], [95:64], [63:32], and [31:0].

48.8.5 SAES key register 0 (SAES_KEYR0)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[31:0]**: Cryptographic key, bits [31:0]

These are bits [31:0] of the write-only bitfield KEY[255:0] AES encryption or decryption key, depending on the MODE[1:0] bitfield of the SAES_CR register.

Writes to SAES_KEYRx registers are ignored when SAES is enabled (EN bit set). When KEYSEL[2:0] is different from 0 and KEYVALID is 0, writes to key registers are also ignored and they result in setting the KEIF bit of the SAES_ISR register.

With KMOD[1:0] at 0x0, a special writing sequence is required. In this sequence, any valid write to AES_KEYRx register clears the KEYVALID flag except for the sequence-completing write that sets it. Also refer to the description of the KEYVALID flag in the AES_SR register.

48.8.6 SAES key register 1 (SAES_KEYR1)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[63:48]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[47:32]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[63:32]**: Cryptographic key, bits [63:32]

Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

48.8.7 SAES key register 2 (SAES_KEYR2)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[95:80]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[79:64]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[95:64]**: Cryptographic key, bits [95:64]

Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

48.8.8 SAES key register 3 (SAES_KEYR3)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[127:112]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[111:96]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[127:96]**: Cryptographic key, bits [127:96]

Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

48.8.9 SAES initialization vector register 0 (SAES_IVR0)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[31:0]**: Initialization vector input, bits [31:0]

SAES_IVRx registers store the 128-bit initialization vector or the nonce, depending on the chaining mode selected. This value is updated by hardware after each computation round (when applicable). Write to this register is ignored when EN bit is set in SAES_CR register

48.8.10 SAES initialization vector register 1 (SAES_IVR1)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[63:48]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[47:32]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[63:32]**: Initialization vector input, bits [63:32]

Refer to the SAES_IVR0 register for description of the IVI[128:0] bitfield.

48.8.11 SAES initialization vector register 2 (SAES_IVR2)

Address offset: 0x028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[95:80]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[79:64]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[95:64]**: Initialization vector input, bits [95:64]

Refer to the SAES_IVR0 register for description of the IVI[128:0] bitfield.

48.8.12 SAES initialization vector register 3 (SAES_IVR3)

Address offset: 0x02C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[127:112]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[111:96]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[127:96]**: Initialization vector input, bits [127:96]
Refer to the SAES_IVR0 register for description of the IVI[128:0] bitfield.

48.8.13 SAES key register 4 (SAES_KEYR4)

Address offset: 0x030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[159:144]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[143:128]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[159:128]**: Cryptographic key, bits [159:128]
Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

48.8.14 SAES key register 5 (SAES_KEYR5)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[191:176]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[175:160]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[191:160]**: Cryptographic key, bits [191:160]
Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

48.8.15 SAES key register 6 (SAES_KEYR6)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[223:208]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[207:192]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[223:192]**: Cryptographic key, bits [223:192]

Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

48.8.16 SAES key register 7 (SAES_KEYR7)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[255:240]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[239:224]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[255:224]**: Cryptographic key, bits [255:224]

Refer to the SAES_KEYR0 register for description of the KEY[255:0] bitfield and for information relative to writing SAES_KEYRx registers.

48.8.17 SAES suspend registers (SAES_SUSPRx)

Address offset: 0x040 + 0x4 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SUSP[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SUSP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SUSP[31:0]**: Suspend data

SAES_SUSPRx registers contain the complete internal register states of the SAES when the CCM processing of the current task is suspended to process a higher-priority task. Refer to *Section 48.4.8: SAES suspend and resume operations* for more details.

Clearing EN bit of the SAES_CR register clears this register to zero.

SAES_SUSPRx registers are not used in other chaining modes than CCM.

48.8.18 SAES interrupt enable register (SAES_IER)

Address offset: 0x300

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGEIE	KEIE	RWEIE	CCFIE
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **RNGEIE:** RNG error interrupt enable

This bit enables or disables (masks) the SAES interrupt generation when RNGEIF (RNG error flag) is set.

0: Disabled (masked)

1: Enabled (not masked)

Bit 2 **KEIE:** Key error interrupt enable

This bit enables or disables (masks) the SAES interrupt generation when KEIF (key error flag) is set.

0: Disabled (masked)

1: Enabled (not masked)

Bit 1 **RWEIE:** Read or write error interrupt enable

This bit enables or disables (masks) the SAES interrupt generation when RWEIF (read and/or write error flag) is set.

0: Disabled (masked)

1: Enabled (not masked)

Bit 0 **CCFIE:** Computation complete flag interrupt enable

This bit enables or disables (masks) the SAES interrupt generation when CCF (computation complete flag) is set.

0: Disabled (masked)

1: Enabled (not masked)

48.8.19 SAES interrupt status register (SAES_ISR)

Address offset: 0x304

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGEIF	KEIF	RWEIF	CCF
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **RNGEIF:** RNG error interrupt flag

This read-only bit is set by hardware when an error is detected on RNG bus interface (for example bad entropy).

0: RNG bus is functional

1: Error detected on RNG bus interface (random seed fetching error)

The flag setting generates an interrupt if the RNGEIE bit of the SAES_IER register is set.

The flag is cleared by setting the corresponding bit of the SAES_ICR register. The clear action triggers the reload of a new random number from the RNG peripheral.

Bit 2 **KEIF:** Key error interrupt flag

This read-only bit is set by hardware when the key information fails to load into key registers or when the key register use is forbidden.

0: No key error detected

1: Key information failed to load into key registers or the key register use is forbidden

The flag setting generates an interrupt if the KEIE bit of the SAES_IER register is set. It also clears the key registers and the KEYVALID flag in the SAES_SR register.

The flag is cleared by setting the corresponding bit of the SAES_ICR register.

KEIF is raised upon any of the following events:

–SAES fails to load the DHUK (KEYSEL[2:0] = 0x1 or 0x4).

–SAES fails to load the BHK (KEYSEL[2:0] = 0x2 or 0x4).

–CRYP fails to load the key shared by SAES peripheral (KMOD[1:0] = 0x2).

–KEYVALID is set and either KEYPROT is set or KEYSEL[2:0] is other than 0x0. The security context of the application that loads the key (secure or nonsecure) does not match the security attribute of the access to SAES_CR or SAES_DOUT. In this case, KEYVALID and EN bits are cleared.

–SAES_KEYRx register write does not respect the correct order. (For KEYSIZE cleared, SAES_KEYR0 then SAES_KEYR1 then SAES_KEYR2 then SAES_KEYR3 register, or reverse. For KEYSIZE set, SAES_KEYR0 then SAES_KEYR1 then SAES_KEYR2 then SAES_KEYR3 then SAES_KEYR4 then SAES_KEYR5 then SAES_KEYR6 then SAES_KEYR7, or reverse).

KEIF must be cleared by the application software, otherwise KEYVALID cannot be set.

Bit 1 **RWEIF:** Read or write error interrupt flag

This read-only bit is set by hardware when a RDERRF or a WRERRF error flag is set in the SAES_SR register.

0: No read or write error detected

1: Read or write error detected

The flag setting generates an interrupt if the RWEIE bit of the SAES_IER register is set.

The flag is cleared by setting the corresponding bit of the SAES_ICR register.

The flags has no meaning when key derivation mode is selected.

See the SAES_SR register for details.

Bit 0 **CCF:** Computation complete flag

This flag indicates whether the computation is completed. It is significant only when the DMAOUTEN bit is cleared, and it may stay high when DMAOUTEN is set.

0: Not completed

1: Completed

The flag setting generates an interrupt if the CCFIE bit of the SAES_IER register is set.

The flag is cleared by setting the corresponding bit of the SAES_ICR register.

48.8.20 SAES interrupt clear register (SAES_ICR)

Address offset: 0x308

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGEIF	KEIF	RWEIF	CCF
												w	w	w	w

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **RNGEIF:** RNG error interrupt flag clear

Application must set this bit to clear the RNGEIF status bit in SAES_ISR register.

Bit 2 **KEIF:** Key error interrupt flag clear

Setting this bit clears the KEIF status bit of the SAES_ISR register.

Bit 1 **RWEIF:** Read or write error interrupt flag clear

Setting this bit clears the RWEIF status bit of the SAES_ISR register, and clears both RDERRF and WRERRF flags in the SAES_SR register.

Bit 0 **CCF:** Computation complete flag clear

Setting this bit clears the CCF status bit of the SAES_ISR register.

48.8.21 SAES register map

Table 414. SAES register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	SAES_CR	IPRST	KEYSEL[2:0]			KSHAREID[1:0]		KMOD[1:0]		NPBLB[3:0]				KEYPROT	KEYSIZE	Res.	CHMOD[2]	Res.	GCMPH[1:0]		DMAOUTEN	DMAINEN	Res.	Res.	Res.	Res.	CHMOD[1]	CHMOD[0]	MODE[1:0]		DATATYPE[1:0]		EN
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0		0	0	0	0					0	0	0	0	0	0	0
0x004	SAES_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEYVALID	Res.	Res.	Res.	BUSY	WRERRF	RDERRF	Res.
	Reset value																									0				0	0	0	
0x008	SAES_DINR	DIN[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	SAES_DOUTR	DOUT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	SAES_KEYR0	KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	SAES_KEYR1	KEY[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	SAES_KEYR2	KEY[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	SAES_KEYR3	KEY[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	SAES_IVR0	IVI[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	SAES_IVR1	IVI[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	SAES_IVR2	IVI[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	SAES_IVR3	IVI[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	SAES_KEYR4	KEY[159:128]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	SAES_KEYR5	KEY[191:160]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	SAES_KEYR6	KEY[223:192]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	SAES_KEYR7	KEY[255:224]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 414. SAES register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x040	SAES_SUSPR0	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	SAES_SUSPR1	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048	SAES_SUSPR2	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	SAES_SUSPR3	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	SAES_SUSPR4	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054	SAES_SUSPR5	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x058	SAES_SUSPR6	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x05C	SAES_SUSPR7	SUSP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x060-0x2FF	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
0x300	SAES_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGEIE	KEIE	RWEIE	CCFIE
	Reset value																													0	0	0	0
0x304	SAES_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGEIF	KEIF	RWEIF	CCF
	Reset value																													0	0	0	0
0x308	SAES_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RNGEIF	KEIF	RWEIF	CCF
	Reset value																													0	0	0	0
0x309-0x3FF	Reserved	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

49 Cryptographic processor (CRYP)

49.1 CRYP introduction

The cryptographic processor (CRYP) encrypts or decrypts data in compliance with the advanced encryption standard (AES) defined by NIST.

CRYP supports ECB, CBC, CTR, GCM, GMAC, and CCM chaining modes for key sizes of 128, 192 or 256 bits. CRYP has the possibility to load by hardware the key stored in SAES peripheral, under SAES control.

The peripheral supports both single and fixed DMA burst transfers for incoming and outgoing data (two DMA channels are required). CRYP also includes input and output FIFOs for better performance.

49.2 CRYP main features

- Compliant with NIST FIPS publication 197 *"Advanced encryption standard (AES)"* (November 2001)
- Encryption and decryption with multiple chaining modes:
  - Electronic codebook (ECB) mode
  - Cipher block chaining (CBC) mode
  - Counter (CTR) mode
  - Galois counter mode (GCM)
  - Galois message authentication code (GMAC) mode
  - Counter with CBC-MAC (CCM) mode
- 16-byte data block processing, supporting cipher key lengths of 128, 192 and 256 bits
  - 14 or 18 clock cycle latency in ECB mode for processing one 16-byte block with 128-bit or 256-bit key, respectively
- Using dedicated key bus, optional key sharing with side-channel resistant SAES peripheral (shared-key mode), controlled by SAES
- Integrated key scheduler to compute the last round key for ECB/CBC decryption
- 256-bit of write-only registers for storing the cryptographic keys (eight 32-bit registers)
- 128-bit of registers for storing initialization vectors (four 32-bit registers)
- 32-bit input buffer associated with an internal input FIFO of eight 32-bit words, corresponding to two AES blocks
- 32-bit output buffer associated with an internal output FIFO of eight 32-bit words, corresponding to two AES blocks
- Automatic data flow control supporting two direct memory access (DMA) channels, one for incoming data, one for processed data. The output FIFO supports both single and burst transfers, while the input FIFO supports only burst transfers.
- Data swapping logic to support 1-, 8-, 16- or 32-bit data
- AMBA AHB target peripheral, accessible through 32-bit word single accesses only. Other access types generates an AHB error, and write accesses are ignored.
- Possibility for software to suspend a message if CRYP needs to process another message with a higher priority, then resume the original message

49.3 CRYP implementation

The devices have one CRYP peripheral, implemented as per the following table.It can use the key generated by the SAES peripheral. For comparison, the SAES peripheral is also included in the table.

Table 415. CRYP versus SAES features

Modes or features(1)	CRYP	SAES
ECB, CBC chaining	X	X
CTR, CCM, GCM chaining	X	X
AES 128-bit ECB encryption in cycles	14	480
DHUK and BHK key selection	-	X
Resistance to side-channel attacks	-	X
Shared key between SAES and CRYP	X	
Key size in bits	128, 192, 256	128, 256

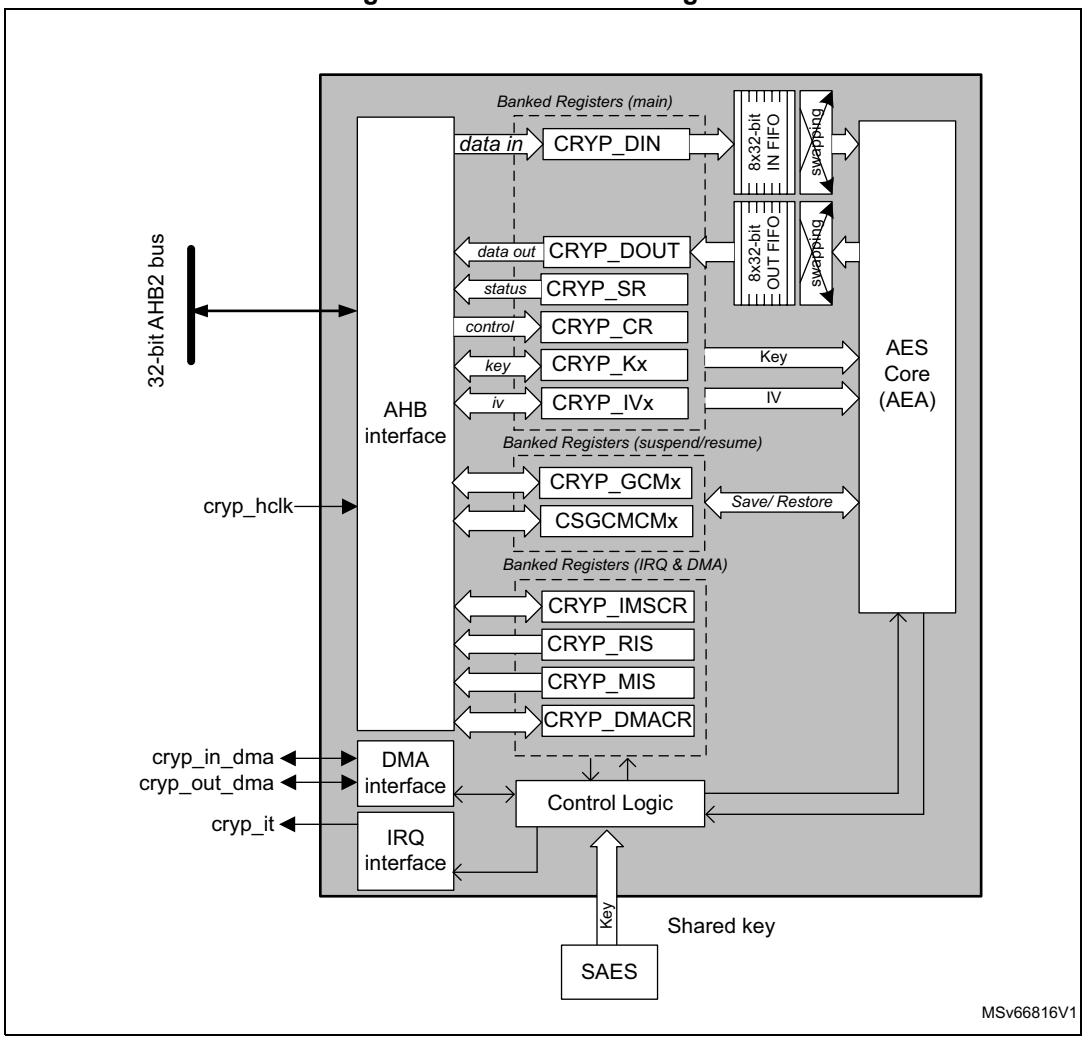
1. X = supported.

49.4 CRYP functional description

49.4.1 CRYP block diagram

The figure below shows the block diagram of the cryptographic processor.

Figure 475. CRYP block diagram



49.4.2 CRYP internal signals

Table 416 describes the user relevant internal signals interfacing the CRYP peripheral.

Table 416. CRYP internal input/output signals

Signal name	Signal type	Description
cryp_hclk	Digital input	AHB clock
cryp_it	Digital output	CRYP interrupt request
cryp_in_dma	Digital input/output	CRYP incoming data FIFO DMA request/ acknowledge
cryp_out_dma	Digital input/output	CRYP processed data FIFO DMA request/ acknowledge

49.4.3 CRYP reset and clocks

The CRYP peripheral is clocked by the AHB clock.

The CRYP has a dedicated reset bit in the RCC.

49.4.4 CRYP symmetric cipher implementation

The cryptographic processor (CRYP) is a 32-bit AHB peripheral that encrypts or decrypts 16-byte blocks of data using the advanced encryption standard (AES). It also implements a set of approved AES symmetric key security functions summarized in *Table 417*. Those functions can be certified NIST PUB 140-3.

Table 417. CRYP approved AES symmetric key functions

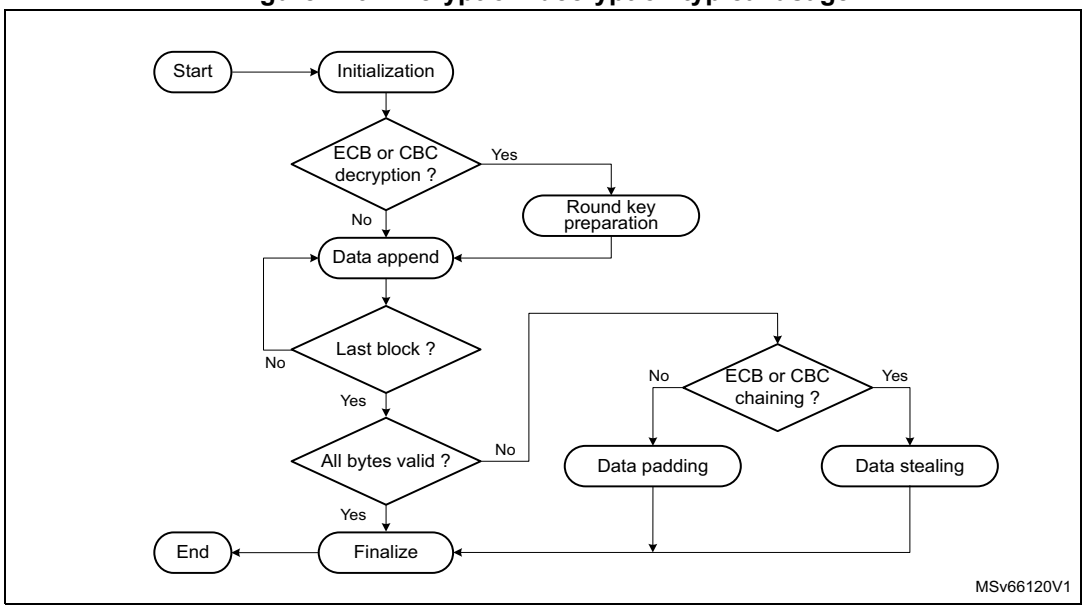
Operations	Algorithm	Specification	Key size in bits	Chaining modes
Encryption, decryption	AES	FIPS PUB 197 NIST SP800-38A	128, 192, 256	ECB, CBC, CTR
Authenticated encryption or decryption		NIST SP800-38C NIST SP800-38D		GCM, CCM
Cipher-based message authentication code		NIST SP800-38D		GMAC

CRYP can be used directly by the CPU, or indirectly using two DMA channels (one for the plaintext, one for the ciphertext).
It is possible to suspend then resume any CRYP processing, following the sequence described in *Section 49.4.8*.

49.4.5 CRYP encryption/ decryption typical usage

The following figure shows a typical operation for encryption or decryption.

Figure 476. Encryption/ decryption typical usage



Initialization

The CRYP peripheral is initialized according to the chaining mode. Refer to *Section 49.4.9: CRYP basic chaining modes (ECB, CBC)* and *Section 49.4.10: CRYP counter mode (CTR)* for details.

Data append

This section describes different ways of appending data for processing. For ECB or CBC chaining modes, refer to *Section 49.4.7: CRYP ciphertext stealing and data padding* if the size of data to process is not a multiple of 16 bytes. The last block management in these cases is more complex than what is described in this section.

General conditions for data processing

The CRYP is busy and processing data (BUSY flag set) when all the conditions below are met:

- CRYPEN is set in CRYP_CR register.
- There are enough data in the input FIFO (at least four words).
- There is enough free-space in the output FIFO (at least four word locations).

It is possible to clear the CRYPEN bit while BUSY bit is set. In this case the ongoing AES processing first completes (that is, the word results are written to the output FIFO) before the BUSY bit is cleared by hardware.

Note: If the application needs to suspend a message to process another one with a higher priority, refer to Section 49.4.8: CRYP suspend and resume operations.

Appending data using the CPU in polling mode

This method uses flag polling to control the data append through the following sequence:

1. When KEYVALID is set, enable CRYP by setting the CRYPEN bit of the CRYP_CR register (if not already done).
2. Repeat the following subsequence until the payload is entirely processed:
   a) Write data in the input FIFO (one block or until the FIFO is full).
   b) Repeat until the second last block of data has been processed:
   – Wait until the not-empty flag OFNE is set, then read the output FIFO (one block or until the FIFO is empty).
   – Wait until the not-full flag IFNF is set, then write the input FIFO (one block or until the FIFO is full), except if it is the last block.
   c) The BUSY bit is set automatically by CRYP. At the end of the processing, the BUSY bit is cleared and both FIFOs are empty (input FIFO empty flag IFEM is set, output FIFO not-empty flag OFNE is cleared).
   d) If the next processing block is the last block, pad the data with zeros to obtain a complete block (when applicable), and specify the number of non-valid bytes (using NPBLB[3:0]) in case of GCM payload encryption or CCM payload decryption (otherwise the tag computation is wrong). This operation must be performed after checking that the BUSY bit is cleared.
3. As the data block just processed is the last block of the message, optionally discard the data that is not part of the message/payload, then disable the CRYP peripheral by clearing CRYPEN.

Note: NPBLB[3:0] bitfield is not used in header phase of GCM, GMAC and CCM chaining modes.

Appending data using the CPU in Interrupt mode

The method uses interrupt from the CRYP peripheral to control the data append, through the following sequence:

1. Enable interrupts from CRYP, by setting the INIM and OUTIM bits of the CRYP_IMSCR register.
2. When KEYVALID is set, enable the CRYP peripheral, by setting CRYPEN (if not already done).
3. Handle the data in the CRYP interrupt service routine that manages the input FIFO. Upon each interrupt:
   a) If the next processing block is the last block of the message, pad (when applicable) the data with zeros to obtain a complete block, and specify the number of non-valid bytes (using NPBLB[3:0] bitfield) in case of GCM payload encryption or CCM payload decryption (otherwise the tag computation is wrong). This operation must be performed after checking that the BUSY bit is cleared. After that load the block into the input FIFO.
   b) If the next processing block is not the last block, load the data into the input FIFO. Application can load only one block (4 words), or load data until the input FIFO is full.
   c) After the last word of data has been written, disable the input FIFO interrupts by clearing the INIM bit (if applicable), then quit the interrupt service routine.

4. Handle the data in the CRYP interrupt service routine that manages the output FIFO. Upon each interrupt:
   a) Read data from the output FIFO. Application can read only one block (4 words), or read data until the FIFO is empty.
   b) When the last word of the message has been read, INIM and BUSY bits are cleared and both FIFOs are empty (IFEM is set, OFNE is cleared). Disable the output FIFO interrupt by clearing the OUTIM bit, and disable the peripheral by clearing the CRYPEN bit.
   c) If the data block just processed is the last block of the message, optionally discard the data that are not part of the message/payload, then quit the interrupt service routine.

Note: The NPBLB[3:0] bitfield is not used in the header phase of GCM, GMAC and CCM chaining modes.

Appending data using the DMA

With this method, all the transfers and processing are managed by DMA and CRYP. Proceed as follows:

1. If the last block of the message to process is shorter than 16 bytes, prepare the last four-word data block by padding the remainder of the block with zeros.
2. Configure the DMA controller so as to transfer the data to process from the memory to the CRYP peripheral input and the processed data from the CRYP peripheral output to the memory, as described in *Section 49.6: CRYP DMA requests*. Configure the DMA controller so as to generate an interrupt on transfer completion. For GCM payload encryption or CCM payload decryption, the DMA transfer **must not** include the last block. The sequence described in *Appending data using the CPU in polling mode* must be used instead for this last block, because the NPBLB[3:0] bitfield must be set up before processing the block, for CRYP to compute a correct tag.
3. When KEYVALID is set, enable the CRYP peripheral, by setting CRYPEN (if not already done).
4. Enable DMA requests, by setting DIEN and DOEN of the CRYP_DMACR register.
5. Upon DMA interrupt indicating the transfer completion, get the CRYP-processed data from the memory. Both FIFOs are normally empty and BUSY flag is cleared.

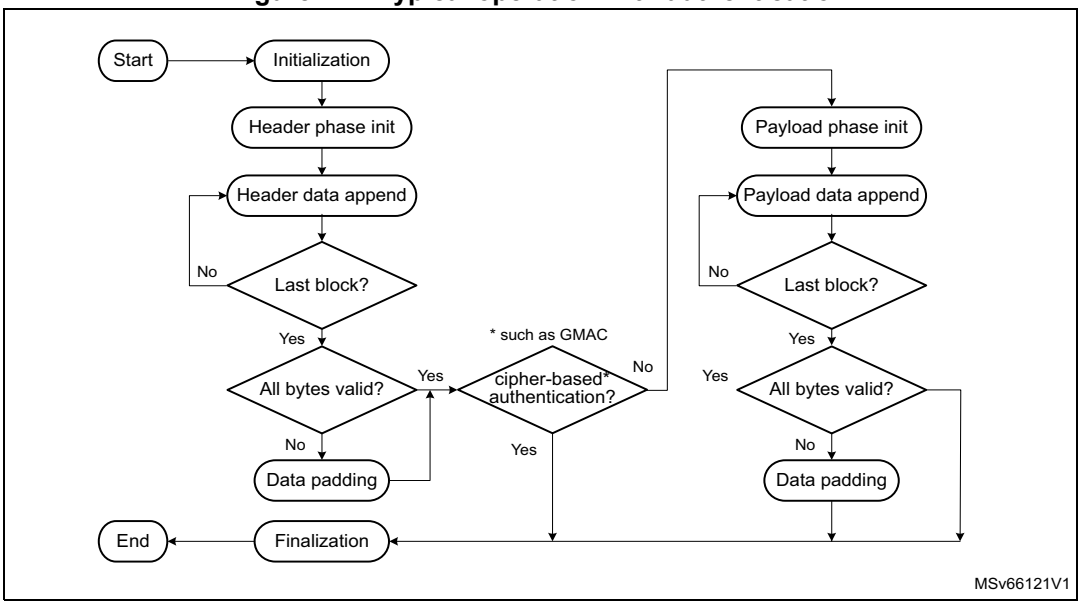
Note: The NPBLB[3:0] bitfield is not used in the header phase of GCM, GMAC and CCM chaining modes.

Caution: It is important for the DMA controller to empty the CRYP output FIFO buffer before filling the CRYP input FIFO buffer. To ensure this, configure the DMA controller to prioritize transfers from CRYP to memory over those from memory to CRYP.

49.4.6 CRYP authenticated encryption, decryption, and cipher-based message authentication

The following figure shows a typical operation for authenticated encryption or decryption, and for cipher-based message authentication.

Figure 477. Typical operation with authentication



Section 49.4.11: CRYP AES Galois/counter mode (GCM) and *Section 49.4.13: CRYP AES Counter with CBC-MAC (CCM)* describe detailed sequences supported by CRYP.

Cipher-based message authentication flow omits the payload phase, as shown in the figure. Detailed sequence supported by CRYP is described in *Section 49.4.12: CRYP AES Galois message authentication code (GMAC)*.

49.4.7 CRYP ciphertext stealing and data padding

When using ECB or CBC modes to manage messages the size of which is not a multiple of the block size (16 bytes), the application must use ciphertext stealing techniques such as those described in NIST *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode*. Since CRYP does not implement such techniques, the application must complete the last block of input data using data from the second last block.

Note: Ciphertext stealing techniques are not documented in this reference manual.

Similarly, in other modes than ECB or CBC, an incomplete input data block (that is, a block with input data shorter than 16 bytes) must be padded with zeros prior to encryption. That is, extra bits must be appended to the trailing end of the data string. After decryption, the extra bits must be discarded. Since CRYP does not implement automatic data padding operation to the last block, the application must follow the recommendation given in this document to manage messages the size of which is not a multiple of 16 bytes.

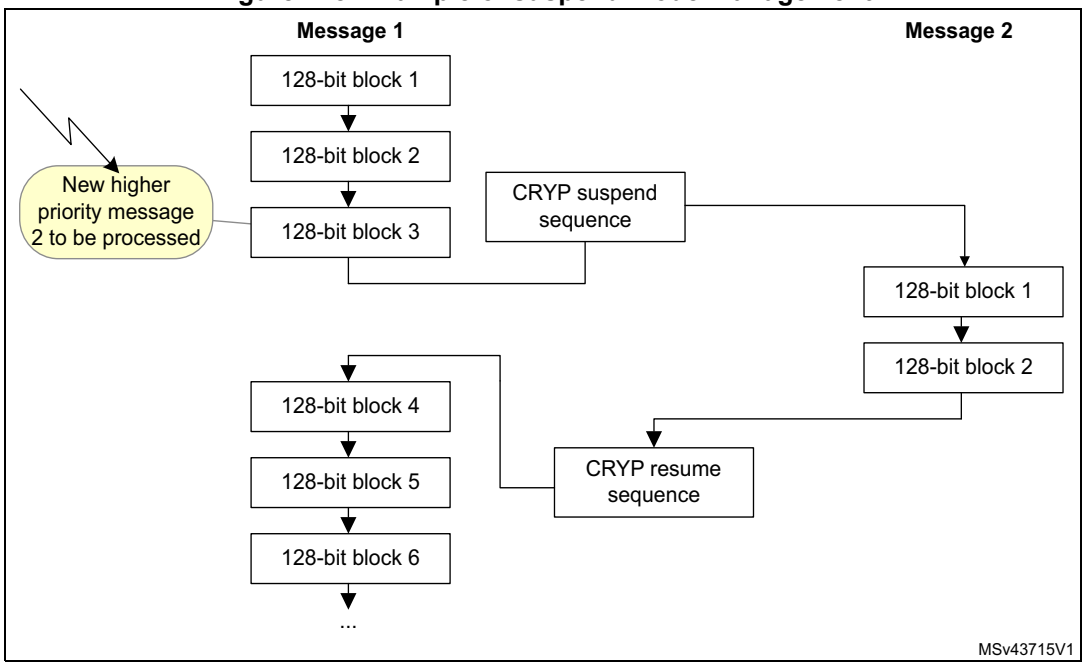
49.4.8 CRYP suspend and resume operations

A message can be suspended to process another message with a higher priority. When the higher-priority message is sent, the suspended message can resume. This applies to both encryption and decryption mode.

Suspend and resume operations do not break the chaining operation. The message processing can resume as soon as CRYP is enabled again, to receive a next data block.

Figure 478 gives an example of suspend and resume operation: message 1 is suspended in order to send a shorter and higher-priority message 2.

Figure 478. Example of suspend mode management



Detailed descriptions of suspend and resume operations are provided in the sections dedicated to each chaining mode.

49.4.9 CRYP basic chaining modes (ECB, CBC)

ECB is the simplest mode of operation. There are no chaining operations, and no special initialization stage. The message is divided into blocks and each block is encrypted or decrypted separately. When decrypting in ECB mode, a special key scheduling is required before processing the first block.

Figure 479 and *Figure 480* describe the electronic codebook (ECB) chaining implementation in encryption and decryption, respectively. To select this chaining mode, write ALGOMODE[3:0] with 0x4.

Figure 479. ECB encryption

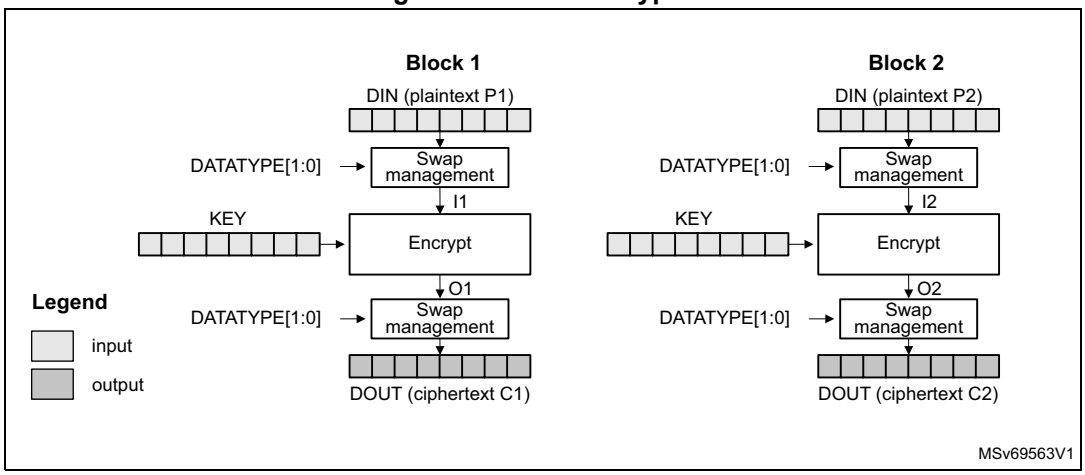
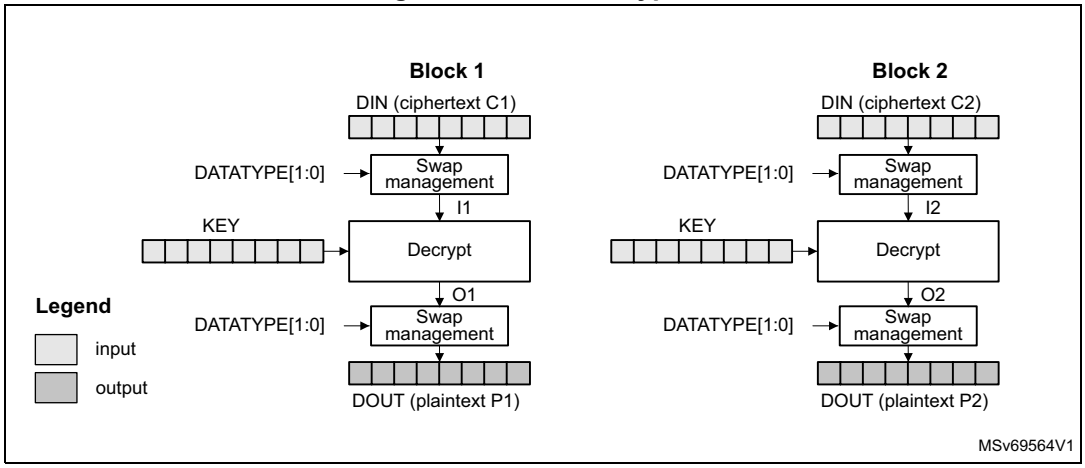


Figure 480. ECB decryption



In CBC encryption mode the output of each block chains with the input of the following block. To make each message unique, an initialization vector is used during the first block processing. When decrypting in CBC mode, a special key scheduling is required before processing the first block.

Figure 481 and *Figure 482* describe the cipher block chaining (CBC) implementation in encryption and decryption, respectively. To select this chaining mode, write ALGOMODE[3:0] with 0x5.

Figure 481. CBC encryption

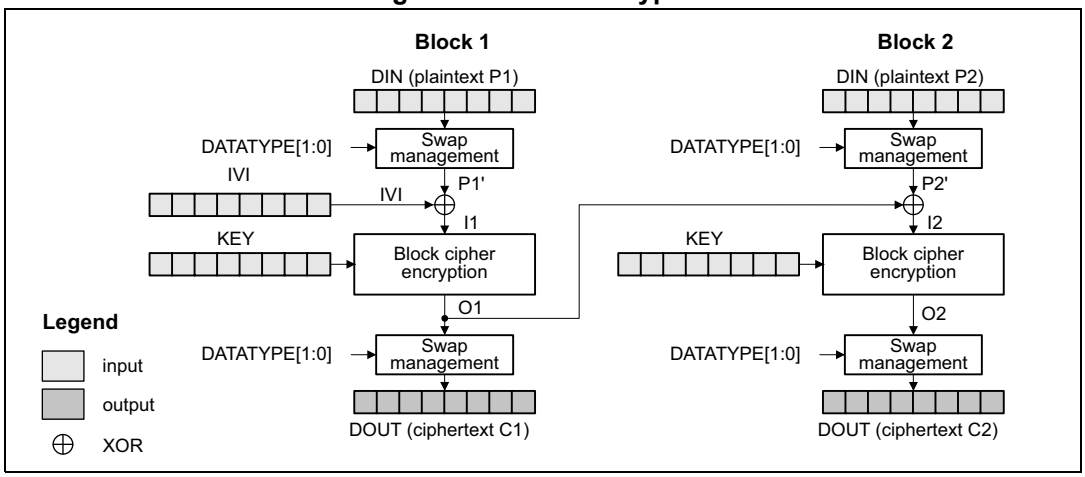
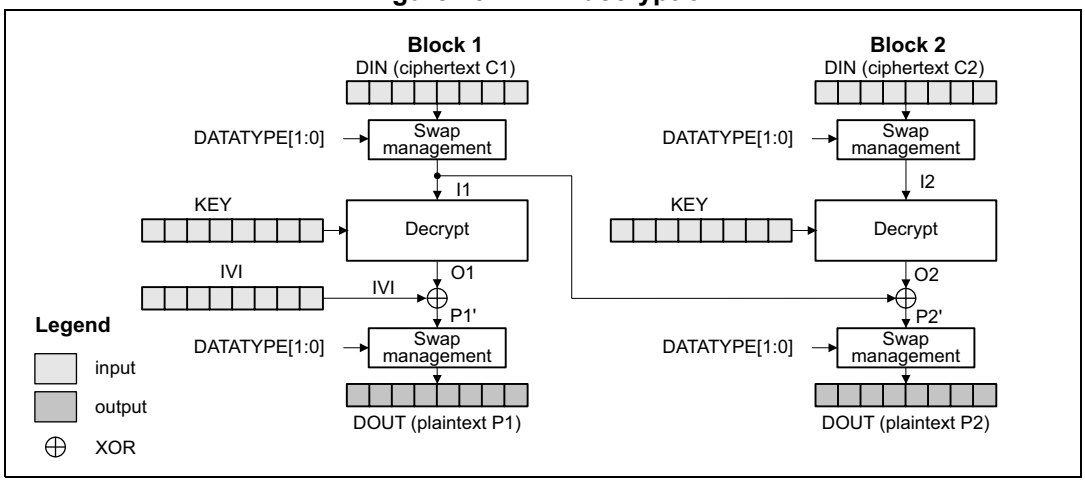


Figure 482. CBC decryption



For more details, refer to NIST *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation.*

ECB/CBC encryption process

This process is described in *Section 49.4.5*, with the following sequence of events:

1. Disable the CRYP peripheral by clearing CRYPEN.
2. Flush the FIFOs by setting FFLUSH.
3. Initialize the CRYP_CR register as follows:
   a) Select ECB or CBC chaining mode (write ALGOMODE[3:0] with 0x4 or 0x5) in encryption mode (ALGODIR = 0).
   b) Configure the data type through DATATYPE[1:0].
   c) Configure the key size through KEYSIZE.
   d) Select the key mode using KMOD[1:0]: if the key is coming from SAES peripheral, set KMOD[1:0] to 0x2, else keep it at 0x0.
4. Write the initialization vector into the CRYP_IVxL/R registers if CBC mode is selected in the previous step.
5. Write the key into the CRYP_KxL/R registers if KMOD[1:0] is 0x0. If KMOD[1:0] = 0x2, the key is being transferred from the SAES peripheral (see *Section 49.4.14*).
6. Wait until KEYVALID is set (the key loading completed).
7. Enable CRYP by setting CRYPEN.
8. Append cleartext data:
   a) If it is the second-last or the last block and the plaintext size of the message is not a multiple of 16 bytes, follow the guidance in *Section 49.4.7*.
   b) Append the cleartext block into CRYP as described in *Section 49.4.5*, then read the CRYP_DOUTR register to save the ciphertext block(s), MSB first.
   c) Repeat the step *b)* until the third-last plaintext block is encrypted. For the last two blocks, follow the steps *a)* and *b)*.
9. Finalize the sequence: disable the CRYP peripheral, by clearing CRYPEN.

Note: KEYSIZE and ALGOMODE[3:0] must be written before the key, as explained above.

KMOD[1:0] = 0x2 (shared key) must be written after ALGOMODE[3:0] or at the same time.

ECB/CBC decryption process

This process is described in *Section 49.4.5*, with the following sequence of events:

1. Disable the CRYP peripheral, by clearing CRYPEN.
2. Flush the FIFOs by setting FFLUSH.
3. Initialize the CRYP_CR register as follows:
   a) Select the key derivation mode (write ALGOMODE[3:0] with 0x7) in encryption mode (ALGODIR= 0).
   b) Configure the data type through DATATYPE[1:0].
   c) Configure the key size through KEYSIZE.
   d) Select the key mode, using KMOD[1:0]: if the key is coming from SAES peripheral set KMOD[1:0] to 0x2, else keep it at 0x0.
4. Write the key into the CRYP_KxL/R registers if KMOD[1:0] is 0x0. If KMOD[1:0]= 0x2, the key is being transferred from the SAES peripheral (see *Section 49.4.14*).
5. Wait until KEYVALID is set (the key loading completed).
6. Enable the CRYP peripheral by setting CRYPEN. The peripheral immediately starts an AES round for key preparation, setting the BUSY bit.
7. Wait until the BUSY bit is cleared, then select ECB or CBC chaining mode (write ALGOMODE[3:0] with 0x4 or 0x5) in decryption mode (ALGODIR = 1). Do not change other parameters.
8. Write the initialization vector into the CRYP_IVxL/R registers if CBC mode is selected in the previous step.
9. Enable CRYP by setting CRYPEN.
10. Append encrypted data:
    a) If it is the second-last or the last block and the ciphertext size of the message is not a multiple of 16 bytes, follow the guidance in *Section 49.4.7*.
    b) Append the ciphertext block into CRYP as described in *Section 49.4.5*, then read the CRYP_DOUTR register to save the cleartext block(s), MSB first.
    c) Repeat the step *b)* until the third-last ciphertext block is decrypted. For the last two blocks, follow the steps *a)* and *b)*.
11. Finalize the sequence: disable the CRYP peripheral, by clearing CRYPEN.

Note: KEYSIZE and ALGOMODE[3:0] must be written before the key, as explained above.

KMOD[1:0] = 0x2 (shared key) must be written after ALGOMODE[3:0] or at the same time.

Suspend and resume operations in ECB/CBC modes

To suspend the processing of a message, proceed as follows:

1. If DMA is used, stop the CRYP DMA transfers to the input FIFO by clearing the DIEN bit of the CRYP_DMACR register.
2. Wait until both FIFOs are empty (IFEM = 1 and OFNE = 0 in CRYP_SR) and the BUSY bit is cleared.
3. If DMA is used, stop the CRYP DMA transfers from the output FIFO, by clearing the DOEN bit of the CRYP_DMACR register.
4. Disable the CRYP peripheral, by clearing CRYPEN.
5. Save the CRYP_CR register and clear the key registers if they are not needed, to process the higher-priority message.

6. Save initialization vector registers (only required in CBC mode as the CRYP_IVxL/R registers are altered during the data processing).
7. If DMA is used, save the DMA controller status (pointers for CRYP input and output data transfers, number of remaining bytes, and so on).

To resume the processing of a message, proceed as follows:

1. If DMA is used, configure the DMA controller so as to complete the remaining input FIFO and output FIFO transfers.
2. Disable the CRYP peripheral, by clearing CRYPEN.
3. Restore the CRYP_CR register (with correct KEYSIZE), then restore the CRYP_KxL/R registers. If KMOD[1:0] = 0x2 the key needs to be transferred again from the SAES peripheral (see *Section 49.4.14*).
4. Prepare the decryption key as described in *ECB/CBC decryption process* (only required for ECB or CBC decryption).
5. Restore the CRYP_IVxL/R registers, using the saved configuration (only required in CBC mode).
6. Enable the CRYP peripheral by setting CRYPEN.
7. If DMA is used, enable CRYP DMA transfers, by setting DIEN and DOEN.

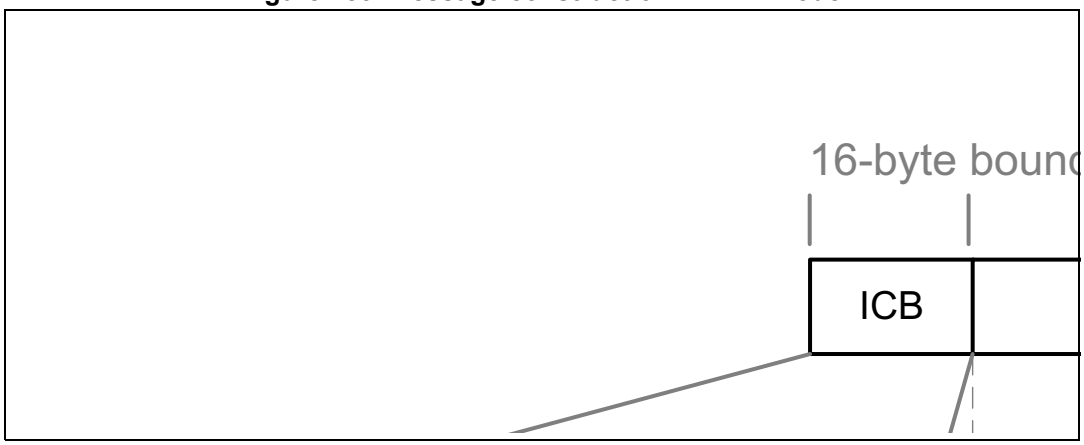
Note: Key registers do not need to be saved as the original key value is known by the application.

49.4.10 CRYP counter mode (CTR)

The CTR mode uses the AES core to generate a key stream. The keys are then XOR-ed with the plaintext to obtain the ciphertext. Unlike with ECB and CBC modes, no key scheduling is required for the CTR decryption since the AES core is always used in encryption mode.

A typical message construction in CTR mode is given in *Figure 483*.

Figure 483. Message construction in CTR mode



The structure of this message is:

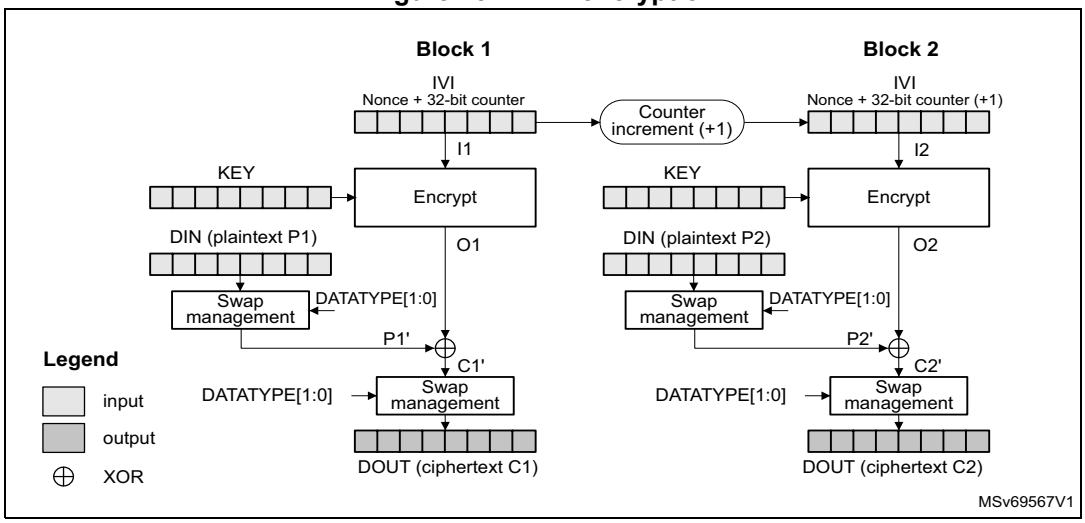
- A 16-byte initial counter block (ICB), composed of two distinct fields:
  - **Initialization vector** (IVI): a 96-bit value that must be unique for each encryption cycle with a given key.
  - **Counter**: a 32-bit big-endian integer that is incremented each time a block processing is complete. The initial value of the counter must be set.
- The plaintext P is encrypted as ciphertext C, with a known length. This length can be non-multiple of 16 bytes, in which case a plaintext padding is required.

For more details, refer to NIST *Special Publication 800-38A, Recommendation for Block Cipher Modes of Operation.*

CTR encryption and decryption

Figure 484 describes the counter (CTR) chaining implementation in the CRYP peripheral (encryption). To select this chaining mode, write ALGOMODE[3:0] with 0x6.

Figure 484. CTR encryption



Initialization vectors in CRYP must be initialized as shown in *Table 418*.

Table 418. Counter mode initialization vector definition

CRYP_IV0LR[31:0]	CRYP_IV0RR[31:0]	CRYP_IV1LR[31:0]	CRYP_IV1RR[31:0]
IVI[127:96]	IVI[95:64]	IVI[63:32]	IVI[31:0] 32-bit counter = 0x1

CTR encryption and decryption process

This process is described in *Section 49.4.5*, with the following sequence of events:

1. Disable the CRYP peripheral, by clearing CRYPEN.
2. Flush the FIFOs by setting FFLUSH.
3. Initialize the CRYP_CR register as follows:
   a) Select CTR chaining mode (write ALGOMODE[3:0] with 0x6) in encryption or decryption mode (write ALGODIR with 0 or 1).
   b) Configure the data type through DATATYPE[1:0].
   c) Configure the key size through KEYSIZE.
   d) Select the key mode, using KMOD[1:0]: if the key is coming from SAES peripheral set KMOD[1:0] to 0x2 else keep it at 0x0.
4. Write the initialization vector into the CRYP_IVxL/R registers according to *Table 418*.
5. Write the key into the CRYP_KxL/R registers if KMOD[1:0] is 0x0. If KMOD[1:0]= 0x2 the key is being transferred from the SAES peripheral (see *Section 49.4.14*).
6. Wait until KEYVALID is set (the key loading completed).
7. Enable CRYP by setting CRYPEN.
8. Append data:
   a) If it is the last block and the plaintext (encryption) or ciphertext (decryption) size in the block is less than 16 bytes, pad the remainder of the block with zeros.
   b) Append the data block into CRYP as described in *Section 49.4.5*, then read the CRYP_DOUTR register to save the resulting block(s), MSB first.
   c) Repeat the step *b)* until the second-last block is processed. For the last block of plaintext (encryption only), follow the steps *a)* and *b)*. For the last block, discard the bits that are not part of the message when the last block is smaller than 16 bytes.
9. Finalize the sequence: disable the CRYP peripheral, by clearing CRYPEN.

Note: KEYSIZE and ALGOMODE[3:0] must be written before the key, as explained above.

KMOD[1:0] = 0x2 (shared key) must be written after ALGOMODE[3:0] or at the same time.

Suspend and resume operations in CTR mode

Like for the CBC mode, it is possible to interrupt a message to send a higher priority message, then resume the interrupted message. Detailed CBC suspend and resume sequence is described in *Section 49.4.9: CRYP basic chaining modes (ECB, CBC)*.

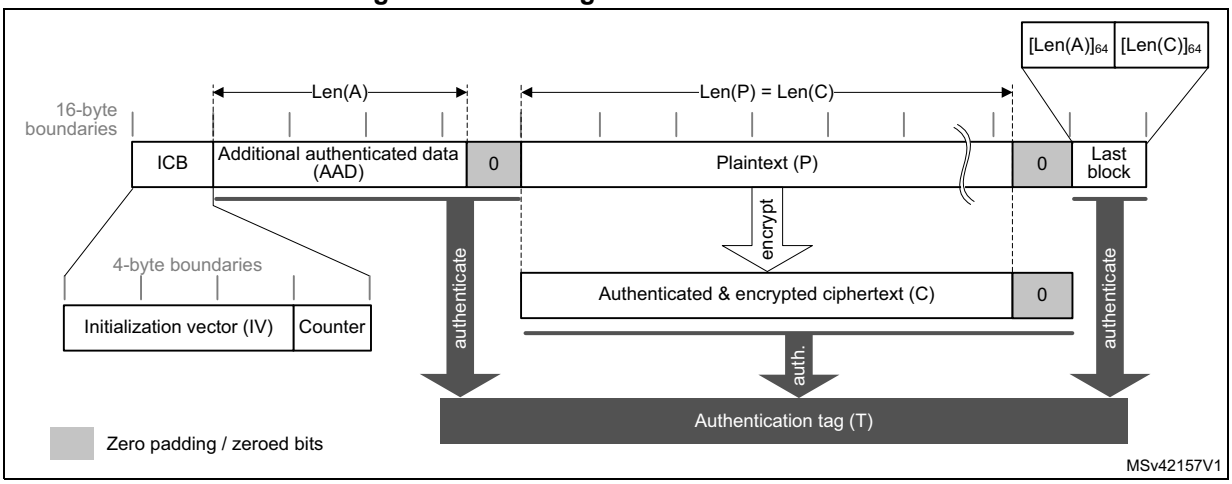
Note: Like for CBC mode, IVI registers must be reloaded during the resume operation.

49.4.11 CRYP AES Galois/counter mode (GCM)

The AES Galois/counter mode (GCM) enables encrypting and authenticating a plaintext message into the corresponding ciphertext and tag (also known as message authentication code).

GCM mode is based on AES in counter mode for confidentiality. It uses a multiplier over a fixed finite field for computing the message authentication code. The following figure shows a typical message construction in GCM mode.

Figure 485. Message construction in GCM



The structure of this message is defined as below:

- **16-byte Initial Counter Block (ICB)**, composed of two distinct fields:
  - **Initialization vector (IVI)**: a 96-bit value that must be unique for each encryption cycle with a given key. The GCM standard supports IVIs with less than 96 bits, but in this case strict rules apply.
  - **Counter***:* a 32-bit big-endian integer that is incremented each time a block processing is complete. According to NIST specification, the counter value is 0x2 when processing the first block of payload.
- **Authenticated header AAD** (also knows as additional authentication data) has a known length Len(A) that may be a non-multiple of 16 bytes, and must not exceed $2^{64} – 1$ bits. This part of the message is only authenticated, not encrypted.
- **Plaintext message P** is both authenticated and encrypted as ciphertext C, with a known length Len(P) that may be non-multiple of 16 bytes, and cannot exceed $2^{32} - 2$ 16-bytes blocks.
- **Last block** contains the AAD header length (bits [32:63]) and the payload length (bits [96:127]) information, as shown in *Table 420*.

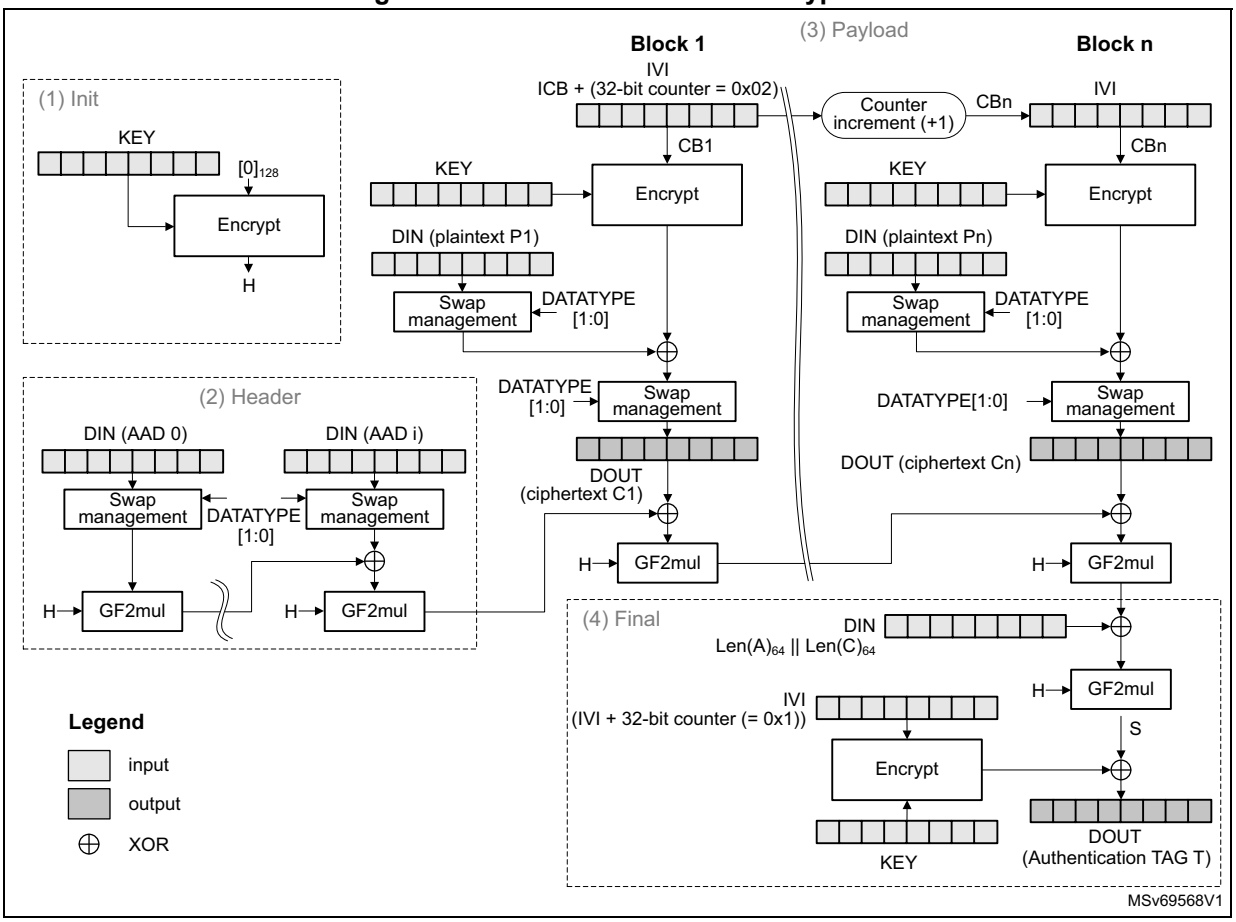
The GCM standard specifies that ciphertext C has the same bit length as the plaintext P.

A special padding scheme is required when a part of the message (AAD or P) has a length that is a non-multiple of 16-bytes.

For more details, refer to NIST *Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.*

Figure 486 describes the GCM chaining implementation in the CRYP peripheral (encryption). To select this chaining mode, write ALGOMODE[3:0] with 0x6.

Figure 486. GCM authenticated encryption



The first counter block (CB1) is derived from the initial counter block ICB by the application software, as defined in *Table 419*.

Table 419. GCM mode IVI registers initialization

Register	CRYP_IV0LR[31:0]	CRYP_IV0RR[31:0]	CRYP_IV1LR[31:0]	CRYP_IV1RR[31:0]
Input data	ICB[127:96]	ICB[95:64]	ICB[63:32]	ICB[31:0] 32-bit counter = 0x2

The last block of a GCM message contains the AAD header length and the payload length information, as shown in *Table 420*.

Table 420. GCM last block definition

Word order to CRYP_DINR	First word	Second word	Third word	Fourth word
Input data	AAD length[63:32]	AAD length[31:0]	Payload length[63:32]	Payload length[31:0]

GCM encryption and decryption process

This process is described in *Section 49.4.6*, with the following sequence of events:

GCM initialize

1. Disable the CRYP peripheral, by clearing CRYPEN.
2. Flush the FIFOs by setting FFLUSH.
3. Initialize the CRYP_CR register:
   a) Select GCM chaining mode (write ALGOMODE[3:0] with 0x8) in encryption or decryption mode (write ALGODIR with 0 or 1).
   b) Configure the data type through DATATYPE[1:0]
   c) Configure the key size through KEYSIZE.
   d) Select the key mode, using KMOD[1:0]: if the key is coming from SAES peripheral set KMOD[1:0] to 0x2 else keep it at 0x0.
   e) Select the GCM initialization phase, by writing 0x0 to GCM_CCMPH[1:0]
4. Write the initialization vector in CRYP_IVxL/R registers according to *Table 419*.
5. Write the key in CRYP_KxL/R registers if KMOD[1:0] is 0x0. If KMOD[1:0]= 0x2, the key is being transferred from the SAES peripheral (see *Section 49.4.14*).
6. Wait until KEYVALID is set (the key loading completed).
7. Set CRYPEN to start the calculation of the hash key. CRYPEN is automatically cleared when the calculation is complete.
8. Wait for the CRYPEN bit to be cleared, indicating that the GCM hash subkey (H) computation is complete.

GCM header phase

9. Initialize header phase:
   a) Select the GCM header phase, by writing 0x1 to GCM_CCMPH[1:0]. Do not change the other configurations written during GCM initialization.
   b) Set CRYPEN to start accepting data.
10. Append header data:
    a) If it is the last block and the AAD in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data into CRYP as described in *Section 49.4.5*.
    c) Repeat the step *b)* until the second-last AAD data block is processed. For the last block follow the steps *a)* and *b)*.
11. Wait until the BUSY flag is cleared.

Note: This phase can be skipped if there is no AAD, that is, Len(A) = 0.

No data are read during header phase.

GCM payload phase

12. Initialize payload phase:
    a) Disable the CRYP peripheral, by clearing CRYPEN.
    b) Select the GCM payload phase, by writing GCM_CCMPH[1:0] with 0x2. Do not change the other configurations written during GCM initialization.
    c) Enable the CRYP peripheral by setting CRYPEN, to start accepting data.
13. Append payload data:
    a) If it is the last block and the message in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data into CRYP as described in *Section 49.4.5*, then read the CRYP_DOUTR register to save the resulting block(s).
    c) Repeat the step *b)* until the second-last plaintext block is encrypted or until the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), follow the steps *a)* and *b)*. For the last block, discard the bits that are not part of the payload when the last block is smaller than 16 bytes.
14. Wait until the BUSY flag is cleared.

Note: This phase can be skipped if there is no payload, that is, Len(C) = 0 (see GMAC mode).

GCM finalization

15. Select the GCM final phase, by writing GCM_CCMPH[1:0] with 0x3, and clear the ALGODIR bit (encryption). Do not change the other configurations written during GCM initialization.
16. Write the final GCM block into the CRYP_DINR register. It is the concatenated AAD bit and payload bit lengths, as shown in *Table 420*.
17. Wait until the OFNE flag in the CRYP_SR register is set.
18. Get the GCM authentication tag, by reading the CRYP_DOUTR register four times.
19. Disable the CRYP peripheral, by clearing CRYPEN. If it is an authenticated decryption, compare the generated tag with the expected tag passed with the message.

Note: In the final phase, data are written to CRYP_DINR normally (no swapping).

Suspend and resume operations in GCM mode

To suspend the processing of a message, proceed as follows:

1. If DMA is used, stop the CRYP DMA transfers to the input FIFO, by clearing DIEN bit of the CRYP_DMACR register.
2. Wait until both FIFOs are empty (IFEM = 1 and OFNE = 0 in CRYP_SR) and the BUSY bit is cleared.
3. In the payload phase, if DMA is used, stop the CRYP DMA transfers from the output FIFO, by clearing the DOEN bit of the CRYP_DMACR register.
4. Disable the CRYP by clearing CRYPEN.
5. Save the current CRYP_CR configuration in the memory. Key registers do not need to be saved as the original key value is known by the application.
6. In the payload phase, save the CRYP_IVxL/R registers as, during the data processing, they changed from their initial values. In the header phase, this step is not required.
7. Save the CRYP_CSGCMCCMRx and CRYP_CSGCMRx registers in memory.
8. If DMA is used, save the DMA controller status (pointer for CRYP input data transfers, number of remaining bytes, and so on). In the payload phase, also save pointer for CRYP output data transfers.

To resume the processing of a message, proceed as follows:

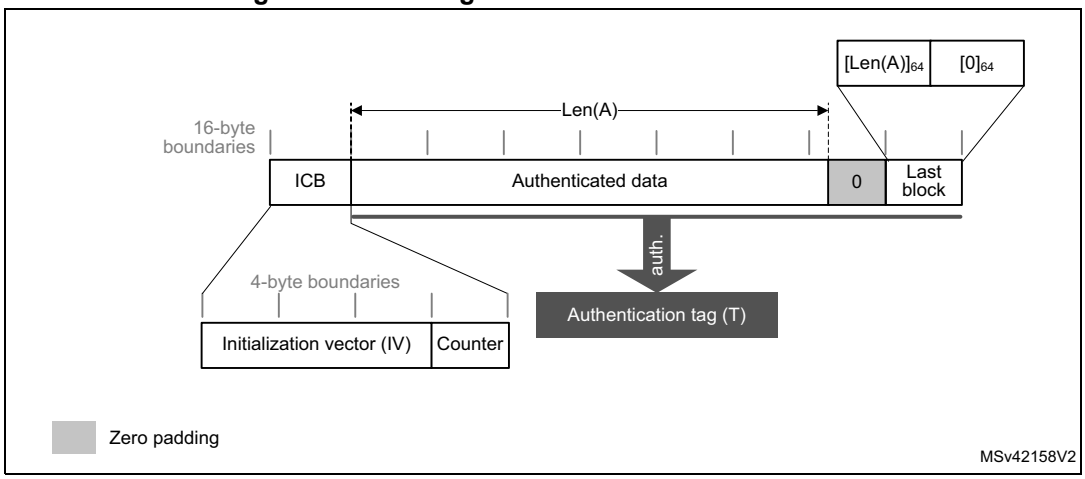
1. If DMA is used, configure the DMA controller in order to complete the remaining input FIFO transfers. In the payload phase, also configure the DMA controller for the remaining output FIFO transfers.
2. Disable the CRYP peripheral, by clearing CRYPEN.
3. Restore the initial setting values in the CRYP_CR and CRYP_KxL/R registers. If KMOD[1:0]= 0x2 the key needs to be transferred again from the SAES peripheral (see *Section 49.4.14*).
4. Write the suspend register values, previously saved in the memory, back into their corresponding CRYP_CSGCMCCMRx and CRYP_CSGCMRx registers.
5. In the payload phase, write the initialization vector register values, previously saved in the memory, back into their corresponding CRYP_IVxL/R registers. In the header phase, write initial setting values back into the CRYP_IVxL/R registers.
6. Enable the CRYP peripheral by setting CRYPEN.
7. If DMA is used, enable CRYP DMA requests, by setting DIEN (and DOEN if in payload phase).

49.4.12 CRYP AES Galois message authentication code (GMAC)

The Galois message authentication code (GMAC) enables the authentication of a plaintext, generating the corresponding tag information (also known as message authentication code).

GMAC is similar to GCM, except that it is applied on a message composed only by plaintext authenticated data (that is, only header, no payload). The following figure shows typical message construction for GMAC.

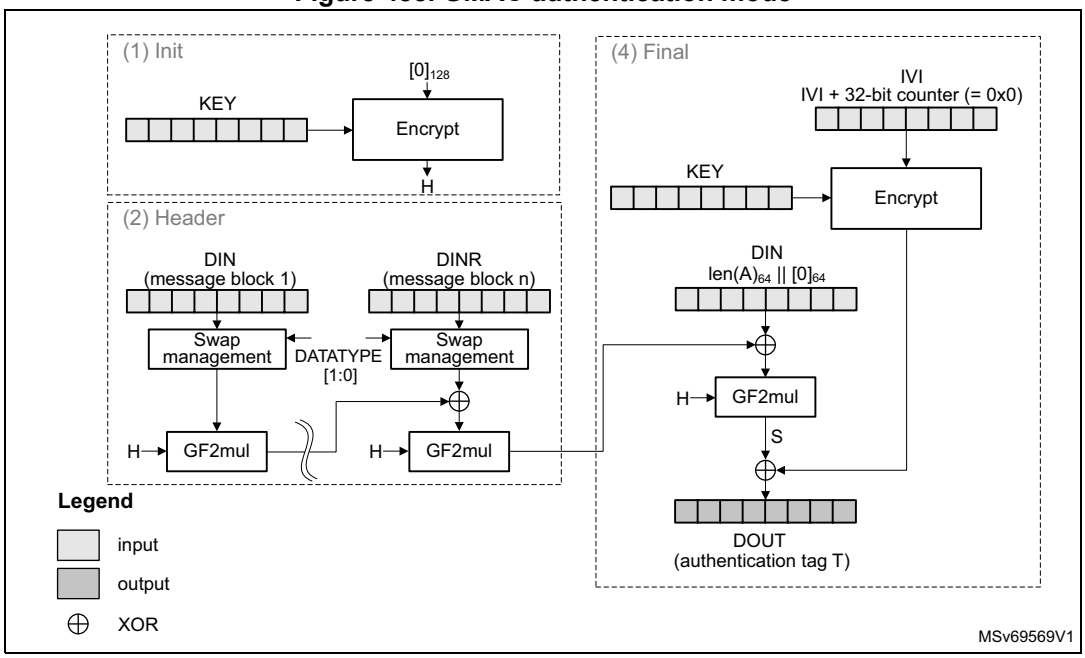
Figure 487. Message construction in GMAC mode



For more details, refer to NIST *Special Publication 800-38D, Recommendation for Block Cipher Modes of Operation - Galois/Counter Mode (GCM) and GMAC.*

Figure 488 describes the GMAC chaining implementation in the CRYP peripheral. To select this chaining mode, write ALGOMODE[3:0] with 0x6.

Figure 488. GMAC authentication mode



The GMAC algorithm corresponds to the GCM algorithm applied on a message only containing a header. As a consequence, all steps and settings are the same as for the GCM, except that the payload phase is omitted.

Suspend and resume operations in GMAC

In GMAC mode, the sequence described for the GCM applies except that only the header phase can be interrupted.

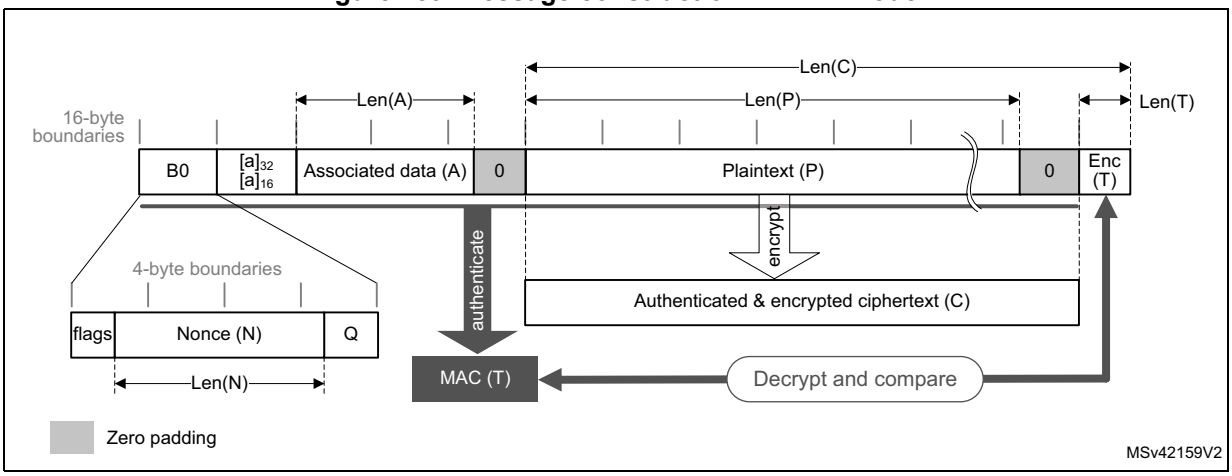
49.4.13 CRYP AES Counter with CBC-MAC (CCM)

The AES counter with cipher block chaining-message authentication code (CCM) algorithm enables encryption and authentication of plaintext, generating the corresponding ciphertext and tag (also known as message authentication code). To ensure confidentiality, the CCM algorithm is based on AES in counter mode. It uses cipher block chaining technique to generate the message authentication code. This is commonly called CBC-MAC.

Note: NIST does not approve CBC-MAC as an authentication mode outside the context of the CCM specification.

The following figure shows typical message construction for CCM.

Figure 489. Message construction in CCM mode



The structure of the message is:

- **16-byte first authentication block (B0)**, composed of three distinct fields:
  - **Q:** a bit string representation of the octet length of P (Len(P))
  - **Nonce (N):** a single-use value (that is, a new nonce must be assigned to each new communication) of Len(N) size. The sum Len(N) + Len(P) must be equal to 15 bytes.
  - **Flags:** most significant octet containing four flags for control information, as specified by the standard. It contains two 3-bit strings to encode the values **t** (MAC length expressed in bytes) and **Q** (plaintext length such that Len(P) < 2^{8Q} bytes). The counter blocks range associated to **Q** is equal to 2^{8Q-4}, that is, if the maximum value of **Q** is 8, the counter blocks used in cipher must be on 60 bits.

- **16-bytes blocks (B)** associated to the Associated Data (A).

  This part of the message is only authenticated, not encrypted. This section has a known length Len(A) that can be a non-multiple of 16 bytes (see *Figure 489*). The standard also states that, on the MSB bits of the first message block (B1), the associated data length expressed in bytes (*a*) must be encoded as follows:

  – If $0 < a < 2^{16} - 2^8$, then it is encoded as $[a]_{16}$, that is, on two bytes.

  – If $2^{16} - 2^8 < a < 2^{32}$, then it is encoded as 0xFF || 0xFE || $[a]_{32}$, that is, on six bytes.

  – If $2^{32} < a < 2^{64}$, then it is encoded as 0xFF || 0xFF || $[a]_{64}$, that is, on ten bytes.
- **16-byte blocks (B)** associated to the plaintext message P, which is both authenticated and encrypted as ciphertext C, with a known length Len(P). This length can be a non-multiple of 16 bytes (see *Figure 489*).
- **Encrypted MAC (T)** of length Len(T) appended to the ciphertext C of overall length Len(C).

When a part of the message (A or P) has a length that is a non-multiple of 16-bytes, a special padding scheme is required.

Note: CCM chaining mode can also be used with associated data only (that is, no payload).

As an example, the C.1 section in *NIST Special Publication 800-38C* gives the following values (hexadecimal numbers):

N: **10111213 141516** (Len(N) = 56 bits or 0x7 bytes)

A: **00010203 04050607** (Len(A) = 64 bits or 0x8 bytes)

P: *20212223* (Len(P) = 32 bits i.e. Q = 0x*4* bytes)

T: 6084341b (Len(T) = 32 bits or t = 4)

B0: 4f**101112 13141516** 00000000 00000004
B1: 0008**0001 02030405 0607**0000 00000000
B2: *20212223* 00000000 00000000 00000000

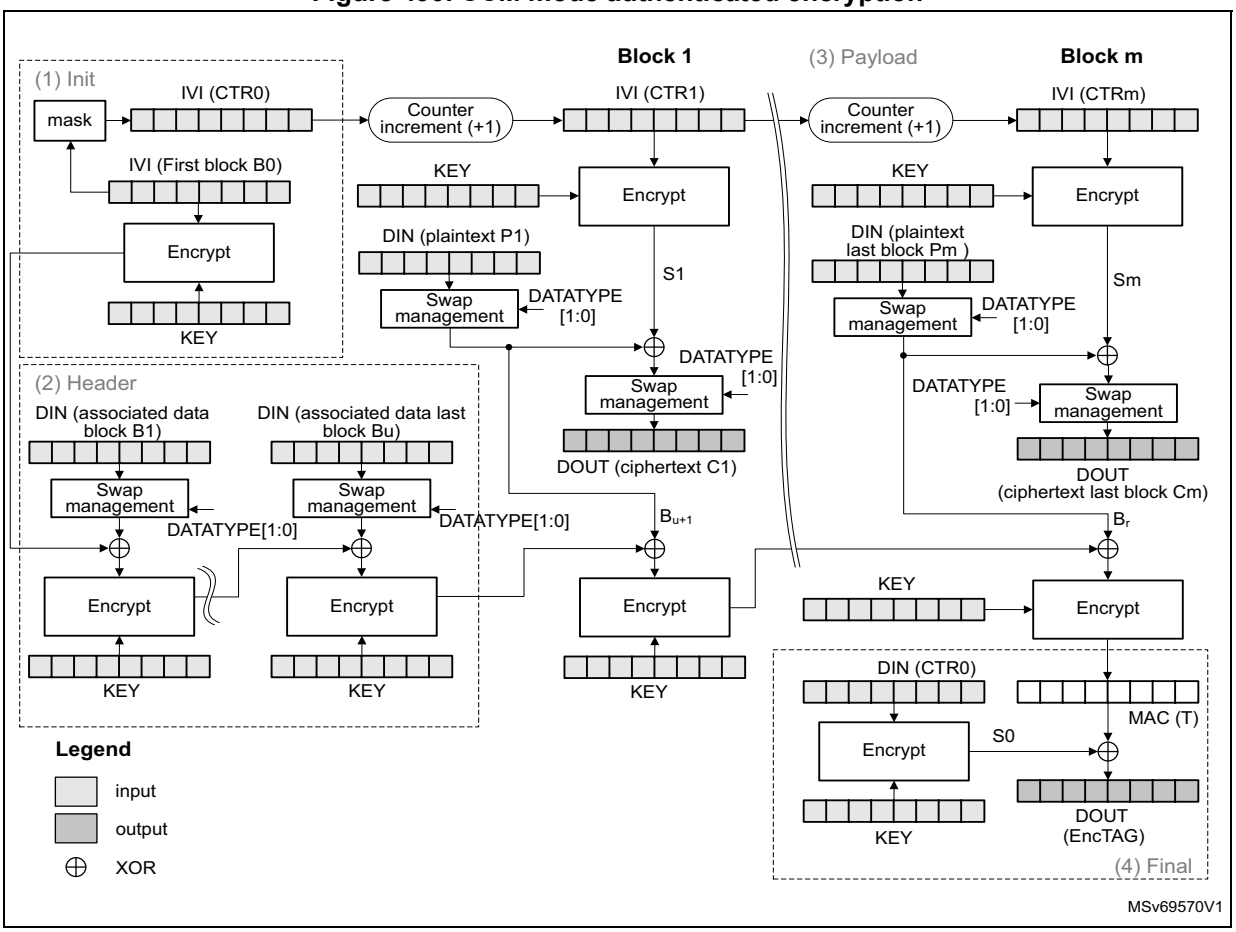
CTR0: 07**10111213 141516** 00000000 00000000

CTR1: 07**10111213 141516** 00000000 00000001

For more details, refer to NIST *Special Publication 800-38C, Recommendation for Block Cipher Modes of Operation - The CCM Mode for Authentication and Confidentiality*.

Figure 490 describes the CCM chaining implementation in the CRYP peripheral (encryption). To select this chaining mode, write ALGOMODE[3:0] with 0x9.

Figure 490. CCM mode authenticated encryption



The first block of a CCM message (B0) must be prepared by the application as defined in *Table 421*.

Table 421. CCM mode IVI registers initialization

Register	CRYP_IV0LR[31:0]	CRYP_IV0RR[31:0]	CRYP_IV1LR[31:0]	CRYP_IV1RR[31:0]
Input data	B0[127:96](1)	B0[95:64]	B0[63:32]	B0[31:0](2)

1. The 5 most significant bits are cleared (flag bits).
2. Q length bits are cleared, except for bit 0 that is set.

CRYP supports counters up to 64 bits, as specified by NIST.

CCM encryption and decryption process

This process is described in *Section 49.4.6*, with the following sequence of events:

CCM initialize

1. Disable the CRYP peripheral, by clearing CRYPEN.
2. Flush the FIFOs by setting FFLUSH.
3. Initialize the CRYP_CR register:
   a) Select CCM chaining mode (write ALGOMODE[3:0] with 0x9) in encryption or decryption mode (write ALGODIR with 0 or 1).
   b) Configure the data type through DATATYPE[1:0]
   c) Configure the key size through KEYSIZE.
   d) Select the key mode, using KMOD[1:0]: if the key is coming from SAES peripheral set KMOD[1:0] to 0x2 else keep it at 0x0.
   e) Select the CCM initialization phase, by writing 0x0 to GCM_CCMPH[1:0]
4. Write the initialization vector into the CRYP_IVxL/R registers with CTR1 information, as defined in *Table 421*
5. Write the key in CRYP_KxL/R registers if KMOD[1:0] is 0x0. If KMOD[1:0]= 0x2 the key is being transferred from the SAES peripheral (see *Section 49.4.14*).
6. Wait until KEYVALID is set (the key loading completed).
7. Enable the CRYP peripheral by setting CRYPEN.
8. Write the B0 data in CRYP_DINR registers according to *Table 421*, then wait for the CRYPEN bit to be cleared.

Note: In this initialization phase, data have to be inserted normally (no swapping).

CCM header phase

9. Initialize header phase:
   a) Prepare the first block of the (B1) data associated with the message, in accordance with CCM chaining rules.
   b) Select the GCM header phase, by writing GCM_CCMPH[1:0] with 0x1. Do not change the other configurations written during the GCM initialization.
   c) Set CRYPEN to start accepting data.
10. Append header data:
    a) If it is the last block and the associated data in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data into CRYP as described in *Section 49.4.5*.
    c) Repeat the step *b)* until the second-last associated data block is processed. For the last block follow the steps *a)* and *b)*.
11. Wait until the BUSY flag is cleared.

Note: This phase can be skipped if there is no associated data, that is, Len(A) = 0

No data are read during the header phase.

CCM payload phase

12. Initialize payload phase:
    a) Disable the CRYP peripheral, by clearing CRYPEN.
    b) Select the GCM payload phase, by writing GCM_CCMPH[1:0] with 0x2. Do not change the other configurations written during GCM initialization.
    c) Enable the CRYP peripheral by setting CRYPEN, to start accepting data.
13. Append payload data:
    a) In encryption only, if it is the last block and the plaintext in the block is smaller than 16 bytes, pad the remainder of the block with zeros.
    b) Append the data into CRYP as described in *Section 49.4.5*, then read the CRYP_DOUTR register to save the resulting block(s).
    c) Repeat the step *b)* until the second-last plaintext block is encrypted or until the last block of ciphertext is decrypted. For the last block of plaintext (encryption only), follow the steps *a)* and *b)*. For the last block, discard the bits that are not part of the payload when the last block is smaller than 16 bytes.
14. Wait until the BUSY flag is cleared.

Note: This phase can be skipped if there is no payload, that is, Len(P) = 0 or Len(C) = Len(T).

Remove $LSB_{Len(T)}(C)$ encrypted tag information when decrypting ciphertext C.

CCM finalization

15. Select the GCM final phase, by writing GCM_CCMPH[1:0] with 0x3, and clear the ALGODIR bit (encryption). Do not change the other configurations written during GCM initialization.
16. Write the CTR0 information into the CRYP_DINR register. CTR0 information is described in *Table 421*, with the bit[0] cleared.
17. Wait until the OFNE flag in the CRYP_SR register is set.
18. Get the CCM authentication tag, by reading the CRYP_DOUTR register four times.
19. Disable the CRYP peripheral by clearing CRYPEN. If it is an authenticated decryption, compare the generated tag with the expected tag passed with the message. Mask the authentication tag output with tag length to obtain a valid tag.

Note: In this final phase, data have to be inserted normally (no swapping).

Suspend and resume operations in CCM mode

To suspend the processing of a message in header or payload phase, proceed as follows:

1. If DMA is used, stop the CRYP DMA transfers to the input FIFO, by clearing DIEN bit of the CRYP_DMACR register.
2. Wait until both FIFOs are empty (IFEM = 1 and OFNE = 0 in CRYP_SR) and the BUSY bit is cleared.
3. In the payload phase, if DMA is used, stop the CRYP DMA transfers from the output FIFO, by clearing the DOEN bit of the CRYP_DMACR register.
4. Disable the CRYP by clearing CRYPEN.
5. Save the current CRYP_CR configuration in the memory. Key registers do not need to be saved as the original key value is known by the application.
6. In the payload phase, save the CRYP_IVxL/R registers as, during the data processing, they changed from their initial values. In the header phase, this step is not required.
7. Save the CRYP_CSGCMCCMRx registers in the memory.
8. If DMA is used, save the DMA controller status (pointer for CRYP input data transfers, number of remaining bytes, and so on). In the payload phase, also save pointer for CRYP output data transfers.

To resume the processing of a message, proceed as follows:

1. If DMA is used, configure the DMA controller in order to complete the remaining input FIFO transfers. In the payload phase, also configure the DMA controller for the remaining output FIFO transfers.
2. Disable the CRYP peripheral, by clearing CRYPEN.
3. Restore the initial setting values in the CRYP_CR and CRYP_KxL/R registers. If KMOD[1:0]= 0x2 the key needs to be transferred again from the SAES peripheral (see *Section 49.4.14*).
4. Write the suspend register values, previously saved in the memory, back into their corresponding CRYP_CSGCMCCMRx registers.
5. In the payload phase, write the initialization vector register values, previously saved in the memory, back into their corresponding CRYP_IVxL/R registers. In the header phase, write initial setting values back into the CRYP_IVxL/R registers.
6. Enable the CRYP peripheral by setting CRYPEN.
7. If DMA is used, enable CRYP DMA requests, by setting DIEN (and DOEN if in payload phase).

49.4.14 AES key sharing with secure AES co-processor

The CRYP peripheral can use the SAES peripheral as security co-processor. The secure application prepares the key in the robust SAES peripheral. When it is ready, the CRYP application can load this prepared key through a dedicated hardware key bus.

The recommended sequence is described hereafter and in section *SAES operations with shared keys* in section *Secure AES coprocessor (SAES)* of this document.

1. In SAES peripheral, the application encrypts the key to share in shared-key mode (KMOD[1:0] at 0x2). When applicable, the proper CRYP instance must be selected through KSHAREID[1:0].
2. Each time the shared key is required in the CRYP peripheral, the application decrypts it in the SAES peripheral in shared-key mode (KMOD[1:0] at 0x2). When applicable, the proper CRYP instance must be selected through KSHAREID[1:0].
3. Once the shared key is decrypted and loaded in SAES_KEYRx registers it can be shared with CRYP. To load the shared key in CRYP, the application sets KEYSIZE as appropriate and writes KMOD[1:0] with 0x2. When KEYVALID is cleared, the key is automatically transfered by hardware into the CRYP_KxL/R registers and the BUSY flag in the CRYP_SR register set.
4. Once the key transfer is complete, the BUSY flag is cleared and the KEYVALID flag set in the CRYP_SR register. If KEYVALID is not set when BUSY bit is cleared, or if the KERF flag is set in the CRYP_SR register, it means that the KEYSIZE value is incorrect or an unexpected event occurred during the transfer (such as DPA error, tamper event or KEYVALID in SAES_SR cleared before the end of the transfer). When such errors occur, reset both peripherals through their IPRST bits and restart the whole key sharing process.

When the key sharing sequence is complete, the CRYP is initialized with a valid, shared key. The application can then process data in normal key mode, by writing KMOD[1:0] with 0x0.

Note: This sequence in CRYP peripheral can be run multiple times (for example, to manage a suspend/resume situation), as long as SAES peripheral is unused and duly remains in key sharing state.

When sharing keys KEYSIZE in CRYP must be identical to SAES value (128 or 256 bits).

49.4.15 CRYP data registers and data swapping

Data input and output

A 16-byte data block enters the CRYP peripheral with four successive 32-bit word writes into the CRYP_DINR register (bitfield DIN[31:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

A 16-byte data block is retrieved from the CRYP peripheral with four successive 32-bit word reads of the CRYP_DOUTR register (bitfield DOUT[31:0]), the most significant word (bits [127:96]) first, the least significant word (bits [31:0]) last.

The four 32-bit words of a 16-byte data block must be stored in the memory consecutively and in big-endian order, that is, with the most significant word on the lowest address. See *Table 422* “no swapping” option for details.

Data swapping

The CRYP peripheral can be configured to perform a bit-, a byte-, a half-word-, or no swapping on the input data word in the CRYP input FIFO, before loading it to the AES processing core, and on the data output from the AES processing core, before sending it to the CRYP output FIFO. The choice depends on the type of data. For example, a byte swapping is used for an ASCII text stream.

The data swap type is selected through the DATATYPE[1:0]. The selection applies both to the input and the output of the CRYP, except in GCM/CCM final phase (data must be inserted normally).

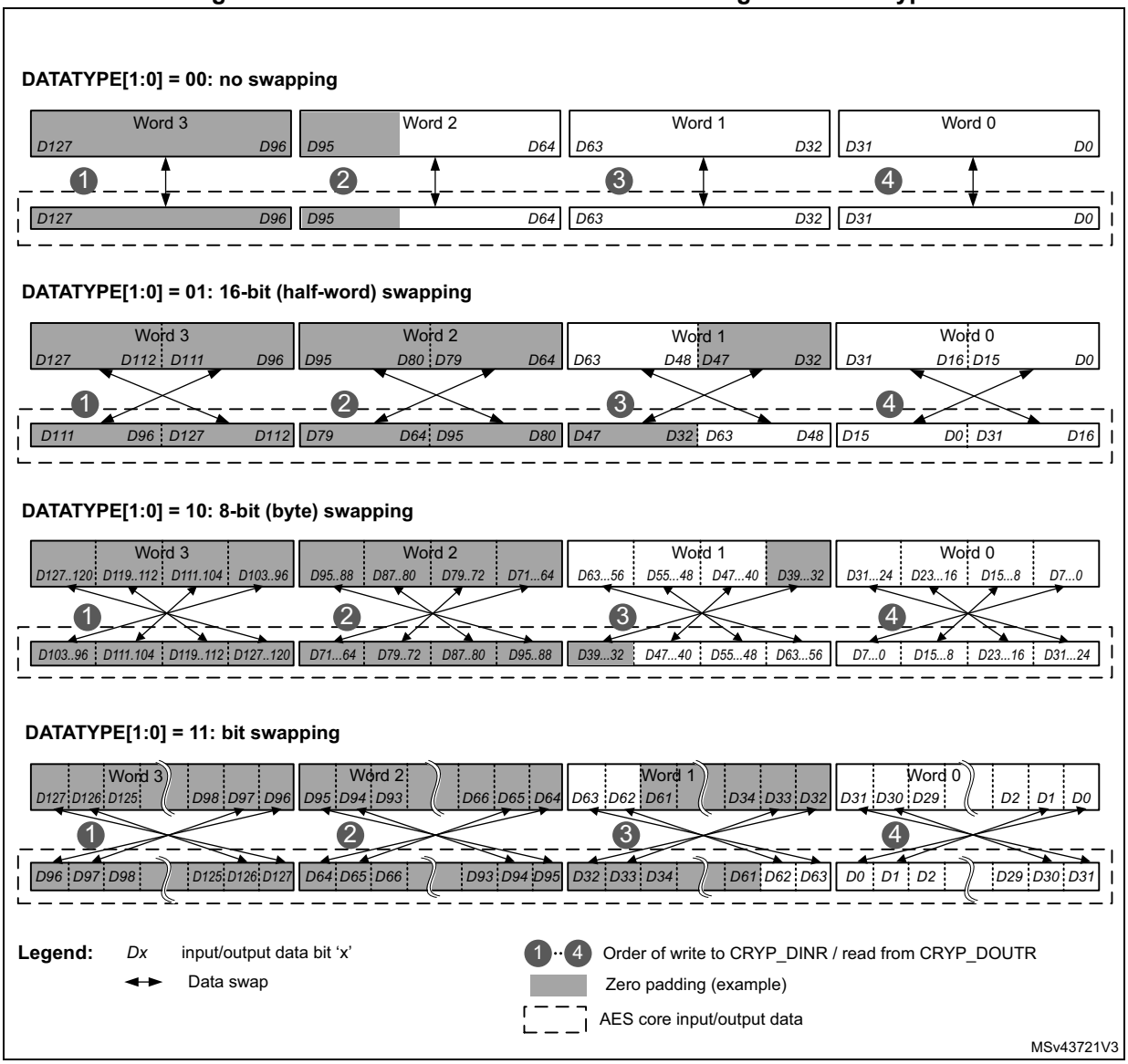
Note: The data in CRYP key registers (CRYP_KxL/R) and initialization vector registers (CRYP_IVxL/R) are not sensitive to the swap mode selection.

The CRYP data swapping feature is summarized in *Table 422* and *Figure 491*.

Table 422. AES data swapping example

DATATYPE[1:0]	Swapping performed	Data block
		System memory data (big-endian)
0x0	No swapping	Block[127..64]: 0x04EEF672 2E04CE96 Block[63..0]: 0x4E6F7720 69732074
		Address @, word[127..96]: 0x04EE F672 Address @ + 0x4, word[95..64]: 0x2E04 CE96 Address @ + 0x8, word[63..32]: 0x4E6F 7720 Address @ + 0xC, word[31..0]: 0x6973 2074
0x1	Half-word (16-bit) swapping	Block[63..0]: 0x**4E6F** 7720 **6973** 2074
		Address @, word[63..32]: 0x7720 **4E6F** Address @ + 0x4, word[31..0]: 0x2074 **6973**
0x2	Byte (8-bit) swapping	Block[63..0]: 0x**4E** 6F **77** 20 **69** 73 **20** 74
		Address @, word[63..32]: 0x20**77** 6F**4E** Address @ + 0x4, word[31..0]: 0x74**20** 73**69**
0x3	Bit swapping	Block[63..32]: 0x4E6F 7720 **0100 1110 0110 1111 0111 0111 0010 0000** Block[31..0]: 0x6973 2074 **0110 1001 0111 0011 0010 0000 0111 0100**
		Address @, word[63..32]: 0x04EE F672 **0000 0100 1110 1110 1111 0110 0111 0010** Address @ + 0x4, word[31..0]: 0x2E04 CE96 **0010 1110 0000 0100 1100 1110 1001 0110**

Figure 491. 128-bit block construction according to the data type



Data padding

Figure 491 also gives an example of memory data block padding with zeros such that the zeroed bits after the data swap form a contiguous zone at the MSB end of the AES core input buffer. The example shows the padding of an input data block containing:

- 84 message bits, with DATATYPE[1:0] = 0x0
- 48 message bits, with DATATYPE[1:0] = 0x1
- 56 message bits, with DATATYPE[1:0] = 0x2
- 34 message bits, with DATATYPE[1:0] = 0x3

49.4.16 CRYP key registers

The eight CRYP_KxR/LR write-only registers store the encryption or decryption key information, as shown in *Table 423*. Reads are not allowed for security reason.

Note: In memory and in CRYP *key registers, keys are stored in big-endian format, with most significant byte on the lowest address.*

Table 423. Key endianness in CRYP_KxR/LR registers (128/192/256-bit keys)

K0LR[31:0]	K0RR[31:0]	K1LR[31:0]	K1RR[31:0]	K2LR[31:0]	K2RR[31:0]	K3LR[31:0]	K3RR[31:0]
-	-	-	-	k[127:96]	k[95:64]	k[63:32]	k[31:0]
K0LR[31:0]	**K0RR[31:0]**	**K1LR[31:0]**	**K1RR[31:0]**	**K2LR[31:0]**	**K2RR[31:0]**	**K3LR[31:0]**	**K3RR[31:0]**
-	-	k[191:160]	k[159:128]	k[127:96]	k[95:64]	k[63:32]	k[31:0]
K0LR[31:0]	**K0RR[31:0]**	**K1LR[31:0]**	**K1RR[31:0]**	**K2LR[31:0]**	**K2RR[31:0]**	**K3LR[31:0]**	**K3RR[31:0]**
k[255:224]	k[223:192]	k[191:160]	k[159:128]	k[127:96]	k[95:64]	k[63:32]	k[31:0]

The key registers are not affected by the data swapping feature controlled by the DATATYPE[1:0] bitfield.

Write operations to the CRYP_KxL/R registers are ignored when BUSY bit is set. The application must check this before modifying key registers.

The entire key must be written before starting an AES computation. In normal key mode (KMOD[1:0] at 0x0), the key registers must always be written in either ascending or descending order. The write sequence becomes (in ascending order):

- K2LR, K2RR, K3LR, then K3RR registers for KEYSIZE = 0x0
- K1LR, K1RR, K2LR, K2RR, K3LR, then K3RR register for KEYSIZE = 0x1
- K0LR, K0RR, K1LR, K1RR, K2LR, K2RR, K3LR, then K3RR register for KEYSIZE = 0x2

Note: KEYSIZE must be written before the key.

As soon as the first key register is written, the KEYVALID bit is cleared. Once the key registers writing sequence is complete, KEYVALID is set and CRYPEN bit becomes writable. If an error occurs, KEYVALID is cleared and KERF set (see *Section 49.4.18*).

49.4.17 CRYP initialization vector registers

The four CRYP_IVxR/LR registers store the initialization vector (IVI) information, as shown in *Table 424*. They can only be written if BUSY bit is cleared.

Note: In memory and in CRYP IVI registers, initialization vectors *are stored in big-endian format, with most significant byte on the lowest address.*

Table 424. Initialization vector endianness in CRYP_IVxR registers (AES)

CRYP_IV0L[31:0]	CRYP_IV0R[31:0]	CRYP_IV1L[31:0]	CRYP_IV1R[31:0]
IVI[127:96]	IVI[95:64]	IVI[63:32]	IVI[31:0]

Initialization vector information depend on the chaining mode selected. When used, CRYP_IVxR/LR registers are updated upon each AES computation cycle (useful for

managing suspend mode).

The initialization vector registers are not affected by the data swapping feature controlled by DATATYPE[1:0].

49.4.18 CRYP error management

The CRYP peripheral manages the errors described in this section.

Key error flag (KERF)

There are multiple sources of errors that set the KERF flag and clear the KEYVALID bit of the CRYP_SR register:

- **Key writing sequence error**: triggered upon detecting an incorrect sequence of writing key registers. See *Section 49.4.16: CRYP key registers* for details.
- **Key sharing size mismatch error**: triggered when KMOD[1:0] is at 0x2 and KEYSIZE in CRYP does not match KEYSIZE in SAES peripheral, or KEYSIZE is at 0x1 in CRYP_CR (192-bit key).
- **Key sharing error**: triggered upon failing transfer of SAES shared key to CRYP. See *Section 49.4.14: AES key sharing with secure AES co-processor* for details.

Upon a key writing sequence error, KERF flag is cleared when a correct key writing sequence starts.

Upon a key sharing error, reset both CRYP and SAES peripherals through the IPRST bit of their corresponding control register, then restart the key sharing sequence.

Note: For any key error, clear KERF flag prior to setting CRYPEN.

49.5 CRYP interrupts

There are two individual maskable interrupt sources generated by the CRYP peripheral to signal the following events:

- Input FIFO empty (IFEM) or not full (IFNF)
- Output FIFO full (OFFU) or not empty (OFNE)

These sources are combined into a common interrupt signal from the CRYP peripheral that connects to the Arm® Cortex® interrupt controller. The application can enable or disable CRYP interrupt sources individually by setting/clearing the corresponding mask bit of the CRYP_IMSCR register. The status of the individual source of event flags can be read from the CRYP_SR register.

The global status of the two source of interrupts can be read either from the CRYP_RISR register (raw, that is unmasked), or from the CRYP_MISR register (masked).

Table 425 gives a summary of the available features.

Table 425. CRYP interrupt requests

Interrupt acronym	Interrupt event	Event flag		Enable bit	Interrupt clear method
		in CRYP_xISR(1)	in CRYP_SR		
CRYP	Output FIFO full	OUTRIS (not masked) OUTMIS (masked)	OFFU	OUTIM(2) and CRYPEN	Read one data from output FIFO
	Output FIFO not empty		OFNE		Read all data from output FIFO
	Input FIFO not full	INRIS (not masked) INMIS (masked)	IFNF	INIM(2) and CRYPEN	Write data until input FIFO is full
	Input FIFO empty		IFEM		Write at least one word in input FIFO

1. Flag of the CRYP_RISR register (unmasked or raw), or CRYP_MISR register (masked).
2. Bit of the CRYP_IMSCR register.

Output FIFO service interrupt (OUTMIS)

The output FIFO service interrupt is asserted when there is one or more (32-bit word) data items in the output FIFO. This interrupt is cleared by reading data from the output FIFO until there is no valid (32-bit) word left (that is, when the interrupt follows the state of the output FIFO not empty flag OFNE).

The output FIFO service interrupt OUTMIS is active also when CRYP is disabled. Consequently, when CRYPEN bit is cleared, the OUTMIS bit is not cleared when the output FIFO is not empty.

Input FIFO service interrupt (INMIS)

The input FIFO service interrupt is asserted when there are less than four words in the input FIFO. It is cleared by performing write operations to the input FIFO until it holds four or more words.

The input FIFO service interrupt INMIS is active only when CRYP is enabled. Consequently, when CRYPEN bit is cleared, the INMIS bit stay low even if the input FIFO is empty.

49.6 CRYP DMA requests

The CRYP peripheral provides an interface to connect to the DMA (direct memory access) controller. The DMA operation is controlled through the DIEN and DOEN bits of the CRYP_DMACR register.

Detailed usage of DMA with CRYP can be found in *Appending data using the DMA* subsection of *Section 49.4.5: CRYP encryption/ decryption typical usage*.

Data input using DMA

Setting DIEN enables DMA writing into CRYP. The CRYP then initiates, during the input phase, a DMA request for each 16-byte data block to write to the CRYP_DINR register (quadruple 32-bit word, MSB first).

The CRYP supports both single and burst DMA transfers. Recommended configuration is fixed burst size of four words.

CRYP issues a DMA transfer request to its input FIFO every time a 16-bytes block has been read from the FIFO.

Note: According to the algorithm and the mode selected, special padding / ciphertext stealing might be required (see Section 49.4.7).

Data output using DMA

Setting DOEN enables DMA reading from CRYP. The CRYP then initiates, during the output phase, a DMA request for each 16-byte data block to read from the CRYP_DOUTR register (quadruple 32-bit word, MSB first).

The CRYP supports both single and burst DMA transfers. Recommended configuration is fixed burst size of four words.

CRYP issues a DMA transfer request from output FIFO every time one 16-byte block has been written into the FIFO.

Note: According to the message size, extra bytes might need to be discarded by application in the last block.

Stopping DMA transfers

All DMA request signals are de-asserted when CRYP is disabled (CRYPEN cleared) or the DMA enable bit (DIEN for input data, DOEN for output data) is cleared.

49.7 CRYP processing time

The following tables provide the 16-byte data block processing latency per operating mode.

Table 426. Processing latency for ECB, CBC and CTR

Key size	Mode of operation	Chaining algorithm	Clock cycles
128 bits	encryption or decryption(1)	ECD, CBC, CTR	14
	key preparation	-	12
192 bits	encryption or decryption(1)	ECD, CBC, CTR	16
	key preparation	-	14
256 bits	encryption or decryption(1)	ECD, CBC, CTR	18
	key preparation	-	16

1. Excluding key preparation time (ECB and CBC only).

Table 427. Processing latency for GCM and CCM (in clock cycles)

Key size	Mode of operation	Chaining algorithm	Initialization phase	Header phase	Payload phase	Final phase
128 bits	encryption or decryption	GCM	24	10	14	14
		CCM	12	14	25	14
192 bits		GCM	28	10	16	16
		CCM	14	16	29	16
256 bits		GCM	32	10	18	18
		CCM	16	18	33	18

49.8 CRYP registers

These registers are accessible through 32-bit word single accesses only (otherwise an AHB error is generated, and write accesses are ignored).

49.8.1 CRYP control register (CRYP_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IPRST	Res.	Res.	Res.	Res.	Res.	KMOD[1:0]		NPBLB[3:0]				ALGOMODE[3]	Res.	GCM_CCMPH[1:0]	
rw						rw	rw	rw	rw	rw	rw	rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRYPEN	FFLUSH	Res.	Res.	Res.	Res.	KEYSIZE[1:0]		DATATYPE[1:0]		ALGOMODE[2:0]			ALGODIR	Res.	Res.
rw	rw					rw	rw	rw	rw	rw	rw	rw	rw		

Bit 31 **IPRST**: CRYP peripheral software reset

Setting the bit resets the CRYP peripheral, putting all registers to their default values, except the IPRST bit itself.

This bit must be kept cleared while writing any configuration registers.

Bits 30:26 Reserved, must be kept at reset value.

Bits 25:24 **KMOD[1:0]**: Key mode selection

This bitfield defines how the CRYP key can be used by the application. KEYSIZE must be correctly initialized when setting KMOD[1:0] different from zero.

0x0: Normal-key mode. Key registers are freely usable.

0x2: Shared-key mode. If shared-key mode is properly initialized in SAES peripheral, the CRYP peripheral automatically loads its key registers with the data stored in the SAES key registers. The key value is available in CRYP key registers when BUSY bit is cleared and KEYVALID is set in the CRYP_SR register. Key error flag KERF is set otherwise in the CRYP_SR register.

Others: Reserved

Attempts to write the bitfield are ignored when BUSY is set.

Bits 23:20 **NPBLB[3:0]:** Number of padding bytes in last block
This padding information must be filled by software before processing the last block of GCM payload encryption or CCM payload decryption, otherwise authentication tag computation is incorrect.
0x0: All bytes are valid (no padding)
0x1: Padding for the last LSB byte
...
0xF: Padding for the 15 LSB bytes of last block.
Attempts to write the bitfield are ignored when BUSY is set.

Bit 18 Reserved, must be kept at reset value.

Bits 17:16 **GCM_CCMPH[1:0]**: GCM or CCM phase selection
This bitfield selects the phase, applicable only with GCM, GMAC or CCM chaining modes.
0x0: Initialization phase
0x1: Header phase
0x2: Payload phase
0x3: Final phase
Attempts to write the bitfield are ignored when BUSY is set.

Bit 15 **CRYPEN:** CRYP enable
This bit enables/disables the CRYP peripheral.
0:CRYP disabled
1: CRYP enabled
This bit is automatically cleared by hardware upon the completion of the key preparation (ALGOMODE[3:0] at 0x7) and upon the completion of GCM/GMAC/CCM initialization phase.
The bit cannot be set as long as KEYVALID is cleared.

Bit 14 **FFLUSH:** FIFO flush
This bit enables/disables the flushing of CRYP input and output FIFOs. Reading this bit always returns 0.
0: No effect
1: FIFO flush enabled
When CRYPEN is cleared, writing this bit to 1 flushes both input and output FIFOs (that is read and write pointers of the FIFOs are reset).
FFLUSH bit must be set when BUSY is cleared, otherwise the FIFO is flushed, but the block being processed may be pushed into the output FIFO just after the flush operation, resulting in a non-empty FIFO condition.
Attempts to write FFLUSH are ignored when CRYPEN is set.

Bits 13:10 Reserved, must be kept at reset value.

Bits 9:8 **KEYSIZE[1:0]**: Key size selection
This bitfield defines the key length in bits of the key used by CRYP.
0x0: 128-bits
0x1: 192 bits
0x2: 256 bits
0x3: Reserved
When KEYSIZE is changed, KEYVALID bit is cleared.
Attempts to write the bitfield are ignored when BUSY is set.

Bits 7:6 **DATATYPE[1:0]**: Data type

This bitfield defines the format of data written in the CRYP_DINR register or read from the CRYP_DOUTR register, through selecting the mode of data swapping. This swapping is defined in *Section 49.4.15: CRYP data registers and data swapping*.

0x0: No swapping (32-bit data).

0x1: Half-word swapping (16-bit data).

0x2: Byte swapping (8-bit data).

0x3: Bit-level swapping.

Attempts to write the bitfield are ignored when BUSY is set.

Bits 19, 5:3 **ALGOMODE[3:0]:** Algorithm mode

This bitfield selects the AES algorithm/chaining mode.

0x4: Electronic codebook (ECB)

0x5: Cipher Block Chaining (CBC)

0x6: Counter mode (CTR)

0x7: AES key preparation for ECB or CBC decryption

0x8: Galois counter mode (GCM) and Galois message authentication code (GMAC)

0x9: Counter with CBC-MAC (CCM)

Others: Reserved

Attempts to write the bitfield are ignored when BUSY is set.

Bit 2 **ALGODIR:** Algorithm direction

This bit selects the algorithm direction.

0: Encryption

1: Decryption

Attempts to write the bitfield are ignored when BUSY is set.

Bits 1:0 Reserved, must be kept at reset value.

49.8.2 CRYP status register (CRYP_SR)

Address offset: 0x04

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY VALID	KERF	Res.	BUSY	OFFU	OFNE	IFNF	IFEM
								r	r		r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **KEYVALID**: Key valid flag

This read-only bit is set by hardware when the key of size defined by KEYSIZE is loaded in CRYP_KxR/LR key registers.

0: Key not valid

1: Key valid

The CRYPEN bit can only be set when KEYVALID is set.

In normal mode when KMOD[1:0] is at zero, the key must be written in the key registers in the correct sequence, otherwise the KERF flag is set and KEYVALID remains cleared.

When KMOD[1:0] is different from zero, the BUSY flag is automatically set by CRYP. When the key is loaded successfully, BUSY is cleared and KEYVALID set. Upon an error, KERF is set, BUSY cleared and KEYVALID remains cleared.

If set, KERF must be cleared, otherwise KEYVALID cannot be set. For further information on key loading, refer to *Section 49.4.16: CRYP key registers*.

Bit 6 **KERF**: Key error flag

This read-only bit is set by hardware when key information failed to load into key registers.

0: No key error detected

1: Key information failed to load into key registers

KERF is triggered upon any of the following errors:

– CRYP_KxR/LR register write does not respect the correct order (refer to *Section 49.4.16: CRYP key registers* for details).

– CRYP fails to load the key shared by SAES peripheral (KMOD = 0x2).

KERF must be cleared by the application software, otherwise KEYVALID cannot be set. It can be done through IPRST bit of CRYP_CR, or when a correct key writing sequence starts.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **BUSY:** Busy bit

This flag indicates whether CRYP is idle or busy.

0: Idle

1: Busy

CRYP is flagged as idle when disabled (CRYPEN = 0) or when the AES core is not processing any data. It happens when the last processing has completed, or CRYP is waiting for enough data in the input FIFO or enough free space in the output FIFO (that is in each case at least 4 words).

CRYP is flagged as busy when processing a block data, preparing a key (ECB or CBC decryption only), or transferring a shared key from SAES peripheral.

Bit 3 **OFFU:** Output FIFO full flag

0: Output FIFO is not full

1: Output FIFO is full

Bit 2 **OFNE:** Output FIFO not empty flag

0: Output FIFO is empty

1: Output FIFO is not empty

Bit 1 **IFNF:** Input FIFO not full flag

0: Input FIFO is full

1: Input FIFO is not full

Bit 0 **IFEM:** Input FIFO empty flag

0: Input FIFO is not empty

1: Input FIFO is empty

49.8.3 CRYP data input register (CRYP_DINR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DIN[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIN[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DIN[31:0]:** Data input

A four-fold sequential write to this bitfield during the Input phase results in pushing a complete 16-byte block into the CRYP input FIFO. From the first to the fourth write, the corresponding data weights are [127:96], [95:64], [63:32], and [31:0].

Input FIFO can receive up to two 16-byte blocks of plaintext (when encrypting) or ciphertext (when decrypting).

If EN bit is set in CRYP_CR register, when at least four 32-bit words have been pushed into the input FIFO, and when at least four 32-bit words are free in the output FIFO, the CRYP automatically starts an encryption or decryption process, setting the BUSY bit.

Reading this register pops data off the input FIFO (oldest value is returned). The data present in the input FIFO are returned only if CRYPEN is cleared (undefined value is returned otherwise). Following one or more reads the FIFO must be flushed (setting the FFLUSH bit) prior to processing new data.

49.8.4 CRYP data output register (CRYP_DOUTR)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DOUT[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DOUT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **DOUT[31:0]:** Data output

A four-fold sequential read to this bitfield during the output phase results in retrieving a complete 16-byte block from the CRYP output FIFO. From the first to the fourth read, the corresponding data weights are [127:96], [95:64], [63:32], and [31:0].

Output FIFO can store up to two 16-byte blocks of plaintext (when decrypting) or ciphertext (when encrypting).

When the output FIFO is empty a read returns an undefined value. Writes are ignored.

49.8.5 CRYP DMA control register (CRYP_DMACR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DOEN	DIEN
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **DOEN:** DMA output enable

When this bit is set, DMA requests are automatically generated by the peripheral during the output data phase.

0: Outgoing data transfer from CRYP via DMA is disabled

1: Outgoing data transfer from CRYP via DMA is enabled

Bit 0 **DIEN:** DMA input enable

When this bit is set, DMA requests are automatically generated by the peripheral during the input data phase.

0: Incoming data transfer to CRYP via DMA is disabled

1: Incoming data transfer to CRYP via DMA is enabled

49.8.6 CRYP interrupt mask set/clear register (CRYP_IMSCR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OUTIM	INIM
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **OUTIM:** Output FIFO service interrupt mask

This bit enables or disables (masks) the CRYP output FIFO service interrupt generation when OUTRIS is set.

0: Output FIFO interrupt is disabled (masked), masked interrupt status (OUTMIS) stays cleared

1: Output FIFO interrupt is enabled (not masked)

Bit 0 **INIM:** Input FIFO service interrupt mask

This bit enables or disables (masks) the CRYP input FIFO service interrupt generation when INRIS is set.

0: Input FIFO interrupt is disabled (masked), masked interrupt status (INMIS) stays cleared

1: Input FIFO interrupt is enabled (not masked)

49.8.7 CRYP raw interrupt status register (CRYP_RISR)

Address offset: 0x18

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OUTRIS	INRIS
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **OUTRIS:** Output FIFO service raw interrupt status

This read-only bit is set by hardware when an output FIFO flag (OFFU or OFNE) is set in CRYP_SR register, regardless of the OUTIM mask bit value in CRYP_IMSCR register.

0: No output FIFO event detected

1: Output FIFO full or not empty detected; an interrupt is generated if OUTIM bit is set in CRYP_IMSCR register

Bit 0 **INRIS:** Input FIFO service raw interrupt status

This read-only bit is set by hardware when an input FIFO flag (IFNF or IFEM) is set in CRYP_SR register, regardless of the INIM mask bit value in CRYP_IMSCR register.

0: No input FIFO event detected

1: Input FIFO empty or not full detected; an interrupt is generated if CRYPEN is set and if INIM bit is set in CRYP_IMSCR register

49.8.8 CRYP masked interrupt status register (CRYP_MISR)

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OUTMIS	INMIS
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **OUTMIS:** Output FIFO service masked interrupt status

This read-only bit is set by hardware when an output FIFO flag (OFFU or OFNE) is set in CRYP_SR register. If the OUTIM mask bit is cleared in CRYP_IMSCR register, the OUTMIS bit stays cleared (masked).

0: No output FIFO event detected or OUTIM mask cleared in CRYP_IMSCR

1: Output FIFO full or not empty detected, with an interrupt pending

The OUTMIS bit is cleared by reading data from the output FIFO until OFNE flag is cleared (output FIFO empty). It is not cleared by disabling CRYP with CRYPEN bit.

Bit 0 **INMIS:** Input FIFO service masked interrupt status

This read-only bit is set by hardware when an input FIFO flag (IFNF or IFEM) is set in CRYP_SR register. If the INIM mask bit is cleared in CRYP_IMSCR register, the INMIS bit stays cleared (masked).

0: No input FIFO event detected or INIM mask cleared in CRYP_IMSCR or CRYPEN bit cleared.

1: Input FIFO empty or not full detected, with an interrupt pending

The INMIS bit is cleared by writing data to the input FIFO until IFEM flag is cleared (there is at least one word in input FIFO), or by clearing CRYPEN,

When CRYP is disabled, INMIS bit stays low even if the input FIFO is empty.

49.8.9 CRYP key register 0L (CRYP_K0LR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
K[255:240]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
K[239:224]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **K[255:224]**: Key bit x (x = 255 to 224)

This write-only bitfield contains the bits [255:224] of the AES encryption or decryption key, depending on the operating mode.

Write to CRYP_KxR/LR registers is ignored when CRYP is busy (BUSY bit set). When key is coming from the SAES peripheral (KMOD[1:0] = 0x2), write is also ignored. With KMOD[1:0] at 0x0, a special writing sequence is required. In this sequence, any valid write to CRYP_KxR/LR register clears the KEYVALID flag except for the sequence-completing write that sets it. Also refer to the description of the KEYVALID flag in the CRYP_SR register.

49.8.10 CRYP key register 0R (CRYP_K0RR)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
K[223:208]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
K[207:192]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **K[223:192]**: Key bit x (x = 223 to 192)

This write-only bitfield contains the bits [223:192] of the AES encryption or decryption key, depending on the operating mode.

Refer to the CRYP_K0LR register for information relative to writing CRYP_KxR/LR registers.

49.8.11 CRYP key register 1L (CRYP_K1LR)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
K[191:176]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
K[175:160]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **K[191:160]**: Key bit x (x = 191 to 160)

This write-only bitfield contains the bits [191:160] of the AES encryption or decryption key, depending on the operating mode.

Refer to the CRYP_K0LR register for information relative to writing CRYP_KxR/LR registers.

49.8.12 CRYP key register 1R (CRYP_K1RR)

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
K[159:144]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
K[143:128]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **K[159:128]**: Key bit x (x = 159 to 128)

This write-only bitfield contains the bits [159:128] of the AES encryption or decryption key, depending on the operating mode.

Refer to the CRYP_K0LR register for information relative to writing CRYP_KxR/LR registers.

49.8.13 CRYP key register 2L (CRYP_K2LR)

Address offset: 0x30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
K[127:112]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
K[111:96]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **K[127:96]**: Key bit x (x = 127 to 96)

This write-only bitfield contains the bits [127:96] of the AES encryption or decryption key, depending on the operating mode.

Refer to the CRYP_K0LR register for information relative to writing CRYP_KxR/LR registers.

49.8.14 CRYP key register 2R (CRYP_K2RR)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
K[95:80]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
K[79:64]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **K[95:64]**: Key bit x (x = 95 to 64)

This write-only bitfield contains the bits [95:64] of the AES encryption or decryption key, depending on the operating mode.

Refer to the CRYP_K0LR register for information relative to writing CRYP_KxR/LR registers.

49.8.15 CRYP key register 3L (CRYP_K3LR)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
K[63:48]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
K[47:32]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **K[63:32]**: Key bit x (x = 63 to 32)

This write-only bitfield contains the bits [63:32] of the AES encryption or decryption key, depending on the operating mode.

Refer to the CRYP_K0LR register for information relative to writing CRYP_KxR/LR registers.

49.8.16 CRYP key register 3R (CRYP_K3RR)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
K[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
K[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **K[31:0]**: Key bit x (x = 31 to 0)

This write-only bitfield contains the bits [31:0] of the AES encryption or decryption key, depending on the operating mode.

Refer to the CRYP_K0LR register for information relative to writing CRYP_KxR/LR registers.

49.8.17 CRYP initialization vector register 0L (CRYP_IV0LR)

Address offset: 0x40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[127:112]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[111:96]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[127:96]**: Initialization vector bit x (x = 127 to 96)

This bitfield stores the initialization vector bits [127:96] for AES chaining modes other than ECB.

The value stored in CRYP_IVxR/LR registers is updated by hardware after each computation round (when applicable).

Write to this register is ignored when CRYP is busy (BUSY bit set).

49.8.18 CRYP initialization vector register 0R (CRYP_IV0RR)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[95:80]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[79:64]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[95:64]**: Initialization vector bit x (x = 95 to 64)

This bitfield stores the initialization vector bits [95:64] for AES chaining modes other than ECB.

The value stored in CRYP_IVxR/LR registers is updated by hardware after each computation round (when applicable).

Write to this register is ignored when CRYP is busy (BUSY bit set).

49.8.19 CRYP initialization vector register 1L (CRYP_IV1LR)

Address offset: 0x48

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[63:48]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[47:32]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[63:32]**: Initialization vector bit x (x = 63 to 32)

This bitfield stores the initialization vector bits [63:32] for AES chaining modes other than ECB.

The value stored in CRYP_IVxR/LR registers is updated by hardware after each computation round (when applicable).

Write to this register is ignored when CRYP is busy (BUSY bit set).

49.8.20 CRYP initialization vector register 1R (CRYP_IV1RR)

Address offset: 0x4C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IVI[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IVI[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **IVI[31:0]**: Initialization vector bit x (x = 31 to 0)

This bitfield stores the initialization vector bits [31:0] for AES chaining modes other than ECB. The value stored in CRYP_IVxR/LR registers is updated by hardware after each computation round (when applicable).

Write to this register is ignored when CRYP is busy (BUSY bit set).

49.8.21 CRYP context swap GCM-CCM registers (CRYP_CSGCMCCMxR)

Address offset: 0x50 + 0x4 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CSGCMCCM[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSGCMCCM[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CSGCMCCM[31:0]**: Context swap for GCM/GMAC and CCM modes

CRYP_CSGCMCCMxR registers contain the complete internal register states of the CRYP when the GCM, GMAC or CCM processing of the current task is suspended to process a higher-priority task. Refer to *Section 49.4.8: CRYP suspend and resume operations* for more details.

CRYP_CSGCMCCMxR registers are not used in other chaining modes than GCM, GMAC or CCM.

49.8.22 CRYP context swap GCM registers (CRYP_CSGCMxR)

Address offset: 0x70 + 0x4 * x, (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CSGCM[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSGCM[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CSGCM[31:0]**: Context swap for GCM/GMAC modes

CRYP_CSGCMxR registers contain the complete internal register states of the CRYP when the GCM or GMAC processing of the current task is suspended to process a higher-priority task. Refer to *Section 49.4.8: CRYP suspend and resume operations* for more details.

CRYP_CSGCMxR registers are not used in other chaining modes than GCM or GMAC.

49.8.23 CRYP register map

Table 428. CRYP register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	CRYP_CR	IPRST	Res.	Res.	Res.	Res.	Res.	KMOD[1:0]		NPBLB[3:0]				ALGOMODE[3]	Res.	GCM_CCMPH[1:0]		CRYPEN	FFLUSH	Res.	Res.	Res.	Res.	KEYSIZE[1:0]		DATATYPE[1:0]		ALGOMODE[2:0]			ALGODIR	Res.	Res.
	Reset value	0						0	0	0	0	0	0	0		0	0	0	0					0	0	0	0	0	0	0	0		
0x04	CRYP_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEYVALID	KERF	Res.	BUSY	OFFU	OFNE	IFNF	IFEM
	Reset value																									0	0		0	0	0	1	1
0x08	CRYP_DINR	DIN[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	CRYP_DOUTR	DOUT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	CRYP_DMACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DOEN	DIEN
	Reset value																															0	0
0x14	CRYP_IMSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OUTIM	INIM
	Reset value																															0	0

Table 428. CRYP register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x18	CRYP_RISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OUTRIS	INRIS
	Reset value																															0	1
0x1C	CRYP_MISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OUTMIS	INMIS
	Reset value																															0	0
0x20	CRYP_K0LR	K[255:224]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x24	CRYP_K0RR	K[223:192]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...																																	
0x38	CRYP_K3LR	K[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x3C	CRYP_K3RR	K[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	CRYP_IV0LR	IVI[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	CRYP_IV0RR	IVI[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	CRYP_IV1LR	IVI[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C	CRYP_IV1RR	IVI[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50 + x * 0x4, (x = 0 to 7), last address 0x6C	CRYP_ CSGCMCCMxR	CSGCMCCMx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x70 + x* 0x4, (x = 0 to 7), last address 0x8C	CRYP_ CSGCMxR	CSGCMx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

50 Hash processor (HASH)

50.1 HASH introduction

The hash processor is a fully compliant implementation of the secure hash algorithm (SHA-1, SHA-2 family) and the HMAC (keyed-hash message authentication code) algorithm. HMAC is suitable for applications requiring message authentication.

The hash processor computes FIPS (Federal Information Processing Standards) approved digests of length of 160, 224, 256 bits, for messages of any length less than 2^{64} bits (for SHA-1, SHA-224 and SHA-256) or less than 2^{128} bits (for SHA-384, SHA-512).

50.2 HASH main features

- Suitable for data authentication applications, compliant with:
  - Federal Information Processing Standards Publication FIPS PUB 180-4, *Secure Hash Standard* (SHA-1 and SHA-2 family)
  - Federal Information Processing Standards Publication FIPS PUB 186-4, *Digital Signature Standard (DSS)*
  - Internet Engineering Task Force (IETF) Request For Comments RFC 2104, *HMAC: Keyed-Hashing for Message Authentication* and Federal Information Processing Standards Publication FIPS PUB 198-1, *The Keyed-Hash Message Authentication Code (HMAC)*
- Fast computation of SHA-1, SHA2-224, SHA2-256, SHA2-384, and SHA2-512
  - 82 (respectively 66) clock cycles for processing one 512-bit block of data using SHA-1 (respectively SHA-256) algorithm
  - 98 clock cycles for processing one 1024-bit block of data using either SHA2-384 or SHA2-512 algorithm
  - Support for SHA-2 truncated outputs (SHA2-512/224, SHA2-512/256)
- Support for HMAC mode with all supported algorithm
- Corresponding 32-bit words of the digest from consecutive message blocks are added to each other to form the digest of the whole message
  - Automatic 32-bit words swapping to comply with the internal little-endian representation of the input bit-string
  - Supported word swapping format: bits, bytes, half-words, and 32-bit words
- Single 32-bit, write-only, input register associated to an internal input FIFO that can receive a single data block of size depending on the selected algorithm
- Automatic padding to complete the input bit string to fit digest minimum block size
- AMBA AHB target peripheral, accessible through 32-bit word single accesses only (otherwise an AHB bus error is generated, and write accesses are ignored)
- 8 × 32-bit words (H0 to H15) for output message digest
- Automatic data flow control supporting direct memory access (DMA) using one channel.
- Support for both single and fixed DMA burst transfers of four words.
- Interruptible message digest computation, on a per-block basis
  - Reloadable digest registers

– Hashing computation suspend/resume mechanism, including DMA

50.3 HASH implementation

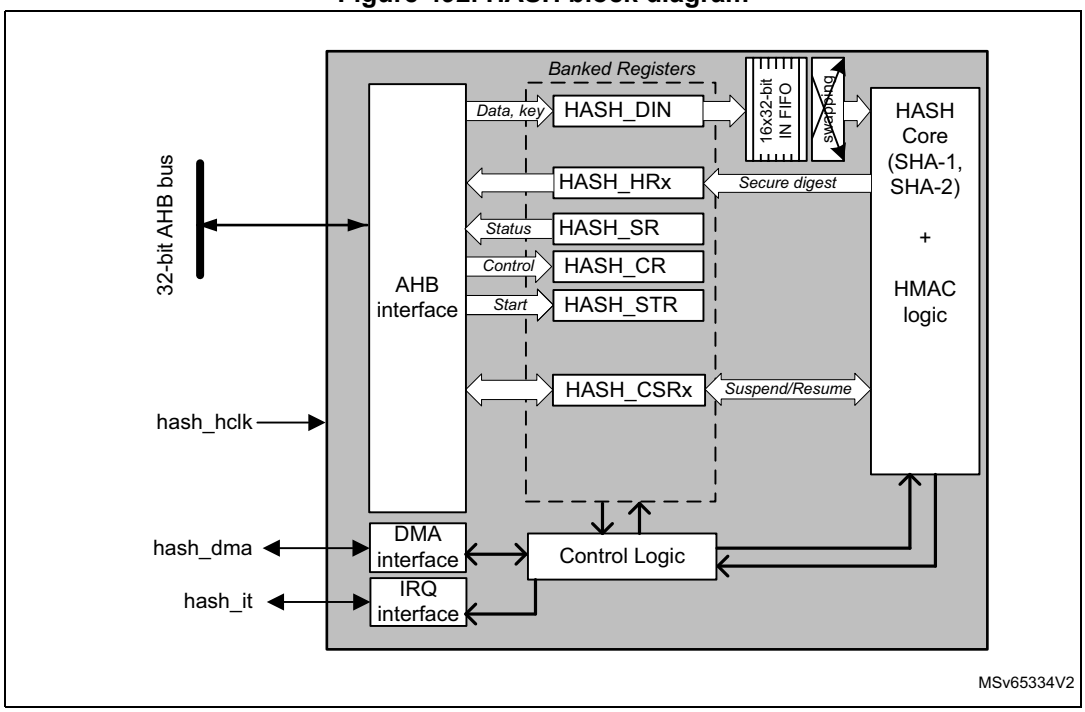
The devices have a single instance of HASH peripheral.

50.4 HASH functional description

50.4.1 HASH block diagram

Figure 492 shows the block diagram of the hash processor.

Figure 492. HASH block diagram



50.4.2 HASH internal signals

Table 429 describes a list of useful to know internal signals available at HASH level, not at product level (on pads).

Table 429. HASH internal input/output signals

Signal name	Signal type	Description
hash_hclk	digital input	AHB bus clock
hash_it	digital output	Hash processor global interrupt request
hash_dma	digital input/output	DMA burst request/ acknowledge

50.4.3 About secure hash algorithms

The hash processor is a fully compliant implementation of the secure hash algorithm defined by FIPS PUB 180-4 standard.

With each algorithm, the HASH computes a condensed representation of a message or data file. More specifically, when a message is presented on the input, the HASH processing core produces a fixed-length output string called a message digest (see *Table 430*).

Table 430. Information on supported hash algorithms

Algorithm	Message digest size (in bits)	Block size (in bytes)(1)	Message length	Bit string message
SHA-1	160	64	$< 2^{64}$ bits	Yes
SHA2-224	224			
SHA2-256	256			
SHA2-384	384	128	$< 2^{128}$ bits	Yes
SHA2-512	512(2)			

1. Block size = (NBWE-1) * 4 bytes. NBWE[4:0] bitfield can be read from the HASH_SR register after ALGO and INIT bits are written in the HASH_CR register.
2. Digest size is 224 bits for SHA2-512/224 and 256 bits for SHA2-512/256 (truncated modes).

The message digest can then be processed with a digital signature algorithm in order to generate or verify the signature for the message.

Signing the message digest rather than the message often improves the efficiency of the process since the message digest is usually much smaller in size than the message. The verifier of a digital signature has to use the same hash algorithm as the one used by the creator of the digital signature.

The SHA-2 functions supported by the hash processor are qualified as “secure” by NIST because it is computationally infeasible to find a message that corresponds to a given message digest, or to find two different messages that produce the same message digest (SHA-1 does not qualify as secure since February 2017).

50.4.4 Message data feeding

The message (or data file) to be processed by the HASH must be considered as a bit string. Per FIPS PUB 180-4 standard this message bit string grows from left to right, with hexadecimal words expressed in “big-endian” convention, so that within each word, the most significant bit is stored in the left-most bit position. For example message string “abc” with a bit string representation of “**01100001 01100010 01100011**” is represented by a 32-bit word **0x00636261**, and 8-bit words **0x61626300**.

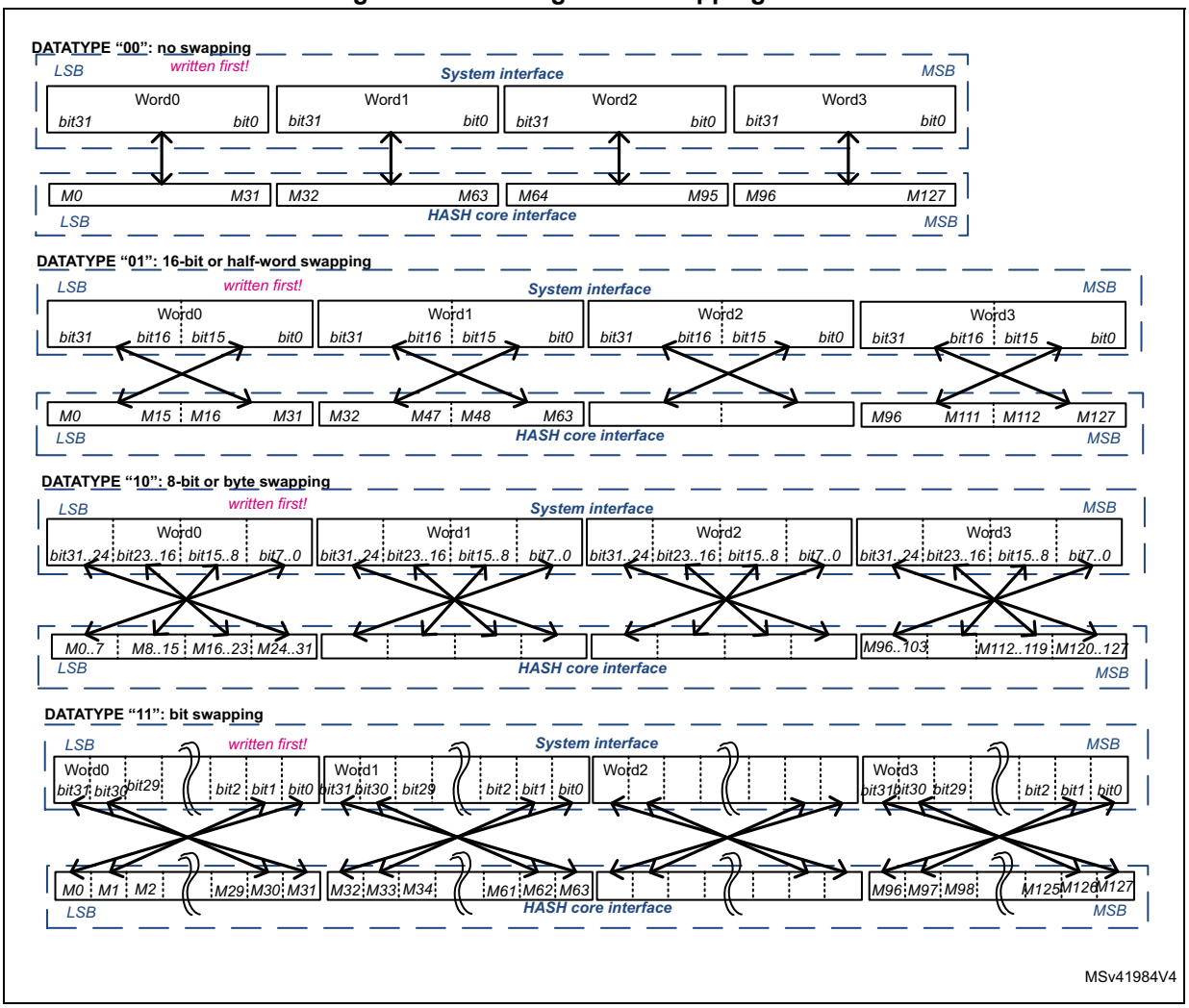
Data are entered into the HASH one 32-bit word at a time, by writing them into the HASH_DIN register. The current contents of the HASH_DIN register are transferred to the 16 words input FIFO each time the register is written with new data. Hence, HASH_DIN and the FIFO form a seventeen 32-bit words length FIFO (named the IN buffer).

In accordance to the kind of data to be processed (for example byte swapping when data are ASCII text stream) there must be a bit, byte, half-word, or no swapping operation to be performed on data from the input FIFO before entering the little-endian hash processing core. *Figure 493* shows how the hash processing core 32-bit data block M0...31 is

constructed from one 32-bit words popped into input FIFO by the driver, according to the DATATYPE bitfield in the HASH control register (HASH_CR)

HASH_DIN data endianness when bit swapping is disabled (DATATYPE = 00) can be described as following: the least significant bit of the message has to be at MSB position in the first word entered into the hash processor, the 32nd bit of the bit string has to be at MSB position in the second word entered into the hash processor and so on.

Figure 493. Message data swapping feature



50.4.5 Message digest computing

The hash processor sequentially processes several blocks when computing the message digest. Block sizes can be found in *Table 430: Information on supported hash algorithms*.

Each time the DMA or the CPU writes a block to the hash processor, the HASH automatically starts computing the message digest. This operation is known as partial digest computation.

As described in *Section 50.4.4: Message data feeding*, the message to be processed is entered into the HASH 32-bit word at a time, writing to the HASH_DIN register to fill the

input FIFO. In order to perform the hash computation on this data the application must follow below sequence.

1. Initialize the hash processor using the HASH_CR register:
   - Write to the HASH_CR register to select the right algorithm using the ALGO bitfield, and set the INIT bit (other bits are kept at zero). Then read the NBWE bitfield from the HASH_SR register to deduce the algorithm block size, which equals **`(NBWE-1) * 4`**. This step is not required if the block size is already known (see *Table 430* for details).
   - Select the right algorithm using the ALGO[3:0] field. If needed, program the correct swapping operation on the message input words using the DATATYPE[1:0] bitfield.
   - When HMAC mode is required, set the MODE bit as well as the LKEY bit if the HMAC key size is greater than the known block size of the algorithm (otherwise keep LKEY cleared). Refer to *Section 50.4.7: HMAC operation* for details.
   - Update NBLW[4:0] in the HASH_STR register to define the number of valid bits in the last word of the message if it is different from 32 bits. NBLW information is used to correctly perform the automatic message padding before the final message digest computation.
2. Complete the initialization by setting the INIT bit in HASH_CR register. Also set the DMAE bit if data are transferred via DMA.

Caution: When programming step 2, it is important that the correct configuration values (ALGO, DATATYPE, HMAC mode, key length, NBLW) are set up before or at the same time.

3. Start filling data by writing to the HASH_DIN register, unless data are automatically transferred via DMA. Note that the processing of a block can start only once the last value of the block has entered the input FIFO. The way the partial or final digest computation is managed depends on the way data are fed into the processor:
   - Data are filled by software:

     Partial digest computations are triggered each time the application writes the first word of the next block, the block size being defined by NBWE bits in HASH_SR. Once the processor is ready again (DINIS = 1 in HASH_SR), the software can write new data to HASH_DIN. This mechanism avoids the introduction of wait states by the HASH.

     The final digest computation is triggered when the last block is entered and the software sets the DCAL bit. If the message length is not an exact multiple of the block size, the NBLW field in HASH_STR register must be written prior to writing DCAL bit (see *Section 50.4.6* for details).
   - Data are filled as a single DMA transfer (MDMAT = 0):

     Partial digest computations are triggered automatically each time the FIFO is full.

     The final digest computation is triggered automatically when the last block has been transferred to the HASH_DIN register by DMA (DCAL bit is set by hardware). If the message length is not an exact multiple of the block size, the NBLW field in HASH_STR register must be written prior to enabling the DMA (see *Section 50.4.6* for details).
   - Data are filled using multiple DMA transfers (MDMAT = 1):

     Partial digest computations are triggered as for single DMA transfers (refer to the above description). However, the final digest computation is not triggered automatically when the last block has been transferred by DMA to the HASH_DIN register (DCAL bit is not set by hardware). It enables the hash processor to

receive a new DMA transfer as part of this digest computation. To launch the final digest computation, the software must clear MDMAT bit before the last DMA transfer in order to trigger the final digest computation as it is done for single DMA transfers.

4. Once the digest calculation is completed (DCIS = 1), the resulting digest can be read from the output registers, as described in *Table 431*. When set, the DMAEN bit is automatically cleared by hardware.

Table 431. Hash processor outputs

Algorithm	Valid output registers	Most significant bit	Digest size (in bits)
SHA-1	HASH_H0 to HASH_H4	HASH_H0[31]	160
SHA2-224	HASH_H0 to HASH_H6	HASH_H0[31]	224
SHA2-256	HASH_H0 to HASH_H7		256
SHA2-384	HASH_H0 to HASH_H11	HASH_H0[31]	384
SHA2-512	HASH_H0 to HASH_H15		512(1)

1. Digest size is 224 bits for SHA2-512/224 and 256 bits for SHA2-512/256 (truncated modes)

For more information about HMAC detailed instructions, refer to *Section 50.4.7: HMAC operation*.

Aborting HASH operation

HASH can be disabled at any time by writing 0x4 to HASH_CR, after stopping any running DMA transfer to HASH (if applicable).

50.4.6 Message padding

Overview

When computing a condensed representation of a message, the process of feeding data into the hash processor (with automatic partial digest computation every block size transfer) loops until the last bits of the original message are written to the HASH_DIN register.

As the length (number of bits) of a message can be any integer value, the last word written to the hash processor may have a valid number of bits between 1 and 32. This number of valid bits in the last word, NBLW, has to be written to the HASH_STR register, so that message padding is correctly performed before the final message digest computation.

Padding processing

Detailed hashing sequences with padding are described in *Section 50.4.5: Message digest computing*. As mentioned in these sequences, the padding is done automatically using the information provided in NBLW[4:0].

Hashing example

Below is an example of SHA-1 digest computation with HASH. Message processing and padding are defined in Federal Information Processing Standards PUB 180-4.

Let us assume that the original message is the ASCII binary-coded form of “abc”, of length L = 24 (‘**U**’ means don’t care):

```
byte 0    byte 1   byte 2   byte 3
01100001 01100010 01100011 UUUUUUUU
<-- 1st word written to HASH_DIN -->
```

NBLW has to be loaded with the value 24: a “**1**” is appended at bit location 24 in the bit string (starting counting from left to right in the above bit string), which corresponds to bit 31 in the HASH_DIN register (little-endian convention):

```
01100001 01100010 01100011 1UUUUUUU
```

Since L = 24, the number of bits in the above bit string is 25, and 423 “0” bits are automatically appended, making now 448 bits.

This gives in hexadecimal (byte words in big-endian format):

```
61626380 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000
```

The message length value, **L**, in two-word format (that is 00000000 00000018) is appended. Hence, the final padded message in hexadecimal (byte words in big-endian format):

```
61626380 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000018
```

If the hash processor is programmed to swap byte within HASH_DIN input register (DATATYPE = 10 in HASH_CR), the above message has to be entered using following sequence:

1. **0xUU636261** is written to the HASH_DIN register (where ‘**U**’ means don’t care)
2. **0x18** is written to the HASH_STR register (the number of valid bits in the last word written to the HASH_DIN register is 24, as the original message length is 24 bits)
3. **0x100** is written to the HASH_STR register to run the final digest computation.
4. The hash computing is complete with the message digest available in the HASH_HRx registers (x = 0 to 4) for the SHA-1 algorithm. For this FIPS example, the expected value is as follows:

```
HASH_HR0 = 0xA9993E36
HASH_HR1 = 0x4706816A
HASH_HR2 = 0xBA3E2571
HASH_HR3 = 0x7850C26C
HASH_HR4 = 0x9CD0D89D
```

50.4.7 HMAC operation

Overview

As specified by Internet Engineering Task Force RFC2104 and NIST FIPS PUB 198-1, the HMAC algorithm is used for message authentication by irreversibly binding the message being processed to a key chosen by the user. The algorithm consists of two nested hash operations:

```
HMAC(message)= Hash((Key | pad) XOR opad |
               Hash((Key | pad) XOR ipad | message))
```

Where:

- **opad** = $[0x5C]_n$ (outer pad) and **ipad** = $[0x36]_n$ (inner pad)
- $[X]_n$ represents a repetition of X *n* times, where *n* equal to the byte size of the underlying hash function data block (n = 64 when block size is 512 bits).
- **pad** is a sequence of zeroes needed to extend the key to the length *n* defined above. If the key length is greater than *n*, the application must first hash the key using Hash() function and then use the resultant byte string as the actual key to HMAC.
- | represents the concatenation operator.

Note: HMAC mode of the hash processor can be used with all supported algorithms.

HMAC processing

Four different steps are required to compute the HMAC:

1. The software sets the INIT bit, with the MODE bit set and the ALGO bits selecting the desired algorithm. The LKEY bit must also be set if the key being used is longer than the block size. In this case, as required by HMAC specifications, the hash processor uses the hash of the key instead of the real key.
2. The software provides the key to be used for the inner hash function, using the same mechanism as the message string loading that is by writing the key data into HASH_DIN register and then completing the transfer by setting DCAL bit and the correct NBLW to HASH_STR register.
3. Once the processor is ready again (DINIS = 1 in HASH_SR), the software can write the message string to HASH_DIN. When the last word of the last block is entered and the software sets DCAL bit in HASH_STR register, the NBLW bitfield must be programmed at the same time to a value different from zero if the message length is not an exact multiple of the block size. Note that the DMA can also be used to feed the message string, as described in *Section 50.4.5: Message digest computing*.
4. Once the processor is ready again (DINIS = 1 in HASH_SR), the software provides the key to be used for the outer hash function, writing the key data into HASH_DIN register, and then completing the transfer by setting DCAL bit and programming the correct NBLW to HASH_STR register. The HMAC result can be found in the valid output registers (HASH_HRx) as soon as DCIS bit is set.

Note: The computation latency of the HMAC primitive depends on the lengths of the keys and message, as described in Section 50.4.11: HASH processing time.

Endianness management details can be found in Section 50.4.4: Message data feeding.

HMAC example

Below is an example of HMAC SHA-1 algorithm (ALGO = 00 and MODE = 1 in HASH_CR) as specified by NIST. SHA-1 block size is 64 bytes.

Let us assume that the original message is the ASCII binary-coded form of "**`Sample message for keylen = blocklen`**", of length L = 34 bytes. If the HASH is programmed in no swapping mode (DATATYPE = 00 in HASH_CR), the following data must be loaded sequentially into HASH_DIN register:

1. **Inner hash key** input (length = 64, that is, no padding), specified by NIST. As key length = 64, LKEY bit is cleared in HASH_CR register

   ```
   00010203 04050607 08090A0B 0C0D0E0F 10111213 14151617
   18191A1B 1C1D1E1F 20212223 24252627 28292A2B 2C2D2E2F
   30313233 34353637 38393A3B 3C3D3E3F
   ```

2. **Message** input (length = 34, that is, padding required). HASH_STR must be set to **`0x20`** to start message padding and inner hash computation (see **'U**' as don't care)

   ```
   53616D70 6C65206D 65737361 67652066 6F72206B 65796C65
   6E3D626C 6F636B6C 656EUUUU
   ```

3. **Outer hash key** input (length = 64, that is, no padding). A key identical to the inner hash key is entered here.
4. **Final outer hash computing** is then performed by the HASH. The HMAC-SHA1 digest result is available in the HASH_HRx registers, as shown below:

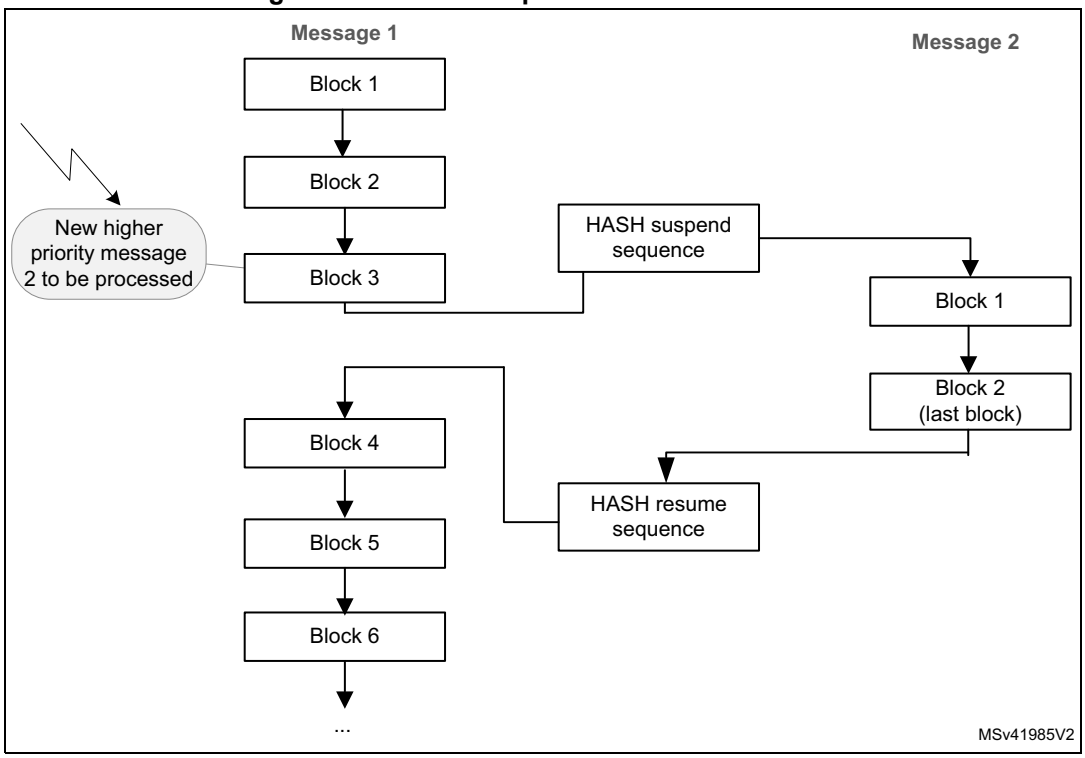
   ```
   HASH_HR0 = 0x5FD596EE
   HASH_HR1 = 0x78D5553C
   HASH_HR2 = 0x8FF4E72D
   HASH_HR3 = 0x266DFD19
   HASH_HR4 = 0x2366DA29
   ```

50.4.8 HASH suspend/resume operations

Overview

It is possible to interrupt a hash/HMAC operation to perform another processing with a higher priority. The interrupted process completes later when the higher-priority task has been processed, as shown in *Figure 494*.

Figure 494. HASH suspend/resume mechanism



To do so, the context of the interrupted task must be saved from the HASH registers to memory, and then be restored from memory to the HASH registers.

The procedures where the data flow is controlled by software or by DMA are described hereafter.

Data loaded by software

When the DMA is not used to load the message into the hash processor, the context can be saved only when no block processing is ongoing.

To suspend the processing of a message, proceed as follows after writing the number of words defined in NBWE:

1. In Polling mode, wait for BUSY = 0 then poll if the DINIS status bit is set.
   In Interrupt mode, implement the next step in DINIS interrupt handler (recommended).
2. Store the contents of the following registers into memory:
   - HASH_IMR
   - HASH_STR
   - HASH_CR
   - HASH_CSR0 to HASH_CSR37, when SHA-1 or SHA2-256 is selected, plus HASH_CSR38 to HASH_CSR53 if an HMAC operation was ongoing
   - HASH_CSR0 to HASH_CSR90, when SHA2-384 or SHA2-512 (truncated or not) is selected, plus HASH_CSR91 to HASH_CSR102 if an HMAC operation was ongoing.

To resume the processing of a message, proceed as follows:

1. Write the following registers with the values saved in memory: HASH_IMR, HASH_STR, HASH_CR.
2. Initialize the hash processor by setting the INIT bit in the HASH_CR register
3. Write the HASH_CSRx registers with the values saved in memory.
4. Restart the processing from the point where it has been interrupted.

Data loaded by DMA

When the DMA is used to load the message into the hash processor, it is recommended to suspend and then restore a secure digest computing as described below.

In this sequence the DMA channel allocated to the hash peripheral remains allocated to the processing of message 1 (see *Figure 494*).

To suspend the processing of a message using DMA, proceed as follows:

1. Clear the DMAE bit to disable the DMA interface. The hash peripheral automatically fetches enough data via the DMA to complete the current burst transfer.
2. Wait until the last DMA transfer is complete (DMAS = 0 in HASH_SR).
3. Disable the DMA channel.
4. In Polling or Interrupt mode (recommended), wait until the hash processor is ready (no block is being processed), that is wait for DINIS = 1 in HASH_SR. If DCIS is also set in HASH_SR, the hash result is available and the context swapping is useless. Else go to step 5.
5. Save HASH_IMR, HASH_STR and HASH_CR registers. Also save a number of HASH_CSRx registers depending on the SHA algorithm that is used:
   - When SHA-1 or SHA2-256 is selected, save HASH_CSR0 to HASH_CSR37, plus HASH_CSR38 to HASH_CSR53 if an HMAC operation was ongoing.
   - When SHA2-384 or SHA2-512 is selected, save HASH_CSR0 to HASH_CSR90, plus HASH_CSR91 to HASH_CSR102 if an HMAC operation was ongoing.

To resume the processing of a message using DMA, proceed as follows:

1. Reconfigure the DMA controller so that it proceeds with the transfer of the message up to the end if it is not interrupted again.
2. Program the values saved in memory to HASH_IMR, HASH_STR and HASH_CR registers.
3. Initialize the hash processor by setting the INIT bit in the HASH_CR register.
4. Program the values saved in memory to the HASH_CSRx registers.
5. Restart the processing from the point where it was interrupted by setting the DMAE bit.

50.4.9 HASH DMA interface

The HASH supports both single and fixed DMA burst transfers of four words.

The hash processor provides an interface to connect to the DMA controller. This DMA can be used to write data to the HASH by setting the DMAE bit in the HASH_CR register. When this bit is set, the HASH initiates a DMA request each time a block has to be written to the HASH_DIN register.

Once four 32-bit words have been received, the HASH automatically triggers a new request to the DMA. For more information, refer to *Section 50.4.5: Message digest computing*.

Before starting the DMA transfer, the software must program the number of valid bits in the last word that is copied into HASH_DIN register. This is done by writing in HASH_STR register the following value:

`NBLW = Len(Message)% 32` where "**`x%32`**" gives the remainder of x divided by 32.

The DMAS bit of the HASH_SR register provides information on the DMA interface activity. This bit is set with DMAE and cleared when DMAE is cleared and no DMA transfer is ongoing.

Note: No interrupt is associated to DMAS bit.

When MDMAT is set, the size of the transfer must be a multiple of four words.

50.4.10 HASH error management

No error flags are generated by the hash processor.

50.4.11 HASH processing time

Table 432 summarizes the time required to process an intermediate block for each mode of operation.

Table 432. Processing time (in clock cycle)

Mode of operation	Block size (in bytes)	FIFO load(1)	Computation phase	Total
SHA-1	64	16	66	**82**
SHA2-224		16	50	**66**
SHA2-256				
SHA2-384	128	32	66	**98**
SHA2-512(2)				

1. Add the time required to load the block into the processor.
2. SHA2-512 includes SHA2-512/224 and SHA2-512/256 modes.

The time required to process the last block of a message (or of a key in HMAC) can be longer. This time depends on the length of the last block and the size of the key (in HMAC mode).

Compared to the processing of an intermediate block, it can be increased by the factor below:

- **1 to 2.5** for a hash message
- **~2.5** for an HMAC input-key
- **1 to 2.5** for an HMAC message
- **~2.5** for an HMAC output key in case of a short key
- **3.5 to 5** for an HMAC output key in case of a long key

50.5 HASH interrupts

Two individual maskable interrupt sources are generated by the hash processor to signal the following events:

- Digest calculation completion (DCIS)
- Data input buffer ready (DINIS)

Both interrupt sources are connected to the same global interrupt request signal (hash_it), which is in turn connected to the device interrupt controller. Each interrupt source can individually be enabled or disabled by changing the mask bits in the HASH_IMR register. Setting the appropriate mask bit enables the interrupt.

The status of each maskable interrupt source can be read from the HASH_SR register. *Table 433* gives a summary of the available features.

Table 433. HASH interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method
HASH	Digest computation completed	DCIS	DCIE	Clear DCIS or set INIT
	Data input buffer ready to get a new block	DINIS	DINIE	Clear DINIS or write to HASH_DIN

50.6 HASH registers

The hash core is associated with several control and status registers and several message digest registers. All these registers are accessible through 32-bit word accesses only, else an AHB error is generated.

50.6.1 HASH control register (HASH_CR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALGO[3:0]				LKEY
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	MDMAT	DINNE	NBW[3:0]				Res.	MODE	DATATYPE[1:0]		DMAE	INIT	Res.	Res.
		rw	r	r	r	r	r		rw	rw	rw	rw	w		

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:17 **ALGO[3:0]**: Algorithm selection

These bits select the hash algorithm. This selection is only taken into account when the INIT bit is set. Changing this bitfield during a computation has no effect

When the ALGO bitfield is updated and INIT bit is set, NBWE in HASH_SR is automatically updated to 0x11.

0000: SHA-1

0001: reserved

0010: SHA2-224

0011: SHA2-256

1100: SHA2-384

1101: SHA2-512/224

1110: SHA2-512/256

1111: SHA2-512

Bit 16 **LKEY**: Long key selection

The application must set this bit if the HMAC key is greater than the block size corresponding to the hash algorithm (see *Table 430: Information on supported hash algorithms* for details). For example the block size is 64 bytes for SHA2-256.

This selection is only taken into account when the INIT and MODE bits are set (HMAC mode selected). Changing this bit during a computation has no effect.

0: HMAC key is shorter or equal to the block size (short key). The actual key value written in HASH_DIN is used during the HMAC computation.

1: HMAC key is longer than the block size (long key). The hash of the key is used instead of the real key during the HMAC computation.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **MDMAT**: Multiple DMA transfers

This bit is set when hashing large files when multiple DMA transfers are needed.

0: DCAL is automatically set at the end of a DMA transfer.

1: DCAL is not automatically set at the end of a DMA transfer.

Bit 12 **DINNE**: DIN not empty

Refer to DINNE bit of HASH_SR for a description of DINNE bit.

This bit is read-only.

Bits 11:8 **NBW[3:0]:** Number of words already pushed

Refer to NBWP[3:0] bitfield of HASH_SR for a description of NBW[3:0] bitfield.

This bitfield is read-only.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **MODE:** Mode selection

This bit selects the normal or the keyed HMAC mode for the selected algorithm. This selection is only taken into account when the INIT bit is set. Changing this bit during a computation has no effect.

0: Hash mode selected

1: HMAC mode selected. LKEY bit must be set if the key being used is longer than the algorithm block size.

Bits 5:4 **DATATYPE[1:0]:** Data type selection

This bitfield defines the format of the data entered into the HASH_DIN register:
00: 32-bit data. The data written into HASH_DIN are directly used by the HASH processing, without reordering.
01: 16-bit data or half-word. The data written into HASH_DIN are considered as two half-words, and are swapped before being used by the HASH processing.
10: 8-bit data or bytes. The data written into HASH_DIN are considered as four bytes, and are swapped before being used by the HASH processing.
11: bit data or bit string. The data written into HASH_DIN are considered as 32 bits (1st bit of the string at position 0), and are swapped before being used by the HASH processing (1st bit of the string at position 31).

Bit 3 **DMAE:** DMA enable

Setting this bit enables DMA transfers. After this bit is set, it is cleared by hardware while the last data of the message is written into the hash processor.
Clearing this bit while a DMA transfer is ongoing does not abort the current transfer. Instead, the DMA interface of the HASH remains internally enabled until the transfer is complete or INIT is set.
Setting INIT bit does not clear DMAE bit.
0: DMA transfers disabled
1: DMA transfers enabled. A DMA request is sent as soon as the hash core is ready to receive data.

Bit 2 **INIT:** Initialize message digest calculation

This is a write-only bit always returning zero upon reading. Setting this bit resets the hash processor core, so that HASH is ready to compute the message digest of a new message. Clearing this bit has no effect.

Bits 1:0 Reserved, must be kept at reset value.

50.6.2 HASH data input register (HASH_DIN)

Address offset: 0x004

Reset value: 0x0000 0000

HASH_DIN is the data input register. It is 32-bit wide. This register is used to enter the message by blocks defined by the hash algorithm (see *Table 430: Information on supported hash algorithms* for details). For example the block size is 64 bytes for SHA2-256.

When the HASH_DIN register is programmed, the value presented on the AHB bus is ‘pushed’ into the hash core and the register takes the new value presented on the AHB bus. To get a correct message format, the DATATYPE bits must have been previously configured in the HASH_CR register.

When a complete block has been written to the HASH_DIN register, an intermediate digest calculation is launched:

- by writing first data of the next block into the HASH_DIN register, if the DMA is not used
- automatically, if the DMA is used

When the last block has been written to the HASH_DIN register, the final digest calculation (including padding) is launched by setting the DCAL bit in the HASH_STR register (final digest calculation). This operation is automatic if the DMA is used and MDMAT bit is cleared.

Reading the HASH_DIN register returns zeros.

Note: When the HASH is busy, a write access to the HASH_DIN register might stall the AHB bus if the digest calculation (intermediate or final) is not complete.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATAIN[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATAIN[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **DATAIN[31:0]:** Data input

Writing this register pushes the current register content into the FIFO, and the register takes the new value presented on the AHB bus.

Do not write to this register when BUSY is set.

Reading this register returns zeros.

50.6.3 HASH start register (HASH_STR)

Address offset: 0x008

Reset value: 0x0000 0000

The HASH_STR register has two functions:

- It is used to define the number of valid bits in the last word of the message entered in the hash processor (that is the number of valid least significant bits in the last data written to the HASH_DIN register)
- It is used to start the processing of the last block in the message by setting the DCAL bit.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCAL	Res.	Res.	Res.	NBLW[4:0]				
							w				rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **DCAL:** Digest calculation

This is a write-only bit always returning zero upon reading. Setting this bit starts the message padding using the previously written value of NBLW, and starts the calculation of the final message digest with all the data words written to the input FIFO since the INIT bit was last set.

Do not set DCAL when BUSY is set.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **NBLW[4:0]:** Number of valid bits in the last word

When the last word of the message bit string is written to HASH_DIN register, the hash processor takes only the valid bits, specified as below, after internal data swapping.

The above mechanism is valid only if DCAL = 0. If NBLW bits are written while DCAL is set, the NBLW bitfield remains unchanged. In other words it is not possible to configure NBLW and set DCAL at the same time.

Reading NBLW bits returns the last value written to NBLW.

0x00: All the 32 bits of the last data written are valid message bits, that is M[31:0]

0x01: Only one bit of the last data written (after swapping) is valid, that is M[0]

0x02: Only two bits of the last data written (after swapping) are valid, that is M[1:0]

0x03: Only three bits of the last data written (after swapping) are valid that is M[2:0]

...

0x1F: Only 31 bits of the last data written (after swapping) are valid that is M[30:0]

Note:

50.6.4 HASH digest registers

These registers contain the message digest result defined as follows:

- HASH_HR0, HASH_HR1, HASH_HR2, HASH_HR3 and HASH_HR4 registers return the SHA-1 digest result.
- HASH_HR0 to HASH_HR6 registers return the SHA2-224, SHA2-512/224 digest result.
- HASH_HR0 to HASH_HR7 registers return the SHA2-256, SHA2-512/256 digest result.
- HASH_HR0 to HASH_HR11 registers return the SHA2-384 digest result.
- HASH_HR0 to HASH_HR15 registers return the SHA2-512 digest result

In all cases, the digest most significant bit is stored in HASH_HR0[31], and unused HASH_HRx registers reads as zero.

If a read access to one of these registers is performed while the hash core is calculating an intermediate digest or a final message digest (DCIS bit equals 0), then the read operation returns zeros.

Note: When starting a digest computation for a new message (by setting the INIT bit), HASH_HRx registers are forced to their reset values.

HASH_HR0 to HASH_HR4 registers can be accessed through two different addresses (register aliasing).

HASH aliased digest register x (HASH_HRAx)

Address offset: 0x00C + 0x4 * x, (x = 0 to 4)

Reset value: 0x0000 0000

The content of the HASH_HRAx registers is identical to the one of the HASH_HRx registers located at address offset 0x310.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Hx[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Hx[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **Hx[31:0]:** Hash data x

Refer to *Section 50.6.4: HASH digest registers* introduction.

HASH digest register x (HASH_HRx)

Address offset: 0x310 + 0x4 * x, (x = 0 to 4)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Hx[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Hx[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **Hx[31:0]:** Hash data x

Refer to *Section 50.6.4: HASH digest registers* introduction.

HASH supplementary digest register x (HASH_HRx)

Address offset: 0x310 + 0x4 * x, (x = 5 to 15)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Hx[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Hx[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **Hx[31:0]:** Hash data x

Refer to *Section 50.6.4: HASH digest registers* introduction.

50.6.5 HASH interrupt enable register (HASH_IMR)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCIE	DINIE
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **DCIE:** Digest calculation completion interrupt enable
0: Digest calculation completion interrupt disabled
1: Digest calculation completion interrupt enabled.

Bit 0 **DINIE:** Data input interrupt enable
0: Data input interrupt disabled
1: Data input interrupt enabled

50.6.6 HASH status register (HASH_SR)

Address offset: 0x024

Reset value: 0x0011 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NBWE[4:0]				
											r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DINNE	Res.	NBWP[4:0]					Res.	Res.	Res.	Res.	Res.	BUSY	DMAS	DCIS	DINIS
r		r	r	r	r	r						r	r	rc_w0	rc_w0

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:16 **NBWE[4:0]:** Number of words expected
This bitfield reflects the number of words in the message that must be pushed into the FIFO to trigger a partial computation. NBWE is decremented by 1 when a write access is performed to the HASH_DIN register.
NBWE is set to the expected block size +1 in words (0x11) when INIT bit is set in HASH_CR. It is set to the expected block size (0x10) when the partial digest calculation ends.

Bit 15 **DINNE:** DIN not empty
This bit is set when the HASH_DIN register holds valid data (that is after being written at least once). It is cleared when either the INIT bit (initialization) or the DCAL bit (completion of the previous message processing) is set.
0: No data are present in the data input buffer
1: The input buffer contains at least one word of data

Bit 14 Reserved, must be kept at reset value.

Bits 13:9 **NBWP[4:0]:** Number of words already pushed

This bitfield is the exact number of words in the message that have already been pushed into the FIFO. NBWP is incremented by 1 when a write access is performed to the HASH_DIN register.

When a digest calculation starts, NBWP is updated to NBWP- block size (in words), and NBWP goes to zero when the INIT bit is set.

Bits 8:4 Reserved, must be kept at reset value.

Bit 3 **BUSY:** Busy bit

0: No block is currently being processed
1: The hash core is processing a block of data

Bit 2 **DMAS:** DMA Status

This bit provides information on the DMA interface activity. It is set with DMAE and cleared when DMAE = 0 and no DMA transfer is ongoing. No interrupt is associated with this bit.

0: DMA interface is disabled (DMAE = 0) and no transfer is ongoing
1: DMA interface is enabled (DMAE = 1) or a transfer is ongoing

Bit 1 **DCIS:** Digest calculation completion interrupt status

This bit is set by hardware when a digest becomes ready (the whole message has been processed). It is cleared by writing it to 0 or by setting the INIT bit in the HASH_CR register.

0: No digest available in the HASH_HRx registers (zeros are returned)

1: Digest calculation complete, a digest is available in the HASH_HRx registers. An interrupt is generated if the DCIE bit is set in the HASH_IMR register.

Bit 0 **DINIS:** Data input interrupt status

This bit is set by hardware when the FIFO is ready to get a new block (16 locations are free). It is cleared by writing it to 0 or by writing the HASH_DIN register.

When DINIS = 0, HASH_CSRx registers reads as zero.

0: Less than 16 locations are free in the input buffer
1: A new block can be entered into the input buffer. An interrupt is generated if the DINIE bit is set in the HASH_IMR register.

50.6.7 HASH context swap registers

These registers contain the complete internal register states of the hash processor. They are useful when a suspend/resume operation has to be performed because a high-priority task needs to use the hash processor while it is already used by another task.

When such an event occurs, the HASH_CSRx registers have to be read and the read values have to be saved in the system memory space. Then the hash processor can be used by the preemptive task. When the hash computation is complete, the saved context can be read from memory and written back into the HASH_CSRx registers.

HASH_CSRx registers can be read only when DINIS equals to 1, otherwise zeros are returned.

HASH context swap register x (HASH_CSRx)

Address offset: 0x0F8 + 0x4 * x, (x = 0 to 102)

Reset value: 0x0022 0002 (HASH_CSR0)

Reset value: 0x0000 0000 (others)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CSx[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSx[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CSx[31:0]:** Context swap x

Refer to *Section 50.6.7: HASH context swap registers* introduction.

50.6.8 HASH register map

Table 434. HASH1 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	HASH_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALGO[3:2]		ALGO[1:0]		LKEY	Res.	Res.	MDMAT	DINNE	NBW[3:0]				Res.	MODE	DATATYPE[1:0]		DMAE	INIT	Res.	Res.
	Reset value												0	0	0	0	0			0	0	0	0	0	0		0	0	0	0	0		
0x004	HASH_DIN	DATAIN[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	HASH_STR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCAL	Res.	Res.	Res.	NBLW[4:0]				
	Reset value																								0				0	0	0	0	0
0x00C + 0x4 * x, (x = 0 to 4) Last address: 0x01C	HASH_HRAx	Hx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	HASH_IMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCIE	DINIE
	Reset value																															0	0
0x024	HASH_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		NBWE[4:0]					DINNE		NBWP[4:0]					Res.	Res.	Res.	Res.	Res.	BUSY	DMAS	DCIS	DINIS
	Reset value												1	0	0	0	1	0		0	0	0	0	0						0	0	0	1
Reserved																																	
0x0F8	HASH_CSR0	CSR0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Table 434. HASH1 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0FC	HASH_CSR1	CSR1[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x100	HASH_CSR2	CSR2[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...																																	
0x1CC	HASH_CSR53	CSR53[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...																																	
0x290	HASH_CSR102	CSR102[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Reserved																																	
0x310 + 0x4 * x, (x = 0 to 15) Last address: 0x34C	HASH_HRx	Hx[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

51 Memory cipher engine (MCE)

51.1 MCE introduction

Memory cipher engine (MCE) defines, in a given address space, multiple regions with specific security setup (encryption). All system bus traffic going through an encrypted region is managed on-the-fly by the MCE, automatically decrypting reads and encrypting writes if authorized.

Multiple ciphering option (stream, block, fast block) are available to offer the best security versus performance trade-off.

Note: When MCE is used in conjunction with XSPI it is mandatory to access the flash memory using the memory map mode of the flash memory controller.

51.2 MCE main features

- System bus in-line encryption (for writes) and decryption (for reads), based on embedded firewall programming
  - Four encryption modes per region (maximum 4 regions available): no encryption (bypass mode), stream cipher, block cipher and fast block cipher modes
  - Start and end of regions defined with 4-Kbytes granularity
  - Default filtering (region 0): any access granted
  - Region 1 to 4 access filtering criteria: none
- Supported block ciphers: AES-128, AES-256 or Noekeon (12 round version), selected at boot
- Supported chaining modes: block and stream
  - Block mode with AES cipher is compatible with ECB mode specified in *NIST FIPS publication 197 Advanced encryption standard (AES)* (normal or fast).
  - Stream mode with AES cipher is compliant with CTR mode specified in *NIST SP800-38A Recommendation for Block Cipher Modes of Operation*.
  - Includes a leakage resilient mode of operation as defense against side channel attacks (SCA).
- One set of write-only and lockable 256-bit master key registers per block cipher (normal, fast)
- Two sets of lockable cipher contexts (128-bit key, IV), usable for stream and block ciphers
- Optimization for XSPI data pre-fetching mechanism (stream cipher only)
- Read-write arbitration scheme, for better read performances
- AHB configuration port
  - Accessible through 32-bit word single accesses only (otherwise an AHB bus error is generated, and write accesses are ignored)
- AXI system bus master/slave interfaces (64-bit)
  - Support for any AXI-64bit read transactions
  - When encryption is enabled, support for AXI-64bit INCRx (x = 1 to 8) and WRAPx (x = 4) write transactions

51.3 MCE implementation

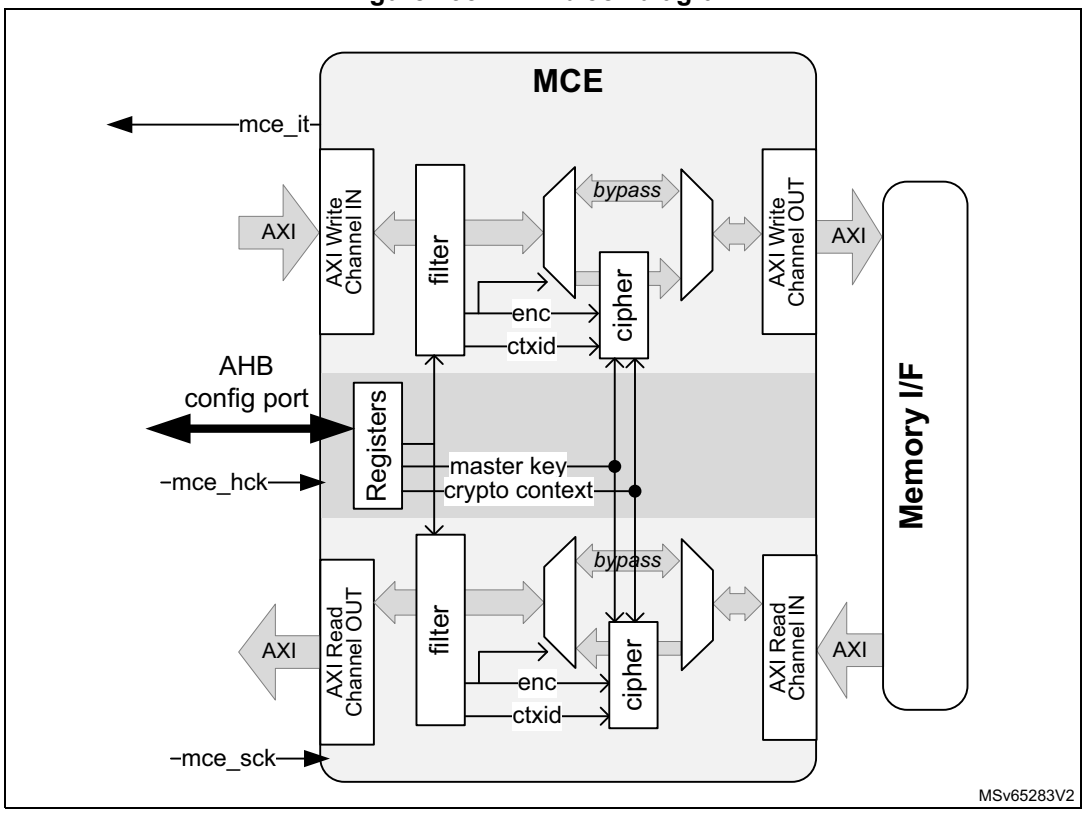
MCE peripheral generates interrupts to the Cortex® CPU interrupt controller (mce_it).

51.4 MCE functional description

51.4.1 MCE block diagram

Figure 495 shows the memory cipher engine block diagram, associated with one or more memory interface peripheral.

Figure 495. MCE block diagram



51.4.2 MCE internal signals

Table 435 describes the user relevant internal signals interfacing the MCE peripheral.

Table 435. MCE internal input/output signals

Signal name	Signal type(1)	Description
mce_sck	input	AXI input clock
mce_hck	input	AHB input clock for MCE
mce_it	output	MCE interrupt request

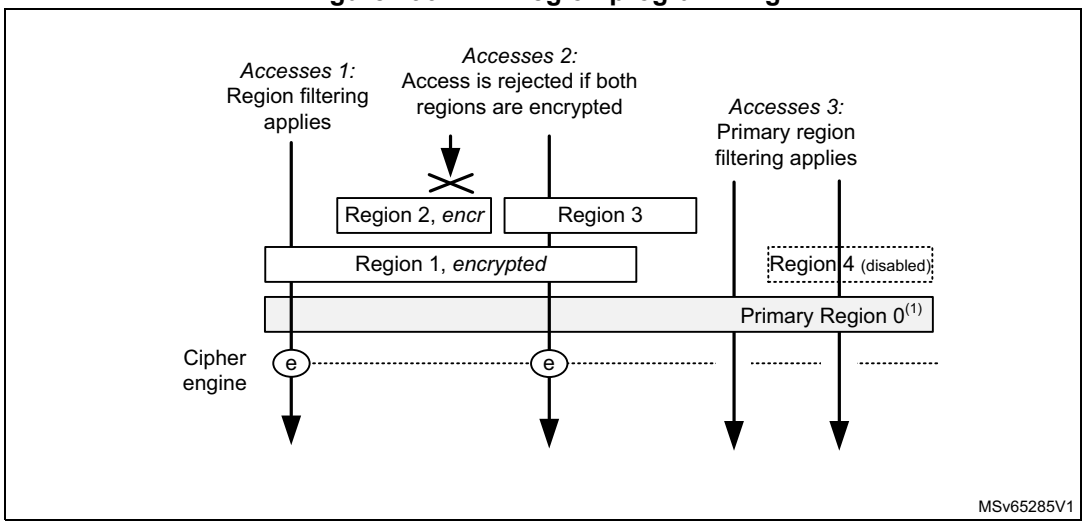
1. All digital.

51.4.3 MCE programming

Overview

Through MCE registers application software defines security regions, as shown on *Figure 496*.

Figure 496. MCE region programming



1. Covers all addressable space.

There are three types of accesses filtered by MCE:

1. Access to an area located in one enabled region (accesses 1). In this cases region filtering rules applies. If encryption is enabled cipher engine is used accordingly.
2. Access to an area overlapping two enabled regions (access 2):
   – If both regions are encrypted, access is automatically rejected with an error
3. Access to an area not belonging to any enabled region (accesses 3). In this case primary region 0 filtering apply (any access granted, no encryption).

Configuring regions

When application needs to use any region x following sequence can be used, for example at boot time.

1. Application selects, once, the correct cipher using CIPHERSEL in MCE_CR register. By default AES-128 is selected. When CIPHERSEL is changed all keys in MCE are zeroed.
2. Application sets a start and end address for each of the required region x, using MCE_SADDRx and MCE_EADDRx registers.
3. Application finalizes the programming by configuring MCE_REGCRx setting up security parameters (encryption). Initializing encryption is detailed in *Section 51.4.5* and *Section 51.4.5*.
4. After final check on above configuration application enables each required region z, setting BREN bit in MCE_REGCRz.

MCE configuration can be fixed until next SoC reset when GLOCK is set in MCE_CR register.

Note: When ENCDIS bit is set in MCE_SR, if ENC bit and BREN bits are set in MCE_REGCRx all writes to this region x are ignored and all read requests return zero. See Section 51.4.8 for details.

Runtime modification of region configuration

Note: When BREN is cleared primary region 0 access rules applies instead of base region rules (see Overview).

GLOCK or KEYLOCK/CCLOCK bits must be used to write lock the following registers: MCE_MKEYRx, MCE_FMKEYRx, MCE_CCxCFGR, MCE_CCxNR0/1 and MCE_CCxKEYR, in order to avoid unexpected changes to the cipher keys/context while used by the MCE.

Encrypting with MCE

MCE automatically encrypts on-the-fly any granted write transaction to a region with encryption properly initialized. When stream cipher is selected it is important to activate write protection as soon as the whole region has been encrypted (read-only region).

When MCE is used in conjunction with XSPI it is mandatory to set the Flash memory controller in memory map mode. It is also required to use the DMA to perform writes to the flash memory using 16 bytes bursts.

51.4.4 MCE reset and clocks

MCE configuration port is clocked by the AHB bus clock.

MCE system bus interface is clocked by the AXI clock. AHB interface does not need to be clocked for the system bus to be usable.

When MCE is kept under reset, no traffic can go through the system bus. Out of reset default firewall configuration applies, as described in *Section 51.4.3*.

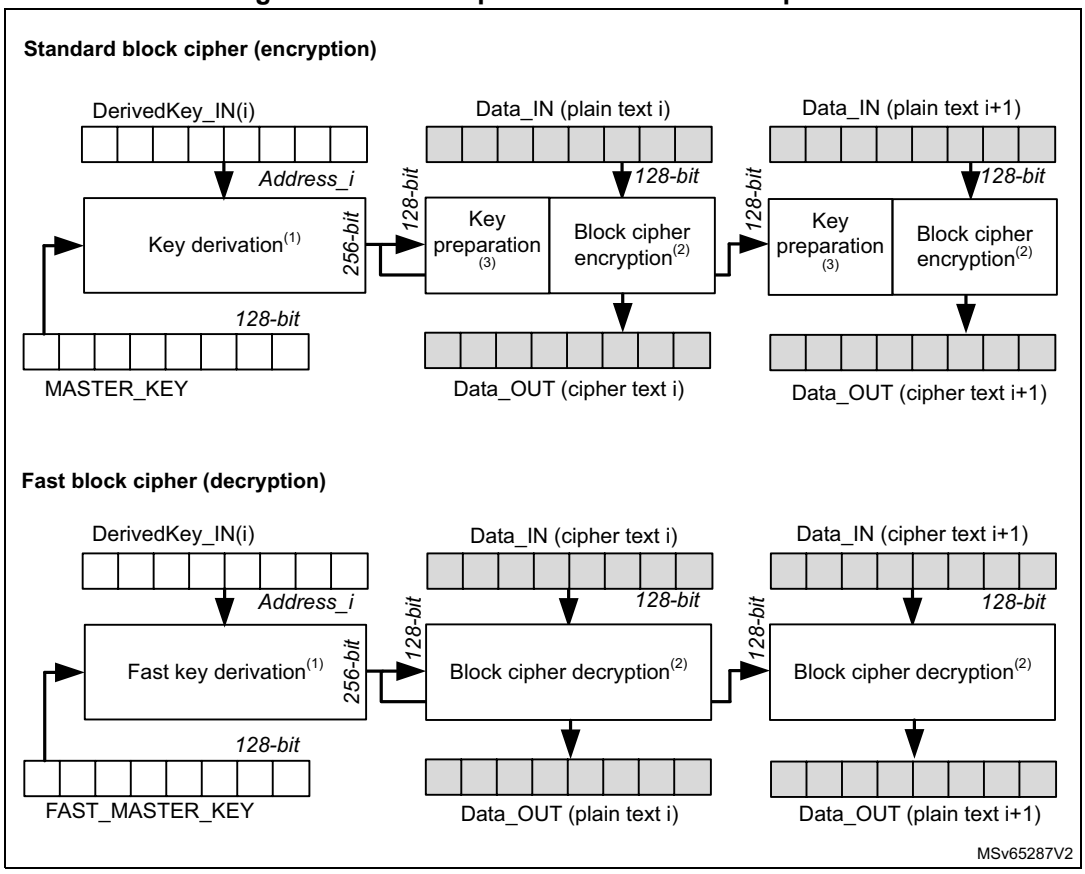
The MCE peripheral clocks are automatically managed by the device, and the peripheral is reset only by a system reset.

51.4.5 MCE block cipher encryption mode

Caution: When MCE is used in conjunction with XSPI it is mandatory to read or write the flash memory using the memory map mode of the flash memory controller.

Figure 497 details the block cipher implementations in the MCE peripheral. Top figure is a standard encryption (for writes), bottom figure is a fast decryption (for reads). Application can select a 256-bit master key using CIPHERSEL bitfield in MCE_CR register (AES-128 selection is shown, multiply by two all key lenghts when AES-256 is selected).

Figure 497. MCE implementation of block ciphers



1. Operation starts from the capture of the address.
2. Operation starts from availability of data input.
3. Only required for AES cipher.

Address_i is most significant bits of physical address of block (i) + block (i+1). Result of key derivation primitive can be used for two consecutive 16-byte data words. Since the normal key derivation function is leakage resilient the master key information is protected against side channel attacks (SCA).

MASTER_KEY must be stored in write-only MCE_MKEYRx registers. MCE_MKEYR4 to 7 registers must be written only if AES-256 cipher is selected with CIPHERSEL. FAST_MASTER_KEY must be stored in write-only MCE_FMKEYRx registers. MCE_FMKEYR4 to 7 registers must be written only if AES-256 cipher is selected with CIPHERSEL. Alternatively, MCE_CCzKEYRx registers can be used if cipher context z is selected in MCE_REGCR (CTXID = z). In this case AES-256 cipher must not be selected with CIPHERSEL.

Each time a key material has been successfully written in MCE, the following flags are set:

- MKVALID bit in MCE_SR is set when a complete key has been written in MCE_MKEYR registers.
- FMKVALID bit in MCE_SR is set when a complete key has been written in MCE_FMKEYR registers.
- KEYCRC[7:0] in MCE_CCzCFGR is loaded with 8-bit checksum when a complete key has been written in MCE_CCzKEYR registers. See KEYCRC bitfield for details.

Bypass mode is selected instead of cipher mode if the selected key is not marked as valid with above flags. See ENC bitfield description for details.

Note: With AES block cipher, writing (encryption) requires extra cycles in order to perform the required key scheduling to obtain the encryption key. See next table for details.

Fast key derivation function may be sensitive against side channel attacks. Hence master key used in normal mode is never used in fast mode.

Relative performances are described on *Table 436*.

Table 436. MCE block cipher latencies

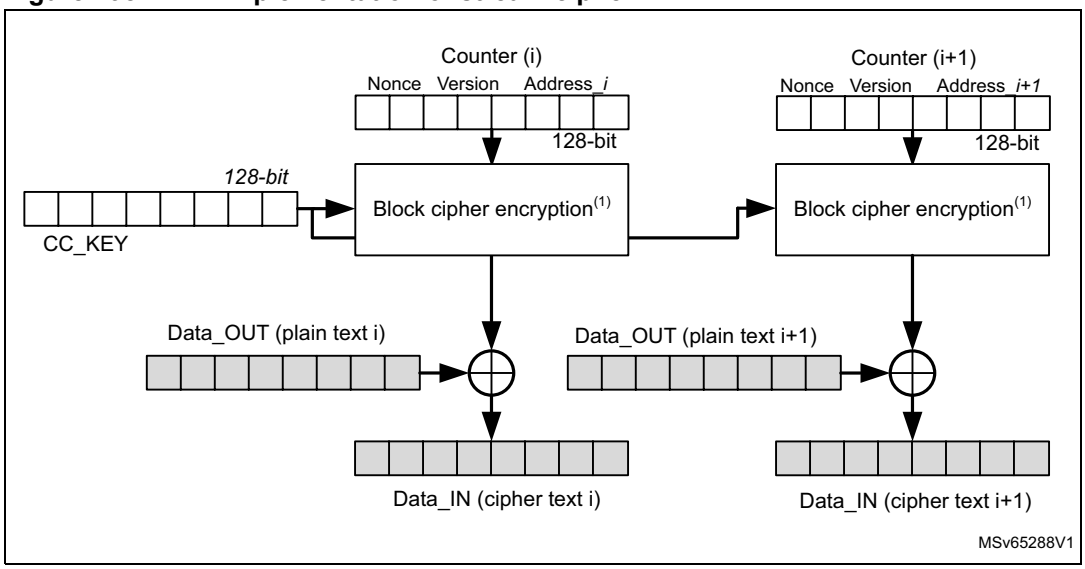
Cipher type	Block mode	ENC bit in MCE_REGCRx	Latency in AXI cycles		Optimization for sequential accesses
			for 16 bytes data	for 32 bytes data	
AES-128(1)	Normal(2)	10	14+11= 25	14+11x2= 36	no
	Fast	11	3+11= 14	3+11x2= 25	
AES-256(3)	Normal(2)	10	20+15= 35	20+15x2= 50	no
	Fast	11	9+15= 24	9+15x2= 39	
Noekeon	Normal	10	14+7= 21	14+7x2= 28	no
	Fast	11	4+7= 11	4+7x2= 18	

1. Additional 10 cycles are required when performing encryption/ writes
2. Leakage resilient mode of operation as defense against side channel attacks.
3. Additional 14 cycles are required when performing encryption/ writes

51.4.6 MCE stream cipher encryption mode

Figure 498 details the stream cipher implementations in the MCE peripheral. It pictures encryptions for AXI writes.

Figure 498. MCE implementation of stream cipher



1. Operation starts from the capture of the address.

Every 128-bit data block (i), a counter (i) information is computed, as defined below:

- Counter[127:0] = MCE_CCzNR1[31:0] || MCE_CCzNR0[31:0] || 0b0000 0000 0000 0000 || MCE_CCzCFGR[31:16] || 0b0000 || Address_i [31:4].

Address_i is the physical address of block (i), modulo 16 bytes.

Version and *Nonce* are stored by application respectively in MCE_CCzCFGR and MCE_CCzNRx registers, with cipher context z selected in MCE_REGCR (CTXID = z).

The least significant byte of Data_IN and Data_OUT are stored in the first 8 bits of the data blocks.

CC_KEY is stored in write-only MCE_CCzKEYRx registers, with cipher context *z* selected in MCE_REGCR (CTXID = *z*). When a complete key have not been written in MCE_CCzKEYR registers bypass mode is selected instead of stream cipher mode.

Note: Stream cipher mode offers no protection against bit-flip attacks, and is sensitive against side channel attacks.

Caution: When MCE is used in conjunction with XSPI it is mandatory to read or write the flash memory using the memory map mode of the flash memory controller.

Relative performances are described on *Table 437*.

Table 437. MCE stream cipher latencies

Cipher type	Stream mode	ENC bit in MCE_REGCRx	Latency for 16 bytes data (in AXI cycles)	Optimization for sequential accesses
AES-128	Normal	01	11	Yes
Noekeon	Normal	01	6	Yes

51.4.7 MCE AXI traffic management

MCE includes a special AXI-64 read-write arbitration scheme, designed to speed-up multiple reads. More specifically, when no write is ongoing in MCE the two cipher cores can be allocated to decrypt two reads in parallel. In this case an incoming write request is stalled until the cipher core allocated to writes is free.

Note: Best performances are obtained when transactions are aligned on two 64-bit words for regions where block cipher is selected. See Section 51.4.5 for details.

51.4.8 MCE encryption disable options

When MCE feature is not available in the product encryption/decryption feature of MCE is not usable, for any region. It means that reads (resp. writes) through MCE are not decrypted (resp. encrypted).

When an unexpected event occurs, the ENCDIS bit is set in MCE_SR register and the following effects apply:

- All previously written key materials are erased.
- If ENC bit and BREN bits are set in MCE_REGCRx, all writes to this region x are ignored and all read requests return zero.

In such an event, the application must reset the MCE peripheral to be able to use encrypted regions again.

51.4.9 MCE error management

When an illegal access *through the system bus* occurs (overlapping encrypted regions, illegal key usage, CIPHERSEL = 0) the corresponding IAEF bit is set in MCE_IASR. Application cleans this flag using IAEF bit in MCE_IACR.

Illegal key usage is triggered when an access to an encrypted region occurs, with the cipher mode in MCE_REGCRx not supported by the selected cipher context key in MCE_CCzCFGR (see MODE[1:0] bitfield).

Additional debug information on system bus illegal accesses can be read in illegal address register (MCE_IADDR). Illegal read (resp. write) through the system bus returns zero (resp. is ignored).

Reading any MCE key register returns zeros, and read to write-only or write to read-only registers do not trigger an interrupt.

51.5 MCE interrupts

There is one individual maskable interrupt source generated by the MCE, signaling the following error event:

- Illegal access through the system bus (IAEF), see *Section 51.4.9*

This interrupt source is connected to the same global interrupt request signal mce_it.

Setting the appropriate mask bit in MCE_IAIER register enables the interrupt. The status of the individual interrupt events can be read from MCE_IASR, and cleared in MCE_IACR register.

Table 438 gives a summary of the available features.

Table 438. MCE interrupt requests

Interrupt acronym	Interrupt event	Event flag(1)	Enable control bit	Interrupt clear method
MCE	Illegal access error	IAEF	IAEIE	Set IAEF bit in MCE_IACR

1. Read in MCE_IASR register

51.6 MCE registers

51.6.1 MCE configuration register (MCE_CR)

Address offset: 0x000

Reset value: 0x0000 0010

Writes are ignored if GLOCK is set in MCE_CR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CIPHERSEL[1:0]		Res.	Res.	MK LOCK	GLOCK
										rw	rw			rs	rs

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:4 **CIPHERSEL[1:0]**: Cipher selection

01: AES-128 cipher selected for all encrypted regions

10: Noekeon cipher selected for all encrypted regions

11: AES-256 cipher selected for all encrypted regions

00: No cipher is selected. Any read (resp. write) to an encrypted region with BREN = 1 returns zero (resp. is ignored), and illegal access flag IAEF bit is set.

When CIPHERSEL value is changed all keys in MCE are zeroed.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **MKLOCK**: Master keys lock

Lock the master key configurations until next reset. This bit is cleared by default and once set it cannot be reset until MCE reset.

0: Writes to MCE_MKEYRx and MCE_FMKEYRx registers are allowed

1: Writes to MCE_MKEYRx and MCE_FMKEYRx registers are ignored until next MCE reset.

Bit 0 **GLOCK**: Global lock

Lock the configuration of most MCE registers until next reset. This bit is cleared by default and once set it cannot be reset until MCE reset.

0: MCE registers are writable

1: All writes to MCE registers are ignored, with the exception of MCE_IACR and MCE_IAIER registers.

51.6.2 MCE status register (MCE_SR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENC DIS	Res.	FMK VALID	Res.	MK VALID
											r		r		r

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **ENCDIS**: encryption disabled

This bit is set by hardware when the encryption feature is not functional.

0: When ENC bit and BREN are set in MCE_REGCRx all allowed write accesses in region x are encrypted, and all allowed read requests are decrypted.

1: When ENC bit and BREN are set in any MCE_REGCRx all write accesses in region x are ignored, and all read requests return zero. All previously written key material are also erased.

When ENCDIS is set application must reset MCE peripheral to be able to use the encryption feature again.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **FMKVALID**: Fast master key valid

0: A valid key has not been written in MCE_FMKEYRx registers, or a valid unlocked fast master key has been erased by hardware (if bit was set before)

1: A valid key has been written in MCE_FMKEYRx registers (write R0 -> R1 -> R2 -> R3 or the reverse)

When this bit is set, any successful write to MCE_FMKEYRx registers clears FMKVALID bit (for example after a write when GLOCK and MKLOCK are cleared).

Bit 1 Reserved, must be kept at reset value.

Bit 0 **MKVALID**: Master key valid

0: A valid key has not been written in MCE_MKEYRx registers, or a valid unlocked master key has been erased by hardware (if bit was set before)

1: A valid key has been written in MCE_MKEYRx registers (write R0 -> R1 -> R2 -> R3 or the reverse)

When this bit is set, any successful write to MCE_MKEYRx registers clears MKVALID bit (for example after a write when GLOCK and MKLOCK are cleared).

51.6.3 MCE illegal access status register (MCE_IASR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IAEF	Res.
														r	

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **IAEF**: Illegal access error flag
This bit is set when an illegal access is detected on the system bus
This bit is cleared by setting corresponding bit in MCE_IACR register.

Bit 0 Reserved, must be kept at reset value.

51.6.4 MCE illegal access clear register (MCE_IACR)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IAEF	Res.
														w	

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **IAEF**: Illegal access error flag clear
Set this bit to clear IAEF bit in MCE_IASR register.

Bit 0 Reserved, must be kept at reset value.

51.6.5 MCE illegal access interrupt enable register (MCE_IAIER)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IAEIE	Res.
														rw	

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **IAEIE**: Illegal access error interrupt enable
0: Interrupt generation on illegal access errors is disabled
1: Interrupt generation when an illegal access error occurs (IAEF = 1)

Bit 0 Reserved, must be kept at reset value.

51.6.6 MCE illegal address register (MCE_IADDR)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IADD[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IADD[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **IADD[31:0]**: Illegal address
When IAEF bit is set in MCE_IASR register, IADD bitfield captures the 32-bit bus address of the erroneous access.

51.6.7 MCE region x configuration register (MCE_REGCRx)

Address offset: 0x040 + 0x10 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

Writes are ignored if GLOCK is set in MCE_CR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ENC[1]	ENC[0]	Res.	Res.	Res.	CTXID [1]	CTXID [0]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BREN
rw	rw				rw	rw									rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:14 **ENC[1:0]**: Encrypted region

Those bits are taken into account only if BREN is set. Write to these bits is ignored if BREN is set.

00: No effect

01: Stream cipher - All allowed read (resp. write) requests are decrypted (resp. encrypted) using the stream cipher, when CCEN bit is set in MCE_CCzCFGR (where CTXID = z).

10: Block cipher - All allowed read (resp. write) requests are decrypted (resp. encrypted) using the SCA resistant block cipher. If CTXID = 0 and MKVALID = 0 bypass mode is selected instead. If CTXID = z and CCEN bit is cleared in MCE_CCzCFGR bypass mode is also selected.

11: Fast block cipher - All allowed read (resp. write) requests are decrypted (resp. encrypted) using the fast block cipher. If CTXID = 0 and FMKVALID = 0 bypass mode is selected instead. If CTXID = z and CCEN bit is cleared in MCE_CCzCFGR bypass mode is also selected.

Bits 13:11 Reserved, must be kept at reset value.

Bits 10:9 **CTXID[1:0]**: Context ID

This bitfield defines the cryptographic context used by the cipher engine assigned to this region. If ENC = 00 bitfield CTXID is ignored. If BREN is set write to this bitfield is ignored.

00: If ENC = 10 (resp. 11) the key stored in MCE_MKEYR (resp. MCE_FMKEYR) registers is used by the block cipher. If ENC = 01 bypass mode is selected instead of stream cipher.

01: If ENC = 10 or 11 the key stored in MCE_CC1KEYR is used by the block cipher.
If ENC = 01 the key stored in MCE_CC1KEYR is used by the stream cipher. The nonce in MCE_CC1NRx registers and the version in MCE_CC1CR register are also used.

10: If ENC = 10 or 11 the key stored in MCE_CC2KEYR is used by the block cipher.
If ENC = 01 the key stored in MCE_CC2KEYR is used by the stream cipher. The nonce in MCE_CC2NRx registers and the version in MCE_CC2CR register are also used.

11: Reserved

When CTXID = z (z different from zero) the selected key is usable only if ENC[1:0] equals to MODE[1:0] in MCE_CCzCFGR register. When selected key is not usable any read to this region returns zero, any write is ignored until a valid key is selected, and IAEF flag is raised each time there is an access to this region with BREN = 1.

Bits 8:1 Reserved, must be kept at reset value.

Bit 0 **BREN**: Base region enable

0: Region x is disabled. Access control of primary region (privileged or unprivileged, no encryption) applies to any access between this region start and end addresses.

1: Region x is enable. Access controls and encryption option defined in this region apply to any access between this region start and end addresses.

BREN cannot be set if BADDRSTART > BADDREND.

51.6.8 MCE start address for region x register (MCE_SADDRx)

Address offset: 0x044 + 0x10 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0000

Writes are ignored if GLOCK is set in MCE_CR register, or if BREN is set in MCE_REGCRx.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BADDSTART[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BADDSTART[15:12]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw												

Bits 31:12 **BADDSTART[31:12]**: Region address start

This bitfield defines the absolute start address of the region x on 4 Kbyte boundary (inclusive).
BREN cannot be set if BADDRSTART > BADDREND.
When MCE determines the region, the first 12 bits (LSB) and the last 4 bits (MSB) in this register are ignored, and when this register is accessed in read the 4 MSB bits and the 12 LSB bits return zeros (reference value in MCE).

Bits 11:0 Reserved, must be kept at reset value.

51.6.9 MCE end address for region x register (MCE_EADDRx)

Address offset: 0x048 + 0x10 * (x - 1), (x = 1 to 4)

Reset value: 0x0000 0FFF

Writes are ignored if GLOCK is set in MCE_CR register, or if BREN is set in MCE_REGCRx.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BADDEND[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BADDEND[15:12]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw												

Bits 31:12 **BADDEND[31:12]**: Region address end

This bitfield defines the absolute end address of the region x on 4 Kbyte boundary (inclusive).
BREN cannot be set if BADDRSTART > BADDREND.
When MCE determines the region, the first 12 bits (LSB) and the last 4 bits (MSB) in this register are ignored, and when this register is accessed in read the 4 MSB bits return zeros and the 12 LSB bits return ones (reference value in MCE).

Bits 11:0 Reserved, must be kept at reset value.

51.6.10 MCE master key x (MCE_MKEYRx)

Address offset: 0x200 + 0x4 * x, (x = 0 to 7)

Reset value: 0x0000 0000

Writes are ignored if MKLOCK or GLOCK bit is set in MCE_CR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MKEY{32 * x + i}															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MKEY{32 * x + i}															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **MKEY{32 * x + i}**: Master key bit (i = 0 to 31)

This key is used by the MCE block cipher in normal, SCA resistant mode, if CTXID = 0x0 in MCE_REGCR register. Register MCE_MKEYR4 to 7 must be written only if CIPHERSEL = 0x3 in MCE_CR, otherwise MKVALID may become invalid.

51.6.11 MCE fast master key x (MCE_FMKEYRx)

Address offset: 0x220 + 0x4 * x, (x = 0 to 7)

Reset value: 0x0000 0000

Writes are ignored if MKLOCK or GLOCK bit is set in MCE_CR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FMKEY{32 * x + i}															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FMKEY{32 * x + i}															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **FMKEY{32 * x + i}**: Fast master key bit (i = 0 to 31)

This key is used by the MCE block cipher in fast mode, if CTXID = 0x0 in MCE_REGCR register. Register MCE_FMKEYR4 to 7 must be written only if CIPHERSEL = 0x3 in MCE_CR, otherwise FMKVALID may become invalid.

51.6.12 MCE cipher context z configuration register (MCE_CCzCFGR)

Address offset: 0x240 + 0x30 *(z - 1) (z = 1 to 2)

Reset value: 0x0000 0000

Writes are ignored if CCLOCK bit is set in this register. Writes are also ignored if GLOCK is set in MCE_CR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
VERSION[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEYCRC[7:0]								Res.	Res.	MODE[1:0]		Res.	KEYLOCK	CCLOCK	CCEN
r	r	r	r	r	r	r	r			rw	rw		rs	rs	rw

Bits 31:16 **VERSION[15:0]**: Version

This 16-bit bitfield must be correctly initialized before CCEN bit is set. Bitfield usage is defined in *Section 51.4.6: MCE stream cipher encryption mode*.

Bits 15:8 **KEYCRC[7:0]**: Key CRC

When KEYLOCK = 0, KEYCRC information is automatically computed by hardware while loading the key of this region in this exact sequence: KEYR0 then KEYR1 then KEYR2 then finally KEYR3 (all written once). A new KEYCRC computation starts as soon as a new valid sequence is initiated. KEYCRC bitfield reads as zero until a valid sequence is completed (after it return the computed CRC value).

When KEYLOCK or GLOCK = 1, KEYCRC bitfield always returns the computed CRC value until the next reset.

CRC computation is an 8-bit checksum using the standard CRC-8-CCITT algorithm X8 + X2 + X + 1 (according the convention).

CRC information is updated, and the key is usable by MCE, only after the last bit of the key has been written.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **MODE[1:0]**: Authorized cipher mode

This bitfield defines the encrypted region type authorized for this cipher context z.

0X: Stream cipher is allowed with this cipher context

10: Block cipher is allowed with this cipher context

11: Fast block cipher is allowed with this cipher context

This cipher mode selection is done in MCE_REGCRx register, by writing ENC[1:0] bitfield, and setting CTXID = z. See CTXID bitfield for details.

When either of the MODE bits are changed, the region key and associated CRC are zeroed.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **KEYLOCK**: Key lock

0: Writes to MCE_CCzKEYR registers are allowed

1: Writes to MCE_CCzKEYR registers are ignored until next MCE reset. KEYCRC bitfield value does not change.

Note: This bit is set once. If this bit is set, it can only be cleared to 0 if MCE is reset.

Bit 1 **CCLOCK**: Cipher context lock

0: Writes to MCE_CCzCFGR and MCE_CCzNR registers are allowed

1: Writes to MCE_CCzCFGR and MCE_CCzNR registers are ignored until next MCE reset.

Note: This bit is set once. If this bit is set, it can only be cleared to 0 if MCE is reset. Setting this bit forces KEYLOCK bit to 1.

Bit 0 **CCEN**: Cipher context enable

0: If an enabled region selects CTXID = z bypass mode is selected by MCE

1: If an enabled region selects CTXID = z with ENC = 01 MCE selects with the stream cipher the information stored in MCE_CCzCFGR, MCE_CCzNR and MCE_CCzKEYR. If the enabled region selects ENC = 10 or 11 instead MCE selects with the block cipher the key information stored in MCE_CCzKEYR. When CCEN is set and MODE[1:0] bits do not allow the configuration written in ENC[1:0] bits (see MODE bitfield) reads return zero and writes are ignored.

51.6.13 MCE cipher context z nonce register 0 (MCE_CCzNR0)

Address offset: 0x244 + 0x30 *(z - 1) (z = 1 to 2)

Reset value: 0x0000 0000

Writes are ignored if CCLOCK is set in MCE_CCzCFGR register, or GLOCK bit is set in MCE_CR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SCNONCE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCNONCE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SCNONCE[31:0]**: Stream cipher nonce, bits [31:0]

This register is used by stream cipher to compute keystream. It must be correctly initialize before CCEN bit is set in MCE_CCzCFGR register. Bitfield usage is defined in *Section 51.4.6: MCE stream cipher encryption mode*.

51.6.14 MCE cipher context z nonce register 1 (MCE_CCzNR1)

Address offset: 0x248 + 0x30 *(z - 1) (z = 1 to 2)

Reset value: 0x0000 0000

Writes are ignored if CCLOCK is set in MCE_CCzCFGR register, or GLOCK bit is set in MCE_CR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SCNONCE[63:48]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCNONCE[47:32]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SCNONCE[63:32]**: Stream cipher nonce, bits [63:32]

Refer to the MCE_CCzNR0 register for description of the SCNONCE[63:0] bitfield.

51.6.15 MCE cipher context z key register 0 (MCE_CCzKEYR0)

Address offset: 0x24C + 0x30 *(z - 1) (z = 1 to 2)

Reset value: 0x0000 0000

Writes are ignored if CCLOCK or KEYLOCK are set in MCE_CCzCFGR register, or GLOCK bit is set in MCE_CR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[31:0]**: cipher key, bits [31:0]

This register is used by the block or stream cipher of MCE when CTXID = z in encrypted region configuration register. KEY[127:0] must be correctly initialize before CCEN bit is set in MCE_CCzCFGR register. When the application selects the block cipher AES-256 must not be selected with CIPHERSEL.

51.6.16 MCE cipher context z key register 1 (MCE_CCzKEYR1)

Address offset: 0x250 + 0x30 *(z - 1) (z = 1 to 2)

Reset value: 0x0000 0000

Writes are ignored if CCLOCK or KEYLOCK are set in MCE_CCzCFGR register, or GLOCK bit is set in MCE_CR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[63:48]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[47:32]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[63:32]**: cipher key, bits [63:32]
Refer to the MCE_CCzKEYR0 register for description of the KEY[127:0] bitfield.

51.6.17 MCE cipher context z key register 2 (MCE_CCzKEYR2)

Address offset: 0x254 + 0x30 *(z - 1) (z = 1 to 2)

Reset value: 0x0000 0000

Writes are ignored if CCLOCK or KEYLOCK are set in MCE_CCzCFGR register, or GLOCK bit is set in MCE_CR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[95:80]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[79:64]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[95:64]**: cipher key, bits [95:64]
Refer to the MCE_CCzKEYR0 register for description of the KEY[127:0] bitfield.

51.6.18 MCE cipher context z key register 3 (MCE_CCzKEYR3)

Address offset: 0x258 + 0x30 * (z - 1) (z =1 to 2)

Reset value: 0x0000 0000

Writes are ignored if CCLOCK or KEYLOCK are set in MCE_CCzCFGR register, or GLOCK bit is set in MCE_CR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[127:112]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[111:96]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[127:96]**: cipher key, bits [127:96]

Refer to the MCE_CCzKEYR0 register for description of the KEY[127:0] bitfield.

51.6.19 MCE register map

Table 439. MCE register map and reset values

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	MCE_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CIPHERSEL[1:0]		Res.	Res.	MKLOCK	GLOCK
	Reset value																											0	1			0	0
0x004	MCE_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENCDIS	Res.	FMKVALID	Res.	MKVALID
	Reset value																												0		0		0
0x008	MCE_IASR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IAEF	Res.
	Reset value																															0	
0x00C	MCE_IACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IAEF	Res.
	Reset value																															0	
0x010	MCE_IAIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IAEIE	Res.
	Reset value																															0	0
0x20	Reserved	Res.																															
0x024	MCE_IADDR	IADD[31:10]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x040+ 0x10 * (x-1), (x = 1 to 4)	MCE_REGCRx	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENC[1]	ENC[0]	Res.	Res.	Res.	CTXID[1]	CTXID[0]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BREN
	Reset value																	0	0				0	0									0
0x044+ 0x10 * (x-1), (x = 1 to 4)	MCE_SADDRx	BADDSTART[31:16]																BADDSTART[15:12]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												
0x048+ 0x10 * (x-1), (x = 1 to 4)	MCE_EADDRx	BADDEND[31:16]																BADDEND[15:12]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												
0x200+0x 4 * x, (x = 0 to 7)	MCE_MKEYRx	MKEY{32 * x + i} (i = 0 to 31)																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x220+0x 4 * x, (x = 0 to 7)	MCE_ FMKEYRx	FMKEY{32 * x + i} (i = 0 to 31)																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x240	MCE_ CC1CFGR	VERSION[15:0]																KEYCRC[7:0]								Res.	Res.	MODE[1]	MODE[0]	Res.	KEYLOCK	CCLOCK	CCEN
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0		0	0	0
0x244	MCE_ CC1NR0	SCNONCE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x248	MCE_ CC1NR1	SCNONCE[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 439. MCE register map and reset values (continued)

Offset	Register name reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x24C	MCE_CC1KEYR0	KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x250	MCE_CC1KEYR1	KEY[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x254	MCE_CC1KEYR2	KEY[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x258	MCE_CC1KEYR3	KEY[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x270	MCE_CC2CFGR	VERSION[15:0]																KEYCRC[7:0]								Res.	Res.	MODE[1]	MODE[0]	Res.	KEYLOCK	CCLOCK	CCEN
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0		0	0	0
0x274	MCE_CC2NR0	SCNONCE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x278	MCE_CC2NR1	SCNONCE[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x27C	MCE_CC2KEYR0	KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x280	MCE_CC2KEYR1	KEY[63:32]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x284	MCE_CC2KEYR2	KEY[95:64]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x288	MCE_CC2KEYR3	KEY[127:96]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

52 Public key accelerator (PKA)

52.1 PKA introduction

PKA (public key accelerator) is intended for the computation of cryptographic public key primitives, specifically those related to RSA, Diffie-Hellmann or ECC (elliptic curve cryptography) over GF(p) (Galois fields). To achieve high performance at a reasonable cost, these operations are executed in the Montgomery domain.

For a given operation, the accelerator performs all computations with no further hardware or software elaboration required to process the inputs or the outputs.

For manipulating secrets, the PKA incorporates a protection against side-channel attacks (SCA), including differential power analysis (DPA), required for SESIP and PSA security assurance level 3 target.

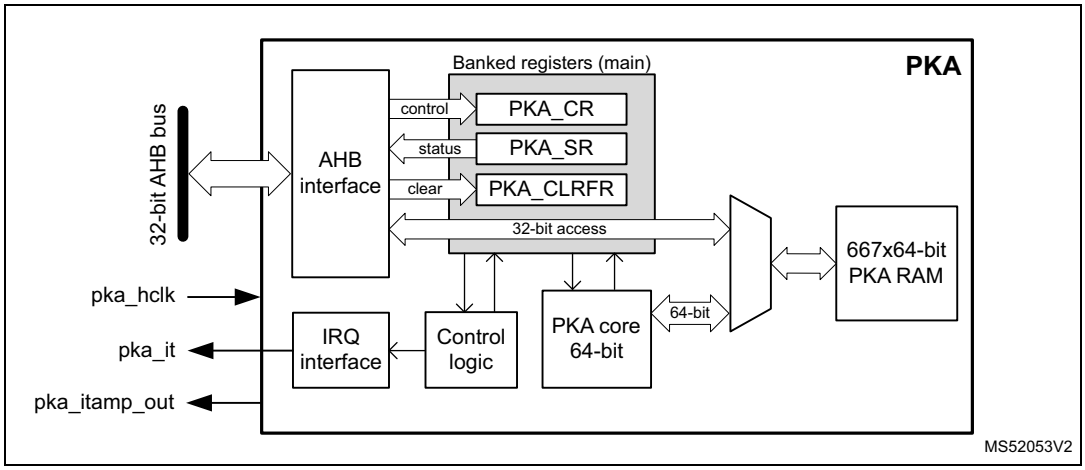
52.2 PKA main features

- Acceleration of RSA, DH and ECC over GF(p) operations, based on the Montgomery method for fast modular multiplications. More specifically:
  - RSA modular exponentiation, RSA chinese remainder theorem (CRT) exponentiation
  - ECC scalar multiplication, point on curve check, complete addition, double base ladder, projective to affine
  - ECDSA signature generation and verification
- Capability to handle operands up to 4160 bits for RSA/DH and 640 bits for ECC
- When manipulating secrets: protection against side-channel attacks (SCA), including differential power analysis (DPA), required for SESIP and PSA security assurance level 3 target.
  - Applicable to modular exponentiation, ECC scalar multiplication and ECDSA signature generation
- Arithmetic and modular operations such as addition, subtraction, multiplication, modular reduction, modular inversion, comparison, and Montgomery multiplication
- Built-in Montgomery domain inward and outward transformations
- AMBA AHB slave peripheral, accessible through 32-bit word single accesses only (otherwise an AHB bus error is generated, and write accesses are ignored)

52.3 PKA functional description

52.3.1 PKA block diagram

Figure 499. PKA block diagram



52.3.2 PKA internal signals

Table 440 lists the internal signals available at the PKA level, not necessarily available on product bonding pads.

Table 440. Internal input/output signals

Signal name	Signal type	Description
pka_hclk	Digital input	AHB bus clock
pka_it	Digital output	Public key accelerator IP global interrupt request
pka_itamp_out	Digital output	PKA internal tamper event signal to TAMP (XOR-ed), triggered when an unexpected fault occurs while PKA manipulates secrets, or when the programmed input point is not found on the input curve (ECDSA signature and ECC scalar multiplication only). This signal is asserted as soon as a fault is detected. When asserted, read access to PKA registers are reset to 0 and writes are ignored. The signal is de-asserted when PKA memory is cleared.

52.3.3 PKA reset and clocks

PKA is clocked on the AHB bus clock. When the PKA peripheral reset signal is released PKA_RAM is cleared automatically, taking 667 clock cycles. During this time the setting of bit EN in PKA_CR register is ignored. Once EN bit is set, refer to *Section 52.3.6* for details about PKA initialization sequence.

According to the security policy applied to the device, PKA RAM can also be reset following a tamper event. Refer to *Tamper detection and response* in the System security section (if applicable to this product).

52.3.4 PKA public key acceleration

Overview

Public key accelerator (PKA) is used to accelerate Rivest, Shamir and Adleman (RSA), Diffie-Hellman (DH) as well as ECC over prime field operations. Supported operand sizes is up to 4160 bits for RSA and DH, and up to 640 bits for ECC.

The PKA supports all non-singular elliptic curves defined over prime fields, that can be described with a short Weierstrass equation $y^2 = x^3 + ax + b$ (mod p). More information can be found in *Section 52.5.1*.

Note: Binary curves, Edwards curves and Curve25519 are not supported by the PKA.

A memory of 5336 bytes (667 words of 64 bits) called PKA RAM is used to provide initial data to the PKA, and to hold the results after computation is completed. Access is done though the PKA AHB interface.

PKA operating modes

The list of operations the PKA can perform is detailed in *Table 441* and *Table 442*, respectively, for integer arithmetic functions and prime field (Fp) elliptic curve functions.

Table 441. PKA integer arithmetic functions list

PKA_CR.MODE[5:0]		Performed operation	Reference
Hex	Binary		
0x01	000001	Montgomery parameter computation R^2 mod *n*	*Section 52.4.2*
0x0E	001110	Modular addition (A+B) mod n	*Section 52.4.3*
0x0F	001111	Modular subtraction (A-B) mod n	*Section 52.4.4*
0x10	010000	Montgomery multiplication (AxB) mod n	*Section 52.4.5*
0x00	000000	Modular exponentiation A^e mod n	*Section 52.4.6*
0x02	000010	Modular exponentiation A^e mod n (fast mode)	
0x03	000011	Modular exponentiation A^e mod n (protected)	*Section 52.4.6*
0x08	001000	Modular inversion A^{-1} mod n	*Section 52.4.7*
0x0D	001101	Modular reduction A mod n	*Section 52.4.8*
0x09	001001	Arithmetic addition A+B	*Section 52.4.9*
0x0A	001010	Arithmetic subtraction A-B	*Section 52.4.10*
0x0B	001011	Arithmetic multiplication AxB	*Section 52.4.11*
0x0C	001100	Arithmetic comparison (A=B, A>B, A<B)	*Section 52.4.12*
0x07	000111	RSA CRT exponentiation	*Section 52.4.13*

Table 442. PKA prime field (Fp) elliptic curve functions list

PKA_CR.MODE[5:0]		Performed operation	Reference
Hex	Binary		
0x28	101000	Point on elliptic curve Fp check	*Section 52.4.14*
0x20	100000	ECC scalar multiplication kP (protected)	*Section 52.4.15*
0x23	100011	ECC complete addition	*Section 52.4.18*
0x24	100100	ECDSA sign (protected)	*Section 52.4.16*
0x26	100110	ECDSA verification	*Section 52.4.17*
0x27	100111	ECC double base ladder	*Section 52.4.19*
0x2F	101111	ECC projective to affine	*Section 52.4.20*

Each of these operating modes has an associated code that has to be written to the MODE[5:0] bitfield in the PKA_CR register. If the application selects any value that is not documented below the write to MODE[5:0] bitfield is ignored, and an operation error (OPERRF) is triggered. When this happens, a new operation must be selected after the error is cleared.

Some operations in *Table 441* and *Table 442* are indicated as protected. Those operations are used when manipulating secret keys (modular exponentiation for RSA decryption, scalar multiplication and signature for ECC). Those secrets (protected against side channel attacks) and any intermediate values are automatically erased when PKA RAM is cleared at the end of the protected operations (BUSY goes low). They are also protected against side channel attacks.

Caution: For security reason it is very important to select protected modular exponentiation (MODE[5:0] = 0x3) when performing RSA decryption.

Montgomery space and fast mode operations

For efficiency reasons, the PKA internally performs modular multiply operations in the Montgomery domain, automatically performing inward and outward transformations.

As Montgomery parameter computation is time consuming, the application can decide to use a faster mode of operation during which the precomputed Montgomery parameter is supplied before starting the operation. The performance improvement is detailed in *Section 52.5.2: Computation times*.

The only operation using fast mode is the modular exponentiation (MODE[5:0] = 0x02).

52.3.5 Typical applications for PKA

Introduction

The PKA can be used to accelerate a number of public key cryptographic functions such as:

- RSA encryption and decryption
- RSA key finalization
- CRT-RSA decryption
- DSA and ECDSA signature generation and verification
- DH and ECDH key agreement

The following publications specify the listed functions:

- FIPS PUB 186-4, Digital Signature Standard (DSS), July 2013 by NIST
- PKCS #1, RSA Cryptography Standard, v1.5, v2.1 and v2.2. by RSA Laboratories
- IEEE1363-2000, IEEE Standard Specifications for Public-Key Cryptography, January 2000
- ANSI X9.62-2005, Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA), November 2005

Refer to these documents for full details. This document limits itself to describing principles.

RSA key pair

For the following RSA operations, a public key and a private key information are defined as follows:

- Alice transmits her public key (*n*, *e*) to Bob. Numbers *n* and *e* are very large positive integers.
- Alice keeps secret her private key d, also a very large positive integer. Alternatively, this private key can also be represented by a quintuple (p, q, dp, dq, qInv).

For more information, refer to the RSA specification.

RSA encryption/decryption principle

As recommended by the PKCS#1 specification, Bob, to encrypt message M using Alice's public key (*n*, *e*), must go through the following steps:

1. Compute the encoded message EM = ENCODE(M), where ENCODE is an encoding method.
2. Turn EM into an integer *m*, with $0 \leq m < n$ and (*m*, *n*) being coprimes.
3. Compute ciphertext $c = m^e \bmod n$.
4. Convert the integer c into a string ciphertext C.

Alice, to decrypt ciphertext c using her private key *d*, follows the steps indicated below:

1. Convert the ciphertext C to an integer ciphertext representative c.
2. If necessary, retrieve the prime factors (p, q) using (n, e, d) information, then compute phi = (p - 1) × (q - 1). Refer to NIST SP800-56B Appendix C for details.
3. Recover plaintext $m = c^d \bmod n = (m^e)^d \bmod n$. If the private key is the quintuple (p, q, dp, dq, qInv), then plaintext *m* is obtained by performing the operations:
   a) $m_1 = c^{dp} \bmod p$
   b) $m_2 = c^{dq} \bmod q$
   c) $h = qInv\ (m_1 - m_2) \bmod p$
   d) $m = m_2 + h\ q$
4. Convert the integer message representative *m* to an encoded message EM.
5. Recover message M = DECODE(EM), where DECODE is a decoding method.

To accelerate these operations with the PKA, use *Modular exponentiation* $A^e \bmod n$ if the private key is *d*, or *RSA CRT exponentiation* if the private key is the quintuple (p, q, dp, dq, qInv).

Note: The decoding operation and the conversion operations between message and integers are specified in PKCS#1 standard.

For the decryption process, the protected version of modular exponentiation (MODE[5:0] = 0x3) is strongly recommended for security reason. For the encryption process, the MODE[5:0] = 0x3 setting cannot be used as it requires the knowledge of the private key.

Elliptic curve selection

For the following ECC operations, the curve parameters are defined as follows:

- Curve corresponds to the elliptic curve field agreed among actors (Alice and Bob). Supported curves parameters are summarized in *Section 52.5.1: Supported elliptic curves*.
- G is the chosen elliptic curve base point (also known as generator), with a large prime order n (that is, $n \times G$ = identity element O).

ECDSA message signature generation

This section describes the ECDSA (elliptic curve digital signature algorithm) signature generation function principle.

Alice, to sign a message *m* using her private key integer d_A, goes through the following steps:

1. Calculate e = HASH(*m*), where HASH is a cryptographic hash function.
2. Let z be the L_n leftmost bits of e, where L_n is the bit length of the group order *n*.
3. Select a cryptographically secure random integer k where $0 < k < n$.
4. Calculate the curve point $(x_1, y_1) = k \times G$.
5. Calculate $r = x_1 \bmod n$. If $r = 0$ go back to step 3.
6. Calculate $s = k^{-1} (z + rd_A) \bmod n$. If $s = 0$ go back to step 3.
7. The signature is the pair (*r*, *s*).

The PKA accelerates the steps *4.* to *7.* by using:

- *ECDSA sign*, or
- All of the following operations:
  - *ECC Fp scalar multiplication* $k \times P$
  - *Modular reduction* A mod n
  - *Modular inversion* A^{-1} mod n
  - *Modular addition* and *Modular and Montgomery multiplication*

ECDSA signature verification

This section describes the ECDSA (elliptic curve digital signature algorithm) signature verification function principle.

Bob, to authenticate Alice's signature, must have a copy of her public key curve point Q_A. Bob can verify that Q_A is a valid curve point by going through the following steps:

1. Check that Q_A is not equal to the identity element O
2. Check that Q_A is on the agreed curve
3. Check that $n \times Q_A = O$.

Then Bob executes the following procedure:

1. Verify that r and s are integer in [1, n-1]
2. Calculate e = HASH(m), where HASH is the agreed cryptographic hash function
3. Let z be the L_n leftmost bits of e
4. Calculate $w = s^{-1}$ mod n
5. Calculate $u_1 = zw$ mod n and $u_2 = rw$ mod n
6. Calculate the curve point $(x_1, y_1) = u_1 \times G + u_2 \times Q_A$
7. The signature is valid if $r = x_1$ (mod n), it is invalid otherwise.

The PKA accelerates the steps *4.* to *7.* by using *ECDSA verification*.

52.3.6 PKA procedure to perform an operation

Enabling/disabling PKA

Setting the EN bit in the PKA_CR register enables the PKA peripheral. The PKA becomes available when the INITOK bit is set in the PKA_SR register. When EN = 0, the PKA peripheral is kept under reset, with its memory still accessible by the application through the AHB interface.

Note: When the PKA is in the process of clearing its memory, the EN bit cannot be set.

When setting the EN bit in PKA_CR, make sure that the MODE[5:0] bitfield value corresponds to an authorized PKA operation (see OPERRF in Section 52.3.7).

Clearing the EN bit while a calculation is in progress aborts the operation. In this case, the content of the PKA memory is not guaranteed, with the exception of the PKA modes 0x03, 0x20 and 0x24. For these operations, the PKA memory is cleared after abort, making the memory unavailable for 667 cycles. During this clearing time only the PKA registers can be accessed, with writes to EN bits ignored.

If the INITOK bit stays at 0, make sure that the RNG peripheral is clocked and properly initialized, then try to enable the PKA again.

Data formats

The format of the input data and the results in the PKA RAM are specified, for each operation, in *Section 52.4*.

Executing a PKA operation

To execute any of the supported PKA operations, proceed as follows:

1. Load initial data into the PKA internal RAM, which is located at address offset 0x400.
2. Write in the MODE[5:0] bitfield of PKA_CR register, specifying the operation which is to be executed and then assert the START bit, also in PKA_CR register.
3. Wait until the PROCENDF bit in the PKA_SR register is set, indicating that the computation is complete.
4. Read the result data from the PKA internal RAM, then clear PROCENDF bit by setting PROCENDFC bit in PKA_CLRFR.

Note: When PKA is busy (BUSY = 1), any access by the application to PKA RAM is ignored, and the RAMERRF flag is set in PKA_SR.

Selecting an illegal or unknown operation in step *2* triggers an OPERRF error and prevents the PROCENDF flag from being set and the procedure step *3* from being completed. See *Section 52.3.7* for details.

Using precomputed Montgomery parameters (PKA fast mode)

As explained in *Section 52.3.4*, when computing many operations with the same modulus, it can be beneficial for the application to compute the corresponding Montgomery parameter only once (see, for example, *Section 52.4.5*). This is referred to as *fast mode*.

To use the PKA peripheral in fast mode, follow this recommended procedure:

1. Load the modulus size and value information into the PKA RAM. Refer to *Section 52.5.1* for detail.
2. Configure the PKA_CR register to *Montgomery parameter computation* mode (MODE[5:0] = 0x1), then assert the START bit.
3. Wait until the PROCENDF bit in the PKA_SR register is set. Then, read the corresponding Montgomery parameter from the PKA memory and clear the PROCENDF bit by setting PROCENDFC in PKA_CLRFR.
4. Continue with the required PKA operation, loading the Montgomery information R2 mod *m* on top of the regular input data. Relevant addresses are provided in *Section 52.4*.

52.3.7 PKA error management

The following error flags in the PKA_SR register are set to indicate errors that may occur when using the PKA:

- **ADDRERRF** flags that the access to PKA RAM falls outside the expected range.
- **RAMERRF** flags an AHB access to the PKA RAM occurring while the PKA core uses the PKA RAM. While the flag is set, reads form the PKA RAM return 0 and writes are ignored.
- **OPERRF** flags that the selected operating mode using MODE[5:0] bitfield is not listed in PKA operating modes (see the bitfield description), or PKA is running in limited mode

(see LMF bit in PKA_SR). While the flag is set, writes to the MODE[5:0] bitfield are ignored.

For each error flag, the PKA generates an interrupt if enabled with the corresponding bit in the PKA_CR register (see *Section 52.6* for details).

The ADDRERRF, OPERRF, and RAMERRF error flags are cleared by setting the ADDRERRFC, OPERRFC, and RAMERRFC bit in PKA_CLRFR, respectively. The OPERRF error flag must be cleared before a new operation is initiated via the PKA_CR register.

Note: The PKA can be re-initialized at any moment by clearing the EN bit in the PKA_CR register.

52.4 PKA operating modes

52.4.1 Introduction

The operations supported by the PKA are detailed in the following subsections, which define the format of both input data and results stored in the PKA RAM.

Warning: Before starting any operation, ensure that all input parameters to the PKA are valid and consistent, as the PKA assumes their correctness. Additionally, input parameters must not exceed the operand sizes specified in the operation tables.

The following information applies to all PKA operations:

- The PKA core processes 64-bit words in its RAM. Accordingly, in the PKA section of this document, 'word' denotes a 64-bit unit.
- When writing an element as input in the PKA RAM, append an additional 64-bit word of zeros after the most significant word, except when the operand size is fixed at one word.
- All RAM storage addresses refer to the least significant word of the data. To obtain the actual address, the application must add the base address of the PKA to the indicated offset.
- Supported operand sizes are:
  - **ROS (RSA operand size)**: Data size is calculated as *(rsa_size / 64) + 1* words, where *rsa_size* is the modulus length in bits. For example, a 1024-bit RSA operand corresponds to 17 words (1088 bits).
  - **EOS (ECC operand size)**: Data size is *(ecc_size / 64) + 1* words, where *ecc_size* is the prime modulus length in bits. For example, a 192-bit ECC operand corresponds to four words (256 bits).

  Note: Both ROS and EOS include the additional zero word.
- Unless indicated otherwise, all operands in the tables are integers.

Note: Any fractional results in the formulas must be rounded to the nearest integer, as the PKA core processes data in 64-bit words.

Note: The maximum ROS is 66 words (4160-bit maximum exponent size), while the maximum EOS is 11 words (640-bit max operand size).

Example 1: Understanding endianness in PKA memory during ECC Fp scalar multiplication preparation

Write the *x*-coordinate of point P for an ECC P256 curve (EOS = 5 words) by placing the least significant bit at bit 0 of the word at address offset 0x578, and the most significant bit at bit 63 of the word at address offset 0x590. Then, write a zero word (0x0000000000000000) at address offset 0x598.

Example 2: Writing the curve coefficient '*a*' for ECC Fp scalar multiplication

To write the value *a* = -3 for a curve with a modulus length of 224 bits (rounded up to five 64-bit words), write the PKA memory with data as in the following table:

Table 443. Example of *'a'* curve coefficient for ECC Fp scalar

RAM offset	Value	Note
0x410	0x0000000000000001	Curve coefficient *a* sign without extra word
0x418	0x0000000000000011	Value of |*a*| LSB
0x420	0x0000000000000000	-
0x428	0x0000000000000000	-
0x430	0x0000000000000000	Value of |*a*| MSB
0x438	0x0000000000000000	Additional zero word

52.4.2 Montgomery parameter computation

This function is used to compute the Montgomery parameter (R^2 mod n) used by the PKA to convert operands into the Montgomery residue system representation. This operation can be very useful for fast mode operation because passing the Montgomery parameter as input saves the time for its computation.

Note: This operation also supports ECC curves when specifying the prime modulus length and EOS size.

The following table summarizes operation instructions for Montgomery parameter computation.

Table 444. Montgomery parameter computation

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x01	PKA_CR	6 bits
	Modulus length	(in bits, 0 ≤ value < 4160)	RAM@0x408	64 bits
	Modulus value *n*	(odd integer only, $n < 2^{4160}$)	RAM@0x1088	ROS
OUT	Result: R^2 mod *n*	-	RAM@0x620	

52.4.3 Modular addition

The following table summarizes operation instructions for modular addition operation that calculates *A* + *B* mod *n*.

Table 445. Modular addition

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x0E	PKA_CR	6 bits
	Operand length	(in bits, not null)	RAM@0x408	64 bits
	Operand *A*	($0 \le A < n$)	RAM@0xA50	ROS
	Operand *B*	($0 \le B < n$)	RAM@0xC68	
	Modulus value *n*	($n < 2^{4160}$)	RAM@0x1088	
OUT	Result: $A + B \bmod n$	($0 \le$ result $< n$)	RAM@0xE78	

52.4.4 Modular subtraction

Modular subtraction operation calculates:

- ***A* - *B* mod *n***, when $A \ge B$
- ***A* + *n* - *B* mod *n***, when $A < B$

The following table summarizes the operation instructions.

Table 446. Modular subtraction

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x0F	PKA_CR	6 bits
	Operand length	(in bits, not null)	RAM@0x408	64 bits
	Operand *A*	($0 \le A < n$)	RAM@0xA50	ROS
	Operand *B*	($0 \le B < n$)	RAM@0xC68	
	Modulus value *n*	($n < 2^{4160}$)	RAM@0x1088	
OUT	Result: $A - B \bmod n$	($0 \le$ result $< n$)	RAM@0xE78	

52.4.5 Modular and Montgomery multiplication

To be more efficient when performing a sequence of multiplications, the PKA accelerates a multiplication that has at least one input in the Montgomery domain. The two main uses of this operation are:

- Map a value from natural domain to Montgomery domain and vice-versa
- Perform a modular multiplication $A \times B \bmod n$

The method to perform these operations is described next. The *x* function is this operation, and the *A*, *B*, and *C* operands are in the natural domain.

1. Inward or outward conversion to or from the Montgomery domain
   a) Assuming that *A* is an integer in the natural domain:
      – Compute *r2modn* using *Montgomery parameter computation*.
      – The result of $AR = A \times r2modn \bmod n$ is *A* in the Montgomery domain.
   b) Assuming that *BR* is an integer in the Montgomery domain:
      – The result of $B = BR \times 1 \bmod n$ is *B* in the natural domain.
      – Similarly, the *AR* value computed in *a)* can be converted into the natural domain by computing $A = AR \times 1 \bmod n$.
2. Simple modular multiplication $A \times B \bmod n$
   a) Compute *r2modn* using *Montgomery parameter computation*.
   b) Compute $AR = A \times r2modn \bmod n$. The output is in the Montgomery domain.
   c) Compute $AB = AR \times B \bmod n$. The output is in the natural domain.
3. Multiple modular multiplication $A \times B \times C \bmod n$
   a) Compute *r2modn* using *Montgomery parameter computation*.
   b) Compute $AR = A \times r2modn \bmod n$. The output is in the Montgomery domain.
   c) Compute $BR = B \times r2modn \bmod n$. The output is in the Montgomery domain.
   d) Compute $ABR = AR \times BR \bmod n$. The output is in the Montgomery domain.
   e) Compute $CR = C \times r2modn \bmod n$. The output is in the Montgomery domain.
   f) Compute $ABCR = ABR \times CR \bmod n$. The output is in the Montgomery domain.
   g) (optional) Repeat the two previous steps if multiplying with additional operands is required.
   h) Compute $ABC = ABCR \times 1 \bmod n$ to retrieve the result in the natural domain.

The following table summarizes the operation instructions.

Table 447. Montgomery multiplication

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x10	PKA_CR	6 bits
	Operand length	(in bits, not null)	RAM@0x408	64 bits
	Operand *A*	($0 \le A < n$)	RAM@0xA50	ROS
	Operand *B*	($0 \le B < n$)	RAM@0xC68	
	Modulus value *n*	(odd integer only, $n < 2^{4160}$)	RAM@0x1088	
OUT	Result: $A \times B \bmod n$(1)	-	RAM@0xE78	

1. Result in Montgomery domain or in natural domain, depending upon the inputs nature (see examples *2* and *3*).

52.4.6 Modular exponentiation

Modular exponentiation operation is commonly used to perform a single-step RSA operation. It calculates of $A^e \bmod n$.

RSA operation involving public information (RSA encryption) can use the normal or fast mode detailed on *Table 448* and *Table 449*. RSA operation involving secret information (RSA decryption) must use the protected mode detailed on *Table 450*, for security reason.

Note: Once this operation is started PKA control register and PKA memory is no more available. Access is restored once BUSY bit is set to 0 by the PKA.

When this operation completes with errors due to unexpected hardware events a PKA tamper event is triggered to TAMP peripheral, and access to PKA RAM becomes blocked until erased by hardware.

Note: When MODE[5:0] = 0x03, the error code is not affected when PKA automatically clears the PKA RAM at the end of this protected operation. When the error output equals 0xD60D, the result output is not affected either.

Operation instructions for modular exponentiation are summarized in *Table 448* (normal mode), *Table 449* (fast mode), and *Table 450* (protected mode). Fast mode usage is explained in *Section 52.3.6*.

Table 448. Modular exponentiation (normal mode)

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x00	PKA_CR	6 bits
IN	Exponent length	(in bits, not null)	RAM@0x400	64 bits
	Operand length	(in bits, not null)	RAM@0x408	
IN/OUT	Operand *A* (base of exponentiation)	($0 \le A < n$)	RAM@0xC68	ROS
IN	Exponent *e*	($0 \le e < n$)	RAM@0xE78	
	Modulus value *n*	(odd integer only, $n < 2^{4160}$)	RAM@0x1088	
OUT	Result: A^e mod *n*	($0 \le$ result $< n$)	RAM@0x838	

Table 449. Modular exponentiation (fast mode)

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x02	PKA_CR	6 bits
IN	Exponent length	(in bits, not null)	RAM@0x400	64 bits
	Operand length	(in bits, not null)	RAM@0x408	
IN/OUT	Operand *A* (base of exponentiation)	($0 \le A < n$)	RAM@0xC68	ROS
IN	Exponent *e*	($0 \le e < n$)	RAM@0xE78	
	Modulus value *n*	(odd integer only, $n < 2^{4160}$)	RAM@0x1088	
IN/OUT	Montgomery parameter R^2 mod *n*	(mandatory)	RAM@0x620	
OUT	Result: A^e mod *n*	($0 \le$ result $< n$)	RAM@0x838	

Table 450. Modular exponentiation (protected mode)

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x03	PKA_CR	6 bits
	Exponent *e* length	(in bits, not null)	RAM@0x400	64 bits
	Modulus or operand length	(in bits, not null)	RAM@0x408	
	Operand *A* (base of exponentiation)	($0 \le A < n$)	RAM@0x16C8	ROS
	Exponent *e*	($0 \le e < n$)	RAM@0x14B8	
	Modulus value *n*	(odd integer only, $n < 2^{4096}$)	RAM@0x0838	
	Phi value(1)	-	RAM@0x0C68	
OUT	Result: A^e mod *n*	($0 \le$ result $< n$)	RAM@0x838	
ERROR	Error A^e mod *n*	No errors: 0xD60D Errors: 0xCBC9	RAM@0x1298	64 bits

1. Euler totient function of n[1] with phi = (p - 1) × (q - 1), where p and q are prime factors of modulus n (see NIST SP800-56B Appendix C or *RSA encryption/decryption principle* for details). As optimization it is recommended to keep phi information as part of the key pair generation. Alternative is to store the key as (p, q, e, d) instead of (N, e, d, *phi*), in this case, to derive N and *phi* using PKA arithmetic multiplier: $N = p \times q$, and *phi* = (p - 1) × (q - 1).

52.4.7 Modular inversion

The modular inversion operation calculates multiplicative inverse A^{-1} mod n. If the modulus n is prime, for all values of A ($1 \le A < n$) modular inversion output is valid. If the modulus n is not prime, A has an inverse only if the greatest common divisor between A and n is 1.

If the operand A is a divisor of the modulus n the result is a multiple of a factor of n.

The following table summarizes the operation instructions.

Table 451. Modular inversion

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x08	PKA_CR	6 bits
	Operand length	(in bits, not null)	RAM@0x408	64 bits
	Operand *A*	($0 \le A < n$)	RAM@0xA50	ROS
	Modulus value *n*	(odd integer only, $n < 2^{4160}$)	RAM@0xC68	
OUT	Result: A^{-1} mod *n*	0 < result < *n*	RAM@0xE78	

52.4.8 Modular reduction

The modular reduction operation calculates the remainder of A divided by n.

The following table summarizes the operation instructions.

Table 452. Modular reduction

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x0D	PKA_CR	6 bits
	Operand length	(in bits, not null)	RAM@0x400	64 bits
	Modulus length	(in bits, 8 < value < 4160)	RAM@0x408	
	Operand *A*	($0 \le A < 2n < 2^{4160}$)	RAM@0xA50	ROS
	Modulus value *n*	(odd integer only, $n < 2^{4160}$)	RAM@0xC68	
OUT	Result *A* mod *n*	($0 <$ result $< n$)	RAM@0xE78	

52.4.9 Arithmetic addition

The arithmetic addition operation calculates *A* + *B*.

The following table summarizes the operation instructions.

Table 453. Arithmetic addition

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x09	PKA_CR	6 bits
	Operand length *M*	(in bits, not null)	RAM@0x408	64 bits
	Operand *A*	($0 \le A < 2^M$)	RAM@0xA50	ROS
	Operand *B*	($0 \le B < 2^M$)	RAM@0xC68	
OUT	Result: *A* + *B*	($0 \le$ result $< 2^{M+1}$)	RAM@0xE78	ROS + 1

52.4.10 Arithmetic subtraction

The arithmetic subtraction operation calculates:

- ***A* - *B***, when $A \ge B$
- $\boldsymbol{A + 2^{int(M / 32) \times 32 + 1} - B}$, when $A < B$ and the residue of $M / 32$ is greater than zero
- $\boldsymbol{A + 2^{int(M / 32) \times 32} - B}$, when $A < B$ and the residue of $M / 32$ is zero

For the last two list items, the 64-bit word following the most significant word of the output is 0xFFFF FFFF FFFF FFFF, as the result is negative.

The following table summarizes the operation instructions.

Table 454. Arithmetic subtraction

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x0A	PKA_CR	6 bits
	Operand length *M*	(in bits, not null)	RAM@0x408	64 bits
	Operand *A*	($0 \le < A < 2^M$)	RAM@0xA50	ROS
	Operand *B*	($0 \le < B < 2^M$)	RAM@0xC68	
OUT	Result: *A* - *B*	($0 \le$ result $< 2^M$)	RAM@0xE78	

52.4.11 Arithmetic multiplication

The arithmetic multiplication operation calculates A × B.

The following table summarizes the operation instructions.

Table 455. Arithmetic multiplication

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x0B	PKA_CR	6 bits
	Operand length *M*	(in bits, not null)	RAM@0x408	64 bits
	Operand *A*	($0 \le A < 2^M$)	RAM@0xA50	ROS
	Operand *B*	($0 \le B < 2^M$)	RAM@0xC68	
OUT	Result: *A* × *B*	($0 \le$ result $< 2^{2M}$)	RAM@0xE78	2xROS

52.4.12 Arithmetic comparison

The arithmetic comparison operation yields the following results:

- **0xED2C**, when *A* = *B*
- **0x7AF8**, when *A* > *B*
- **0x916A**, when *A* < *B*

The following table summarizes the operation instructions.

Table 456. Arithmetic comparison

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x0C	PKA_CR	6 bits
	Operand length *M*	(in bits, not null)	RAM@0x408	64 bits
	Operand *A*	($0 \le A < 2^M$)	RAM@0xA50	ROS
	Operand *B*	($0 \le B < 2^M$)	RAM@0xC68	
OUT	If result *A* = *B*	0xED2C, 0x7AF8, or 0x916A	RAM@0xE78	64 bits

52.4.13 RSA CRT exponentiation

For efficiency many popular crypto libraries such as OpenSSL RSA use the following optimization for decryption and signing based on the chinese remainder theorem (CRT):

- *p* and *q* are precomputed primes, stored as part of the private key
- $d_P = d \bmod (p - 1)$
- $d_Q = d \bmod (q - 1)$ and
- $q_{inv} = q^{-1} \bmod p$

These values allow the recipient to compute the exponentiation m = A^d (mod pq) more efficiently as follows:

- $m_1 = A^{dP} \bmod p$
- $m_2 = A^{dQ} \bmod q$
- $h = q_{inv} (m_1 - m_2) \bmod p$, with $m_1 > m_2$
- $m = m_2 + hq \bmod pq$

Operation instructions for computing CRT exponentiation A^d mod pq are summarized in *Table 457*.

Table 457. CRT exponentiation

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x07	PKA_CR	6 bits
IN	Operand length	(in bits, not null)	RAM@0x408	64 bits
IN	Operand d_P	($0 < d_P < 2^{M/2}$)	RAM@0x730	ROS / 2
	Operand d_Q	($0 < d_Q < 2^{M/2}$)	RAM@0xE78	
	Operand q_{inv}	($0 < q_{inv} < 2^{M/2}$)	RAM@0x948	
	Prime p(1)	($0 < p < 2^{M/2}$)	RAM@0xB60	
	Prime q(1)	($0 < q < 2^{M/2}$)	RAM@0x1088	
IN	Operand A	($0 < A < 2^M$)	RAM@0x12A0	ROS
OUT	Result: A^d mod pq	($0 \leq$ result < pq)	RAM@0x838	

1. Must be different from 2.

52.4.14 Point on elliptic curve Fp check

This operation checks whether a given point P (x, y) satisfies or not the curves over prime fields equation $y^2 = (x^3 + ax + b) \bmod p$, where a and b are elements of the curve.

The following table summarizes the operation instructions.

Table 458. Point on elliptic curve *Fp* check

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x28	PKA_CR	6 bits
	Modulus length	(in bits, not null, 8 < value < 640)	RAM@0x408	64 bits
	Curve coefficient *a* sign	0x0: positive 0x1: negative	RAM@0x410	
	Curve coefficient |*a*|	(absolute value, |*a*| < *p*)	RAM@0x418	EOS
	Curve coefficient *b*	(|*b*| < *p*)	RAM@0x520	
	Curve modulus value *p*	(odd integer prime, 0 < *p* < 2640)	RAM@0x470	
	Point P coordinate *x*	(*x* < p)	RAM@0x578	
	Point P coordinate *y*	(*y* < *p*)	RAM@0x5D0	
	Montgomery parameter R^2 mod *n*	-	RAM@0x4C8	
OUT	Result: point *P* on curve	0xD60D: point on curve 0xA3B7: point not on curve 0xF946: *x* or *y* coordinate is not smaller than modulus *p*	RAM@0x680	64 bits

52.4.15 ECC Fp scalar multiplication

This operation calculates $k \times P\,(x_P, y_P)$, where *P* is a point on a curve over prime fields and *x* is the elliptic curve scalar point multiplication. The result is a point that belongs to the same curve or a point at infinity.

Note: Once this operation is started PKA control register and PKA memory is no more available. Access is restored once BUSY bit is set to 0 by the PKA.

When this operation completes with errors due to unexpected hardware events, a PKA tamper event is triggered to TAMP peripheral, and access to PKA RAM becomes blocked until erased by hardware. PKA tamper is also triggered when the programmed input point is not found on the input ECC curve. PKA operation *point on elliptic curve* can be used to avoid this.

The following table summarizes the operation instructions.

Table 459. ECC Fp scalar multiplication

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x20	PKA_CR	6 bits
IN	Curve prime order *n* length	(in bits, not null,)	RAM@0x400	64 bits
	Curve modulus *p* length	(in bits, not null, 8 < value < 640)	RAM@0x408	
	Curve coefficient *a* sign	0x0: positive 0x1: negative	RAM@0x410	

Table 459. ECC Fp scalar multiplication (continued)

Parameters with direction		Value (note)	Storage	Size
IN	Curve coefficient |*a*|	(absolute value, |*a*| < *p*)	RAM@0x418	EOS
	Curve coefficient *b*	(positive integer)	RAM@0x520	
	Curve modulus value *p*	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x1088	
	Scalar multiplier *k*	($0 \leq k < n$)	RAM@0x12A0	
	Point P coordinate x_P	($x_P < p$)	RAM@0x578	
	Point P coordinate y_P	($y_P < p$)	RAM@0x470	
	Curve prime order *n*	(integer prime)	RAM@0xF88	
OUT	Result: *k* × *P* coordinate *x'*	(result < *p*)	RAM@0x578	EOS
	Result: *k* × *P* coordinate *y'*	(result < *p*)	RAM@0x5D0	
ERROR	Error *k* × *P*	No errors: 0xD60D Errors: 0xCBC9	RAM@0x680	64 bits

When performing this operation the following special cases must be noted:

- For *k* = 0 this function returns a point at infinity (0, 0) if curve parameter *b* is nonzero, (0, 1) otherwise. For *k* different from 0 it might happen that a point at infinity is returned. When the application detects this behavior, a new computation must be carried out.
- For *k* < 0 (that is, a negative scalar multiplication is required) the multiplier absolute value
  k = |-*k*| must be provided to the PKA. After the computation completion, the formula -*P* = (*x*, -*y*) can be used to compute the *y* coordinate of the effective final result (the *x* coordinate remains the same).

Note: The error code is not affected when PKA automatically clears the PKA RAM at the end of this protected operation. When the error output equals 0xD60D, the result output is not affected either.

52.4.16 ECDSA sign

Table 460 (input parameters) and *Table 461* (output parameters) summarize the ECDSA signing operation.

Upon an output error different from 0xD60D, the application must generate a new *k* and repeat the ECDSA sign operation.

Note: As long as this operation is ongoing (BUSY = 1), the PKA control register and the PKA memory are not accessible. The access is restored when the PKA clears the BUSY bit.

The operation completion with errors due to unexpected hardware events triggers a PKA tamper event in the TAMP peripheral and blocks access to the PKA RAM until the hardware erases the event. Missing programmed input point on the input ECC curve also triggers a PKA tamper event. The use of the *point on elliptic curve* PKA operation prevents this.

Table 460. ECDSA sign - Inputs

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x24	PKA_CR	6 bits
	Curve prime order *n* length (*nlen*)	(in bits, not null)	RAM@0x400	64 bits
	Curve modulus *p* length	(in bits, 8 < value < 640)	RAM@0x408	
	Curve coefficient *a* sign	0x0: positive 0x1: negative	RAM@0x410	
	Curve coefficient |*a*|	(absolute value, |*a*| < *p*)	RAM@0x418	EOS
	Curve coefficient *b*	(positive integer)	RAM@0x520	
	Curve modulus value *p*	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x1088	
	Integer *k*(1)	($0 \leq k <$ n)	RAM@0x12A0	
	Curve base point *G* coordinate *x*	($x < p$)	RAM@0x578	
	Curve base point *G* coordinate *y*	($y < p$)	RAM@0x470	
	Hash of message *z*	(hash size equal to *nlen*)(2)	RAM@0xFE8	
	Private key *d*	($0 < d$)	RAM@0xF28	
	Curve prime order *n*	(integer prime)	RAM@0xF88	

1. This integer is usually a cryptographically secure random number, but in some cases *k* can be deterministically generated.
2. To match the hash parameter size to the length of the curve prime order *n,* pad the result with zeros or use the hash truncation.

Table 461. ECDSA sign - Outputs

Parameters with direction		Value (note)	Storage	Size
OUT	Signature part *r*	($0 < r < n$)	RAM@0x730	EOS
	Signature part *s*	($0 < s < n$)	RAM@0x788	
ERROR	Result of signature	0xD60D: successful computation, no error 0xCBC9: failed computation 0xA3B7: signature part r is equal to 0 0xF946: signature part s is equal to 0	RAM@0xFE0	64 bits

Note: The error code is not affected when the PKA automatically clears the PKA RAM at the end of this protected operation. The error output 0xD60D does not affect the result output either.

Extended ECDSA support

The PKA also supports the extended ECDSA signature, which accepts the same inputs as the standard ECDSA signature (see *Table 460*) and produces similar outputs (see *Table 461*), with the addition of the point *kG* coordinates. The following table defines these additional outputs.

Table 462. Extended ECDSA sign - additional outputs

Parameters with direction		Value (note)	Storage	Size
OUT	Curve point *kG* coordinate x_1	($0 \le x_1 < p$)	RAM@0x1400	EOS
	Curve point *kG* coordinate y_1	($0 \le y_1 < p$)	RAM@0x1458	

52.4.17 ECDSA verification

Table 463 (input parameters) and *Table 464* (output parameters) summarize the ECDSA verification operation.

Upon an output error different from 0xD60D, the signature is not verified.

Table 463. ECDSA verification - inputs

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x26	PKA_CR	6 bits
	Curve prime order *n* length (*nlen*)	(in bits, not null)	RAM@0x408	64 bits
	Curve modulus *p* length	(in bits, not null, 8 < value < 640)	RAM@0x4C8	
	Curve coefficient *a* sign	0x0: positive 0x1: negative	RAM@0x468	
	Curve coefficient |*a*|	(absolute value, $\|a\| < p$)	RAM@0x470	EOS
	Curve modulus value *p*	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x4D0	
	Curve base point *G* coordinate *x*	($x < p$)	RAM@0x678	
	Curve base point *G* coordinate *y*	($y < p$)	RAM@0x6D0	
	Public-key curve point *Q* coordinate x_Q	($x_Q < p$)	RAM@0x12F8	
	Public-key curve point *Q* coordinate y_Q	($y_Q < p$)	RAM@0x1350	
	Signature part *r*	($0 < r < n$)	RAM@0x10E0	
	Signature part *s*	($0 < s < n$)	RAM@0xC68	
	Hash of message *z*	(hash size equal to *nlen*)(1)	RAM@0x13A8	
	Curve prime order *n*	(integer prime)	RAM@0x1088	

1. To match the hash parameter size to the length of the curve prime order *n,* pad the result with zeros or use the hash truncation.

Table 464. ECDSA verification - outputs

Parameters with direction		Value (note)	Storage	Size
OUT	Result: ECDSA verify	– 0xD60D: valid signature – 0xA3B7: invalid signature	RAM@0x5D0	64 bits
	Computed signature part *r*	– ($0 < r < n$)	RAM@0x578	EOS

52.4.18 ECC complete addition

The ECC complete addition operation calculates the sum of two points on an elliptic curve.

Operation instructions are summarized in *Table 465*.

Note: The two input points and the resulting point are represented in Jacobian coordinates (X,Y,Z). To input a point in affine coordinates (x, y) conversion (X, Y, Z) = (x, y, 1) can be used. To convert resulting point to Jacobian coordinates conversion (x, y)= (X / Z^2, Y / Z^3) can be used.

Table 465. ECC complete addition

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x23	PKA_CR	6 bits
	Curve modulus *p* length	(in bits, not null, 8 < value < 640)	RAM@0x408	64 bits
	Curve coefficient *a* sign	– 0x0: positive 0x1: negative	RAM@0x410	
	Curve modulus value *p*	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x470	EOS
	Curve coefficient $\|a\|$	(absolute value, $\|a\| < p$)	RAM@0x418	
	First point P coordinate *X*	($x < p$)	RAM@0x628	
	First point P coordinate *Y*	($y < p$)	RAM@0x680	
	First point P coordinate *Z*	($z < p$)	RAM@0x6D8	
	Second point Q coordinate *X*	($x < p$)	RAM@0x730	
	Second point Q coordinate *Y*	($y < p$)	RAM@0x788	
	Second point Q coordinate *Z*	($z < p$)	RAM@0x7E0	
OUT	Result coordinate *X*	($x < p$)	RAM@0xD60	
	Result coordinate *Y*	($y < p$)	RAM@0xDB8	
	Result coordinate *Z*	($z < p$)	RAM@0xE10	

52.4.19 ECC double base ladder

ECC double base ladder operation consists in the computation of $k \times P + m \times Q$, where (P, Q) are two points on an elliptic curve and (k, m) are two scalars. Operation instructions are summarized in *Table 466*.

If the resulting point is the point at infinity (error code 0xA3B7), the resulting coordinate equals to (0, 0).

Note: The two input points are represented in Jacobian coordinates (X, Y , Z). To input a point in affine coordinates (x, y) conversion (X, Y, Z) = (x, y, 1) can be used. The result is represented in affine coordinates (x ,y)

This operation requires the input point Z coordinate to be equal to 1 (input point represented in affine coordinates).

Table 466. ECC double base ladder

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x27	PKA_CR	6 bits
	Curve prime order *n* length	(in bits, not null)	RAM@0x400	64 bits
	Curve modulus *p* length	(in bits, not null, 8 < value < 640)	RAM@0x408	
	Curve coefficient *a* sign	– 0x0: positive – 0x1: negative	RAM@0x410	EOS
	Curve coefficient |*a*|	(absolute value, |a| < p)	RAM@0x418	
	Curve modulus value *p*	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x470	
	Integer *k*	($0 < k < 2^{640}$)	RAM@0x520	
	Integer *m*	($0 < m < 2^{640}$)	RAM@0x578	
	First point *P* coordinate *X*	($x < p$)	RAM@0x628	
	First point *P* coordinate *Y*	($y < p$)	RAM@0x680	
	First point *P* coordinate *Z*	($z < p$)	RAM@0x6D8	
	Second point *Q* coordinate *X*	($x < p$)	RAM@0x730	
	Second point *Q* coordinate *Y*	($y < p$)	RAM@0x788	
	Second point *Q* coordinate *Z*	($z < p$)	RAM@0x7E0	
OUT	Result coordinate *x*	($x < p$)	RAM@0x578	EOS
	Result coordinate *y*	(y < *p*)	RAM@0x5D0	
	Error code	– Point not at infinity: 0xD60D – Point at infinity: 0xA3B7	RAM@0x520	64 bits

52.4.20 ECC projective to affine

ECC projective to affine operation calculates the conversion between the representation of a point P in homogeneous projective coordinates and the representation of the point P in affine coordinates. Namely, if the point is represented by the triple (*X*, *Y*, *Z*), it calculates the affine coordinates (*x*, *y*) = (*X* / *Z*, *Y* / *Z*).

All the operations are performed modulo the modulus p of the curve, which the point belongs to. If the resulting point is the point at infinity (error code 0xA3B7), resulting coordinate equals (0,0).

Operation instructions are summarized in *Table 467*.

Table 467. ECC projective to affine

Parameters with direction		Value (note)	Storage	Size
IN	MODE[5:0]	0x2F	PKA_CR	6 bits
	Curve modulus *p* length	(in bits, 8 < value < 640)	RAM@0x408	64 bits
	Curve modulus value *p*	(odd integer prime, $0 < p < 2^{640}$)	RAM@0x470	EOS
	Point P coordinate *X* (projective)	($x < p$)	RAM@0xD60	
	Point P coordinate *Y* (projective)	($y < p$)	RAM@0xDB8	
	Point P coordinate *Z* (projective)	($z < p$)	RAM@0xE10	
	Montgomery parameter R^2 mod *n*	-	RAM@0x4C8	
OUT	Point P coordinate *x* (affine)	($x < p$)	RAM@0x578	
	Point P coordinate *y* (affine)	($y < p$)	RAM@0x5D0	
ERROR	Error code	Point not at infinity: 0xD60D Point at infinity: 0xA3B7	RAM@0x680	64 bits

52.5 Example of configurations and processing times

52.5.1 Supported elliptic curves

The PKA supports all non-singular elliptic curves defined over prime fields. Those curves can be described with a short Weierstrass equation, $y^2 = x^3 + ax + b$ (mod p).

Note: Binary curves, Edwards curves and Curve25519 are not supported by the PKA.
The maximum supported operand size for ECC operations is 640 bits.

When publishing the ECC domain parameters of those elliptic curves, standard bodies define the following parameters:

- the prime integer *p*, used as the modulus for all point arithmetic in the finite field *GF(p)*
- the (usually prime) integer *n*, the order of the group generated by *G*, defined below
- the base point of the curve *G*, defined by its coordinates (*Gx*, *Gy*)
- the integers *a* and *b*, coefficients of the short Weierstrass equation.

For the last bullet, when standard bodies define *a* as negative, PKA supports two representations:

1. ***a* defined as *p*-|*a*|** in the finite field *GF(p)*, for example ***p*-3**:
   Curve coefficient p = 0xFFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFF**F**
   Curve coefficient a sign= 0x0 (positive)
   Curve coefficient a = 0xFFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFF**C**
2. ***a* defined as negative**, for example **-3**:
   Curve coefficient p = 0xFFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF 00000000 FFFFFFFF FFFFFFFF
   Curve coefficient a sign= 0x**1** (negative)
   Curve coefficient a = 0x00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 0000000**3**

Table 468 summarizes the family of curves supported by PKA for ECC operations.

Table 468. Family of supported curves for ECC operations

Curve name	Standard	Reference	
P-192	NIST	*Digital Signature Standard (DSS)*, NIST FIPS 186-4	
P-224			
P-256			
P-384			
P-521			
brainpoolP224r1, brainpoolP224t1	IETF	– *Brainpool Elliptic Curves*, IETF RFC 5639 – *Brainpool Elliptic Curves for the Internet Key Exchange (IKE) Group Description Registry*, IETF RFC 6932	https://tools.ietf.org
brainpoolP256r1, brainpoolP256t1			
brainpoolP320r1, brainpoolP320t1			
brainpoolP384r1, brainpoolP384t1			
brainpoolP512r1, brainpoolP512t1			
secp192k1, secp192r1	SEC	*Standards for Efficient Cryptography SEC 2 curves*	https://www.secg.org
secp224k1, secp224r1			
secp256k1, secp256r1			
secp384r1			
secp521r1			
Recommended curve parameters for public key cryptographic algorithm SM2	OSCCA	– *Public key cryptographic algorithm SM2 based on elliptic curves*, Organization of State Commercial Administration of China OSCCA SM2, December 2010 – *Digital signatures - Part 3 Discrete logarithm based mechanisms*, ISO/IEC 14888-3, November 2018	

52.5.2 Computation times

The following tables summarize the PKA computation times, expressed in AHB clock cycles.

Table 469. Modular exponentiation

Exponent length (in bits)	Mode	Modulus length (in bits)			
		1024	2048	3072	4096
3	Normal	66000	242000	310000	503000
	Fast	7000	23000	48000	82000
17	Normal	69000	253000	333000	543000
	Fast	24000	34000	72000	123000
2^{16} + 1	Normal	82000	296000	427000	705000
	Fast	23000	78000	165000	284000
1024	Protected	2960000	-	-	-
	Normal	1725000	-	-	-
	Fast	1666000	-	-	-
	CRT(1)	595000	-	-	-
2048	Protected	-	17643000	-	-
	Normal	-	11553000	-	-
	Fast	-	11334000	-	-
	CRT(1)	-	3454000	-	-
3072	Protected	-	-	53390000	-
	Normal	-	-	36280000	-
	Fast	-	-	36019000	-
	CRT(1)	-	-	10328000	-
4096	Protected	-	-	-	119824000
	Normal	-	-	-	82619000
	Fast	-	-	-	82198000
	CRT(1)	-	-	-	23061000

1. CRT stands for chinese remainder theorem optimization (MODE[5:0] bitfield= 0x07).

Table 470. ECC scalar multiplication(1)

Modulus length (in bits)							
160	192	256	320	384	512	521	640
-	742000	1220000	2116000	3031000	5981000	7844000	10200000

1. These times depend on the number of 1s included in the scalar parameter, and include the computation of Montgomery parameter R2.

Table 471. ECDSA signature average computation time(1) (2)

Modulus length (in bits)							
160	**192**	**256**	**320**	**384**	**512**	**521**	**640**
-	633000	1087000	1708000	2548000	4856000	5978000	8030000

1. These values are average execution times of random moduli of given length, as they depend upon the length and the value of the modulus.
2. The execution time for the moduli that define the finite field of NIST elliptic curves is shorter than that needed for the moduli used for Brainpool elliptic curves or for random moduli of the same size.

Table 472. ECDSA verification average computation times

Modulus length (in bits)							
160	**192**	**256**	**320**	**384**	**512**	**521**	**640**
566000	676000	1221000	1957000	2967000	5825000	7096000	9773000

Table 473. ECC double base ladder average computation times

Modulus length (in bits)							
160	**192**	**256**	**320**	**384**	**512**	**521**	**640**
541000	644000	1152000	1873000	2814000	5506000	6753000	9350000

Table 474. ECC projective to affine average computation times

Modulus length (in bits)						
160	**192**	**256**	**320**	**384**	**512**	**640**
24700	32000	56000	90000	144000	271000	326000

Table 475. ECC complete addition average computation times

Modulus length (in bits)						
160	**192**	**256**	**320**	**384**	**512**	**640**
8000	8200	10500	13000	19400	29600	39000

Table 476. Point on elliptic curve Fp check average computation times

Modulus length (in bits)							
160	**192**	**256**	**320**	**384**	**512**	**521**	**640**
2200	2240	2870	3550	4400	6260	7460	8500

Table 477. Montgomery parameters average computation times(1)

Modulus length (in bits)							
192	**256**	**320**	**512**	**1024**	**2048**	**3072**	**4096**
4200	5200	5900	12600	35500	126500	290000	420000

1. The computation times depend upon the length and the value of the modulus, hence these values are average execution times of random moduli of given length.

52.6 PKA interrupts

There are four individual maskable interrupt sources generated by the public key accelerator, signaling the following events:

1. PKA unsupported operation error (OPERRF), see *Section 52.3.7*
2. Access to unmapped address (ADDRERRF), see *Section 52.3.7*
3. PKA RAM access while PKA operation is in progress (RAMERRF), see *Section 52.3.7*
4. PKA end of operation (PROCENDF)

The interrupt sources are connected to the same global interrupt request signal pka_it.

The user can enable or disable above interrupt sources individually by changing the mask bits in the *PKA control register (PKA_CR)*. Setting the appropriate mask bit to 1 enables the interrupt. The status of the individual interrupt events can be read from the PKA status register (PKA_SR), and it is cleared in PKA_CLRFR register.

Table 478 gives a summary of the available features.

Table 478. PKA interrupt requests

Acronym	Event	Event flag	Enable control bit	Clear method
PKA	Unsupported operation	OPERRF	OPERRIE	Set OPERRFC bit
	Access to unmapped address error	ADDRERRF	ADDRERRIE	Set ADDRERRFC bit
	PKA RAM access error	RAMERRF	RAMERRIE	Set RAMERRFC bit
	PKA end of operation	PROCENDF	PROCENDIE	Set PROCENDFC bit

52.7 PKA registers

52.7.1 PKA control register (PKA_CR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPERRIE	ADDRERRIE	RAMERRIE	Res.	PROCENDIE	Res.
										rw	rw	rw		rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	MODE[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	START	EN
		rw	rw	rw	rw	rw	rw							rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **OPERRIE**: Operation error interrupt enable
0: No interrupt is generated when OPERRF flag is set in PKA_SR.
1: An interrupt is generated when OPERRF flag is set in PKA_SR.

Bit 20 **ADDRERRIE**: Address error interrupt enable
0: No interrupt is generated when ADDRERRF flag is set in PKA_SR.
1: An interrupt is generated when ADDRERRF flag is set in PKA_SR.

Bit 19 **RAMERRIE**: RAM error interrupt enable
0: No interrupt is generated when RAMERRF flag is set in PKA_SR.
1: An interrupt is generated when RAMERRF flag is set in PKA_SR.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **PROCENDIE**: End of operation interrupt enable
0: No interrupt is generated when PROCENDF flag is set in PKA_SR.
1: An interrupt is generated when PROCENDF flag is set in PKA_SR.

Bits 16:14 Reserved, must be kept at reset value.

Bits 13:8 **MODE[5:0]**: PKA operation code

000000: Montgomery parameter computation then modular exponentiation
000001: Montgomery parameter computation only
000010: Modular exponentiation only (Montgomery parameter must be loaded first)
000011: Modular exponentiation (protected, used when manipulating secrets)
100000: Montgomery parameter computation then ECC scalar multiplication (protected)
100100: ECDSA sign (protected)
100110: ECDSA verification
101000: Point on elliptic curve Fp check
000111: RSA CRT exponentiation
001000: Modular inversion
001001: Arithmetic addition
001010: Arithmetic subtraction
001011: Arithmetic multiplication
001100: Arithmetic comparison
001101: Modular reduction
001110: Modular addition
001111: Modular subtraction
010000: Montgomery multiplication
100011: ECC complete addition
100111: ECC double base ladder
101111: ECC projective to affine

When an operation not listed here is written by the application with EN bit set, OPERRF bit is set in PKA_SR register, and the write to MODE[5:0] bitfield is ignored. When PKA is configured in limited mode (LMF = 1 in PKA_SR), writing a MODE[5:0] different from 0x26 with EN bit to 1 triggers OPERRF bit to be set and write to MODE[5:0] bit is ignored.

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **START**: start the operation

Set this bit to start the operation selected by the MODE[5:0] bitfield, using the operands and data already written to the PKA RAM. This bit is always read as 0.

When an illegal operation is selected while START bit is set no operation is started, and OPERRF bit is set in PKA_SR.

Note: START is ignored if PKA is busy.

Bit 0 **EN**: PKA enable

0: Disable PKA

1: Enable PKA; PKA becomes functional when INITOK is set by hardware in PKA_SR. When an illegal operation is selected while EN = 1, OPERRF bit is set in PKA_SR. See PKA_CR.MODE[5:0] bitfield for details.

Note: When EN = 0, PKA RAM can still be accessed by the application.

52.7.2 PKA status register (PKA_SR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPERRF	ADDRERRF	RAMERRF	Res.	PROCENDF	BUSY
										r	r	r		r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LMF	INITOK
														r	r

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **OPERRF**: Operation error flag

0: No event error

1: An illegal or unknown operation has been selected in PKA_CR register

This bit is cleared using OPERRFC bit in PKA_CLRFR.

Bit 20 **ADDRERRF**: Address error flag

0: No address error

1: Address access is out of range (unmapped address)

This bit is cleared using ADDRERRFC bit in PKA_CLRFR.

Bit 19 **RAMERRF**: PKA RAM error flag

0: No PKA RAM access error

1: An AHB access to the PKA RAM occurred while the PKA core was computing and using its internal RAM (AHB PKA_RAM access are not allowed while PKA operation is in progress).

This bit is cleared using RAMERRFC bit in PKA_CLRFR.

Bit 18 Reserved, must be kept at reset value.

Bit 17 **PROCENDF**: PKA end of operation flag

0: Operation in progress

1: PKA operation is completed. This flag is set when the BUSY bit is deasserted.

Bit 16 **BUSY**: Busy flag

This bit is set whenever a PKA operation is in progress (START = 1 in PKA_CR). It is automatically cleared when the computation is complete, making PKA RAM accessible again.

0: No operation is in progress (default)

1: An operation is in progress

If PKA is started with a wrong opcode, it stays busy for a couple of cycles, then it aborts automatically the operation and goes back to ready (BUSY = 0).

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **LMF**: Limited mode flag

This bit is updated when EN bit in PKA_CR is set

0: All values documented in MODE[5:0] bitfield can be used.

1: Only ECDSA verification (MODE[5:0] = 0x26) is supported by the PKA.

Bit 0 **INITOK**: PKA initialization OK
This bit is asserted when PKA initialization is complete.
0: PKA is not initialized correctly. START bit cannot be set.
1: PKA is initialized correctly and can be used normally.
When RNG is not able to output proper random numbers or when SAES is using the RNG bus, INITOK remains at 0. If it happens, make sure the RNG peripheral is properly clocked and initialized, and the SAES peripheral is clocked with BUSY bit cleared in SAES_SR to release the RNG bus, then enable PKA again.

52.7.3 PKA clear flag register (PKA_CLRFR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPERRFC	ADDRERRFC	RAMERRFC	Res.	PROCENDFC	Res.
										w	w	w		w	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **OPERRFC**: Clear operation error flag
0: No action
1: Clear the OPERRF flag in PKA_SR

Bit 20 **ADDRERRFC**: Clear address error flag
0: No action
1: Clear the ADDRERRF flag in PKA_SR

Bit 19 **RAMERRFC**: Clear PKA RAM error flag
0: No action
1: Clear the RAMERRF flag in PKA_SR

Bit 18 Reserved, must be kept at reset value.

Bit 17 **PROCENDFC**: Clear PKA end of operation flag
0: No action
1: Clear the PROCENDF flag in PKA_SR

Bits 16:0 Reserved, must be kept at reset value.

Note: Reading PKA_CLRFR returns all 0s.

52.7.4 PKA RAM

The PKA RAM is mapped at the offset address of 0x0400 compared to the PKA base address. Only 32-bit word single accesses are supported, through PKA.AHB interface.

RAM size is 5336 bytes (max word offset: 0x14D0)

Note: PKA RAM cannot be used just after a PKA reset or a product reset, as described in Section 52.3.3: PKA reset and clocks.

52.7.5 PKA register map

Table 479. PKA register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	PKA_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPERRIE	ADDRERRIE	RAMERRIE	Res.	PROCENDIE	Res.	Res.	Res.	MODE[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	START	EN
	Reset value											0	0	0		0				0	0	0	0	0	0							0	0
0x004	PKA_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPERRF	ADDRERRF	RAMERRF	Res.	PROCENDF	BUSY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LMF	INITOK
	Reset value											0	0	0		0	0															0	0
0x008	PKA_CLRFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OPERRFC	ADDRERRFC	RAMERRFC	Res.	PROCENDFC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value											0	0	0		0																	

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

53 Advanced-control timers (TIM1/TIM8)

53.1 TIM1/TIM8 introduction

The advanced-control timers (TIM1/TIM8) consist of a 16-bit autoreload counter driven by a programmable prescaler.

It may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The advanced-control (TIM1/TIM8) and general-purpose (TIMy) timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 53.3.30: Timer synchronization*.

53.2 TIM1/TIM8 main features

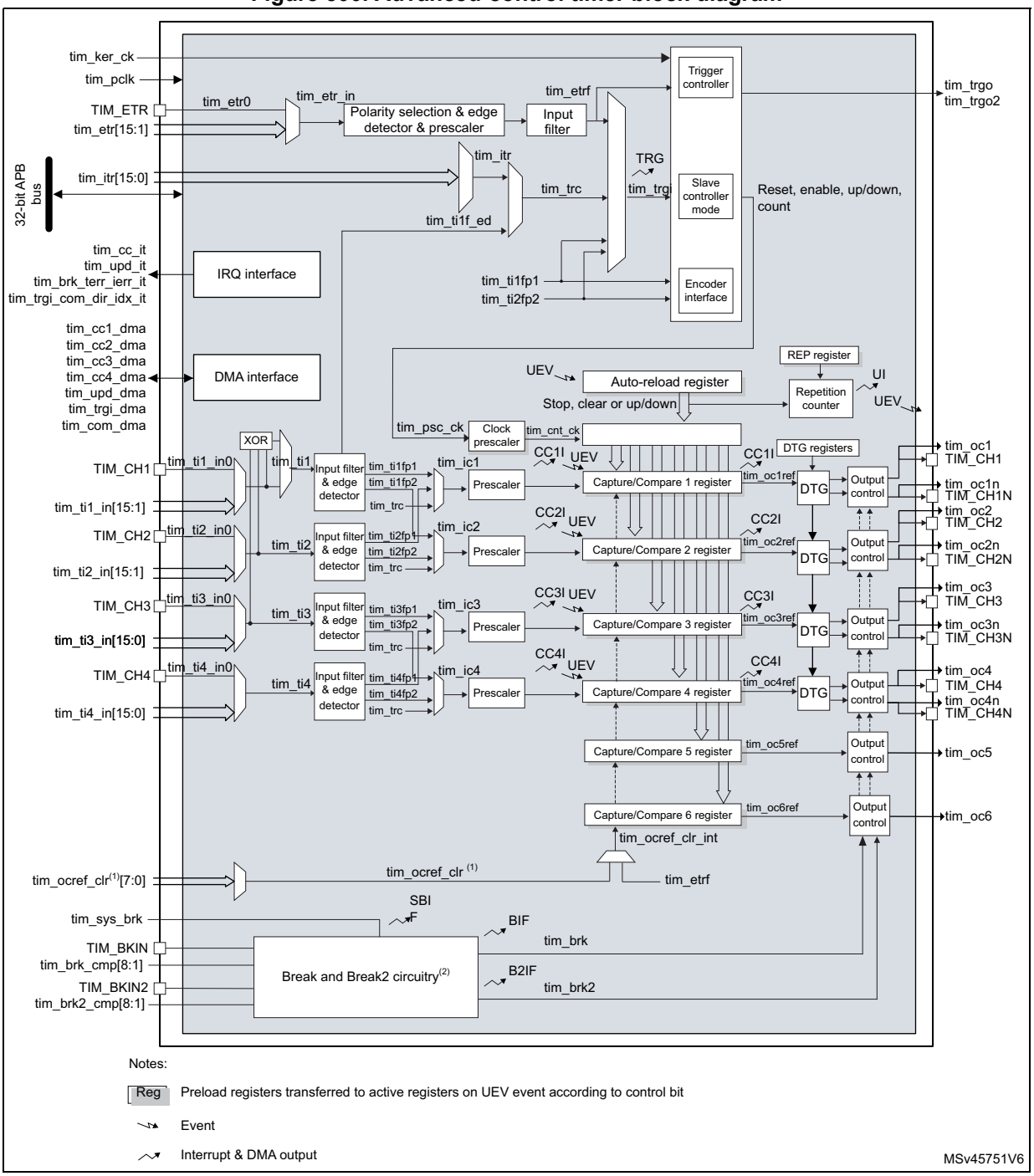
TIM1/TIM8 timer features include:

- 16-bit up, down, up/down autoreload counter.
- 16-bit programmable prescaler allowing dividing (also “on the fly”) the counter clock frequency by any factor from 1 to 65536.
- Up to six independent channels for:
  - Input capture (but channels 5 and 6)
  - Output compare
  - PWM generation (edge and center-aligned mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time
- Synchronization circuit to control the timer with external signals and to interconnect several timers together.
- Repetition counter to update the timer registers only after a given number of cycles of the counter.
- 2 break inputs to put the timer’s output signals in a safe user selectable configuration.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization, or count by internal/external trigger)
  - Input capture
  - Output compare
- Supports incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes
- Trigger input for external clock or cycle-by-cycle current management
- ADC synchronization for jitter-free sampling points

53.3 TIM1/TIM8 functional description

53.3.1 Block diagram

Figure 500. Advanced-control timer block diagram



1. This feature is not available on all timers, refer to *Section 53.3.2: TIM1/TIM8 pins and internal signals*.
2. See *Figure 547: Break and Break2 circuitry overview* for details.

53.3.2 TIM1/TIM8 pins and internal signals

The tables in this section summarize the TIM inputs and outputs

Table 480. TIM input/output pins

Pin name	Signal type	Description
TIM_CH1 TIM_CH2 TIM_CH3 TIM_CH4	Input/output	Timer multi-purpose channels. Each channel can be used for capture, compare or PWM. TIM_CH1 and TIM_CH2 can also be used as external clock (below 1/4 of the tim_ker_ck clock), external trigger and quadrature encoder inputs. TIM_CH1, TIM_CH2 and TIM_CH3 can be used to interface with digital hall effect sensors.
TIM_CH1N TIM_CH2N TIM_CH3N TIM_CH4N	Output	Timer complementary outputs, derived from TIM_CHx outputs with the possibility to have deadtime insertion.
TIM_ETR	Input	External trigger input. This input can be used as external trigger or as external clock source. This input can receive a clock with a frequency higher than the tim_ker_ck if the tim_etr_in prescaler is used.
TIM_BKIN TIM_BKIN2	Input/output	Break and Break2 inputs. These inputs can also be configured in bidirectional mode.

Table 481. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_ti1_in[15:0] tim_ti2_in[15:0] tim_ti3_in[15:0] tim_ti4_in[15:0]	Input	Internal timer inputs bus. The tim_ti1_in[15:0] and tim_ti2_in[15:0] inputs can be used for capture or as external clock (below 1/4 of the tim_ker_ck clock) and for quadrature encoder signals.
tim_etr[15:0]	Input	External trigger internal input bus. These inputs can be used as trigger, external clock or for hardware cycle-by-cycle pulsewidth control. These inputs can receive clock with a frequency higher than the tim_ker_ck if the tim_etr_in prescaler is used.
tim_itr[15:0]	Input	Internal trigger input bus. These inputs can be used for the slave mode controller or as a input clock (below 1/4 of the tim_ker_ck clock).
tim_trgo/tim_trgo2	Output	Internal trigger outputs. These triggers are used by other timers and /or other peripherals.

Table 481. TIM internal input/output signals (continued)

Internal signal name	Signal type	Description
tim_ocref_clr[7:0]	Input	Timer tim_ocref_clr input bus. These inputs can be used to clear the tim ocxref signals, typically for hardware cycle-by-cycle pulsewidth control.
tim_brk_cmp[8:1]	Input	Break input for internal signals
tim_brk2_cmp[8:1]	Input	Break2 input for internal signals
tim_sys_brk[n:0]	Input	System break input. This input gathers the MCU's system level errors.
tim_pclk	Input	Timer APB clock
tim_ker_ck	Input	Timer kernel clock
tim_cc_it	Output	Timer capture/compare interrupt
tim_upd_it	Output	Timer update event interrupt
tim_brk_terr_ierr_it	Output	Timer break, break2, transition error and index error interrupt
tim_trgi_com_dir_idx_it	Output	Timer trigger, commutation, direction and index interrupt
tim_cc1_dma tim_cc2_dma tim_cc3_dma tim_cc4_dma	Output	Timer capture / compare 1..4 dma requests
tim_upd_dma	Output	Timer update dma request
tim_trgi_dma	Output	Timer trigger dma request
tim_com_dma	Output	Timer commutation dma request

Table 482, *Table 483*, *Table 484* and *Table 485* list the sources connected to the tim_ti[4:1] input multiplexers.

Table 482. Interconnect to the tim_ti1 input multiplexer

tim_ti1 inputs	Sources	
	TIM1	TIM8
tim_ti1_in0	TIM1_CH1	TIM8_CH1
tim_ti1_in[15:1]	Reserved	Reserved

Table 483. Interconnect to the tim_ti2 input multiplexer

tim_ti2 inputs	Sources	
	TIM1	TIM8
tim_ti2_in0	TIM1_CH2	TIM8_CH2
tim_ti2_in[15:1]	Reserved	Reserved

Table 484. Interconnect to the tim_ti3 input multiplexer

tim_ti3 inputs	Sources	
	TIM1	TIM8
tim_ti3_in0	TIM1_CH3	TIM8_CH3
tim_ti3_in[15:1]	Reserved	Reserved

Table 485. Interconnect to the tim_ti4 input multiplexer

tim_ti4 inputs	Sources	
	TIM1	TIM8
tim_ti4_in0	TIM1_CH4	TIM8_CH4
tim_ti4_in[15:1]	Reserved	Reserved

Table 486 lists the internal sources connected to the tim_itr input multiplexer.

Table 486. Internal trigger connection

Timer internal trigger input signal	TIM1	TIM8
tim_itr0	Reserved	tim1_trgo
tim_itr1	tim2_trgo	tim2_trgo
tim_itr2	tim3_trgo	tim3_trgo
tim_itr3	tim4_trgo	tim4_trgo
tim_itr4	tim5_trgo	tim5_trgo
tim_itr5	tim8_trgo	Reserved
tim_itr6	tim9_trgo	tim9_trgo
tim_itr7	tim12_trgo	tim12_trgo
tim_itr8	tim13_oc1	tim13_oc1
tim_itr9	tim14_oc1	tim14_oc1
tim_itr10	tim15_trgo	tim15_trgo
tim_itr11	tim16_oc1	tim16_oc1
tim_itr12	tim17_oc1	tim17_oc1
tim_itr[15:13]	Reserved	Reserved

Table 487 lists the internal sources connected to the tim_etr input multiplexer.

Table 487. Interconnect to the tim_etr input multiplexer

Timer external trigger input signal	Timer external trigger signals assignment	
	TIM1	TIM8
tim_etr0	TIM1_ETR	TIM8_ETR
tim_etr[2:1]	Reserved	Reserved
tim_etr3	adc1_awd1	adc2_awd1
tim_etr4	adc1_awd2	adc2_awd2
tim_etr5	adc1_awd3	adc2_awd3
tim_etr6	adc2_awd1	Reserved
tim_etr7	adc2_awd2	
tim_etr8	adc2_awd3	
tim_etr[15:9]	Reserved	

Table 488, *Table 489* and *Table 490* list the sources connected to the tim_brk and tim_brk2inputs.

Table 488. Timer break interconnect

tim_brk inputs	TIM1	TIM8
TIM_BKIN	TIM1_BKIN pin	TIM8_BKIN pin
tim_brk_cmp[8:1]	Reserved	Reserved

Table 489. Timer break2 interconnect

tim_brk2 inputs	TIM1	TIM8
TIM_BKIN2	TIM1_BKIN2 pin	TIM8_BKIN2 pin
tim_brk2_cmp[8:1]	Reserved	Reserved

Table 490. System break interconnect

tim_sys_brk inputs	TIM1/TIM8	Enable bit in SYSCFG_BKLOCKR register
tim_sys_brk0	Programmable Voltage Detector (PVD)	PVD_BL
tim_sys_brk1	Cortex®-M55 LOCKUP	CM55LU_BL
tim_sys_brk2	Backup RAM double ECC error	BKPRAM_BL
tim_sys_brk3	Cortex®-M55 TCM double ECC error	CM55TCM_BL
tim_sys_brk4	Cortex®-M55 cache double ECC error	CM55CACHE_BL
CSS	Clock Security System	None (always enabled)

53.3.3 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit counter with its related autoreload register. The counter can count up, down or both up and down. The counter clock can be divided by a prescaler.

The counter, the autoreload register and the prescaler register can be written or read by software, even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Autoreload register (TIMx_ARR)
- Repetition counter register (TIMx_RCR)

The autoreload register is preloaded. Writing to or reading from the autoreload register accesses the preload register. The content of the preload register are transferred into the shadow register permanently or at each update event (UEV), depending on the autoreload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when downcounting) and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note: The counter starts counting 1 clock cycle after setting the CEN bit in the TIMx_CR1 register.

Prescaler description

The prescaler divides the counter clock frequency by any factor from 1 to 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 501 and *Figure 502* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 501. Counter timing diagram with prescaler division change from 1 to 2

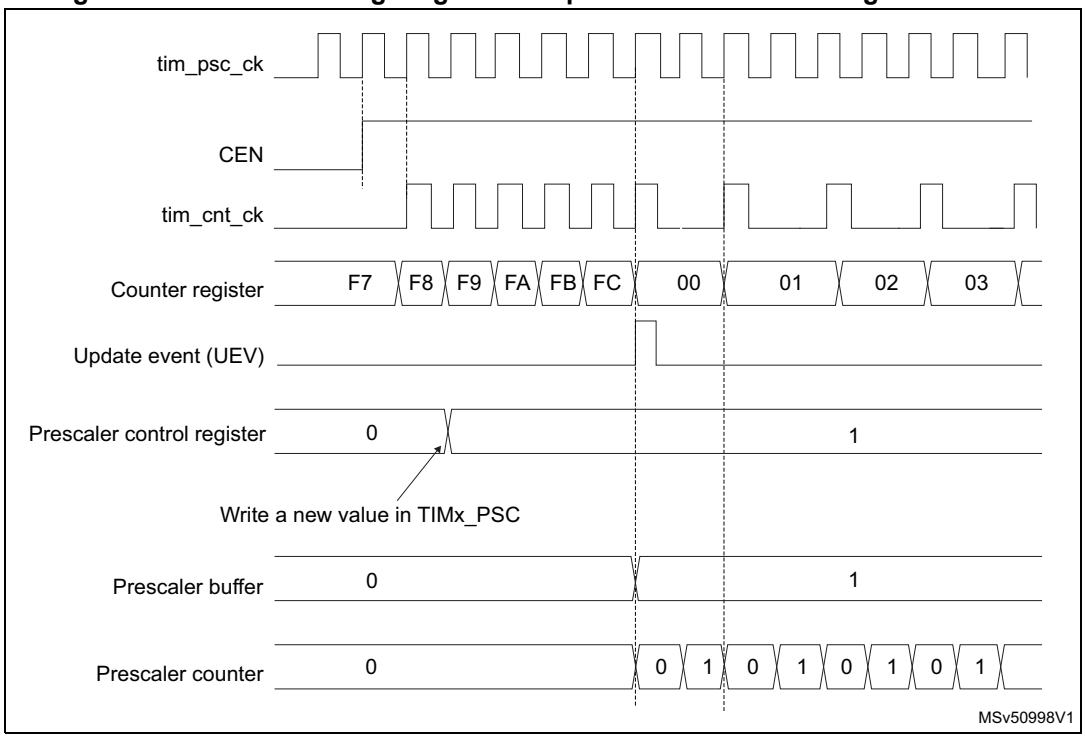
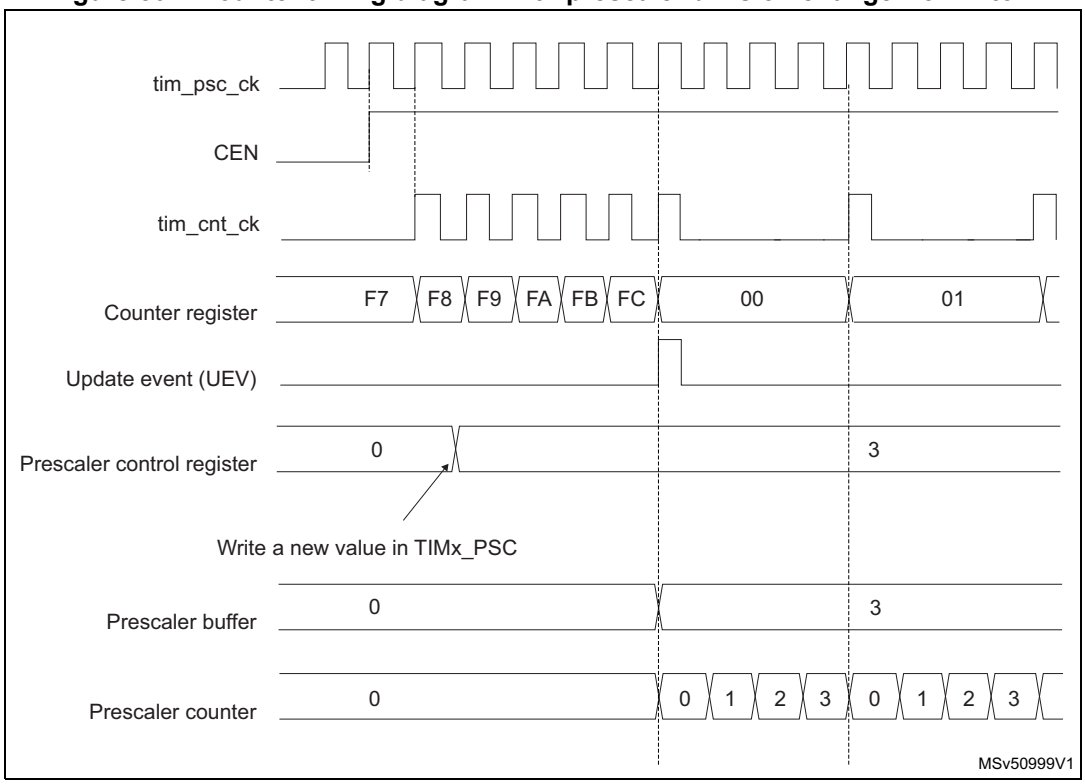


Figure 502. Counter timing diagram with prescaler division change from 1 to 4



53.3.4 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the autoreload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR) + 1. Else the update event is generated at each counter overflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register,
- The autoreload shadow register is updated with the preload value (TIMx_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.

Figure 503. Counter timing diagram, internal clock divided by 1

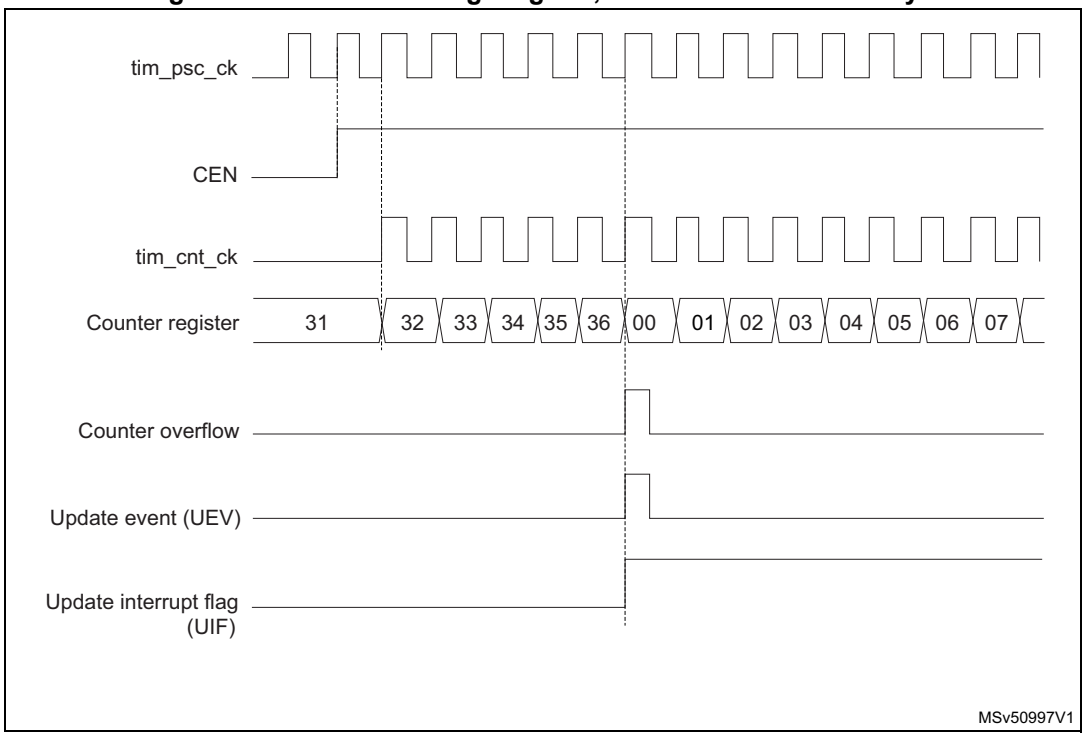


Figure 504. Counter timing diagram, internal clock divided by 2

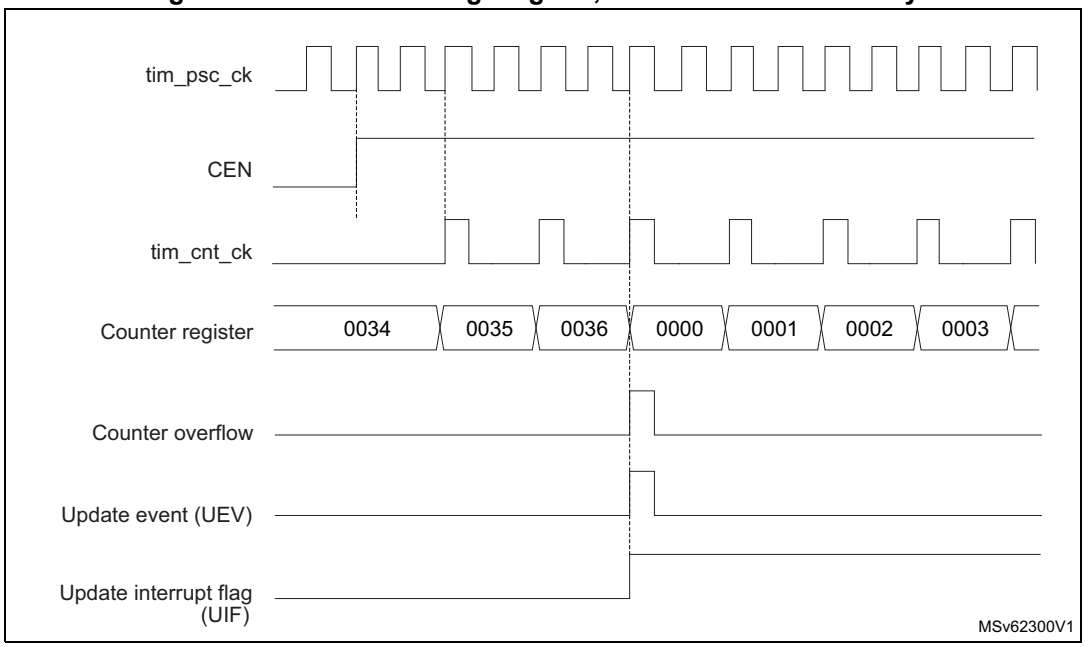


Figure 505. Counter timing diagram, internal clock divided by 4

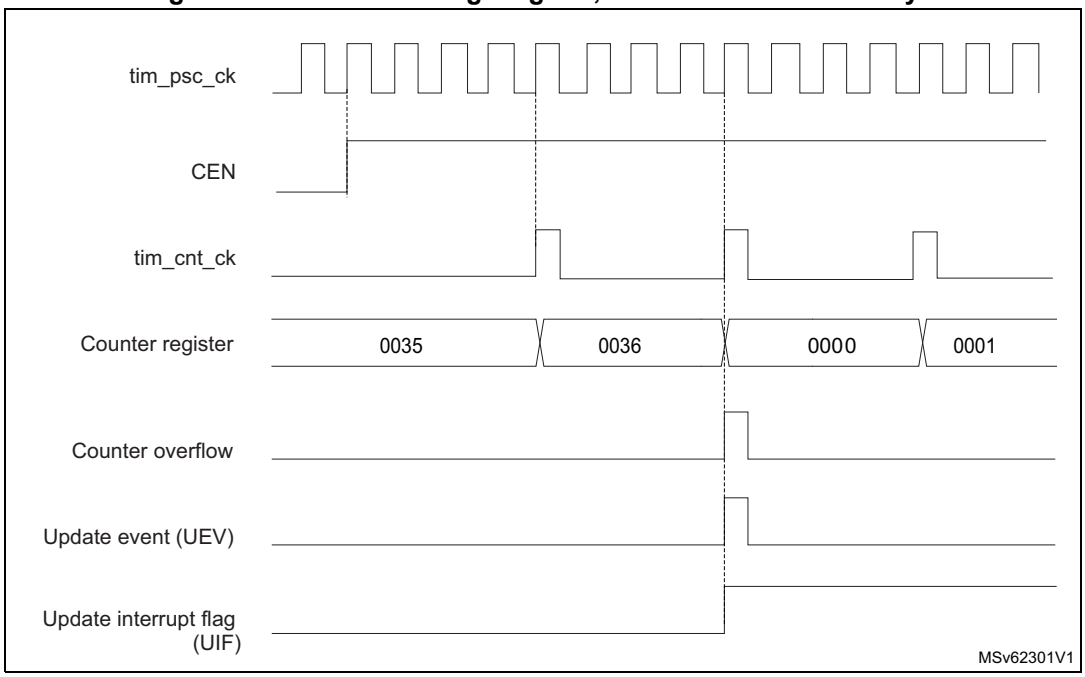


Figure 506. Counter timing diagram, internal clock divided by N

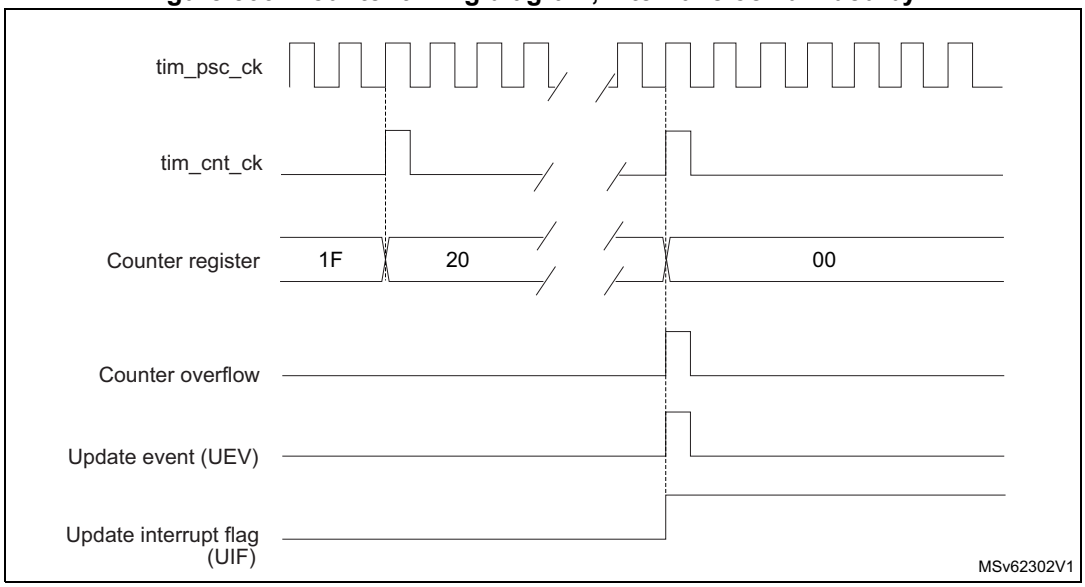


Figure 507. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded)

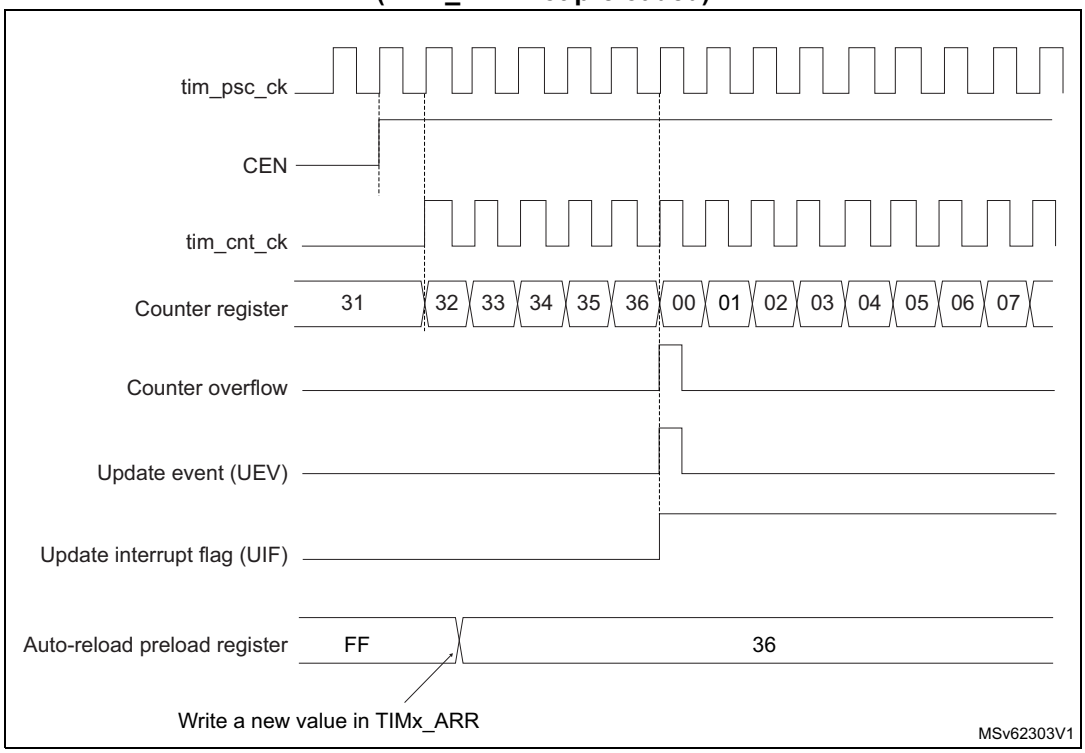
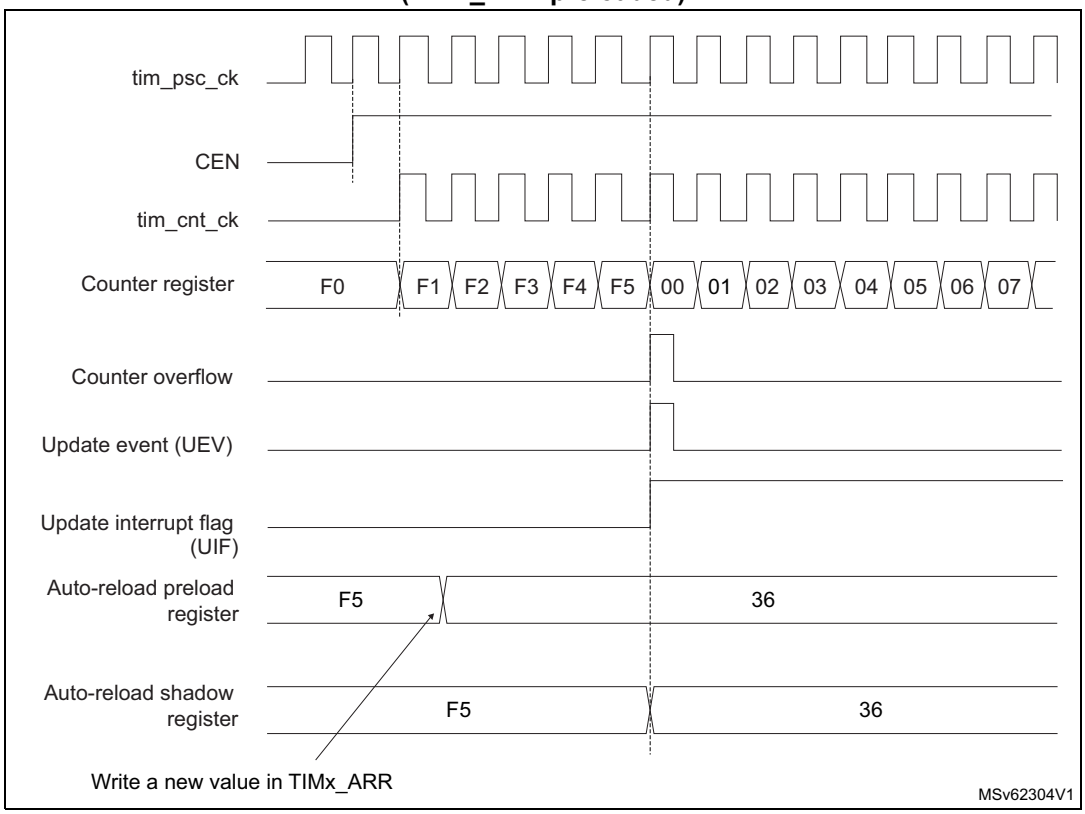


Figure 508. Counter timing diagram, update event when ARPE = 1 (TIMx_ARR preloaded)



Downcounting mode

In downcounting mode, the counter counts from the autoreload value (content of the TIMx_ARR register) down to 0, then restarts from the autoreload value and generates a counter underflow event.

If the repetition counter is used, the update event (UEV) is generated after downcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR) + 1. Else the update event is generated at each counter underflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current autoreload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate doesn't change).

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register.
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The autoreload active register is updated with the preload value (content of the TIMx_ARR register). Note that the autoreload is updated before the counter is reloaded, so that the next period is the expected one.

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.

Figure 509. Counter timing diagram, internal clock divided by 1

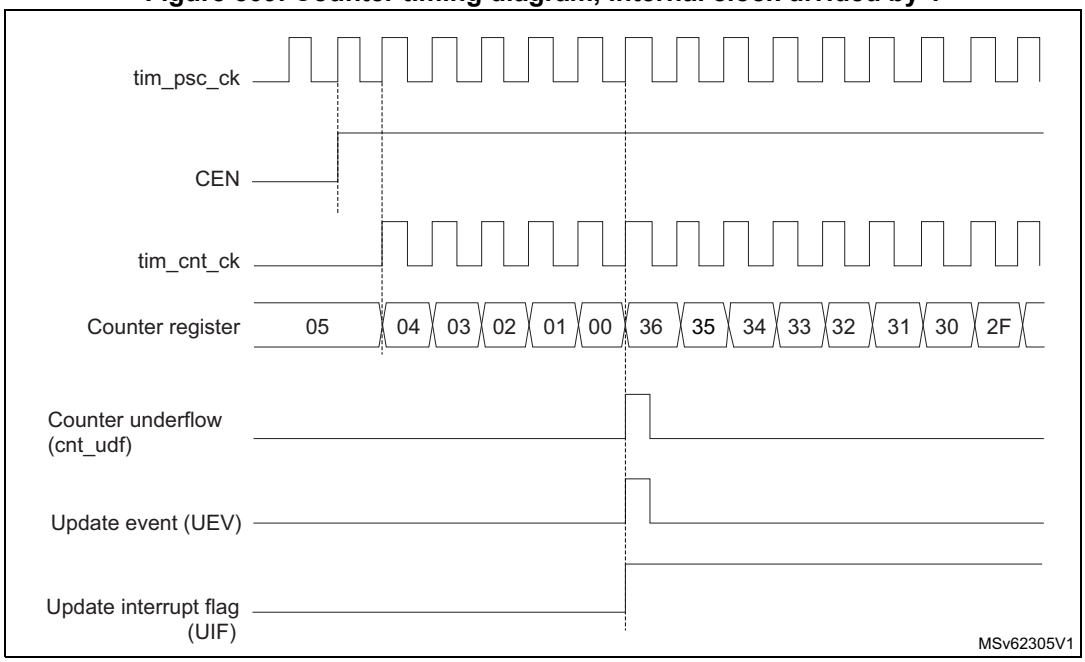


Figure 510. Counter timing diagram, internal clock divided by 2

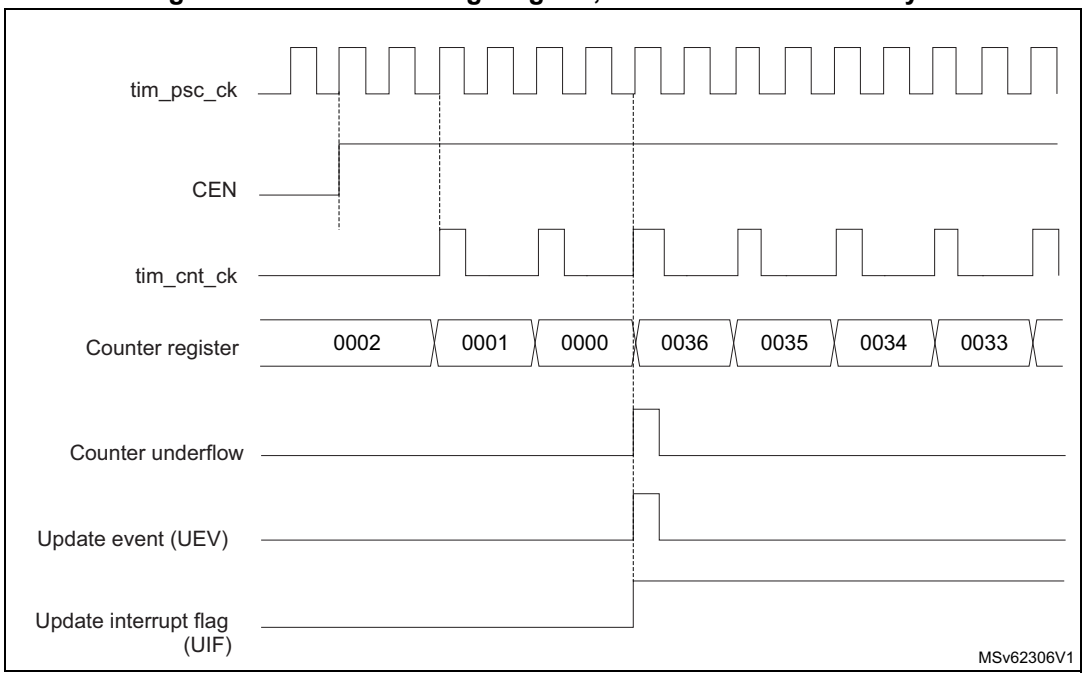


Figure 511. Counter timing diagram, internal clock divided by 4

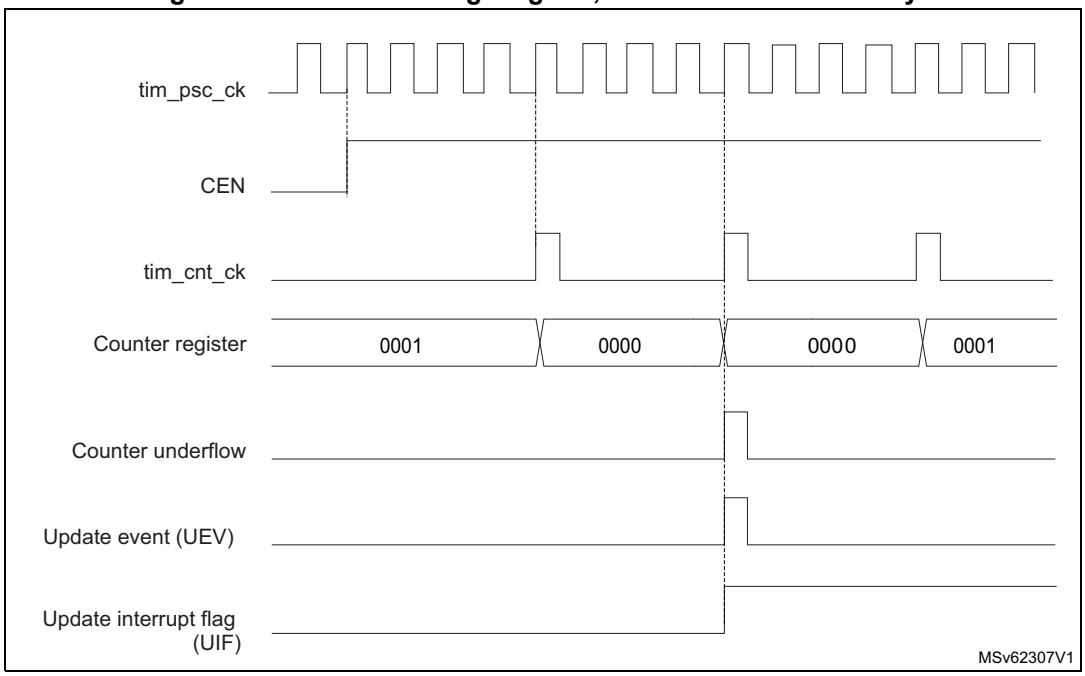


Figure 512. Counter timing diagram, internal clock divided by N

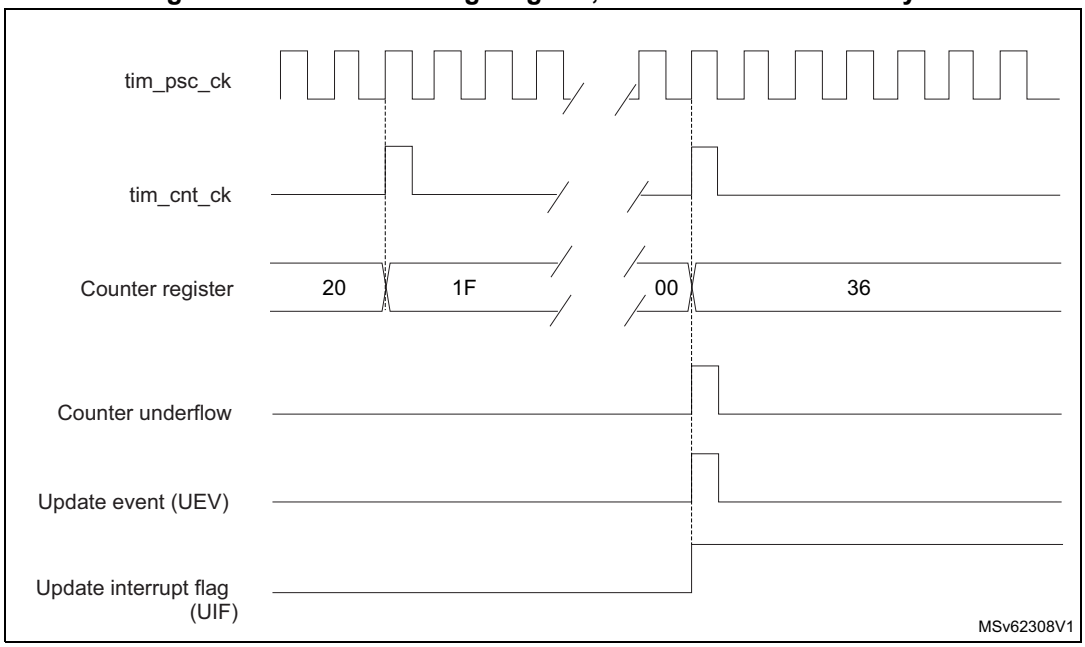
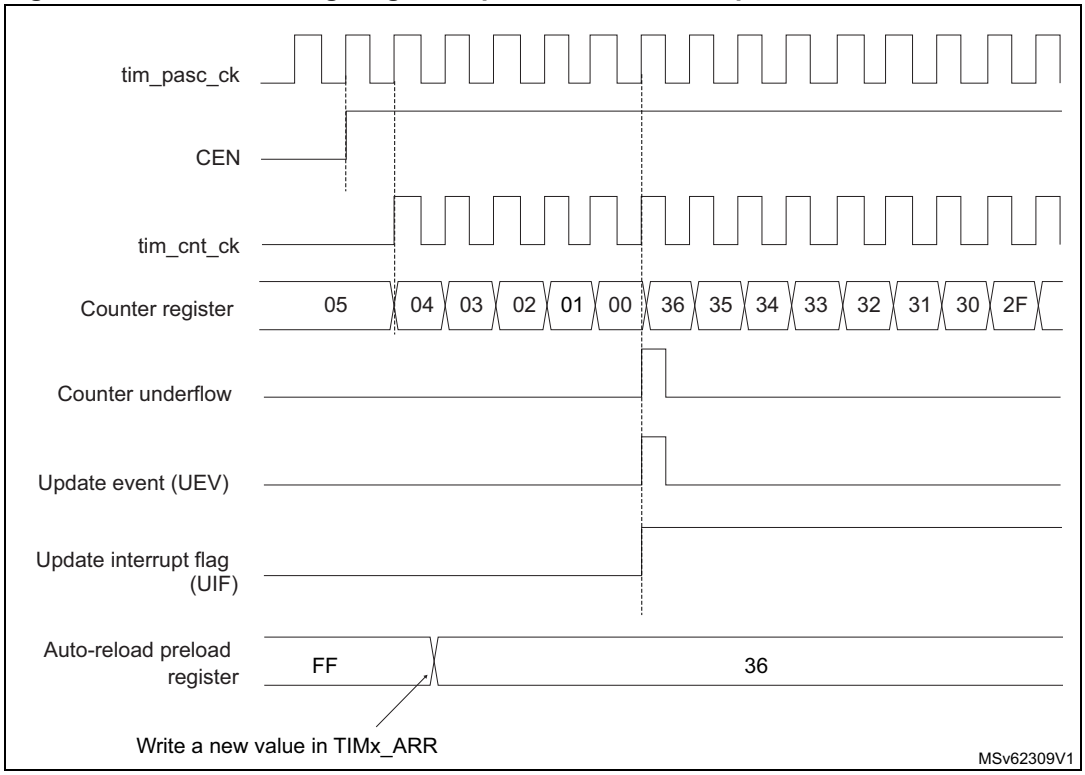


Figure 513. Counter timing diagram, update event when repetition counter is not used



Center-aligned mode (up/down counting)

In center-aligned mode, the counter counts from 0 to the autoreload value (content of the TIMx_ARR register) – 1, generates a counter overflow event, then counts from the

autoreload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are not equal to 00. The Output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = 01), the counter counts up (Center aligned mode 2, CMS = 10) the counter counts up and down (Center aligned mode 3, CMS = 11).

In this mode, the DIR direction bit in the TIMx_CR1 register cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current autoreload value.

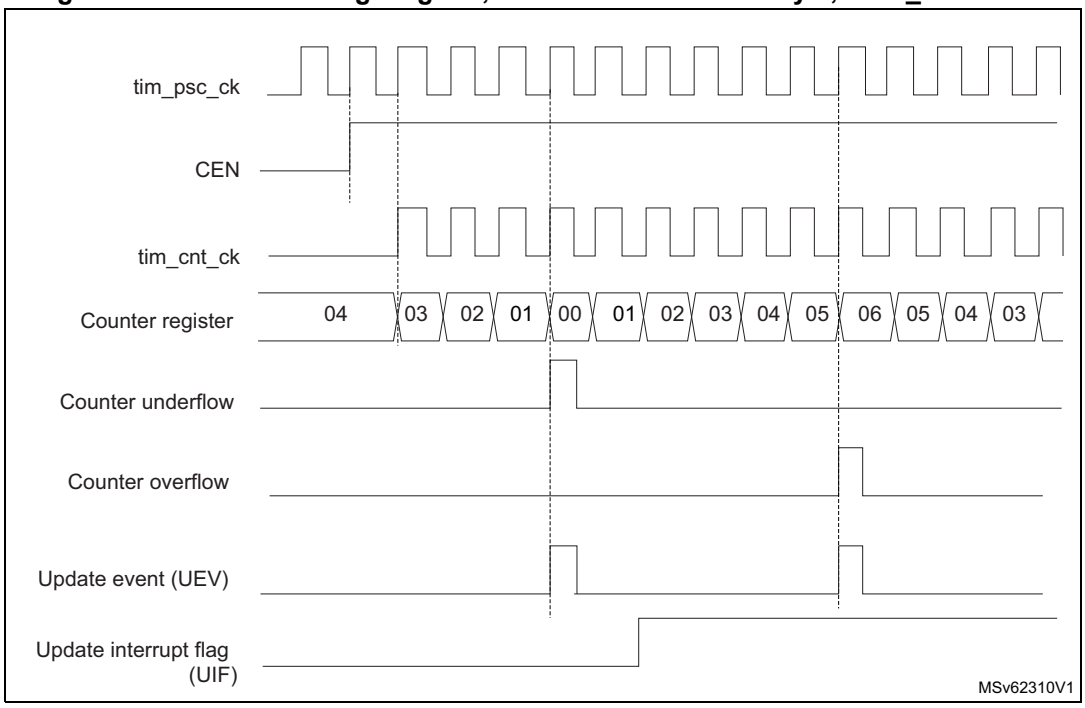
In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an UEV update event but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register)
- The autoreload active register is updated with the preload value (content of the TIMx_ARR register). Note that if the update source is a counter overflow, the autoreload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.

Figure 514. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6



1. Here, center-aligned mode 1 is used (for more details refer to *Section 53.6: TIM1/TIM8 registers*).

Figure 515. Counter timing diagram, internal clock divided by 2

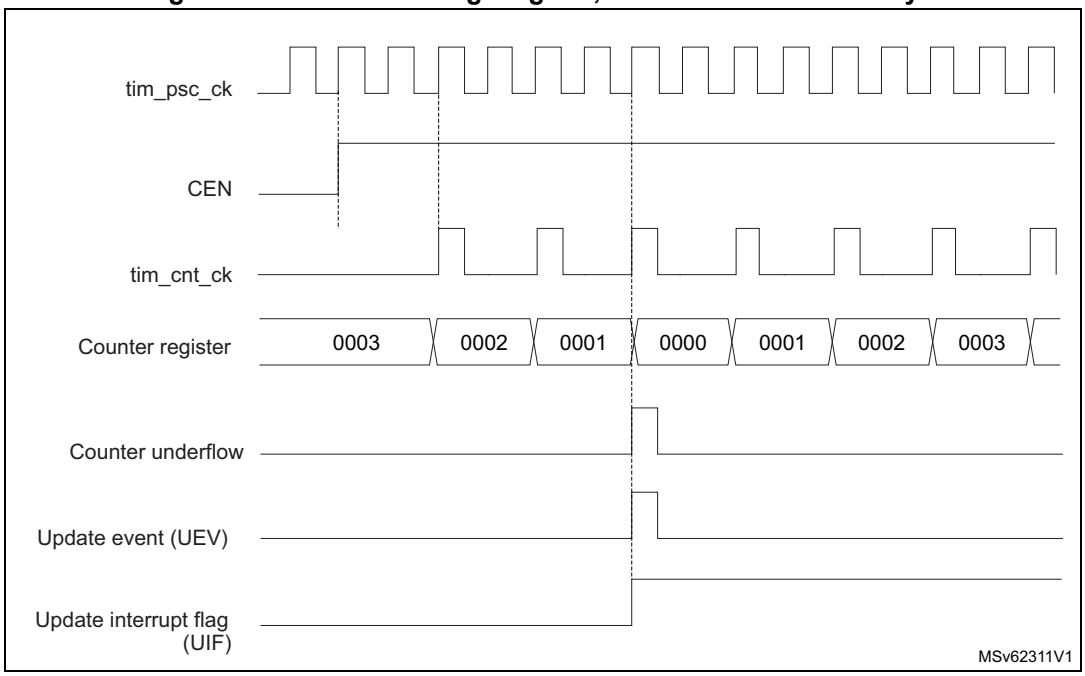


Figure 516. Counter timing diagram, internal clock divided by 4, TIMx_ARR = 0x36

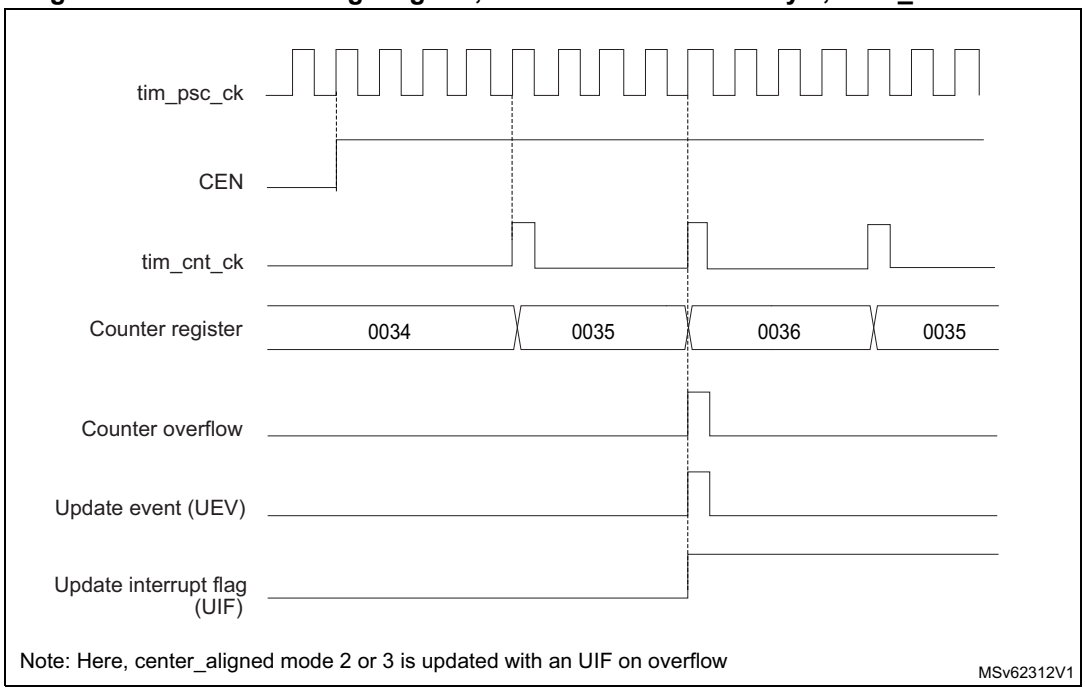


Figure 517. Counter timing diagram, internal clock divided by N

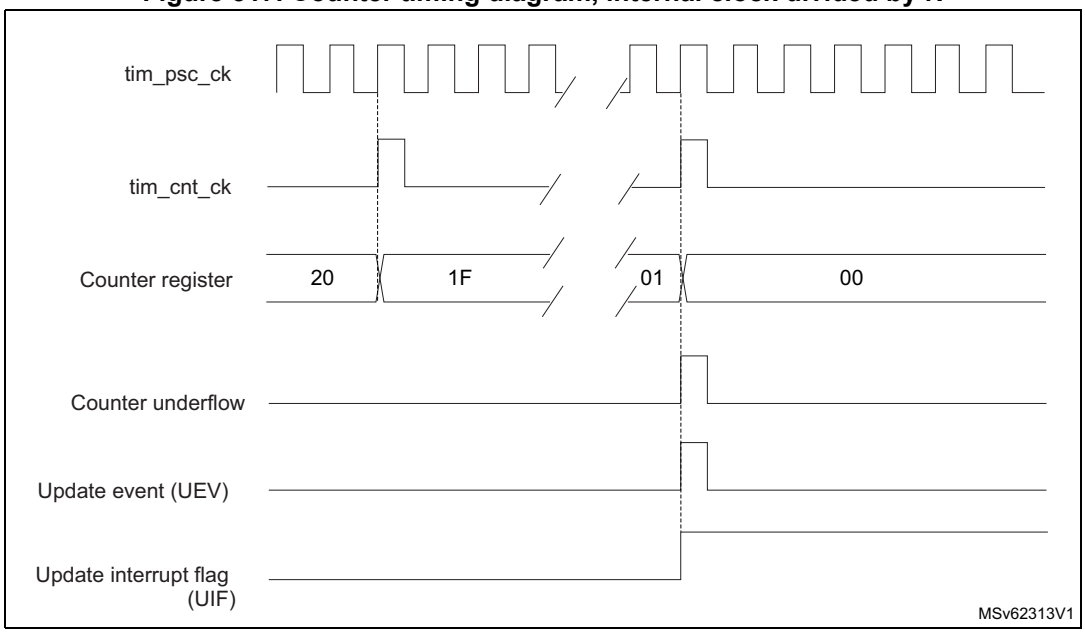


Figure 518. Counter timing diagram, update event with ARPE = 1 (counter underflow)

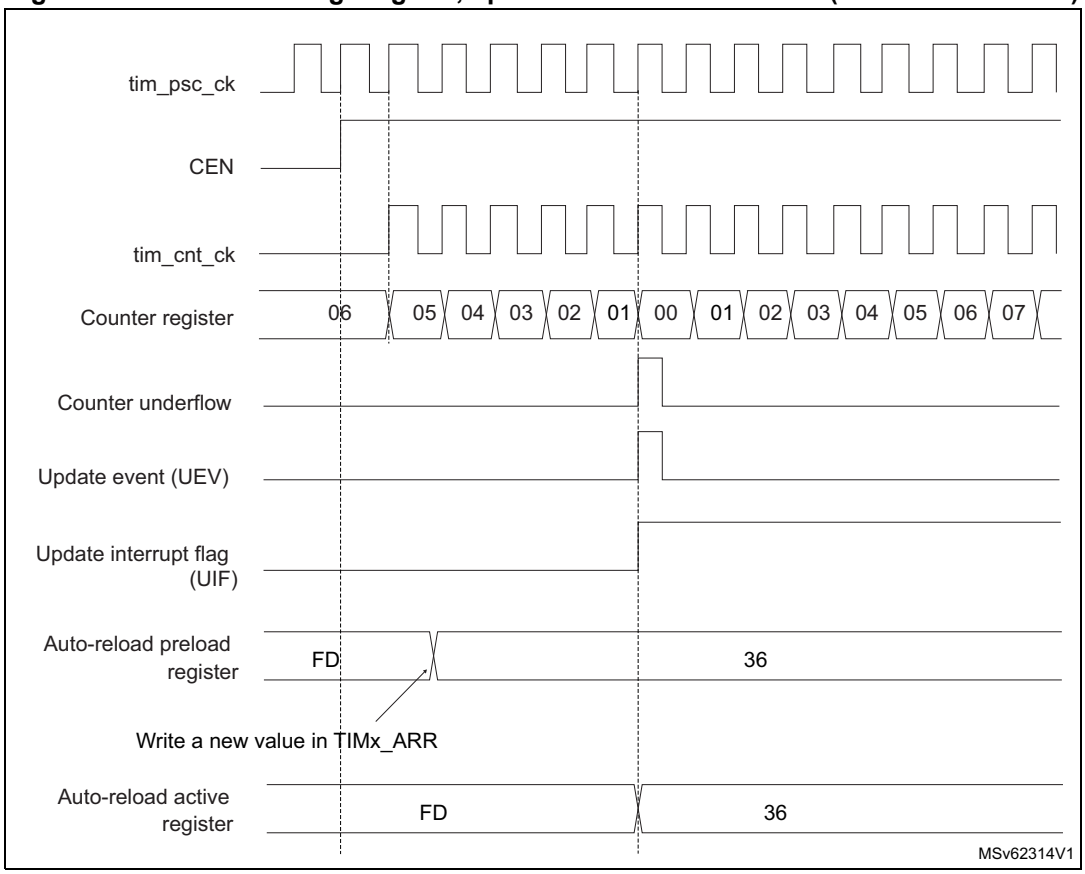
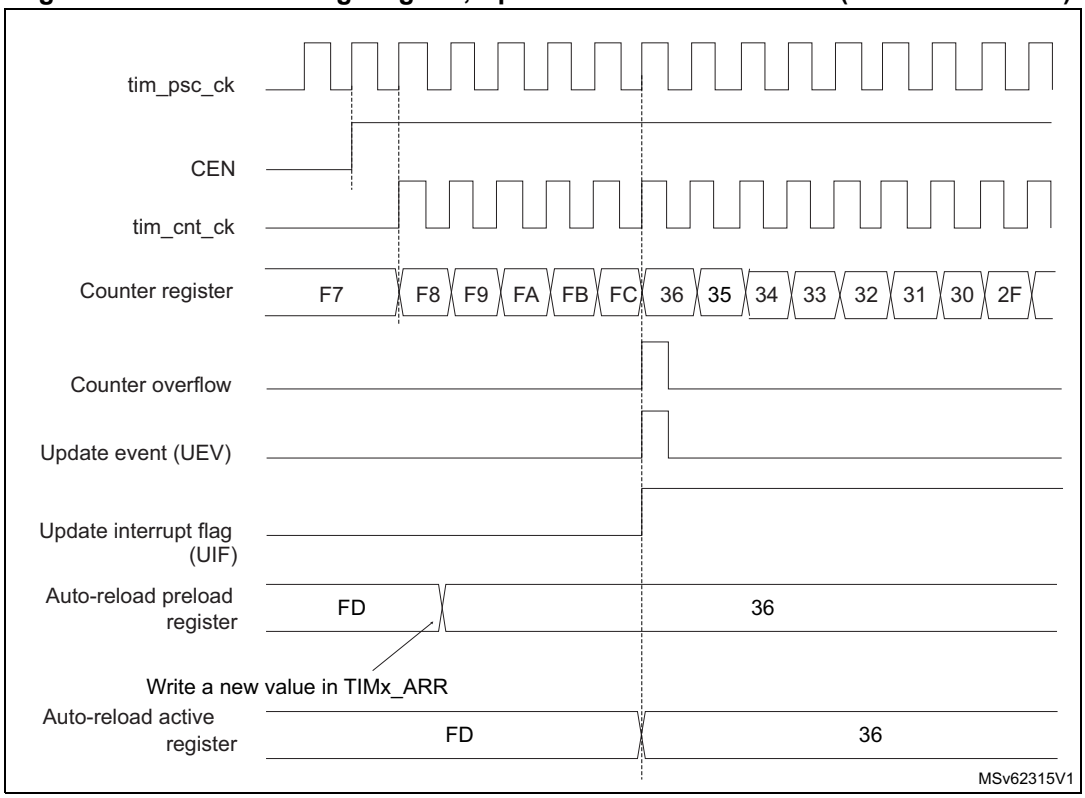


Figure 519. Counter timing diagram, Update event with ARPE = 1 (counter overflow)



53.3.5 Repetition counter

Section 53.3.3: Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows/underflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIMx_ARR autoreload register, TIMx_PSC prescaler register, but also TIMx_CCRx capture/compare registers in compare mode) every N+1 counter overflows or underflows, where N is the value in the TIMx_RCR repetition counter register.

The repetition counter is decremented:

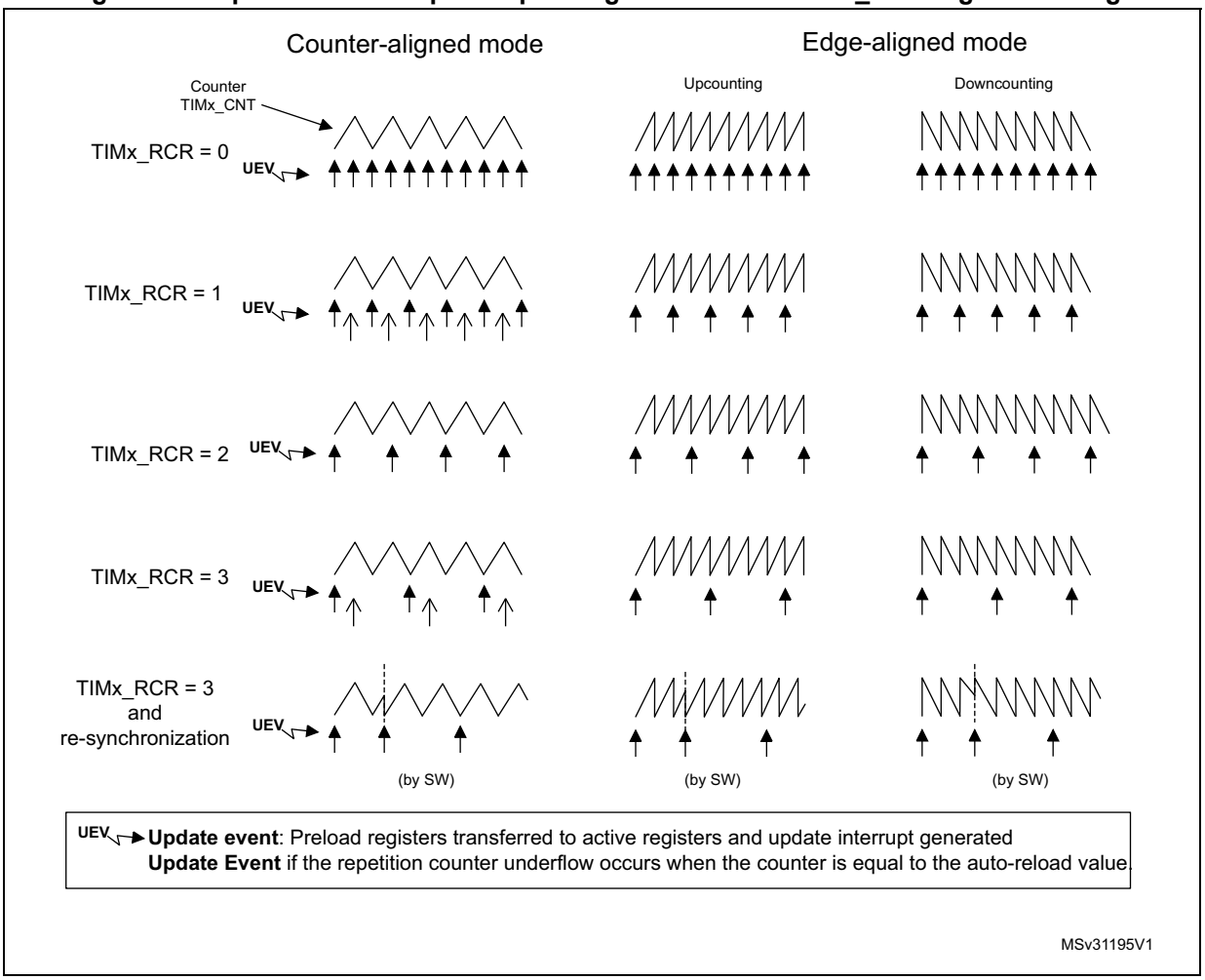
- At each counter overflow in upcounting mode,
- At each counter underflow in downcounting mode,
- At each counter overflow and at each counter underflow in center-aligned mode. Although this limits the maximum number of repetition to 32768 PWM cycles, it makes it possible to update the duty cycle twice per PWM period. When refreshing compare registers only once per PWM period in center-aligned mode, maximum resolution is $2xT_{ck}$, due to the symmetry of the pattern.

The repetition counter is an autoreload type; the repetition rate is maintained as defined by the TIMx_RCR register value (refer to *Figure 520*). When the update event is generated by software (by setting the UG bit in TIMx_EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the TIMx_RCR register.

In Center aligned mode, for odd values of RCR, the update event occurs either on the overflow or on the underflow depending on when the RCR register was written and when the counter was launched: if the RCR was written before launching the counter, the UEV occurs on the underflow. If the RCR was written after launching the counter, the UEV occurs on the overflow.

For example, for RCR = 3, the UEV is generated each 4th overflow or underflow event depending on when the RCR was written.

Figure 520. Update rate examples depending on mode and TIMx_RCR register settings



53.3.6 External trigger input

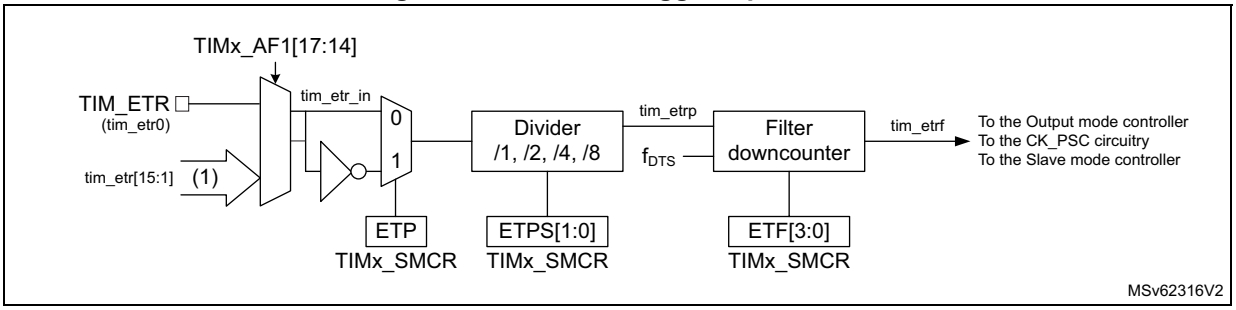
The timer features an external trigger input tim_etr_in. It can be used as:

- external clock (external clock mode 2, see *Section 53.3.7*)
- trigger for the slave mode (see *Section 53.3.30*)
- PWM reset input for cycle-by-cycle current regulation (see *Section 53.3.9*)

Figure 521 below describes the tim_etr_in input conditioning. The input polarity is defined with the ETP bit in TIMxSMCR register. The trigger can be prescaled with the divider programmed by the ETPS[1:0] bitfield and digitally filtered with the ETF[3:0] bitfield. The resulting signal (tim_etrf) is available for three purposes: as an external clock, to condition

the output (typically to reset a PWM output for a current limitation), and as a trigger for the Slave mode controller.

Figure 521. External trigger input block



The tim_etr_in input comes from multiple sources: input pins (default configuration), or internal sources. The selection is done with the ETRSEL[3:0] bitfield in the TIMx_AF1 register.

Refer to *Section 53.3.2: TIM1/TIM8 pins and internal signals* for the list of sources connected to the etr_in input in the product.

53.3.7 Clock selection

The counter clock can be provided by the following clock sources:

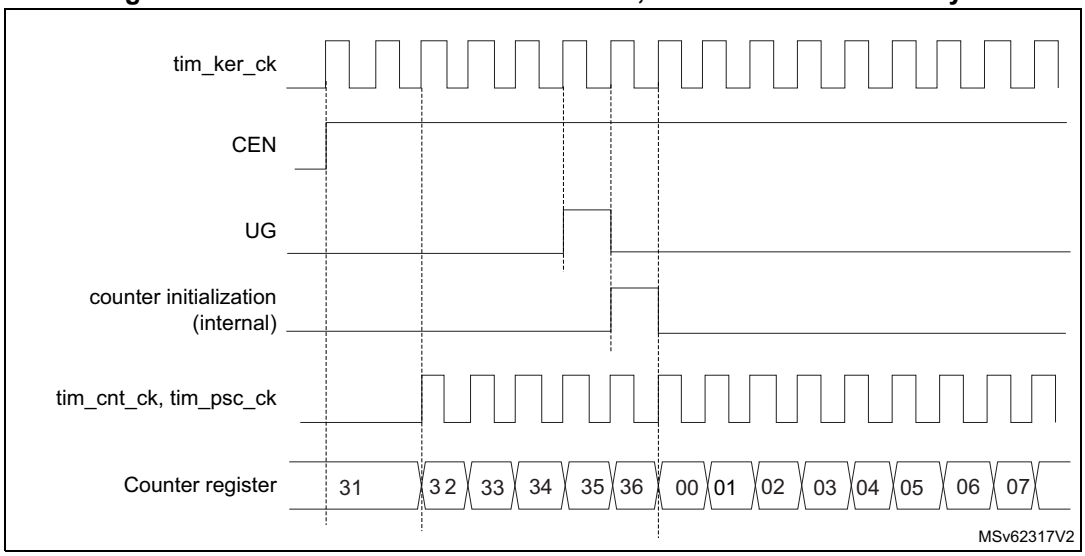
- Internal clock (tim_ker_ck)
- External clock mode1: external input pin (tim_ti1 or tim_ti2)
- External clock mode2: external trigger input (tim_etr_in)
- Encoder mode

Internal clock source (tim_ker_ck)

If the slave mode controller is disabled (SMS = 000), then the CEN, DIR (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 522 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

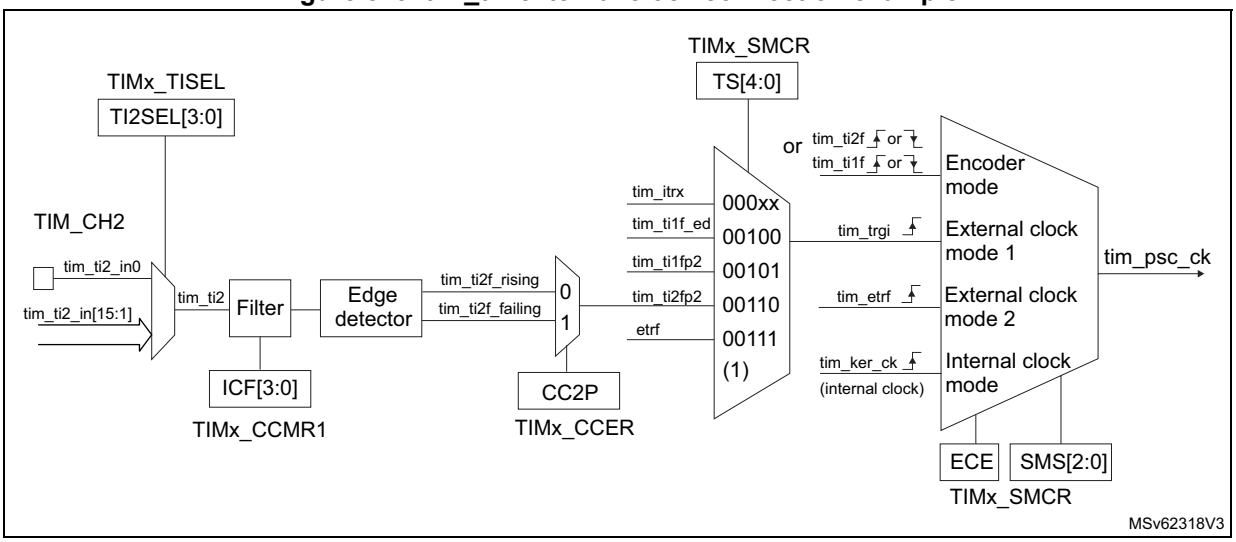
Figure 522. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1

This mode is selected when SMS = 111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 523. tim_ti2 external clock connection example



1. Codes ranging from 01000 to 11111 are reserved.

For example, to configure the upcounter to count in response to a rising edge on the tim_ti2 input, use the following procedure:

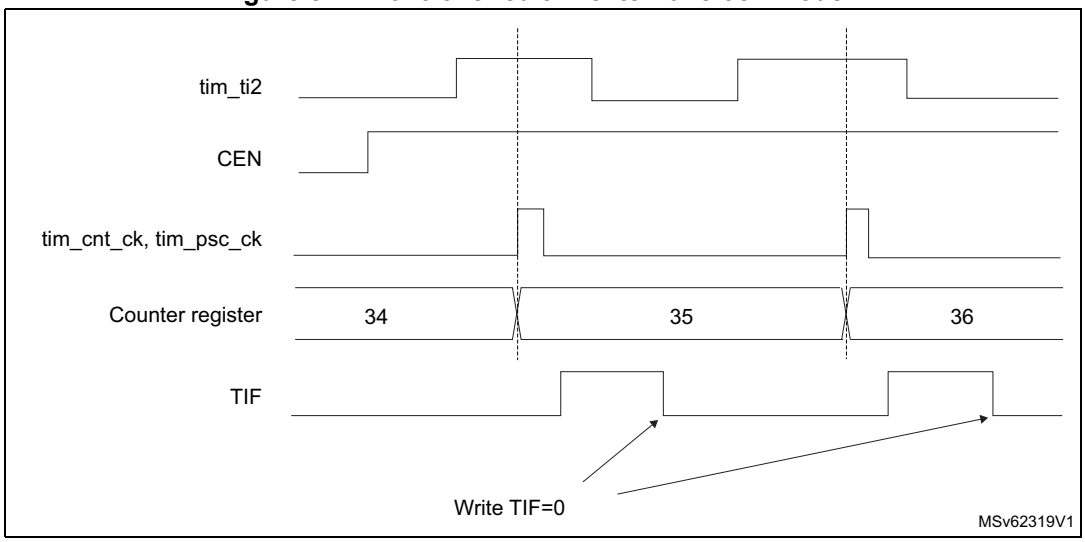
1. Configure channel 2 to detect rising edges on the tim_ti2 input by writing CC2S = 01 in the TIMx_CCMR1 register.
2. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F = 0000).
3. Select rising edge polarity by writing CC2P = 0 and CC2NP = 0 in the TIMx_CCER register.
4. Configure the timer in external clock mode 1 by writing SMS = 111 in the TIMx_SMCR register.
5. Select tim_ti2 as the trigger input source by writing TS = 00110 in the TIMx_SMCR register.
6. Enable the counter by writing CEN = 1 in the TIMx_CR1 register.

Note: The capture prescaler is not used for triggering, it is not necessary to configure it.

When a rising edge occurs on tim_ti2, the counter counts once and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual clock of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 524. Control circuit in external clock mode 1



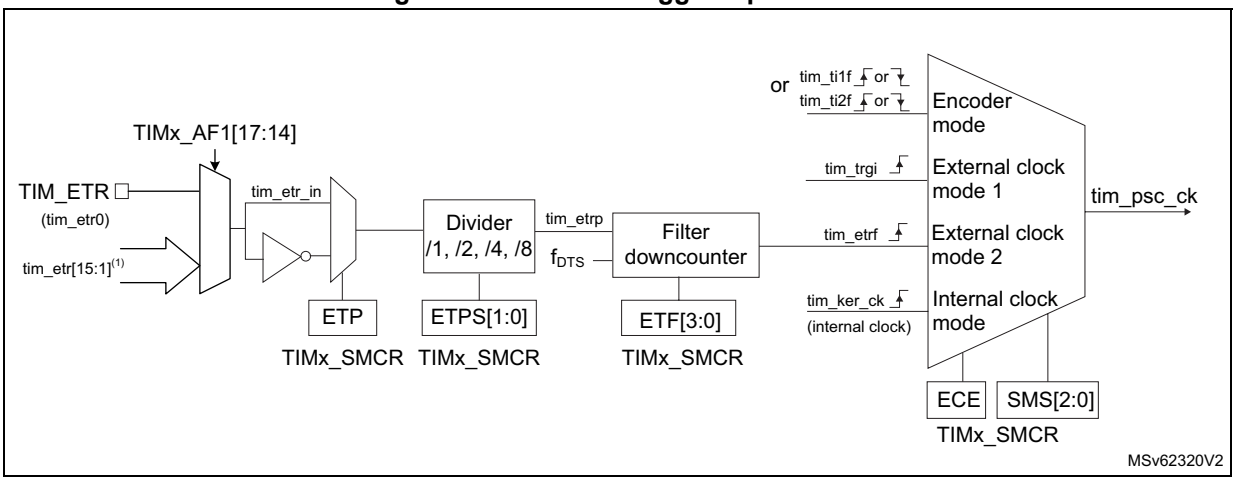
External clock source mode 2

This mode is selected by writing ECE = 1 in the TIMx_SMCR register.

The counter counts at each rising or falling edge on the external trigger input tim_etr_in.

The *Figure 525* gives an overview of the external trigger input block.

Figure 525. External trigger input block



1. Refer to *Section 53.3.2: TIM1/TIM8 pins and internal signals*.

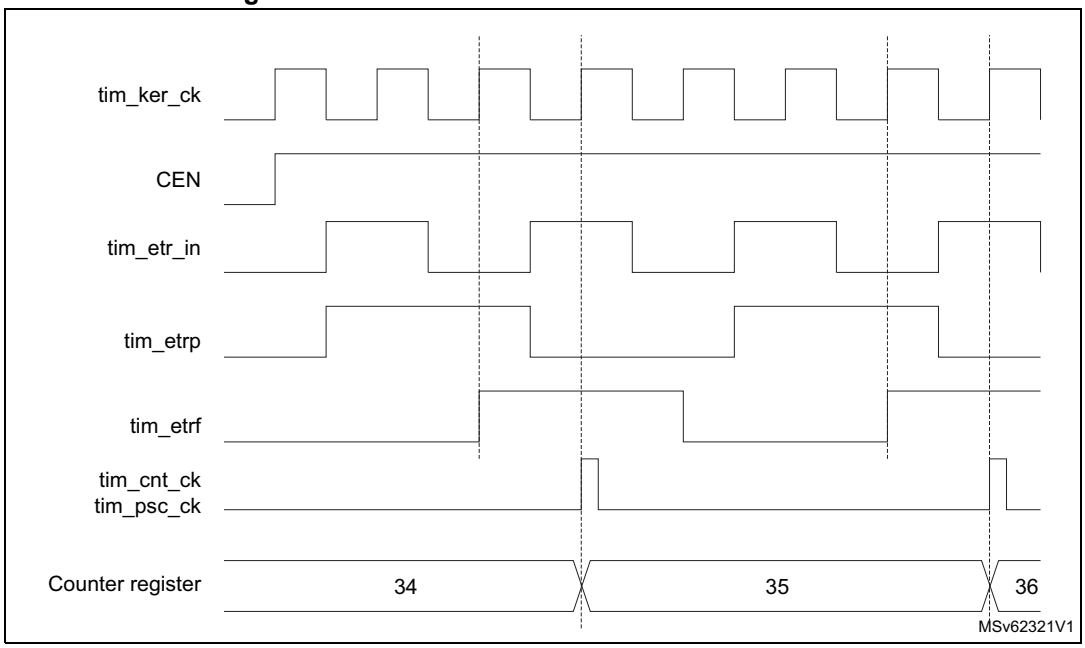
For example, to configure the upcounter to count each 2 rising edges on tim_etr_in, use the following procedure:

1. As no filter is needed in this example, write ETF[3:0] = 0000 in the TIMx_SMCR register.
2. Set the prescaler by writing ETPS[1:0] = 01 in the TIMx_SMCR register
3. Select rising edge detection on the tim_etr_in input by writing ETP = 0 in the TIMx_SMCR register
4. Enable external clock mode 2 by writing ECE = 1 in the TIMx_SMCR register.
5. Enable the counter by writing CEN = 1 in the TIMx_CR1 register.

The counter counts once each 2 tim_etr_in rising edges.

The delay between the rising edge on tim_etr_in and the actual clock of the counter is due to the resynchronization circuit on the tim_etrp signal. As a consequence, the maximum frequency which can be correctly captured by the counter is at most ¼ of tim_ker_ck frequency. When the ETRP signal is faster, the user must apply a division of the external signal by a proper ETPS prescaler setting.

Figure 526. Control circuit in external clock mode 2



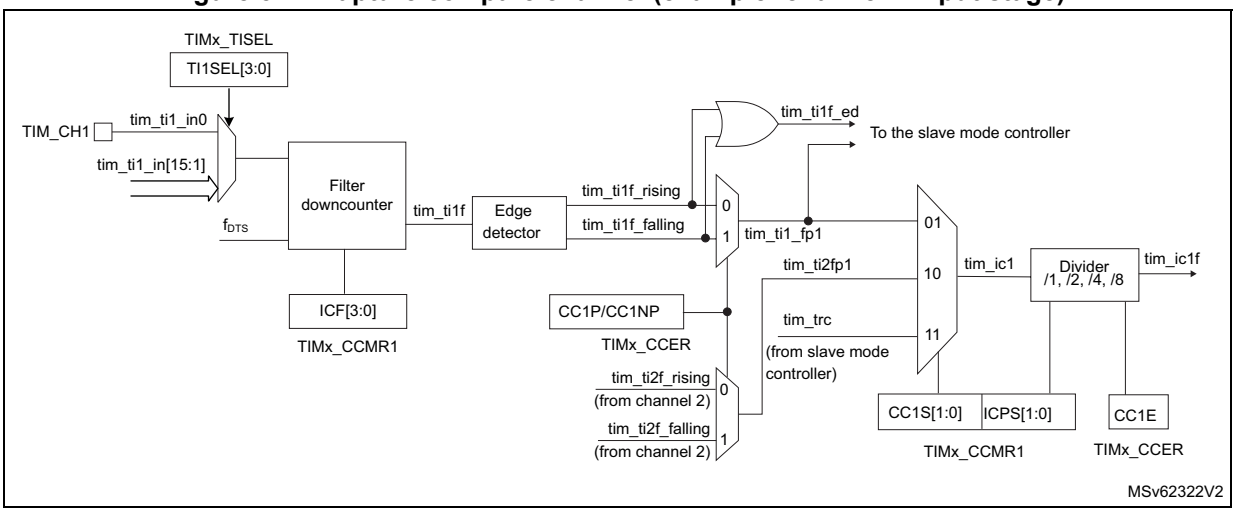
53.3.8 Capture/compare channels

Each capture/compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing, and prescaler, except for channels 5 and 6) and an output stage (with comparator and output control).

Figure 527 to *Figure 530* give an overview of one capture/compare channel.

The input stage samples the corresponding tim_tix input to generate a filtered signal tim_tixf. Then, an edge detector with polarity selection generates a signal (tim_tixfpy) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 527. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: tim_ocxref (active high). The polarity acts at the end of the chain.

Figure 528. Capture/compare channel 1 main circuit

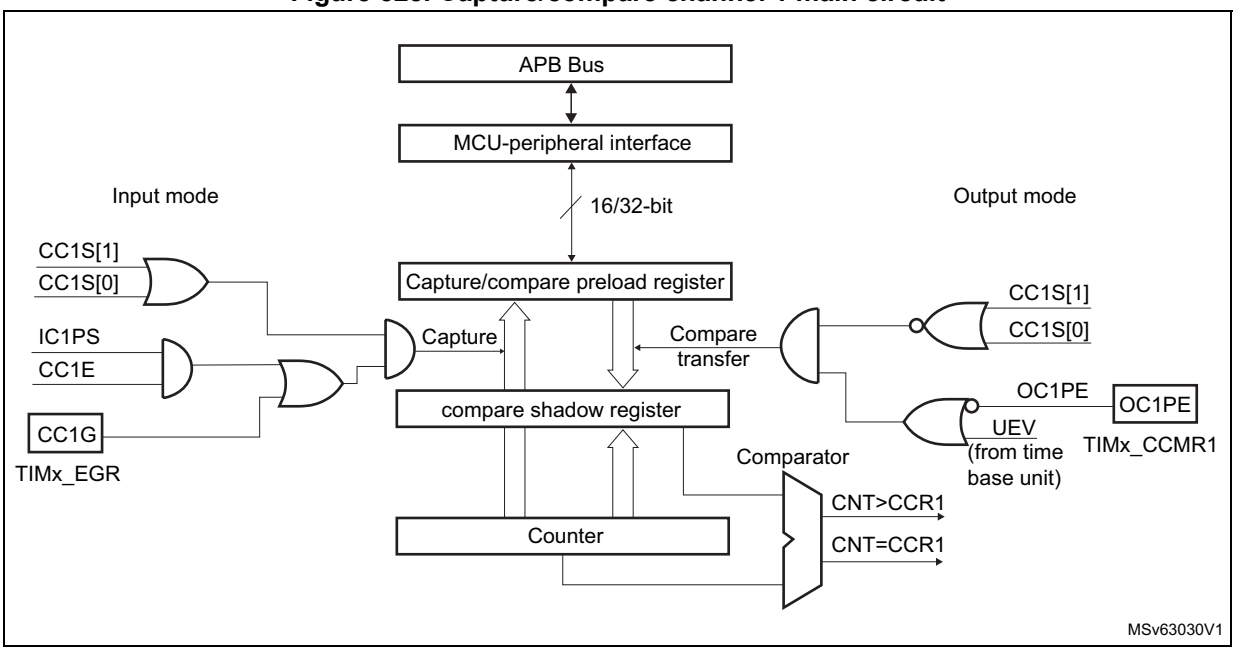
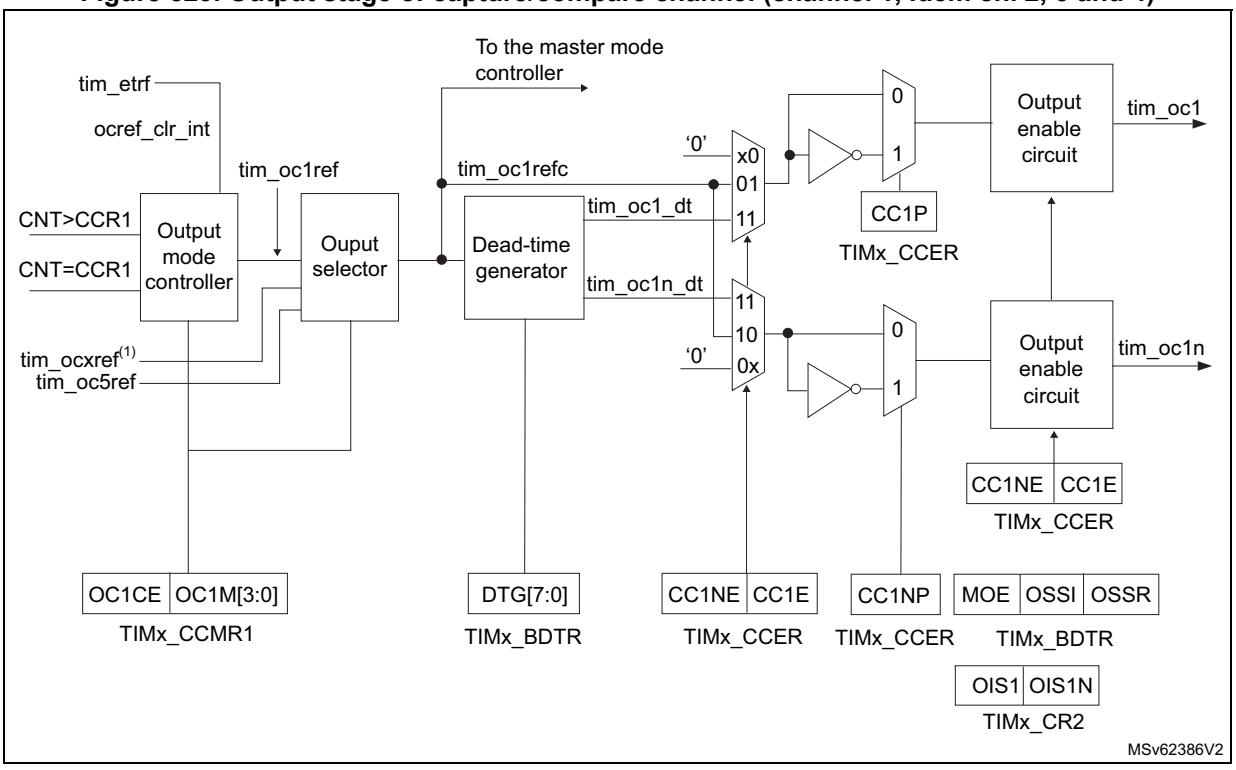
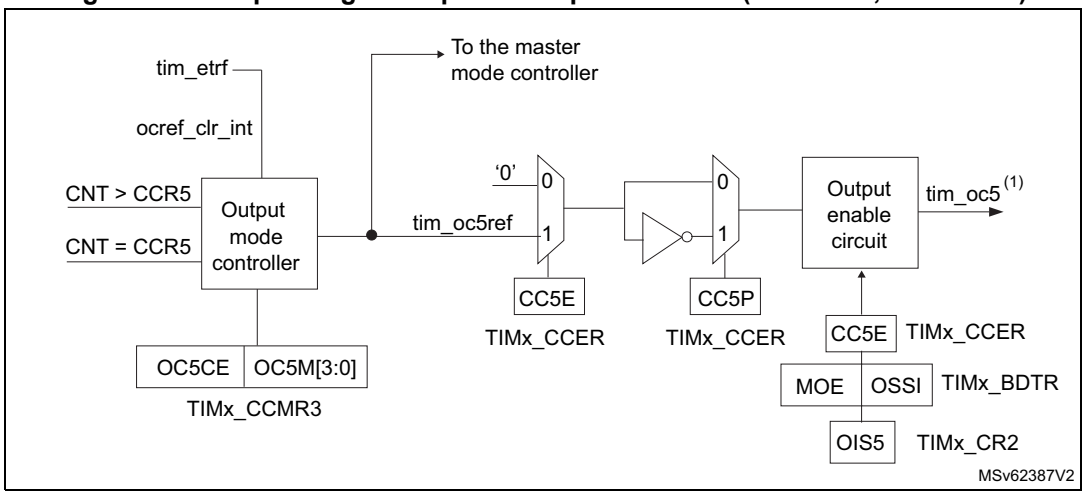


Figure 529. Output stage of capture/compare channel (channel 1, idem ch. 2, 3 and 4)



1. tim_ocxref, where x is the rank of the complementary channel

Figure 530. Output stage of capture/compare channel (channel 5, idem ch. 6)



1. Not available externally.

The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

53.3.9 Input capture mode

In Input capture mode, the capture/compare registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the overcapture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to 0 or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx_CCR1 when tim_ti1 input rises. To do this, use the following procedure:

- Select the active input: TIMx_CCR1 must be linked to the tim_ti1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input, and the TIMx_CCR1 register becomes read-only.
- Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the tim_tix (ICxF bits in the TIMx_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at must five internal clock cycles. We must program a filter duration longer than these five clock cycles. We can validate a transition on tim_ti1 when eight consecutive samples with the new level

have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.

- Select the edge of the active transition on the tim_ti1 channel by writing CC1P and CC1NP bits to 0 in the TIMx_CCER register (rising edge in this case).
- Program the input prescaler. In our example, we wish the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIMx_CCMR1 register).
- Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
- If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which may happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

53.3.10 PWM input mode

This mode is used to measure both the period and the duty cycle of a PWM signal connected to single tim_tix input:

- The TIMx_CCR1 register holds the period value (interval between two consecutive rising edges).
- The TIM_CCR2 register holds the pulsewidth (interval between two consecutive rising and falling edges.

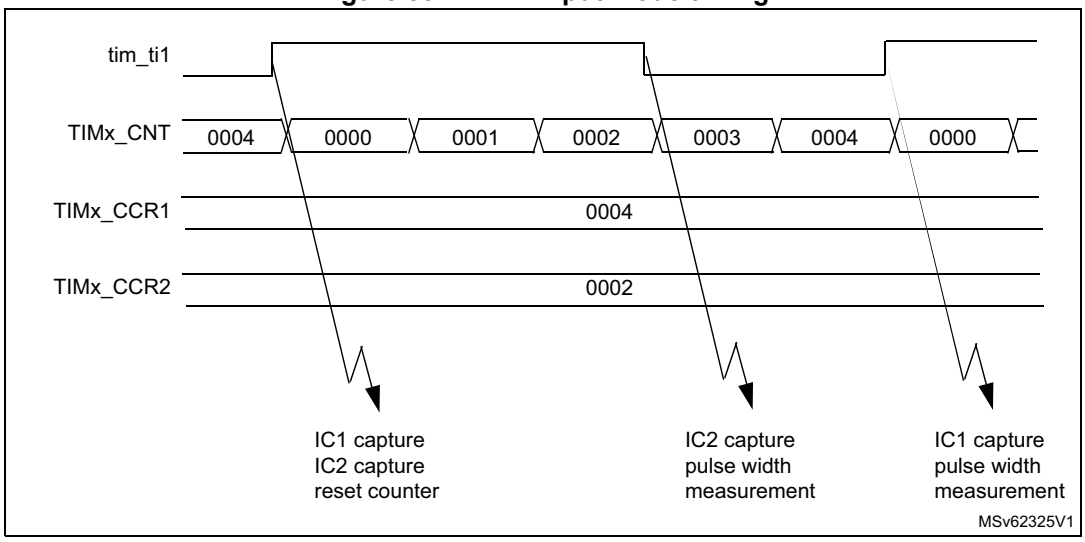
This mode is a particular case of input capture mode. The set-up procedure is similar with the following differences:

- Two ICx signals are mapped on the same tim_tixfp1 input.
- These two ICx signals are active on edges with opposite polarity.
- One of the two tim_tixfp signals is selected as trigger input and the slave mode controller is configured in reset mode.

The period and the pulsewidth of a PWM signal applied on tim_ti1 can be measured using the following procedure:

- Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (tim_ti1 selected).
- Select the active polarity for tim_ti1fp1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P and CC1NP bits to 0 (active on rising edge).
- Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (tim_ti1 selected).
- Select the active polarity for tim_ti1fp2 (used for capture in TIMx_CCR2): write the CC2P and CC2NP bits to CC2P/CC2NP = 10 (active on falling edge).
- Select the valid trigger input: write the TS bits to 00101 in the TIMx_SMCR register (tim_ti1fp1 selected).
- Configure the slave mode controller in reset mode: write the SMS bits to 0100 in the TIMx_SMCR register.
- Enable the captures: write the CC1E and CC2E bits to 1 in the TIMx_CCER register.

Figure 531. PWM input mode timing



53.3.11 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (tim_ocxref and then tim_ocx/tim_ocxn) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (tim_ocxref/tim_ocx) to its active level, user just needs to write 0101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus tim_ocxref is forced high (tim_ocxref is always active high) and tim_ocx get opposite value to CCxP polarity bit.

For example: CCxP = 0 (tim_ocx active high) => tim_ocx is forced to high level.

The tim_ocxref signal can be forced low by writing the OCxM bits to 0100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

53.3.12 Output compare mode

This function is used to control an output waveform or indicate when a period of time has elapsed. Channels 1 to 4 can be output, while channel 5 and 6 are only available inside the microcontroller (for instance, for compound waveform generation or for ADC triggering).

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM = 0000), be set active (OCxM = 0001), be set inactive (OCxM = 0010) or can toggle (OCxM = 0011) on match.
- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

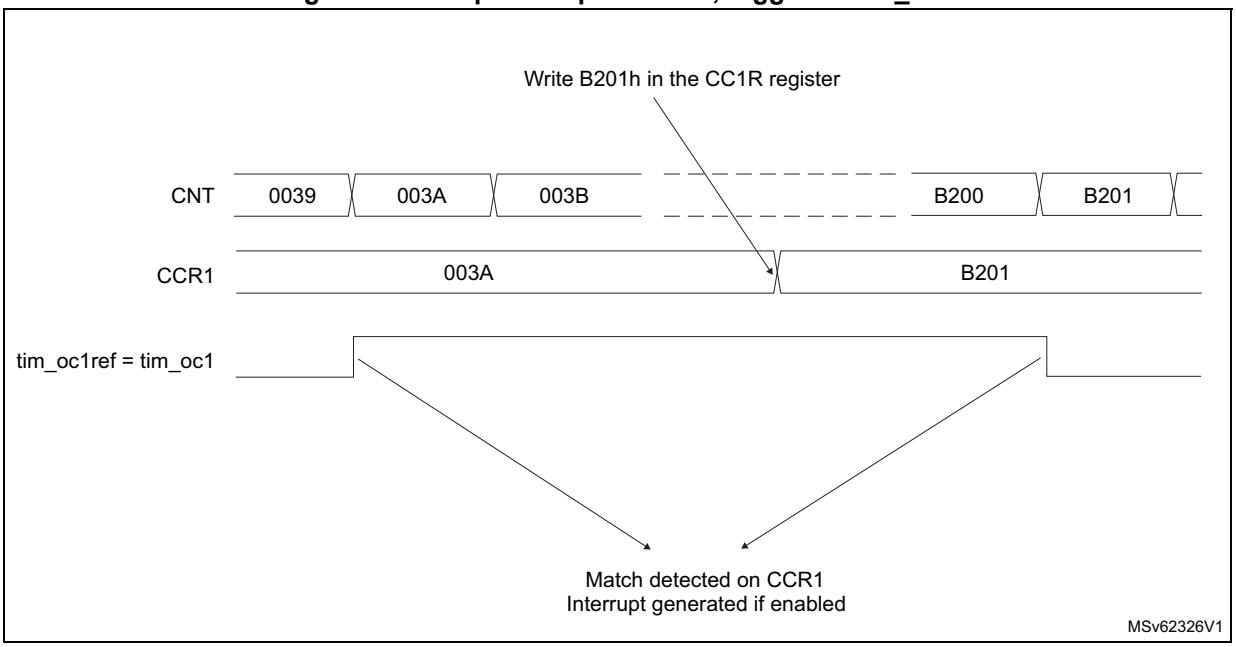
In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 0011 to toggle tim_ocx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register
   - Write CCxP = 0 to select active high polarity
   - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE = 0, else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 532*.

Figure 532. Output compare mode, toggle on tim_oc1



53.3.13 PWM mode

Pulse width modulation mode is used to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per tim_ocx output) by writing 0110 (PWM mode 1) or 0111 (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the autoreload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

tim_ocx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. tim_ocx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI, and OSSR bits (TIMx_CCER and TIMx_BDTR registers). Refer to the TIMx_CCER register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter).

The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx_CR1 register.

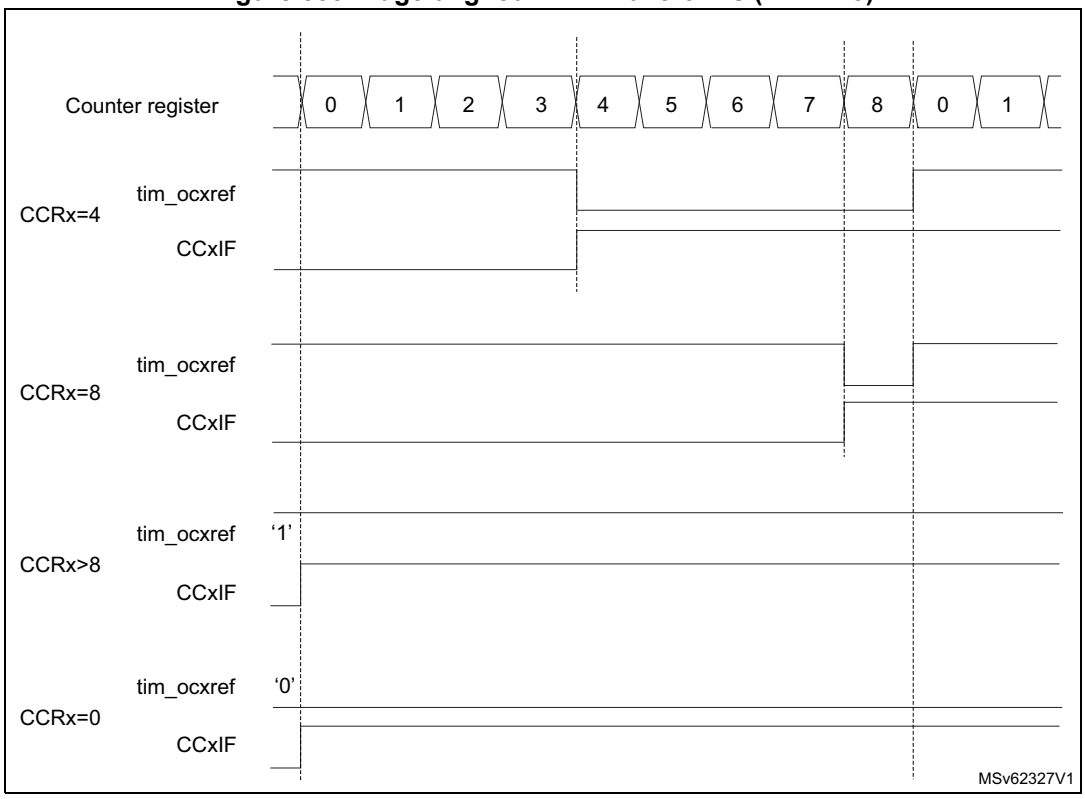
PWM edge-aligned mode

- Upcounting configuration

  Upcounting is active when the DIR bit in the TIMx_CR1 register is low. Refer to *Upcounting mode*.

  In the following example, the mode is PWM mode 1. The reference PWM signal tim_ocxref is high as long as TIMx_CNT < TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the autoreload value (in TIMx_ARR) then tim_ocxref is held at 1. If the compare value is zero then tim_ocxref is held at 0. *Figure 533* shows some edge-aligned PWM waveforms in an example where TIMx_ARR = 8.

Figure 533. Edge-aligned PWM waveforms (ARR = 8)



- Downcounting configuration

  Downcounting is active when DIR bit in TIMx_CR1 register is high. Refer to the *Downcounting mode*

  In PWM mode 1, the reference signal tim_ocxref is low as long as TIMx_CNT > TIMx_CCRx else it becomes high. If the compare value in TIMx_CCRx is greater than the autoreload value in TIMx_ARR, then tim_ocxref is held at 1. 0% PWM is not possible in this mode.

PWM center-aligned mode

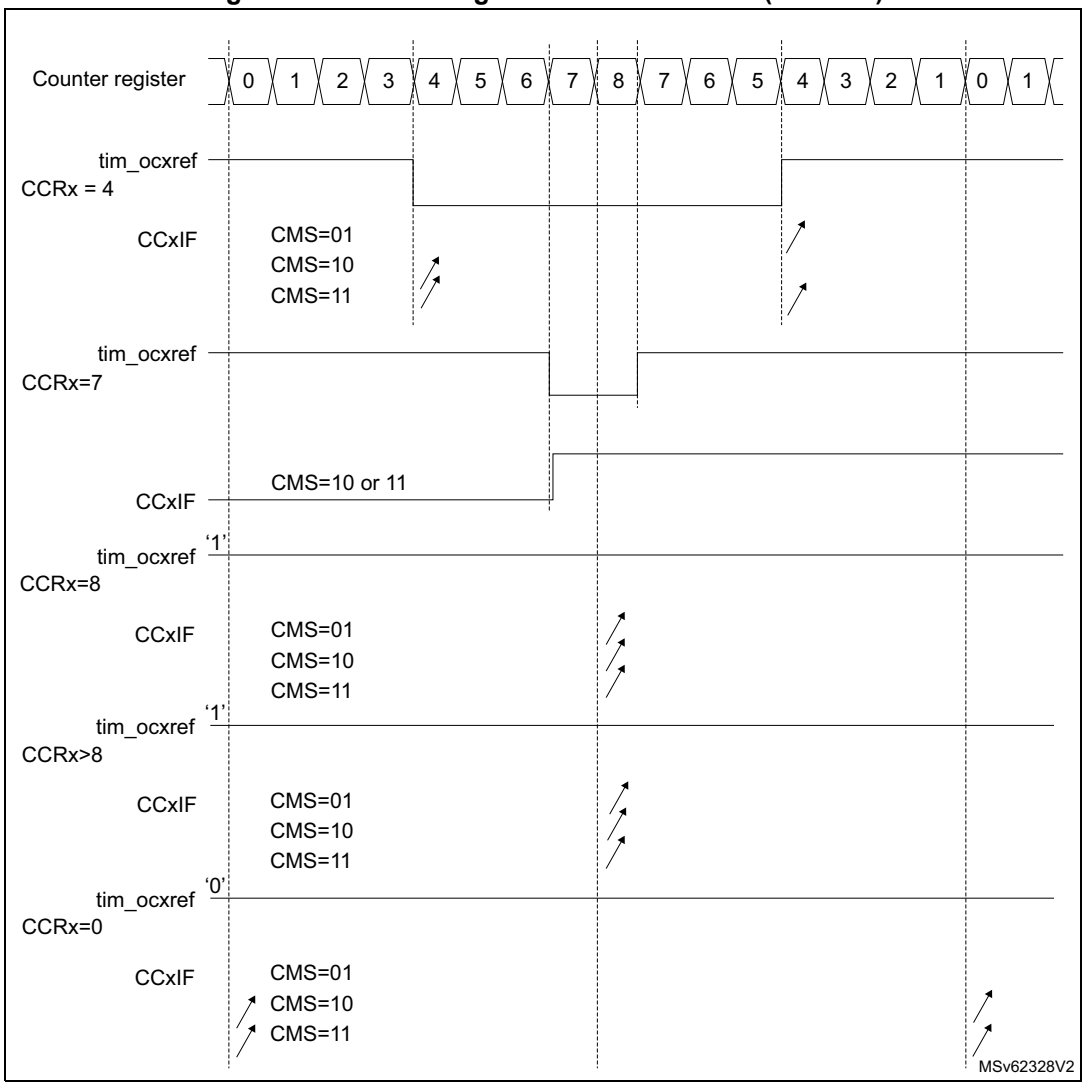
Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from 00 (all the remaining configurations having the same effect on the tim_ocxref/tim_ocx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit

(DIR) in the TIMx_CR1 register is updated by hardware and must not be changed by software. Refer to *Center-aligned mode (up/down counting)*.

Figure 534 shows some center-aligned PWM waveforms in an example where:

- TIMx_ARR = 8
- PWM mode is the PWM mode 1
- The flag is set when the counter counts down corresponding to the center-aligned mode 1 selected for CMS = 01 in TIMx_CR1 register.

Figure 534. Center-aligned PWM waveforms (ARR = 8)



Hints on using center-aligned mode:

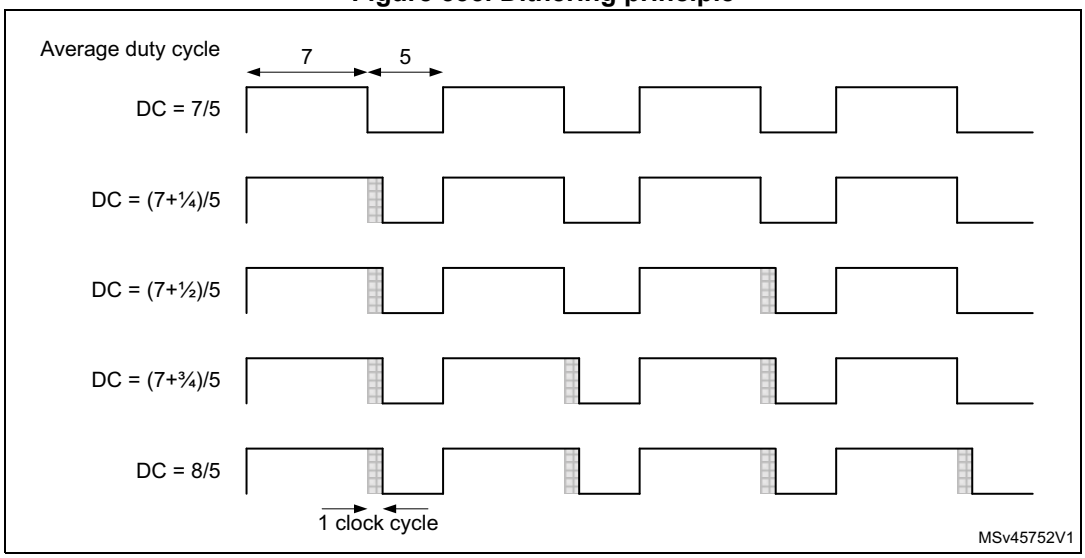
- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit in the TIMx_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if a value greater than the autoreload value is written in the counter (TIMx_CNT > TIMx_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if 0 or the TIMx_ARR value is written in the counter but no update event UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIMx_EGR register) just before starting the counter and not to write the counter while it is running.

Dithering mode

The PWM mode effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIMx_CR1 register. This applies to both the CCR (for duty cycle resolution increase) and ARR (for PWM frequency resolution increase).

The operating principle is to have the actual CCR (or ARR) value slightly changed (adding or not one timer clock period) over 16 consecutive PWM periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average duty cycle or PWM period. *Figure 535* presents the dithering principle applied to four consecutive PWM cycles.

Figure 535. Dithering principle

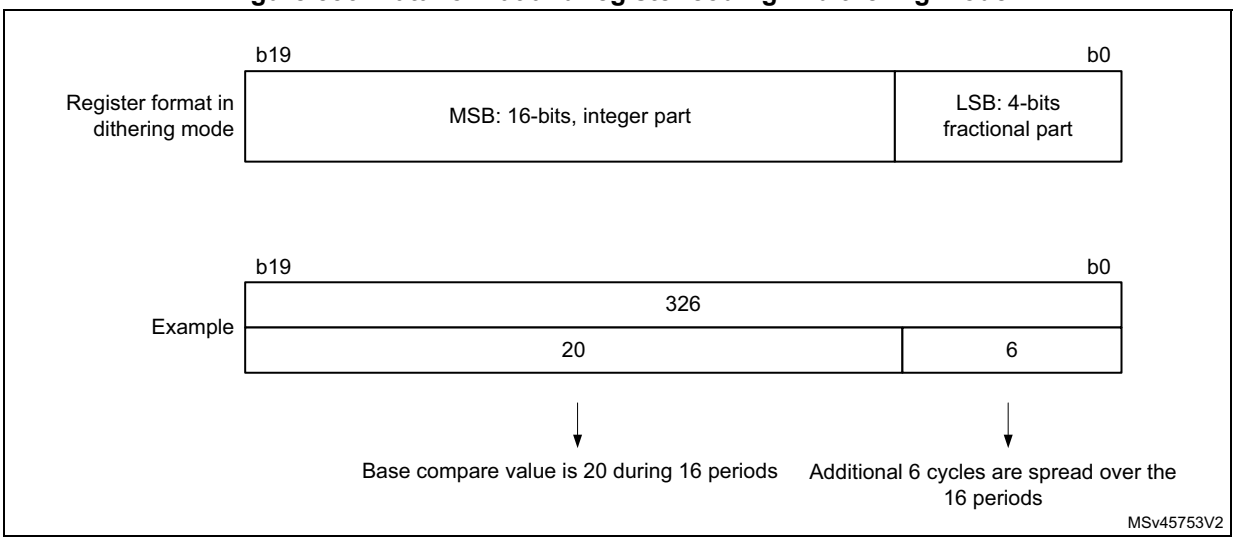


When the dithering mode is enabled, the register coding is changed as follows (see *Figure 536* for example):

- The four LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted to the bits 19:4 and are coding for the base value.

Note: The following sequence must be followed when resetting the DITHEN bit:
1. CEN and ARPE bits must be reset.
2. The DITHEN bit must be reset.
3. The CCIF flags must be cleared.
4. The CEN bit can be set (eventually with ARPE = 1).

Figure 536. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

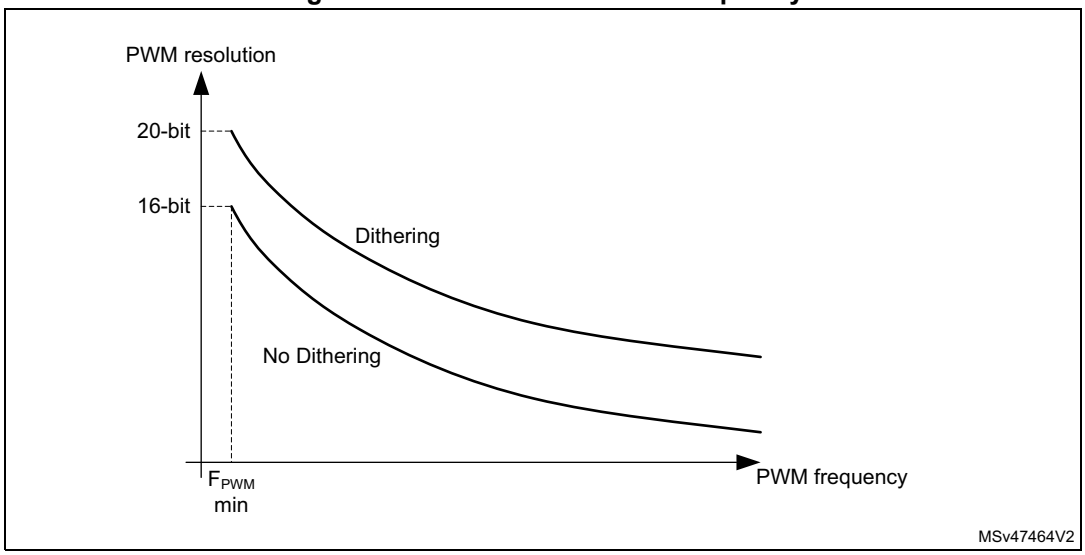
$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode enabled: } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

Note: The maximum TIMx_ARR and TIMxCCRy values are limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).

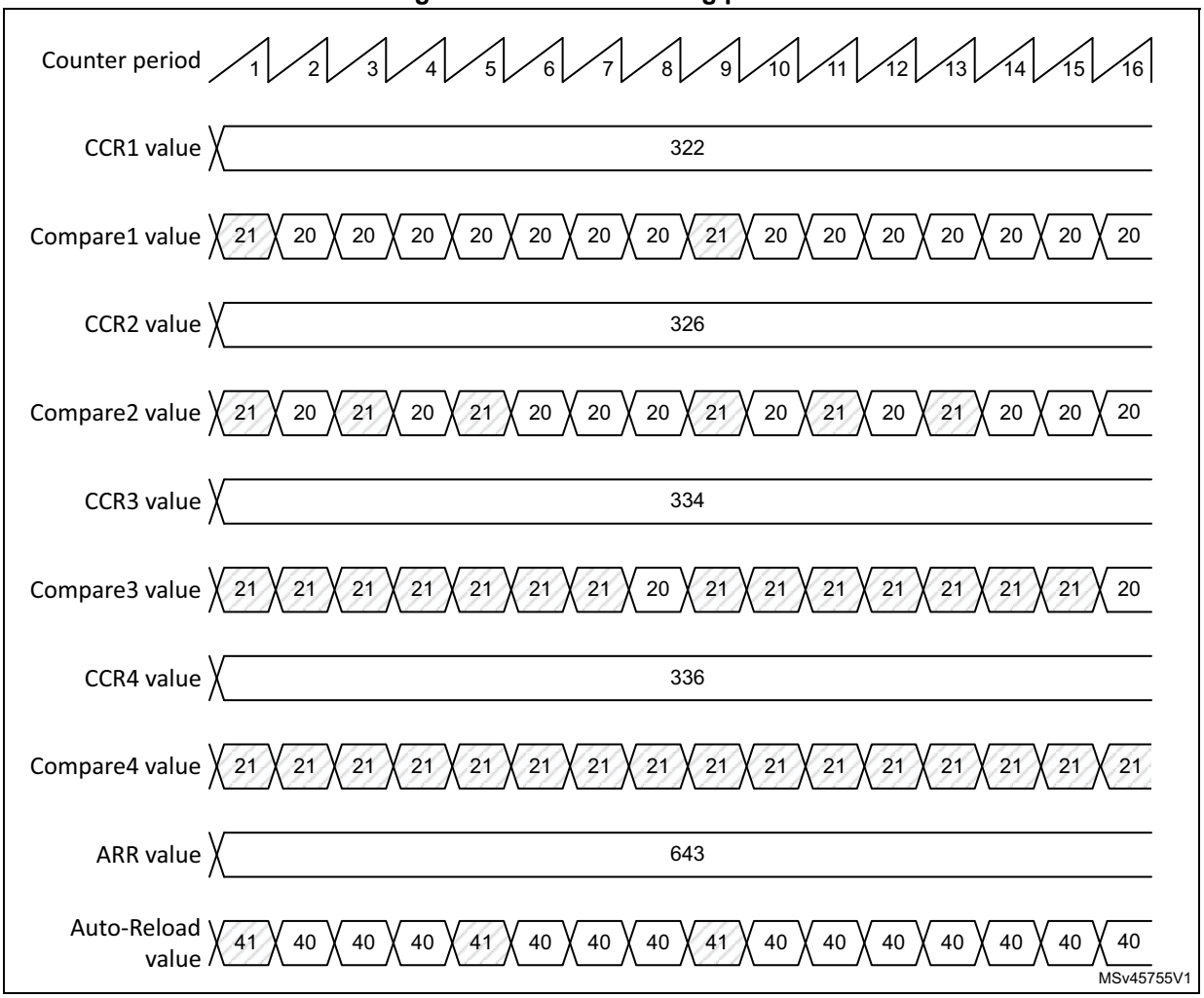
As shown on *Figure 537*, the dithering mode is used to increase the PWM resolution whatever the PWM frequency.

Figure 537. PWM resolution vs frequency



The duty cycle and/or period changes are spread over 16 consecutive periods, as described in *Figure 538*.

Figure 538. PWM dithering pattern



The autoreload and compare values increments are spread following specific patterns described in *Table 491*. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 491. CCR and ARR register change dithering pattern

LSB value	PWM period															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-

Table 491. CCR and ARR register change dithering pattern (continued)

LSB value	PWM period															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

The dithering mode is also available in center-aligned PWM mode (CMS bits in TIMx_CR1 register are not equal to 00). In this case, the dithering pattern is applied over eight consecutive PWM periods, considering the up and down counting phases as shown in *Figure 539*.

Figure 539. Dithering effect on duty cycle in center-aligned PWM mode

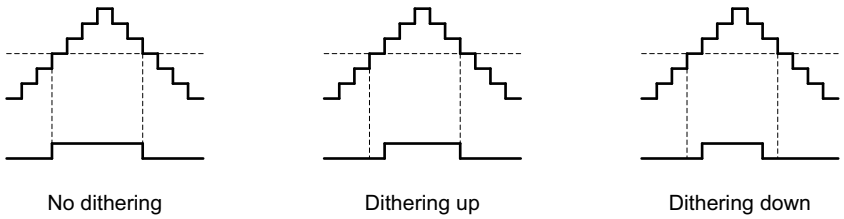


Table 492 shows how the dithering pattern is added in center-aligned PWM mode.

Table 492. CCR register change dithering pattern in center-aligned PWM mode

LSB value	PWM period															
	1		2		3		4		5		6		7		8	
	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-

Table 492. CCR register change dithering pattern in center-aligned PWM mode (continued)

LSB value	PWM period															
	1		2		3		4		5		6		7		8	
	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

53.3.14 Asymmetric PWM mode

Asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and the phase-shift are determined by a pair of TIMx_CCRx register. One register controls the PWM during up-counting, the second during down counting, so that PWM is adjusted every half PWM cycle:

- tim_oc1refc (or tim_oc2refc) is controlled by TIMx_CCR1 and TIMx_CCR2
- tim_oc3refc (or tim_oc4refc) is controlled by TIMx_CCR3 and TIMx_CCR4

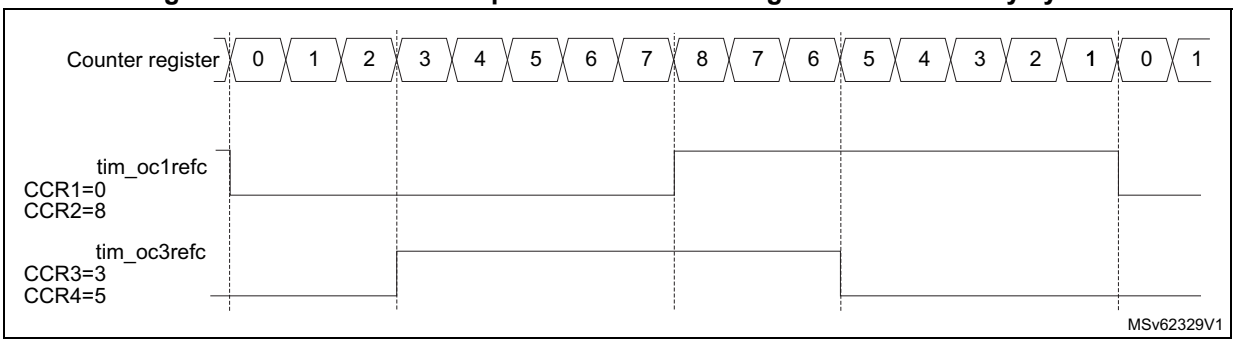
Asymmetric PWM mode can be selected independently on two channel (one tim_ocx output per pair of CCR registers) by writing 1110 (Asymmetric PWM mode 1) or 1111 (Asymmetric PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

Note: The OCxM[3:0] bitfield is split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

When a given channel is used as asymmetric PWM channel, its complementary channel can also be used. For instance, if an tim_oc1refc signal is generated on channel 1 (Asymmetric PWM mode 1), it is possible to output either the tim_oc2ref signal on channel 2, or an tim_oc2refc signal resulting from asymmetric PWM mode 1.

Figure 540 represents an example of signals that can be generated using asymmetric PWM mode (channels 1 to 4 are configured in asymmetric PWM mode 2). Together with the deadtime generator, this allows a full-bridge phase-shifted DC to DC converter to be controlled.

Figure 540. Generation of 2 phase-shifted PWM signals with 50% duty cycle



53.3.15 Combined PWM mode

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, tim_ocxrefc, are made of an OR or AND logical combination of two reference PWMs:

- tim_oc1refc (or tim_oc2refc) is controlled by TIMx_CCR1 and TIMx_CCR2
- tim_oc3refc (or tim_oc4refc) is controlled by TIMx_CCR3 and TIMx_CCR4

Combined PWM mode can be selected independently on two channels (one tim_ocx output per pair of CCR registers) by writing 1100 (Combined PWM mode 1) or 1101 (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

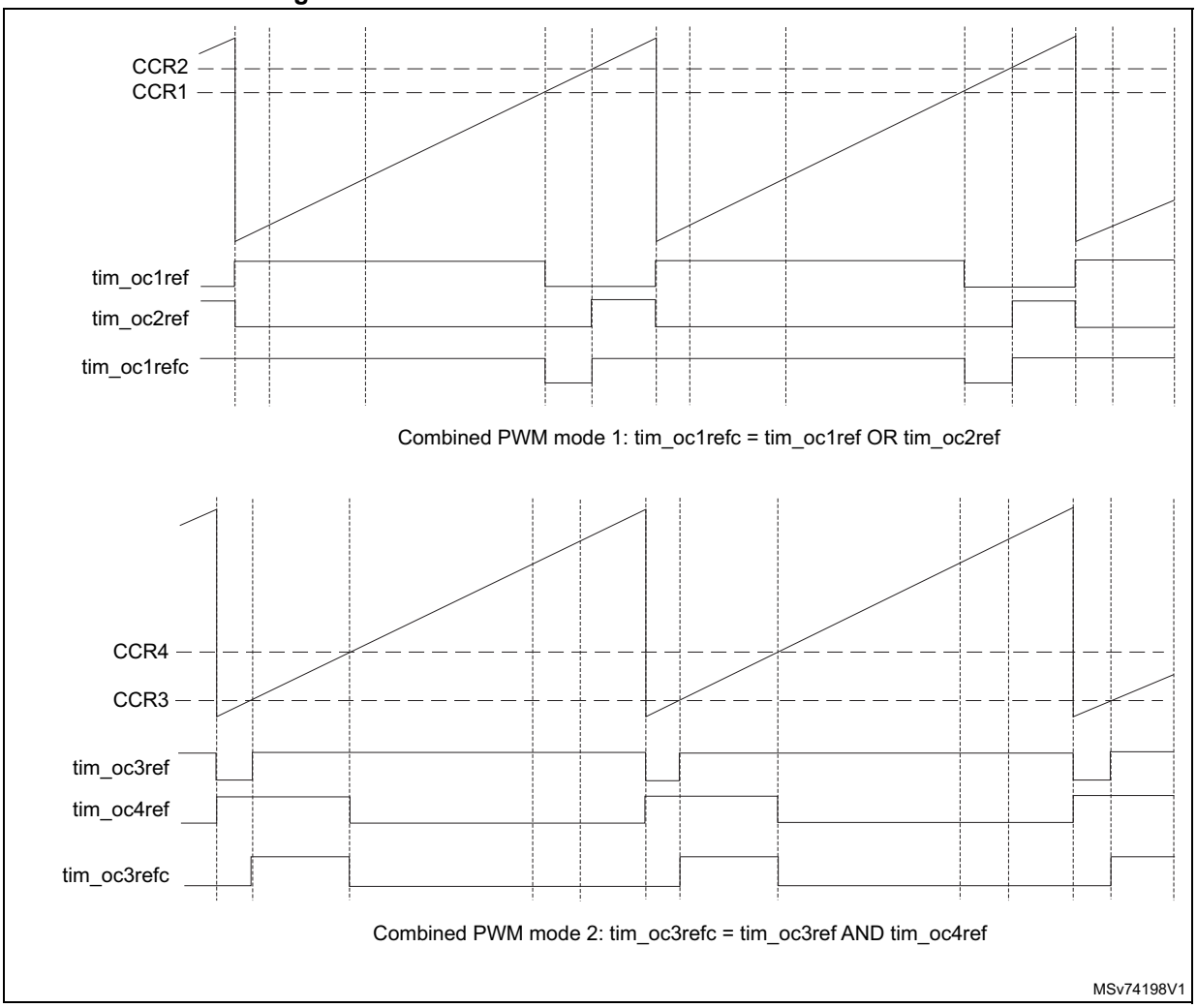
When a given channel is used as combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

Note: The OCxM[3:0] bitfield is split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

Figure 541 represents an example of signals that can be generated using combined PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2.
- Channel 2 is configured in PWM mode 1.
- Channel 3 is configured in Combined PWM mode 2.
- Channel 4 is configured in PWM mode 1.

Figure 541. Combined PWM mode on channel 1 and 3



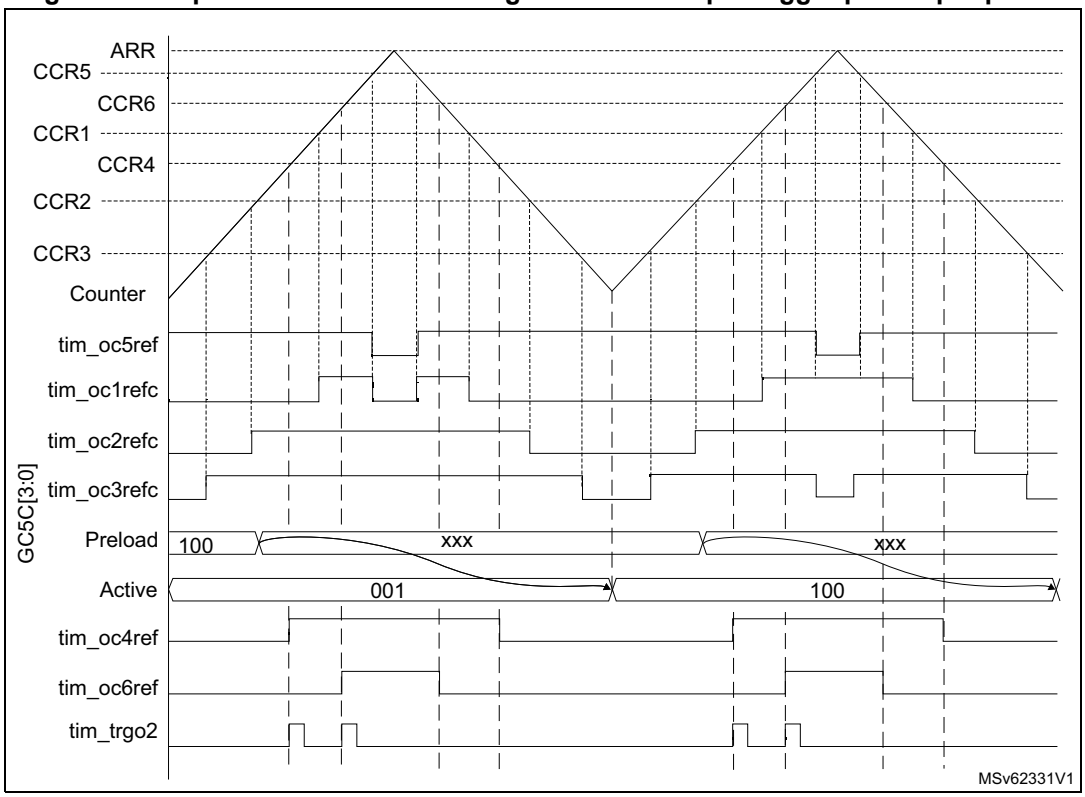
53.3.16 Combined 3-phase PWM mode

Combined 3-phase PWM mode allows one to three center-aligned PWM signals to be generated with a single programmable signal ANDed in the middle of the pulses. The tim_oc5ref signal is used to define the resulting combined signal. The 3-bits GC5C[3:1] in the TIMx_CCR5 allow selection on which reference signal the tim_oc5ref is combined. The resulting signals, tim_ocxrefc, are made of an AND logical combination of two reference PWMs:

- If GC5C1 is set, tim_oc1refc is controlled by TIMx_CCR1 and TIMx_CCR5.
- If GC5C2 is set, tim_oc2refc is controlled by TIMx_CCR2 and TIMx_CCR5.
- If GC5C3 is set, tim_oc3refc is controlled by TIMx_CCR3 and TIMx_CCR5.

Combined 3-phase PWM mode can be selected independently on channels 1 to 3 by setting at least one of the 3-bits GC5C[3:1].

Figure 542. 3-phase combined PWM signals with multiple trigger pulses per period



The tim_trgo2 waveform shows how the ADC can be synchronized on given 3-phase PWM signals. Refer to *Section 53.3.31: ADC triggers* for more details.

53.3.17 Complementary outputs and dead-time insertion

The advanced-control timers (TIM1/TIM8) can output two complementary signals and manage the switching-off and the switching-on instants of the outputs.

This time is generally known as dead-time and it has to be adjusted depending on the devices that are connected to the outputs and their characteristics (such as intrinsic delays of level-shifters, or delays due to power switches).

The polarity of the outputs (main output tim_ocx or complementary tim_ocxn) can be selected independently for each output. This is done by writing to the CCxP and CCxNP bits in the TIMx_CCER register.

The complementary signals tim_ocx and tim_ocxn are activated by a combination of several control bits: the CCxE and CCxNE bits in the TIMx_CCER register and the MOE, OISx, OISxN, OSSI, and OSSR bits in the TIMx_BDTR and TIMx_CR2 registers. Refer to *Table 500: Output control bits for complementary tim_ocx and tim_ocxn channels with break feature* for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a

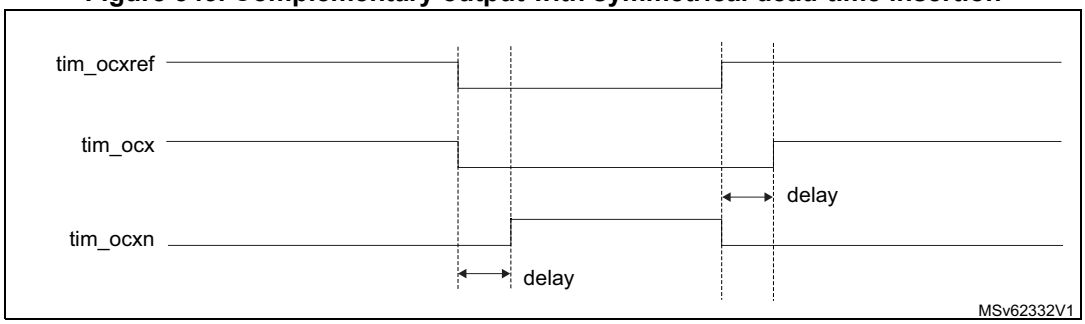
reference waveform tim_ocxref, it generates two outputs tim_ocx and tim_ocxn. If tim_ocx and tim_ocxn are active high:

- The tim_ocx output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The tim_ocxn output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

If the delay is greater than the width of the active output (tim_ocx or tim_ocxn) then the corresponding pulse is not generated.

The following figures show the relationships between the output signals of the dead-time generator and the reference signal tim_ocxref considering CCxP = 0, CCxNP = 0, MOE = 1, CCxE = 1 and CCxNE = 1 in these examples.

Figure 543. Complementary output with symmetrical dead-time insertion



The DTAE bit in the TIMx_DTR2 is used to differentiate the deadtime values for rising and falling edges of the reference signal, as shown on *Figure 544*.

In asymmetrical mode (DTAE = 1), the rising edge-referred deadtime is defined by the DTG[7:0] bitfield in the TIMx_BDTR register, while the falling edge-referred is defined by the DTGF[7:0] bitfield in the TIMx_DTR2 register. The DTAE bit must be written before enabling the counter and must not be modified while CEN = 1.

It is possible to have the deadtime value updated on-the-fly during pwm operation, using a preload mechanism. The deadtime bitfield DTG[7:0] and DTGF[7:0] are preloaded when the DTPE bit is set, in the TIMX_DTR2 register. The preload value is loaded in the active register on the next update event.

Note: If the DTPE bit is enabled while the counter is enabled, any new value written since last update is discarded and previous value is used.

Figure 544. Asymmetrical deadtime

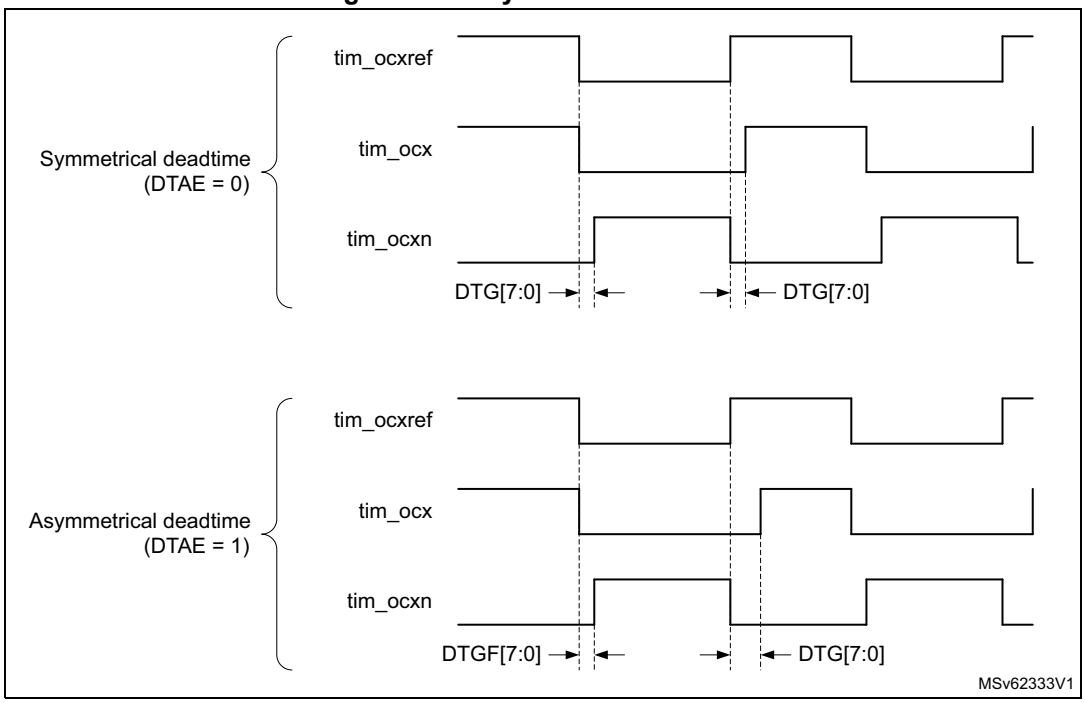


Figure 545. Dead-time waveforms with delay greater than the negative pulse

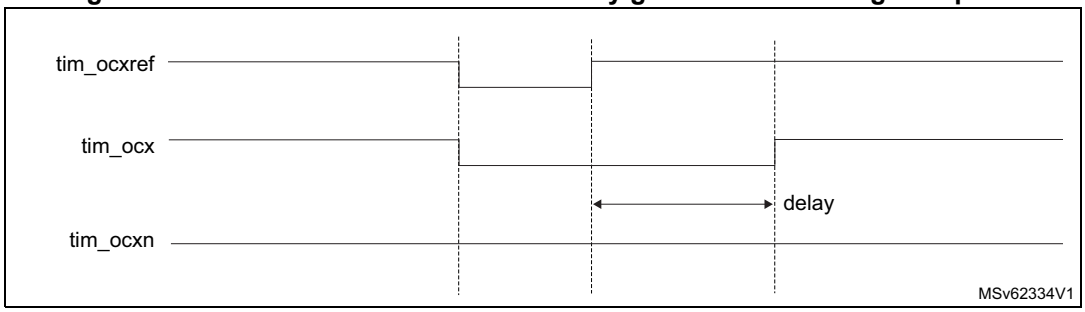
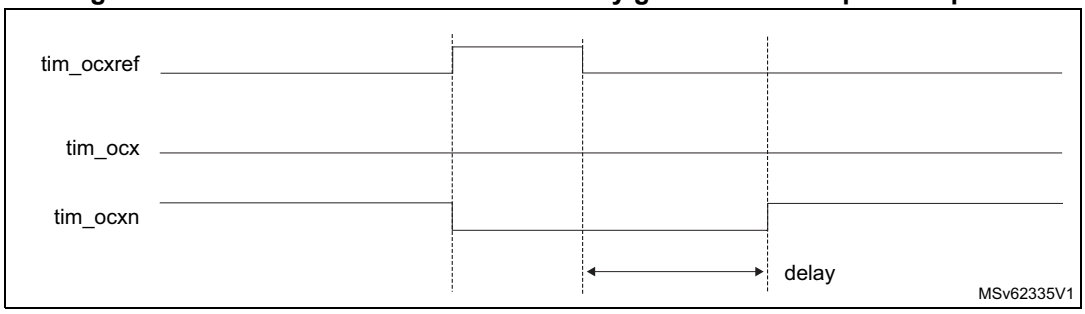


Figure 546. Dead-time waveforms with delay greater than the positive pulse



The dead-time delay is the same for each of the channels and is programmable with the DTG bits in the TIMx_BDTR register. Refer to *Section 53.6.20: TIMx break and dead-time register (TIMx_BDTR)(x = 1, 8)* for delay calculation.

Redirecting tim_ocxref to tim_ocx or tim_ocxn

In output mode (forced, output compare or PWM), tim_ocxref can be redirected to the tim_ocx output or to tim_ocxn output by configuring the CCxE and CCxNE bits in the TIMx_CCER register.

This is used to send a specific waveform (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

Note: When only tim_ocxn is enabled (CCxE = 0, CCxNE = 1), it is not complemented and becomes active as soon as tim_ocxref is high. For example, if CCxNP = 0 then tim_ocxn = tim_ocxref. On the other hand, when both tim_ocx and tim_ocxn are enabled (CCxE = CCxNE = 1) tim_ocx becomes active when tim_ocxref is high whereas tim_ocxn is complemented and becomes active when tim_ocxref is low.

53.3.18 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the timers. The two break inputs are usually connected to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry shuts down the PWM outputs and forces them to a predefined safe state. A number of internal MCU events can also be selected to trigger an output shut-down.

The break features two channels. A break channel which gathers both system-level fault (clock failure, ECC/parity errors,...) and application fault (from input pins and built-in comparator), and can force the outputs to a predefined level (either active or inactive) after a deadtime duration. A break2 channel which only includes application faults and is able to force the outputs to an inactive state.

The output enable signal and output levels during break are depending on several control bits:

- The MOE bit in TIMx_BDTR register is used to enable/disable the outputs by software and is reset in case of break or break2 event.
- The OSSI bit in the TIMx_BDTR register defines whether the timer controls the output in inactive state or releases the control to the GPIO controller (typically to have it in Hi-Z mode)
- The OISx and OISxN bits in the TIMx_CR2 register which are setting the output shut-down level, either active or inactive. The tim_ocx and tim_ocxn outputs cannot be set both to active level at a given time, whatever the OISx and OISxN values. Refer to *Table 500: Output control bits for complementary tim_ocx and tim_ocxn channels with break feature* for more details.

When exiting from reset, the break circuit is disabled and the MOE bit is low. The break functions can be enabled by setting the BKE and BK2E bits in the TIMx_BDTR register. The break input polarities can be selected by configuring the BKP and BK2P bits in the same register. BKEx and BKPx can be modified at the same time. When the BKEx and BKPx bits are written, a delay of one APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait one APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIMx_BDTR register). It results in some delays between the asynchronous

and the synchronous signals. In particular, if MOE is set to 1 whereas it was low, a delay must be inserted (dummy instruction) before reading it correctly. This is because the write acts on the asynchronous signal whereas the read reflects the synchronous signal.

The sources for break (tim_brk) channel are:

- External sources connected to one of the TIMx_BKIN pin (as per selection done in the GPIO alternate function selection registers), with polarity selection and optional digital filtering
- Internal sources:
  - coming from a tim_brk_cmpx input (refer to *Section 53.3.2: TIM1/TIM8 pins and internal signals* for product specific implementation)
  - coming from a system break request (refer to *Section 53.3.2: TIM1/TIM8 pins and internal signals* for product specific implementation)

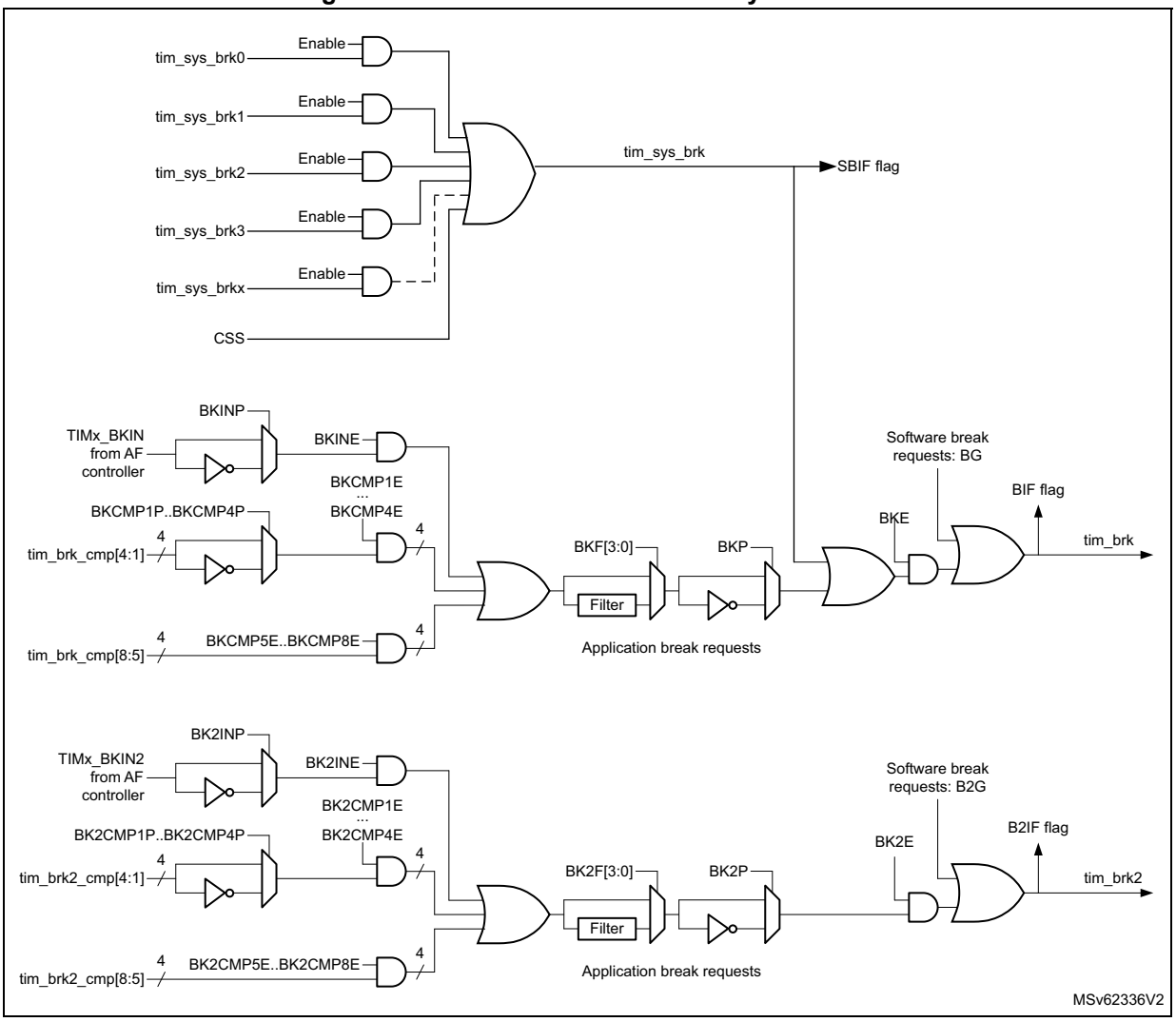
The sources for break2 (tim_brk2) are:

- External sources connected to one of the TIMx_BKIN2 pin (as per selection done in the GPIO alternate function selection registers), with polarity selection and optional digital filtering
- Internal sources coming from a tim_brk2_cmpx input (refer to *Section 53.3.2: TIM1/TIM8 pins and internal signals* for product specific implementation)

Break events can also be generated by software using BG and B2G bits in the TIMx_EGR register.

All sources are ORed before entering the timer tim_brk or tim_brk2 inputs, as per *Figure 547* below.

Figure 547. Break and Break2 circuitry overview



Note: An asynchronous (clockless) operation is only guaranteed when the programmable filter is disabled. If it is enabled, a fail safe clock mode (for example by using the internal PLL and/or the CSS) must be used to guarantee that break events are handled.

When one of the breaks occurs (selected level on one of the break inputs):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state, or even releasing the control to the GPIO controller (selected by the OSSI bit). This feature is enabled even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the OISx bit in the TIMx_CR2 register as soon as MOE = 0. If OSSI = 0, the timer releases the output control (taken over by the GPIO controller), otherwise the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, tim_ocx and tim_ocxn cannot be driven to

their active level together. Note that because of the resynchronization on MOE, the dead-time duration is slightly longer than usual (around 2 tim_ker_ck clock cycles).

  - If OSSI = 0, the timer releases the output control (taken over by the GPIO controller which forces a Hi-Z state), otherwise the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.

- The break status flag (SBIF, BIF, and B2IF bits in the TIMx_SR register) is set. An interrupt is generated if the BIE bit in the TIMx_DIER register is set. A DMA request can be sent if the BDE bit in the TIMx_DIER register is set.
- If the AOE bit in the TIMx_BDTR register is set, the MOE bit is automatically set again at the next update event (UEV). As an example, this can be used to perform a regulation. Otherwise, MOE remains low until the application sets it to 1 again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors, or any security components.

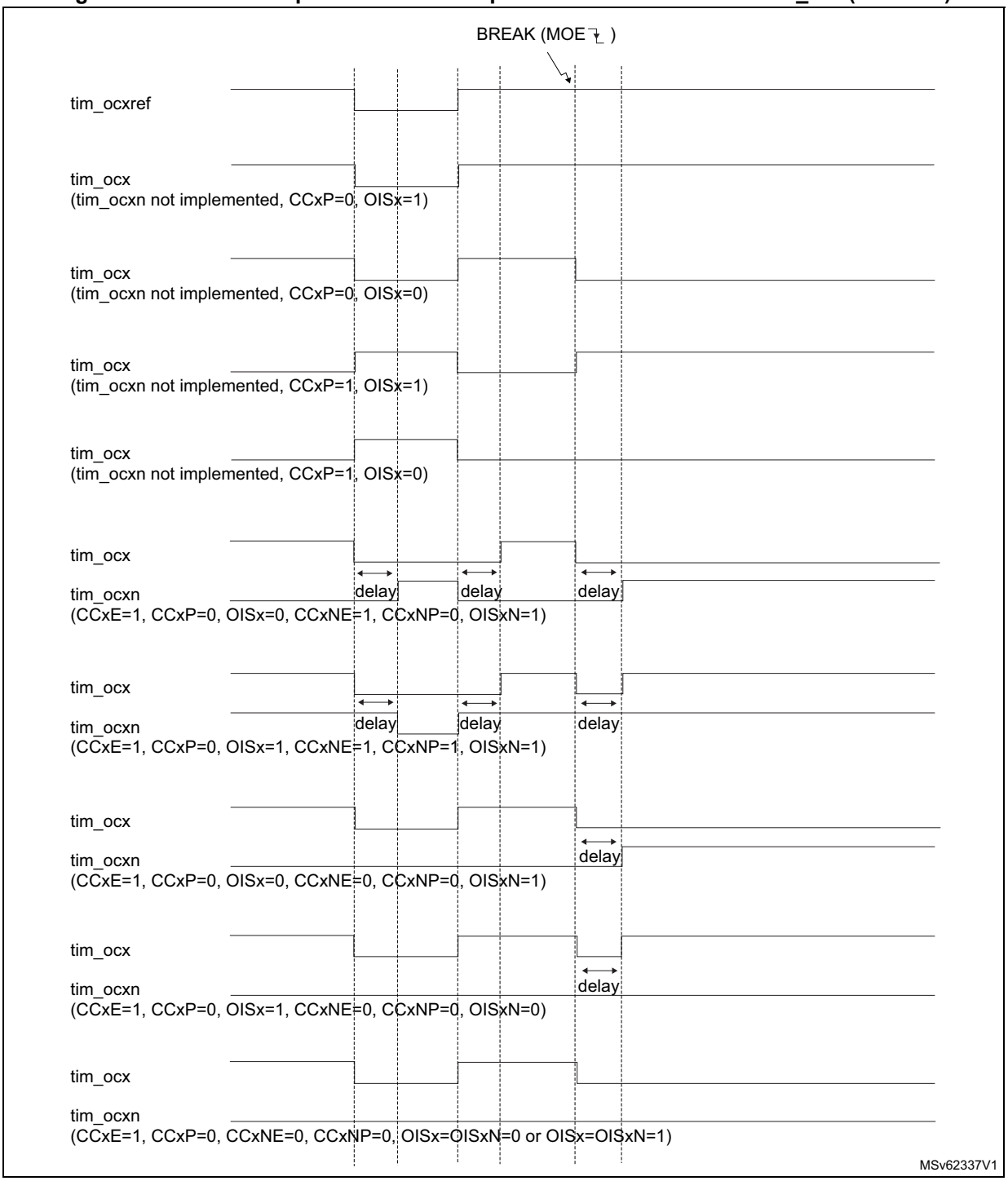
Note: If the MOE is reset by the CPU while the AOE bit is set, the outputs are in idle state and forced to inactive level or Hi-Z depending on OSSI value. If both the MOE and AOE bits are reset by the CPU, the outputs are in disabled state and driven with the level programmed in the OISx bit in the TIMx_CR2 register.

The break inputs are active on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF and B2IF cannot be cleared.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It is used to freeze the configuration of several parameters (dead-time duration, tim_ocx/tim_ocxn polarities and state when disabled, OCxM configurations, break enable, and polarity). The application can choose from three levels of protection selected by the LOCK bits in the TIMx_BDTR register. Refer to *Section 53.6.20*. The LOCK bits can be written only once after an MCU reset.

Figure 548 shows an example of behavior of the outputs in response to a break.

Figure 548. Various output behavior in response to a break event on tim_brk (OSSI = 1)



The two break inputs have different behaviors on timer outputs:

- The tim_brk input can either disable (inactive state) or force the PWM outputs to a predefined safe state.
- tim_brk2 can only disable (inactive state) the PWM outputs.

The tim_brk has a higher priority than tim_brk2 input, as described in *Table 493*.

Note: tim_brk2 must only be used with OSSR = OSSI = 1.

Table 493. Behavior of timer outputs versus tim_brk/tim_brk2 inputs

tim_brk	tim_brk2	Timer outputs state	Typical use case	
			tim_ocxn output (low side switches)	tim_ocx output (high side switches)
Active	X	– Inactive then forced output state (after a deadtime) – Outputs disabled if OSSI = 0 (control taken over by GPIO logic)	ON after deadtime insertion	OFF
Inactive	Active	Inactive	OFF	OFF

Figure 549 gives an example of tim_ocx and tim_ocxn output behavior in case of active signals on tim_brk and tim_brk2 inputs. In this case, both outputs have active high polarities (CCxP = CCxNP = 0 in TIMx_CCER register).

Figure 549. PWM output state following tim_brk and tim_brk2 assertion (OSSI = 1)

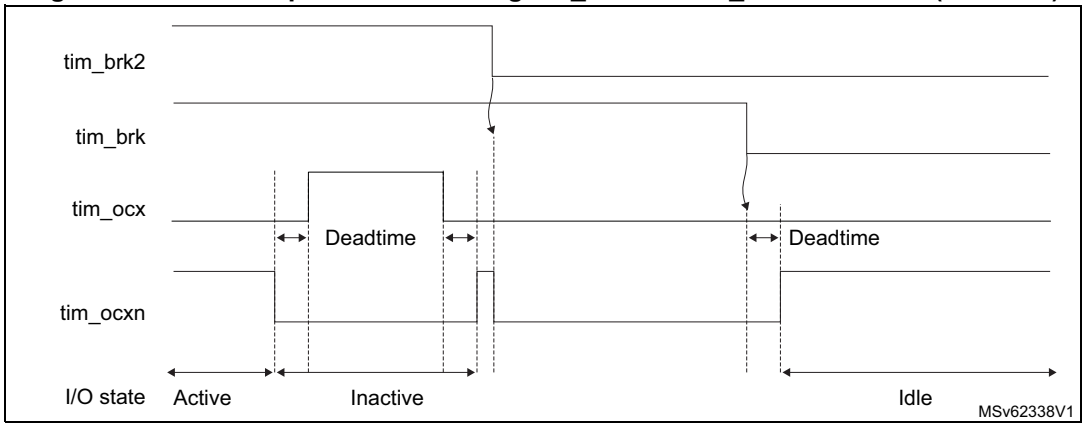
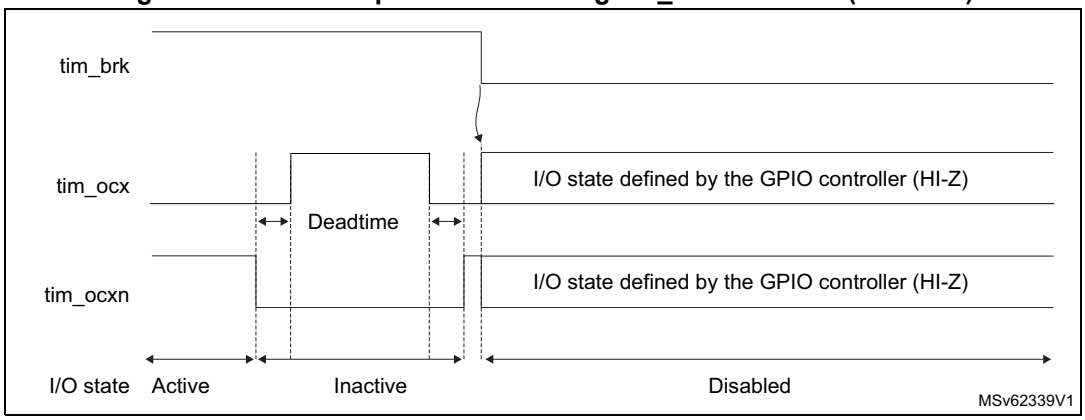


Figure 550. PWM output state following tim_brk assertion (OSSI = 0)



53.3.19 Bidirectional break inputs

The TIM1/TIM8 feature bidirectional break I/Os, as represented on *Figure 551*.

This provides support for:

- A board-level global break signal available for signaling faults to external MCUs or gate drivers, with a unique pin being both an input and an output status pin.
- Internal break sources and multiple external open drain sources ORed together to trigger a unique break event, when multiple internal and external break sources must be merged.

The tim_brk and tim_brk2 inputs are configured in bidirectional mode using the BKBID and BK2BID bits in the TIMxBDTR register. The BKBID programming bits can be locked in read-only mode using the LOCK bits in the TIMxBDTR register (in LOCK level 1 or above).

The bidirectional mode is available for both the tim_brk and tim_brk2 inputs, and require the I/O to be configured in open-drain mode with active low polarity (using BKINP, BKP, BK2INP and BK2P bits). Any break request coming either from system (for example CSS), from on-chip peripherals, or from break inputs forces a low level on the break input to signal the fault event. The bidirectional mode is inhibited if the polarity bits are not correctly set (active high polarity), for safety purposes.

The break software events (BG and B2G) also cause the break I/O to be forced to 0 to indicate to the external components that the timer is entered in break state. However, this is valid only if the break is enabled (BKE or B2KE = 1). When a software break event is generated with BKE or B2KE = 0), the outputs are put in safe state and the break flag is set, but there is no effect on the TIMx_BKIN and TIMx_BKIN2 I/Os.

A safe disarming mechanism prevents the system to be definitively locked-up (a low level on the break input triggers a break which enforces a low level on the same input).

When the BKDSRM (BK2DSRM) bit is set to 1, this releases the break output to clear a fault signal and to give the possibility to re-arm the system.

At no point the break protection circuitry can be disabled:

- The break input path is always active: a break event is active even if the BKDSRM (BK2DSRM) bit is set and the open drain control is released. This prevents the PWM output to be restarted as long as the break condition is present.
- The BKDSRM (BK2DSRM) bit cannot disarm the break protection as long as the outputs are enabled (MOE bit is set) (see *Table 494*).

Table 494. Break protection disarming conditions

MOE	BKBID (BK2BID)	BKDSRM (BK2DSRM)	Break protection state
0	0	X	Armed
0	1	0	Armed
0	1	1	Disarmed
1	X	X	Armed

Arming and rearming break circuitry

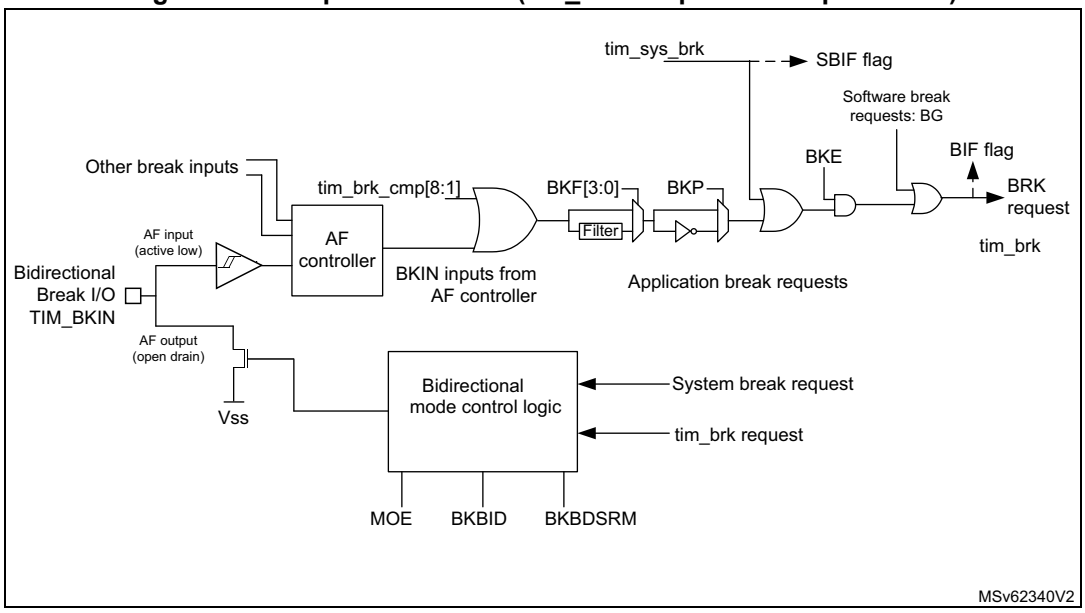
The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).

The following procedure must be followed to re-arm the protection after a break (break2) event:

- The BKDSRM (BK2DSRM) bit must be set to release the output control.
- The software must wait until the system break condition disappears (if any) and clear the SBIF status flag (or clear it systematically before rearming).
- The software must poll the BKDSRM (BK2DSRM) bit until it is cleared by hardware (when the application break condition disappears).

From this point, the break circuitry is armed and active, and the MOE bit can be set to re-enable the PWM outputs.

Figure 551. Output redirection (tim_brk2 request not represented)



53.3.20 Clearing the tim_ocxref signal on an external event

The tim_ocxref signal of a given channel can be cleared when a high level is applied on the tim_ocref_clr_int input (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1). tim_ocxref remains low until the next transition to the active state, on the following PWM

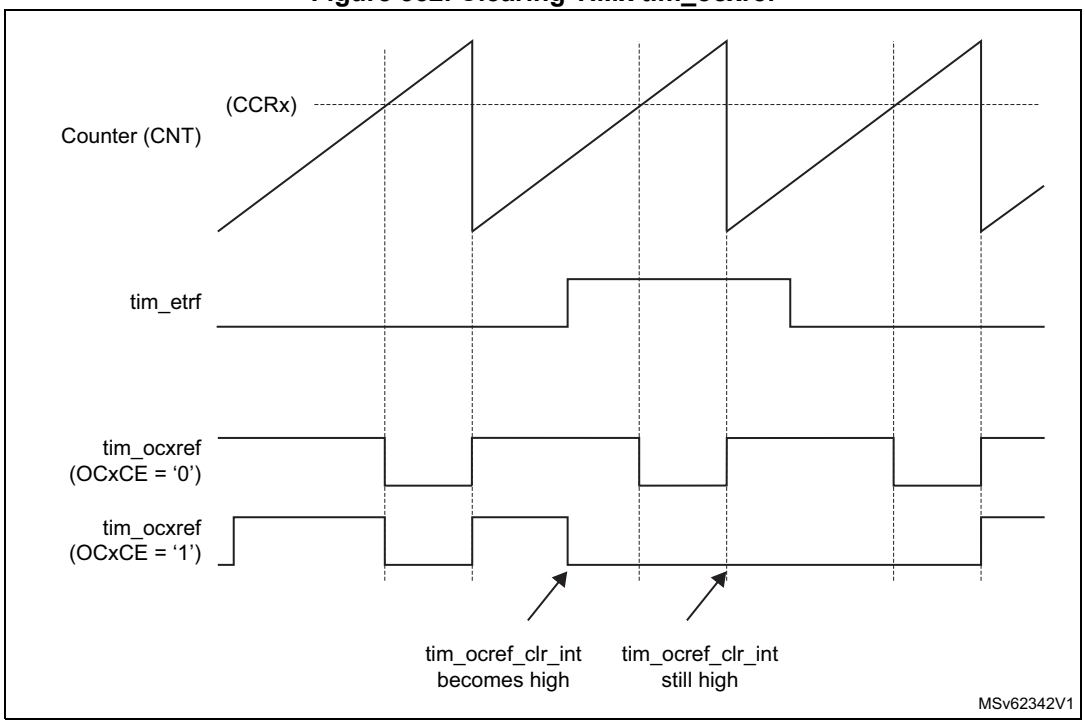
cycle. This function can only be used in Output compare and PWM modes. It does not work in Forced mode.

tim_etr_in must be configured as follows:

1. The external trigger prescaler must be kept off: bits ETPS[1:0] of the TIMx_SMCR register set to 00.
2. The external clock mode 2 must be disabled: bit ECE of the TIMx_SMCR register set to 0.
3. The external trigger polarity (ETP) and the external trigger filter (ETF) can be configured according to application needs (as per polarity of the source connected to the trigger and eventual need to remove noise using the filter).

Figure 552 shows the behavior of the tim_ocxref signal when the tim_etrf input becomes high, for both values of the enable bit OCxCE. In this example, the timer TIMx is programmed in PWM mode.

Figure 552. Clearing TIMx tim_ocxref



Note: In case of a PWM with a 100% duty cycle (if CCRx>ARR), then tim_ocxref is enabled again at the next counter overflow.

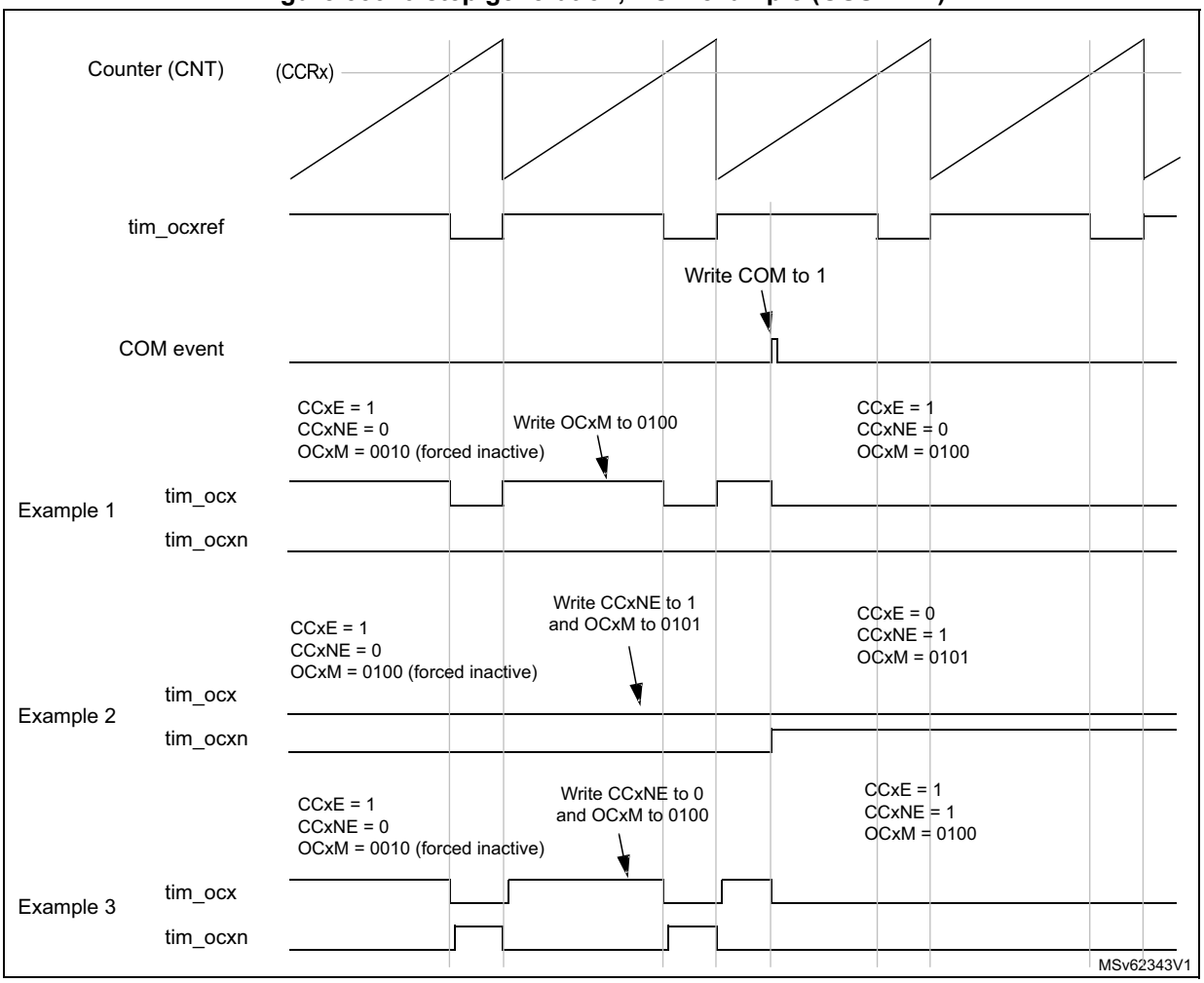
53.3.21 6-step PWM generation

When complementary outputs are used on a channel, preload bits are available on the OCxM, CCxE, and CCxNE bits. The preload bits are transferred to the shadow bits at the COM commutation event. Thus one can program in advance the configuration for the next step and change the configuration of all the channels at the same time. COM can be generated by software by setting the COM bit in the TIMx_EGR register or by hardware (on tim_trgi rising edge).

A flag is set when the COM event occurs (COMIF bit in the TIMx_SR register), which can generate an interrupt (if the COMIE bit is set in the TIMx_DIER register) or a DMA request (if the COMDE bit is set in the TIMx_DIER register).

Figure 553 describes the behavior of the tim_ocx and tim_ocxn outputs when a COM event occurs, in three different examples of programmed configurations.

Figure 553. 6-step generation, COM example (OSSR = 1)



53.3.22 One-pulse mode

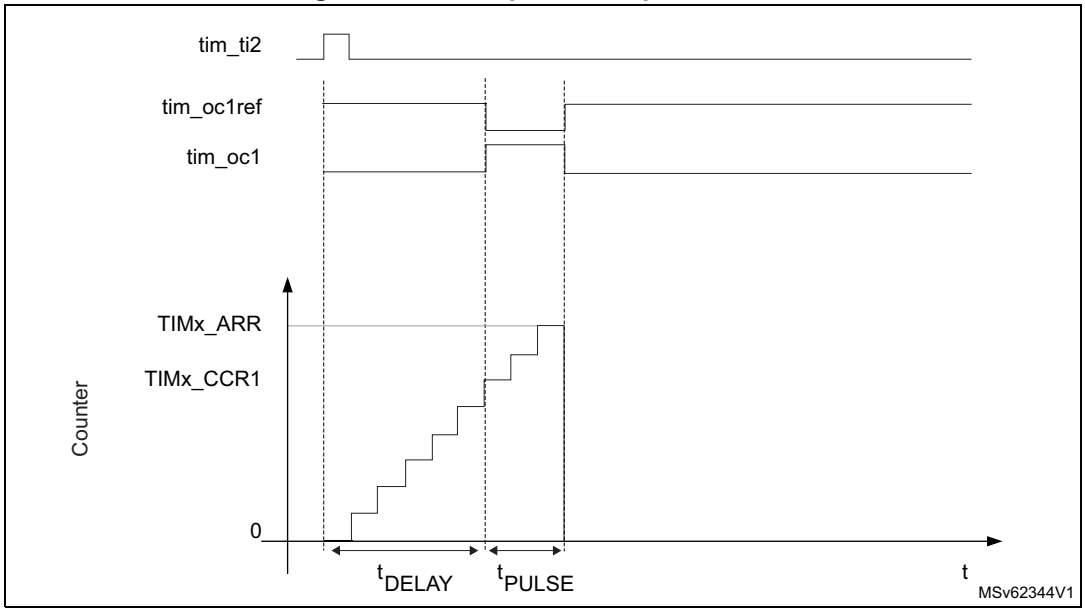
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- In upcounting: CNT < CCRx ≤ ARR (in particular, 0 < CCRx)
- In downcounting: CNT > CCRx

Figure 554. Example of one pulse mode.



In the following example, the user wants to generate a positive pulse on tim_oc1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the tim_ti2 input pin.

Use tim_ti2fp2 as trigger 1:

- Map tim_ti2fp2 to tim_ti2 by writing CC2S = 01 in the TIMx_CCMR1 register.
- tim_ti2fp2 must detect a rising edge, write CC2P = 0 and CC2NP = 0 in the TIMx_CCER register.
- Configure tim_ti2fp2 as trigger for the slave mode controller (tim_trgi) by writing TS = 00110 in the TIMx_SMCR register.
- tim_ti2fp2 is used to start the counter by writing SMS to 110 in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the autoreload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Suppose the user wants to build a waveform with a transition from 0 to 1 when a compare match occurs and a transition from 1 to 0 when the counter reaches the auto-reload value. This is achieved by enabling PWM mode 2 (OC1M = 111 in TIMx_CCMR1). Optionally the preload registers can be enabled by writing OC1PE = 1 in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the autoreload value in the

TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on tim_ti2. CC1P is written to 0 in this example.

In this example, the DIR and CMS bits in the TIMx_CR1 register must be low.

Since only one pulse (Single mode) is needed, a 1 must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the autoreload value back to 0). When OPM bit in the TIMx_CR1 register is set to 0, so the Repetitive mode is selected.

Particular case: tim_ocx fast enable:

In One-pulse mode, the edge detection on tim_tix input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min that can be achieved.

To output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then tim_ocxref (and tim_ocx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

53.3.23 Retriggerable One-pulse mode

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with nonretriggerable one-pulse mode described in *Section 53.3.22*:

- The pulse starts as soon as the trigger occurs (no programmable delay).
- The pulse is extended if a new trigger occurs before the previous one is completed.

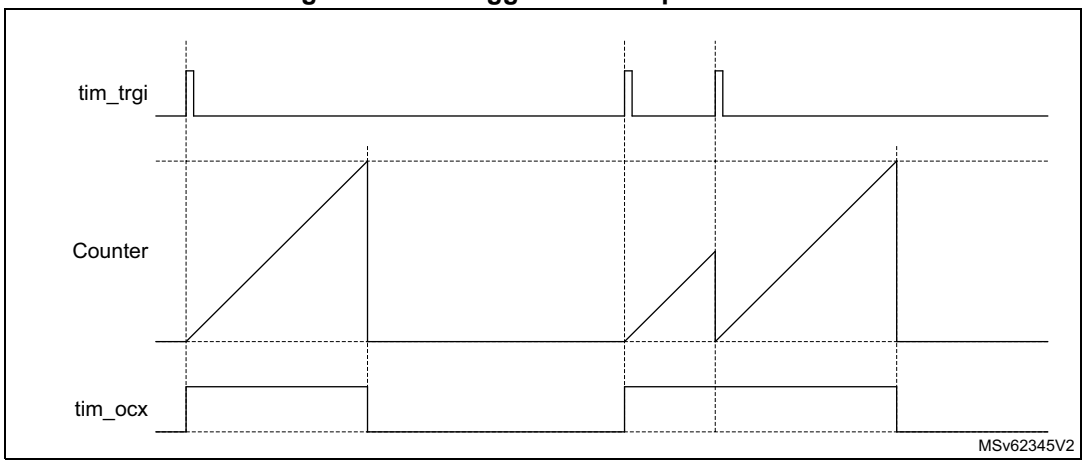
The timer must be in Slave mode, with the bits SMS[3:0] = 1000 (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to 1000 or 1001 for retriggerable OPM mode 1 or 2.

If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode, CCRx must be above or equal to ARR.

Note: The OCxM[3:0] and SMS[3:0] bitfields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the three least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.

Figure 555. Retriggerable one-pulse mode



53.3.24 Pulse on compare mode

A pulse can be generated upon compare match event. A signal with a programmable pulsewidth generated when the counter value equals a given compare value, for debugging or synchronization purposes.

This mode is available for any slave mode selection, including encoder modes, in edge and center aligned counting modes. It is solely available for channel 3 and channel 4. The pulse generator is unique and is shared by the two channels, as shown on *Figure 556*.

Figure 556. Pulse generator circuitry

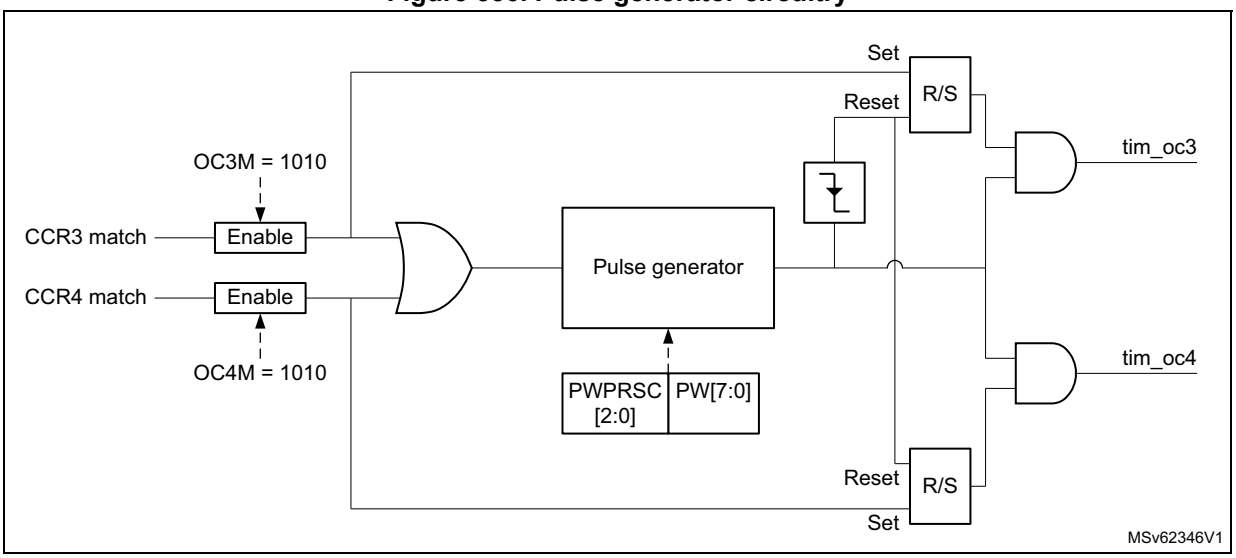
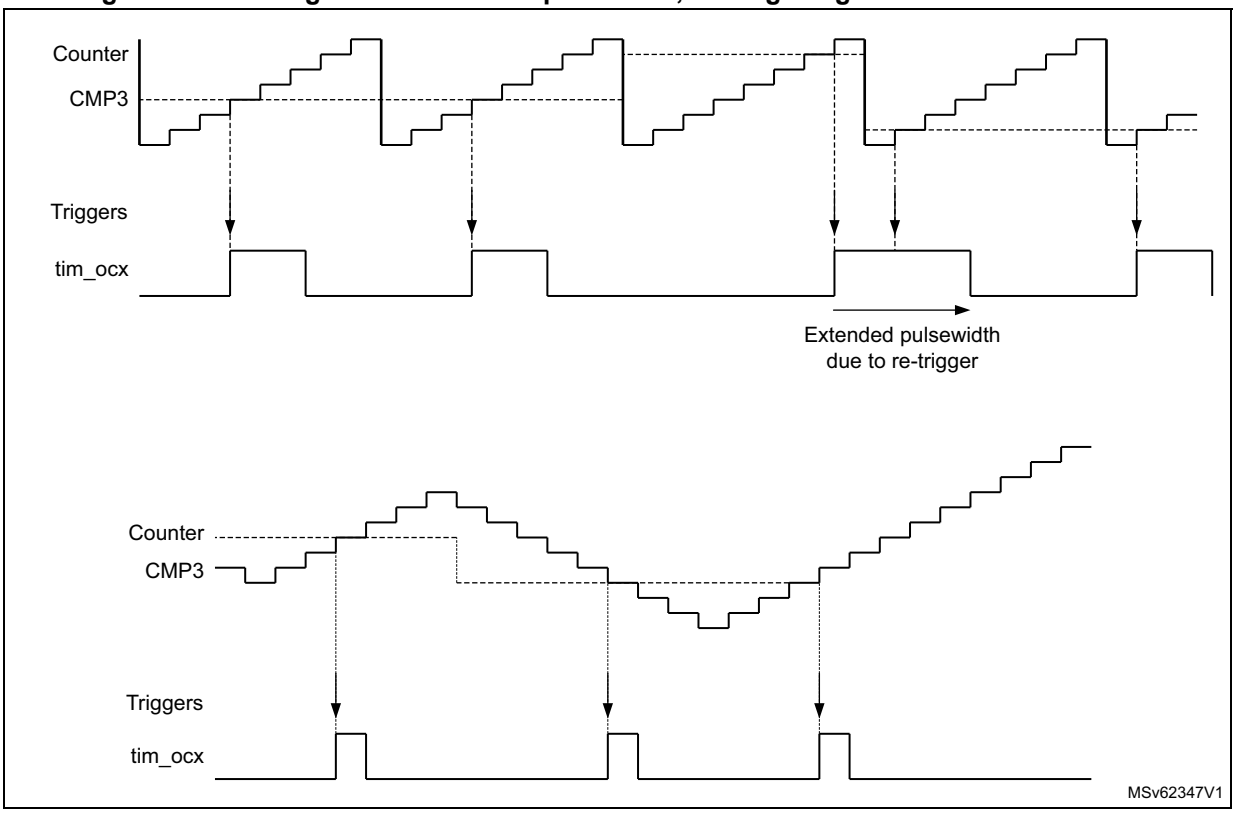


Figure 557 shows how the pulse is generated for edge-aligned and encoder operating modes.

Figure 557. Pulse generation on compare event, for edge-aligned and encoder modes



This output compare mode is selected using the OC3M[3:0] and OC4M[3:0] bitfields in TIMx_CCMR2 register.

The pulsewidth is programmed using the PW[7:0] bitfield in the register, using a specific clock prescaled according to PWPRSC[2:0] bits, as follows:

t_{PW} = PW[7:0] x t_{PWG}

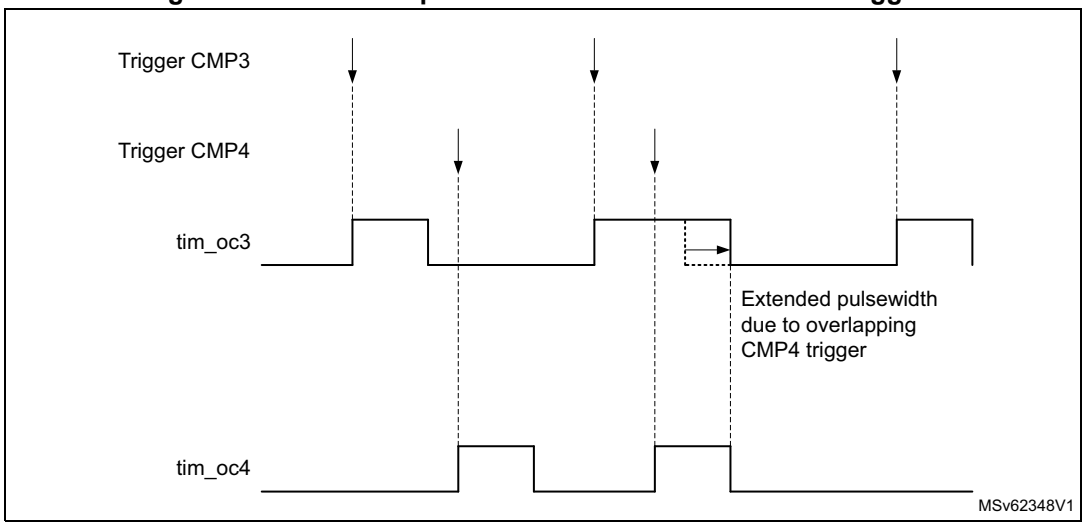
where $t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim_ker_ck}$

gives the resolution and maximum values depending on the prescaler value.

The pulse is retriggerable: a new trigger while the pulse is ongoing, causes the pulse to be extended.

Note: If the two channels are enabled simultaneously, the pulses are issued independently as long as the trigger on one channel is not overlapping the pulse generated on the concurrent output. On the opposite, if the two triggers are overlapping, the pulse width related to the first arriving trigger is extended (because of the retrigger), while the pulse width of the last arriving trigger is correct (as shown on Figure 558).

Figure 558. Extended pulsewidth in case of concurrent triggers



53.3.25 Encoder interface mode

Quadrature encoder

To select Encoder Interface mode write SMS = 0001 in the TIMx_SMCR register if the counter is counting on tim_ti1 edges only, SMS = 0010 if it is counting on tim_ti2 edges only and SMS = 0011 if it is counting on both tim_ti1 and tim_ti2 edges.

Select the tim_ti1 and tim_ti2 polarity by programming the CC1P and CC2P bits in the TIMx_CCER register. When needed, the input filter can be programmed as well. CC1NP and CC2NP must be kept low.

The two inputs tim_ti1 and tim_ti2 are used to interface to an quadrature encoder. Refer to *Table 495*. The counter is clocked by each valid transition on tim_ti1fp1 or tim_ti2fp2 (tim_ti1 and tim_ti2 after input filter and polarity selection, tim_ti1fp1 = tim_ti1 if not filtered and not inverted, tim_ti2fp2 = tim_ti2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx_CR1 register written to 1). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (tim_ti1 or tim_ti2), whatever the counter is counting on tim_ti1 only, tim_ti2 only or both tim_ti1 and tim_ti2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the autoreload value in the TIMx_ARR register (0 to ARR or ARR down to 0 depending on the direction). So the TIMx_ARR must be configured before starting. In the same way, the capture, compare, prescaler, repetition counter, trigger output features continue to work as normal. Encoder mode and External clock mode 2 are not compatible and must not be selected together.

In this mode, the counter is modified automatically following the speed and the direction of the quadrature encoder and its content, therefore, always represents the encoder's position. The count direction correspond to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming tim_ti1 and tim_ti2 do not switch at the same time.

Table 495. Counting direction versus encoder signals (CC1P = CC2P = 0)

Active edge	SMS[3:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
Counting on tim_ti1 only x1 mode	1110	High	Down	Up	No count	No count
		Low	No count	No count	No count	No count
Counting on tim_ti2 only x1 mode	1111	High	No count	No count	Up	Down
		Low	No count	No count	No count	No count
Counting on tim_ti1 only x2 mode	0001	High	Down	Up	No count	No count
		Low	Up	Down	No count	No count
Counting on tim_ti2 only x2 mode	0010	High	No count	No count	Up	Down
		Low	No count	No count	Down	Up
Counting on tim_ti1 and tim_ti2 x4 mode	0011	High	Down	Up	Up	Down
		Low	Up	Down	Down	Up

A quadrature encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicate the mechanical zero position, may be connected to the external trigger input and trigger a counter reset.

Figure 559 gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are selected. This might occur if the sensor is positioned near to one of the switching points. For this example the configuration is the following:

- CC1S = 01 (TIMx_CCMR1 register, tim_ti1fp1 mapped on tim_ti1).
- CC2S = 01 (TIMx_CCMR1 register, tim_ti2fp2 mapped on tim_ti2).
- CC1P = 0 and CC1NP = 0 (TIMx_CCER register, tim_ti1fp1 noninverted, tim_ti1fp1 = tim_ti1).
- CC2P = 0 and CC2NP = 0 (TIMx_CCER register, tim_ti1fp2 noninverted, tim_ti1fp2= tim_ti2).
- SMS = 0011 (TIMx_SMCR register, both inputs are active on both rising and falling edges).
- CEN = 1 (TIMx_CR1 register, Counter enabled).

Figure 559. Example of counter operation in encoder interface mode.

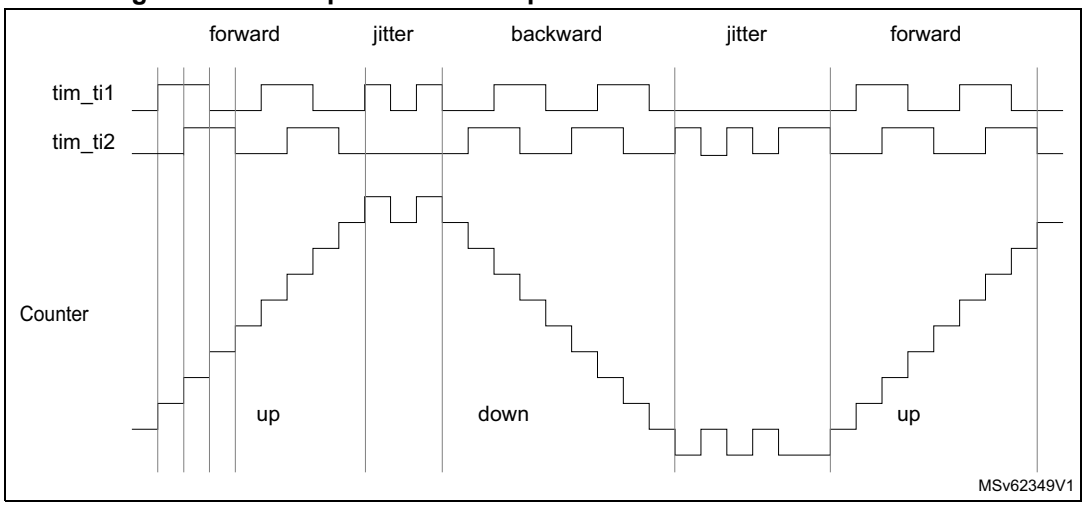


Figure 560 gives an example of counter behavior when tim_ti1fp1 polarity is inverted (same configuration as above except CC1P = 1).

Figure 560. Example of encoder interface mode with tim_ti1fp1 polarity inverted.

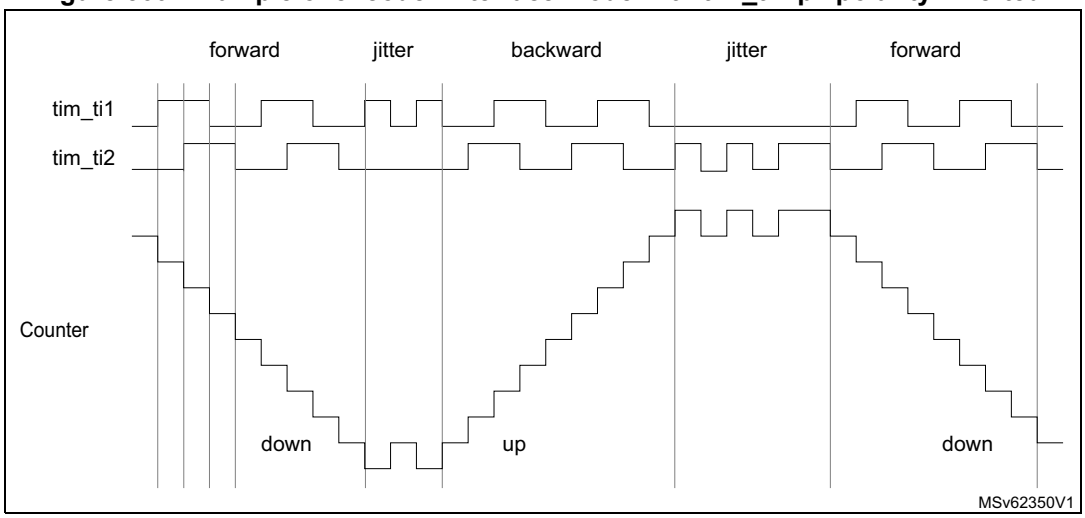
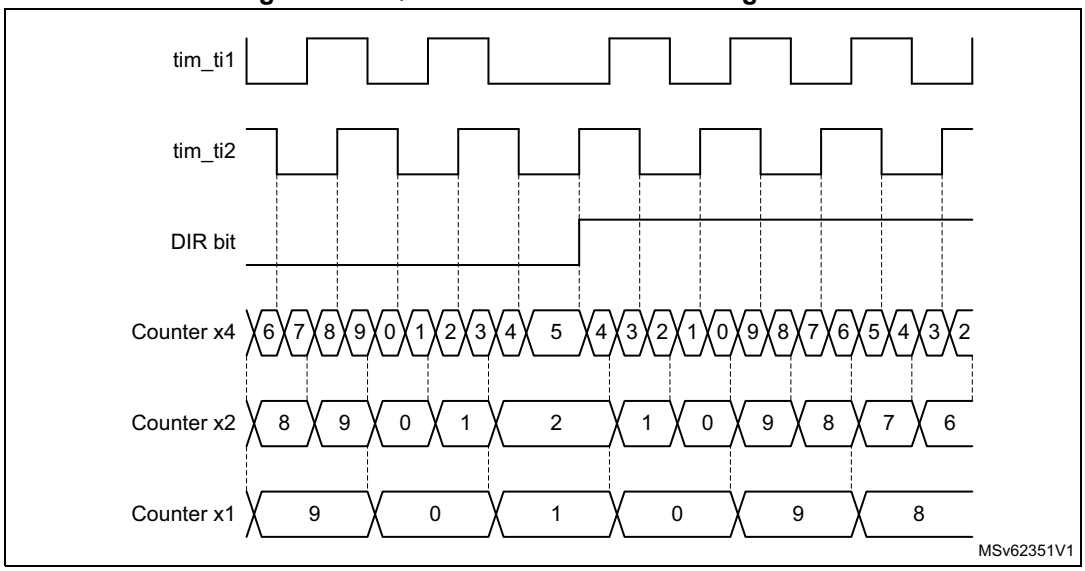


Figure 561 shows the timer counter value during a speed reversal, for various counting modes.

Figure 561. Quadrature encoder counting modes



The timer, when configured in Encoder Interface mode provides information on the sensor's current position. Dynamic information can be obtained (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). When available, it is also possible to read its value through a DMA request.

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag (UIF) into the timer counter register's bit 31 (TIMxCNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

Clock plus direction encoder mode

In addition to the quadrature encoder mode, the timer offers support for other types of encoders.

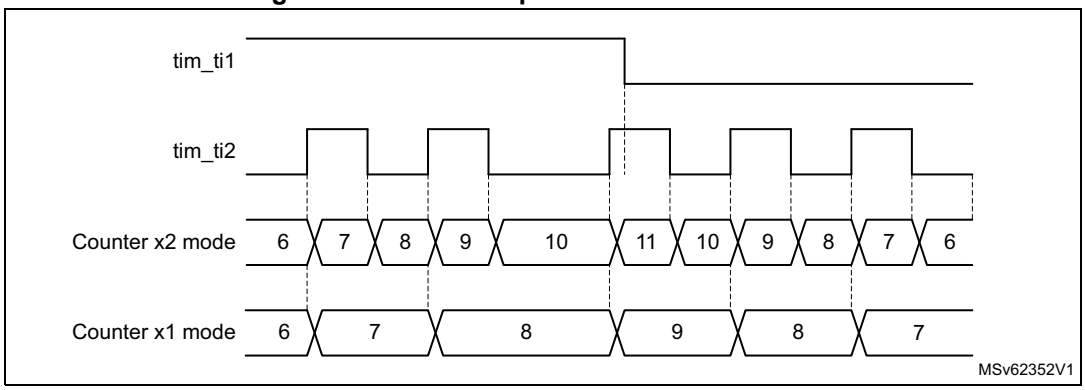
In the clock plus direction mode shown on *Figure 562*, the clock is provided on a single line, on tim_ti2, while the direction is forced using the tim_ti1 input.

This mode is enabled with the SMS[3:0] bitfield in the TIMx_SMCR register, as following:

- 1010: x2 mode, the counter is updated on both rising and falling edges of the clock
- 1011: x1 mode, the counter is updated on a single clock edge, as per CC2P bit value: CC2P = 0 corresponds to rising edge sensitivity and CC2P = 1 corresponds to falling edge sensitivity

The polarity of the direction signal on tim_ti1 is set with the CC1P bit: 0 corresponds to positive polarity (up-counting when tim_ti1 is high and down-counting when tim_ti1 is low) and CC1P = 1 corresponds to negative polarity (up-counting when tim_ti1 is low).

Figure 562. Direction plus clock encoder mode



Directional clock encoder mode

In the directional clock mode on *Figure 563*, the clocks are provided on two lines, with a single one at once, depending on the direction, so as to have one up-counting clock line and one down-counting clock line.

This mode is enabled with the SMS[3:0] bitfield in the TIMx_SMCR register, as following:

- 1100: x2 mode, the counter is updated on both rising and falling edges of any of the two clock line. The CC1P and CC2P bits are coding for the clock idle state. CCxP = 0 corresponds to high-level idle state (refer to *Figure 563*) and CCxP = 1 corresponds to low-level idle state (refer to *Figure 564*).
- 1101: x1 mode, the counter is updated on a single clock edge, as per CC1P and CC2P bit value. CCxP = 0 corresponds to falling edge sensitivity and high-level idle state (refer to *Figure 563*), CCxP = 1 corresponds to rising edge sensitivity and low-level idle state (refer to *Figure 564*).

Figure 563. Directional clock encoder mode (CC1P = CC2P = 0)

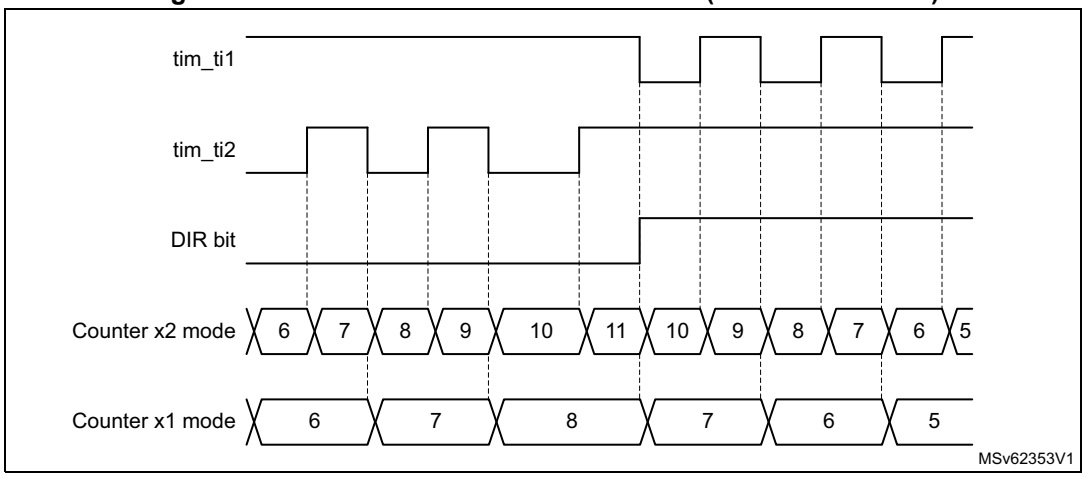


Figure 564. Directional clock encoder mode (CC1P = CC2P = 1)

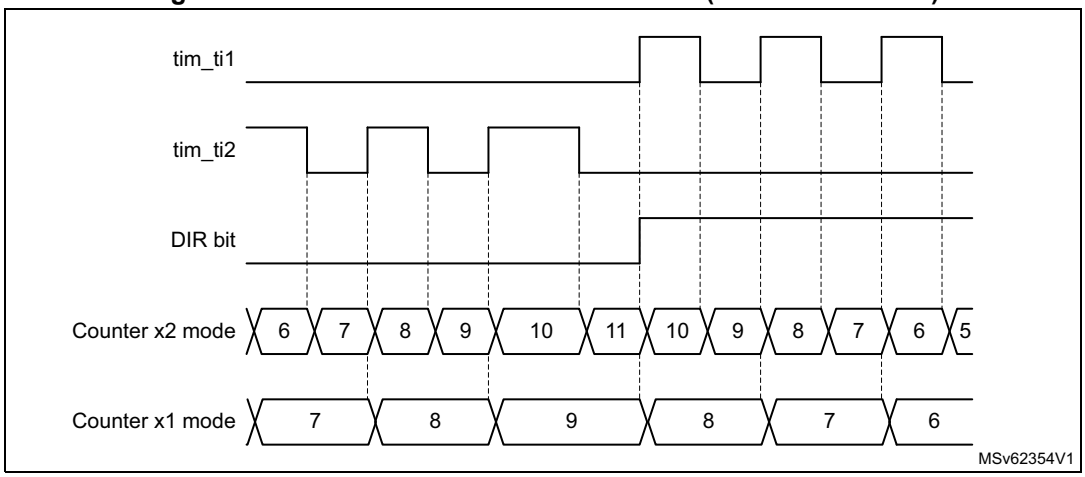


Table 496 here-below details how the directional clock mode operates, for any input transition.

Table 496. Counting direction versus encoder signals and polarity settings

Directional clock mode	SMS[3:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
x2 mode CCxP = 0	1100	High	Down	Down	Up	Up
		Low	No count	No count	No count	No count
x2 mode CCxP = 1	1100	High	No count	No count	No count	No count
		Low	Down	Down	Up	Up
x1 mode CCxP = 0	1101	High	No count	Down	No count	Up
		Low	No count	No count	No count	No count
x1 mode CCxP = 1	1101	High	No count	No count	No count	No count
		Low	Down	No count	Up	No count

Index input

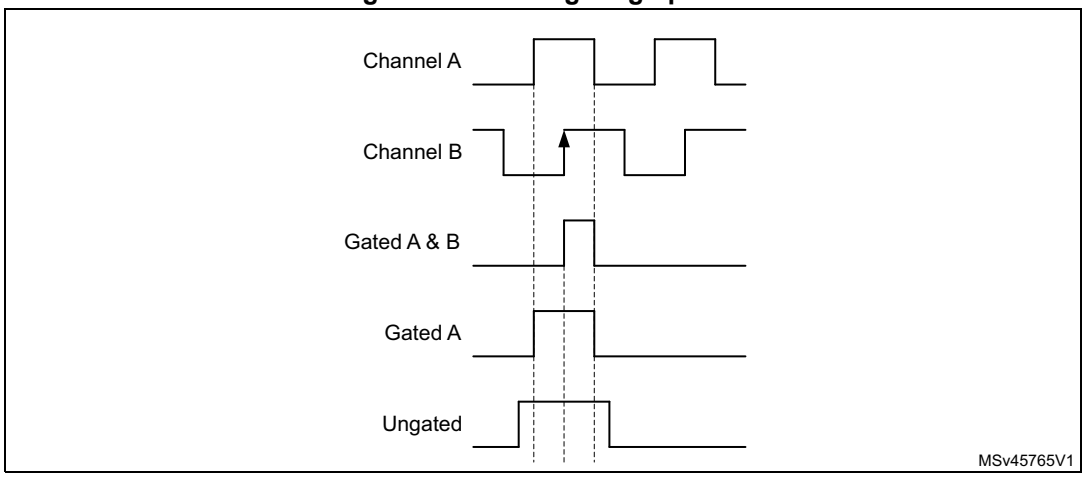
The counter can be reset by an index signal coming from the encoder, indicating an absolute reference position. The index signal must be connected to the tim_etr_in input. It can be filtered using the digital input filter.

The index functionality is enabled with the IE bit in the TIMX_ECR register. The IE bit must be set only in encoder mode, when the SMS[3:0] bitfield has the following values: 0001, 0010, 011, 1010, 1011, 1100, 1101, 1110, 1111.

Available encoders are proposed with several options for index pulse conditioning, as per *Figure 565*:

- gated with A and B: the pulsewidth is 1/4 of one channel period, aligned with both A and B edges
- gated with A (or gated with B): the pulsewidth is 1/2 of one channel period, aligned with the two edges on channel A (resp. channel B)
- ungated: the pulsewidth is up to one channel period, without any alignment to the edges

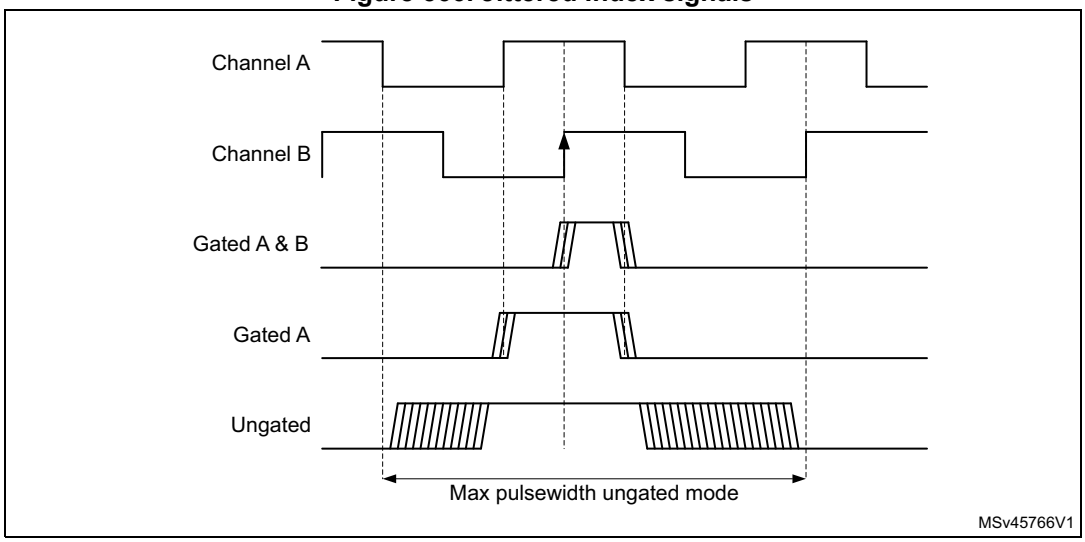
Figure 565. Index gating options



The circuitry tolerates jitter on index signal, whatever the gating mode, as show on *Figure 566*.

In ungated mode, the signal must be strictly below two encoder periods. If the pulsewidth is greater or equal to two encoder period, the counter is reset multiple times.

Figure 566. Jittered Index signals



The timer supports the three gating options identically, without any specific programming needed. It is only necessary to define on which encoder state (for example channel A and

channel B state combination) the index must be synchronized, using the IPOS[1:0] bitfield in the TIMx_ECR register.

The index detection event acts differently depending on counting direction to ensure symmetrical operation during speed reversal:

- The counter is reset during up-counting (DIR bit = 0).
- The counter is set to TIMx_ARR when down counting.

This allows the index to be generated on the very same mechanical angular position whatever the counting direction. *Figure 567* shows at which position is the index generated, for a simplistic example (an encoder providing four edges par mechanical rotation).

Figure 567. Index generation for IPOS[1:0] = 11

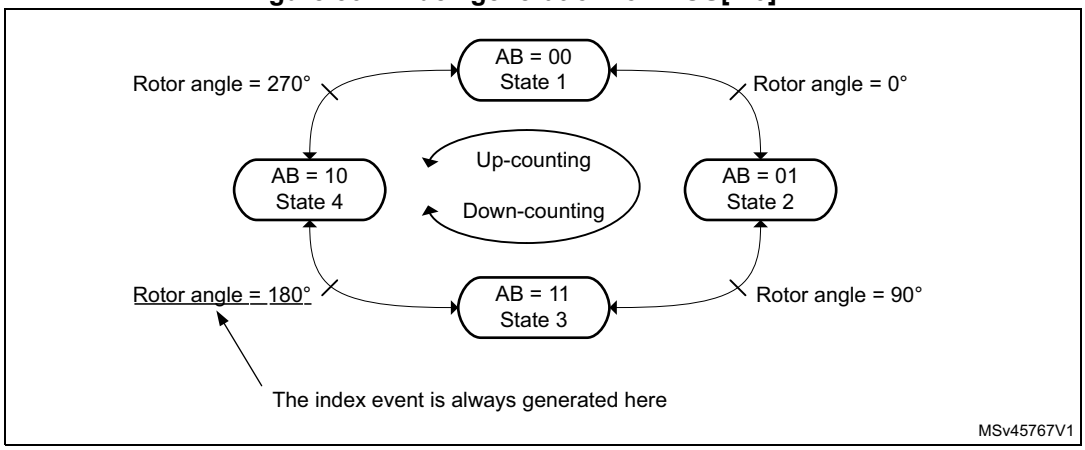


Figure 568 presents waveforms and corresponding values for IPOS[1:0] = 11. It shows that the instant at which the counter value is forced is automatically adjusted depending on the counting direction:

- Counter set to 0 when encoder state is 11 (ChA = 1, ChB = 1), when up-counting (DIR bit = 0).
- Counter set to TIMx_ARR when exiting the 11 state, when down-counting (DIR bit = 1).

An interrupt can be issued upon index detection event.

The arrows are indicating on which transition is the index event interrupt generated.

Figure 568. Counter reading with index gated on channel A (IPOS[1:0] = 11)

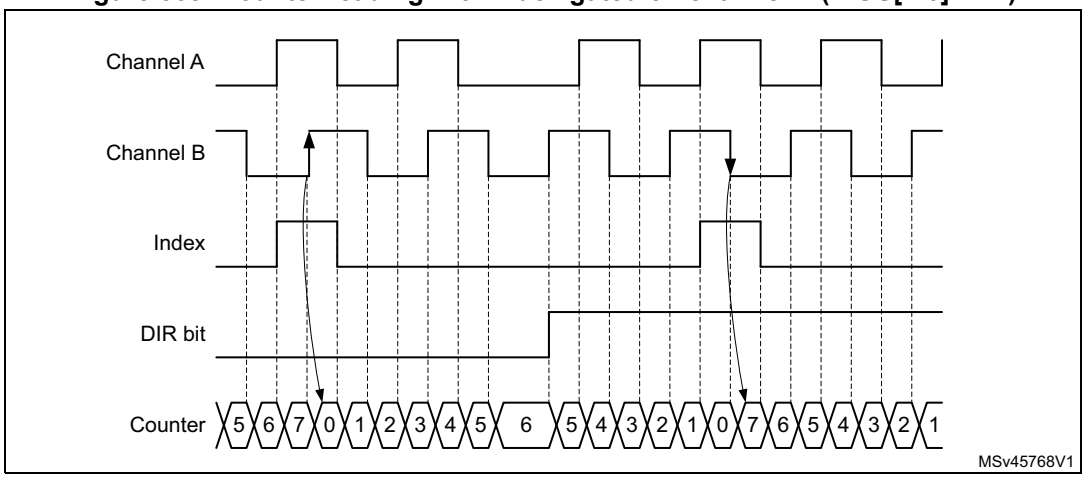


Figure 569. presents waveforms and corresponding values for the ungated mode. The arrows are indicating on which transition is the index event generated.

Figure 569. Counter reading with index ungated (IPOS[1:0] = 00)

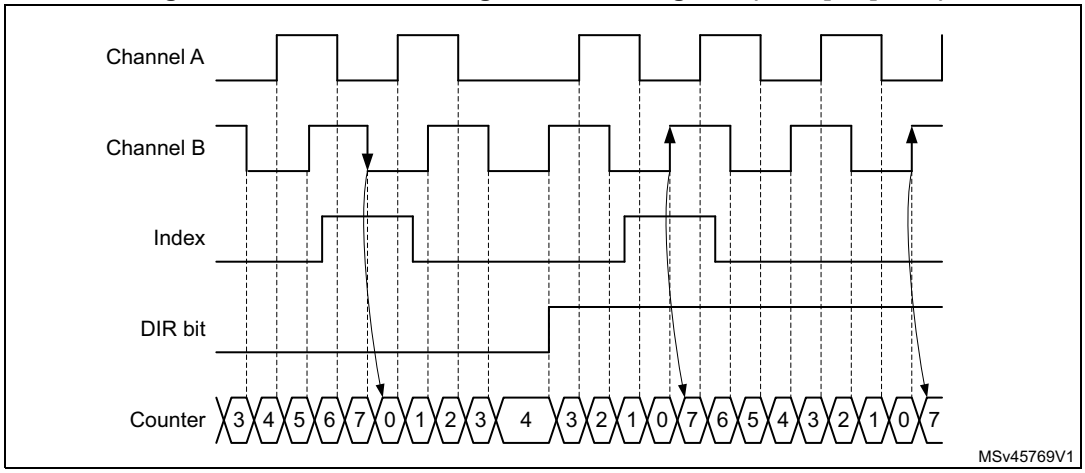


Figure 570. shows how the 'gated on A & B' mode is handled, for various pulse alignment scenario. The arrows are indicating on which transition is the index event generated.

Figure 570. Counter reading with index gated on channel A and B

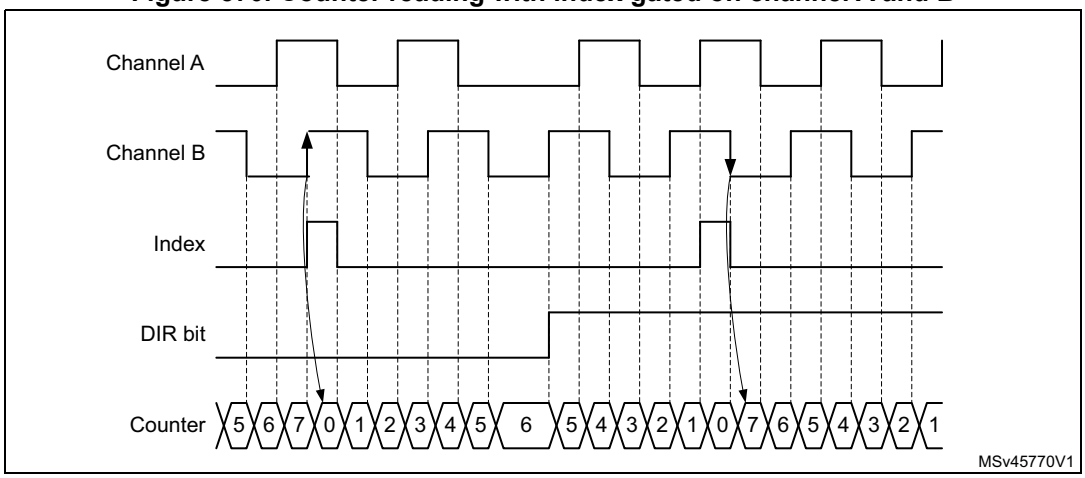


Figure 571 and *Figure 572* detail the case where the subsequent index pulse may be narrower than one quarter of the encoder clock period.

Figure 571. Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11)

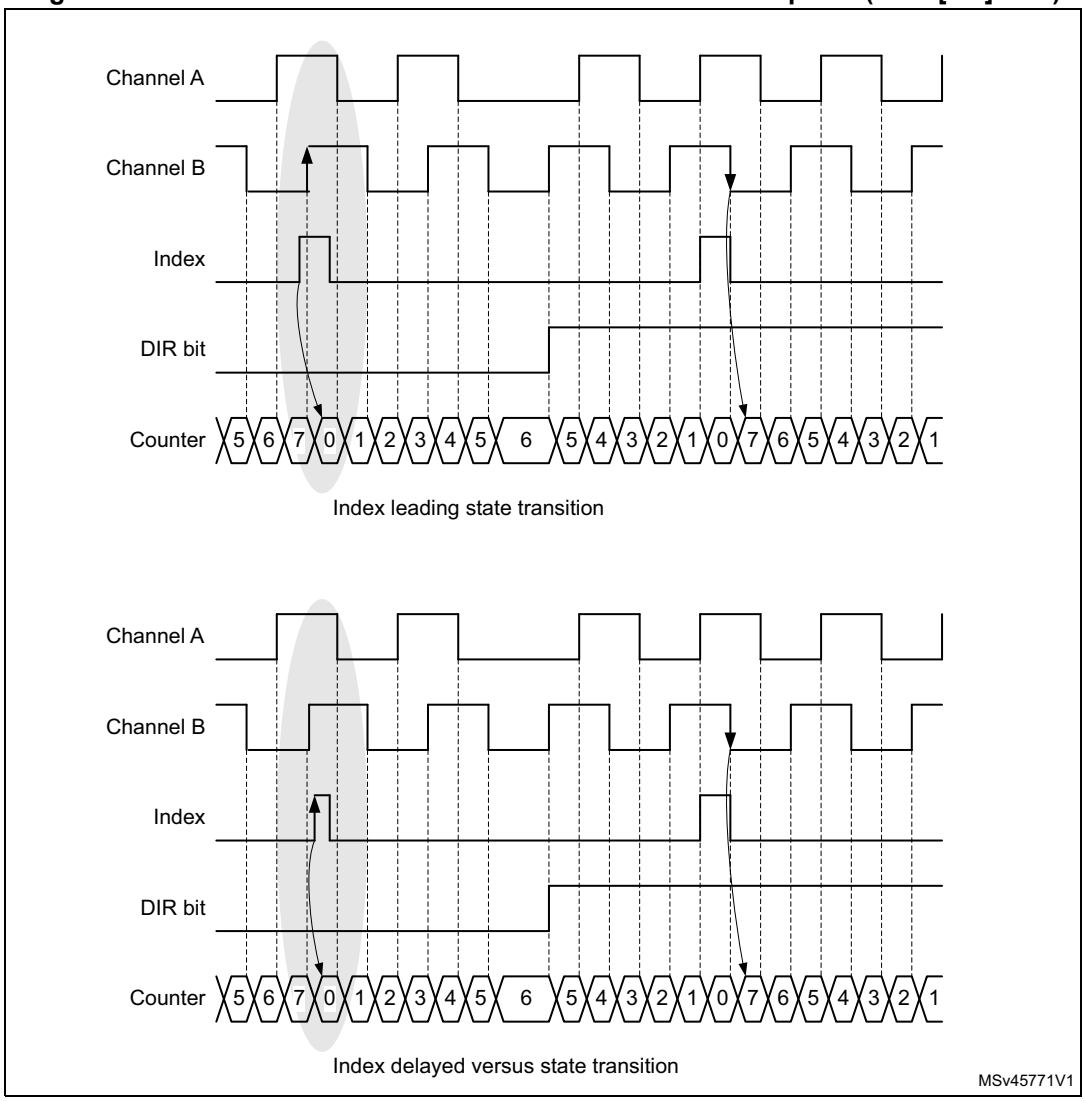


Figure 572. Counter reset Narrow index pulse (closer view, ARR = 0x07)

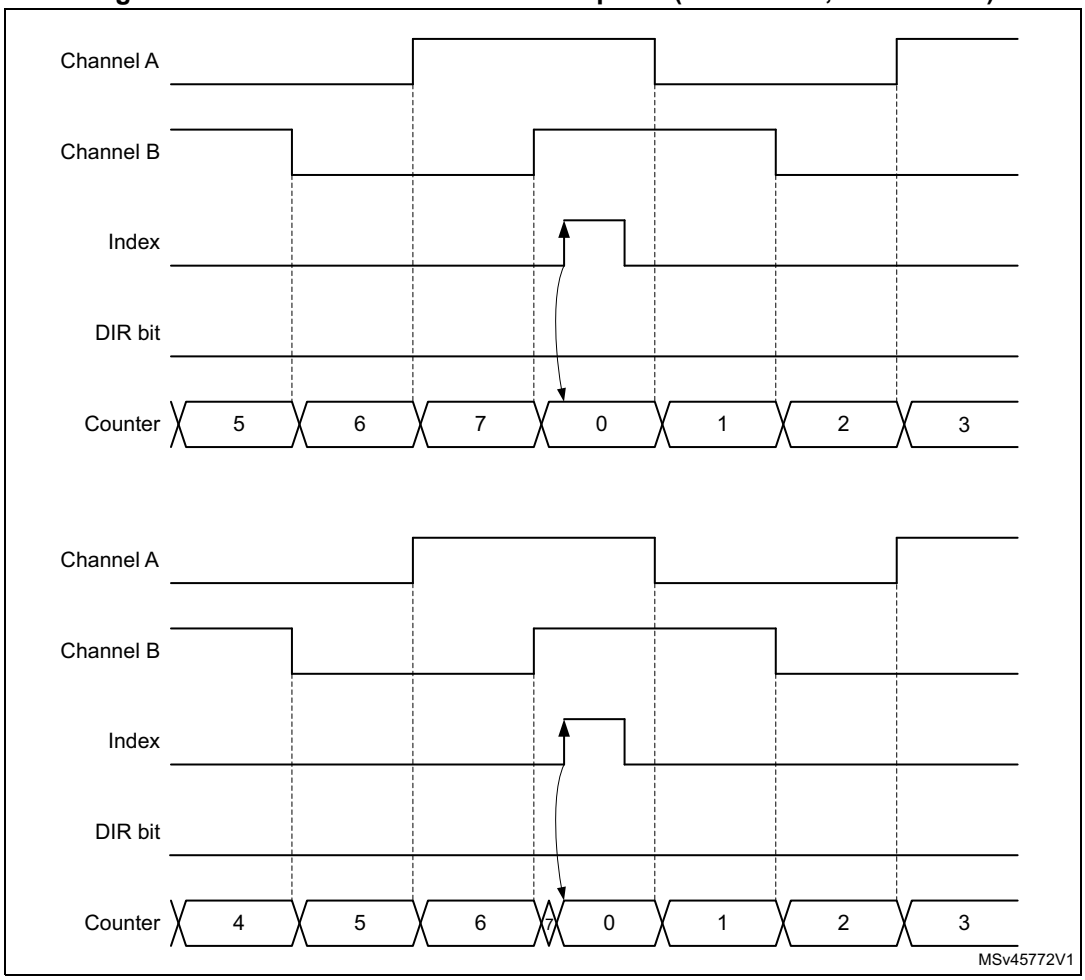
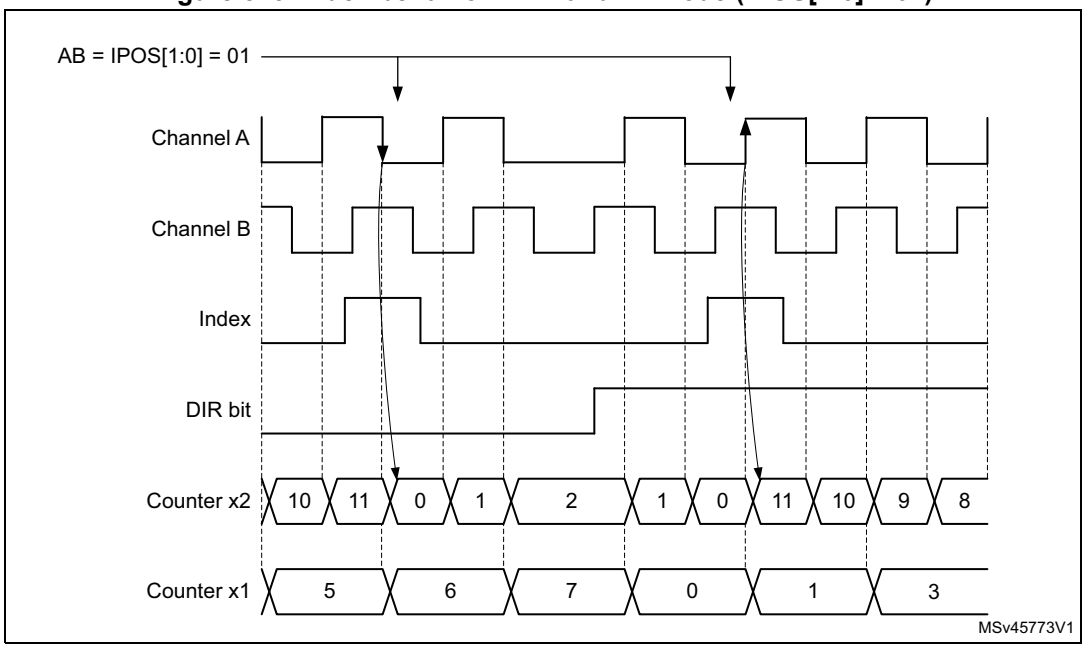


Figure 573 shows how the index is managed in x1 and x2 modes.

Figure 573. Index behavior in x1 and x2 mode (IPOS[1:0] = 01)

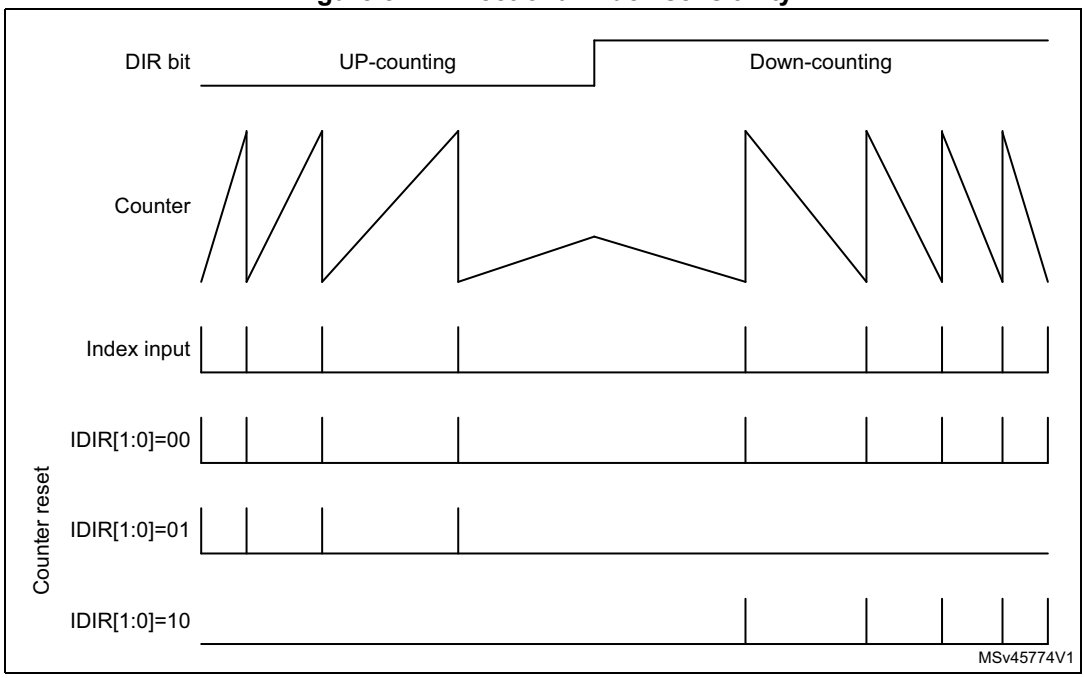


Directional index sensitivity

The IDIR[1:0] bitfield in the TIMx_ECR register allows the index to be active only in a selected counting direction.

Figure 574 shows the relationship between index and counter reset events, depending on IDIR[1:0] value.

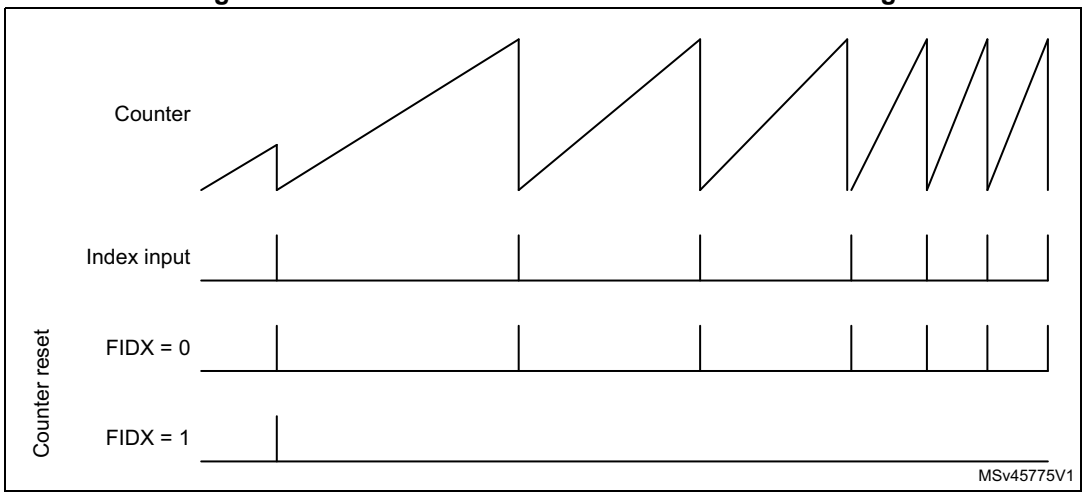
Figure 574. Directional index sensitivity



Special first index event management

The FIDX bit in the TIMx_ECR register allows the index to be taken only once, as shown on *Figure 575*. Once the first index has arrived, any subsequent index is ignored. If needed, the circuitry can be rearmed by writing the FIDX bit to 0 and setting it again to 1.

Figure 575. Counter reset as function of FIDX bit setting



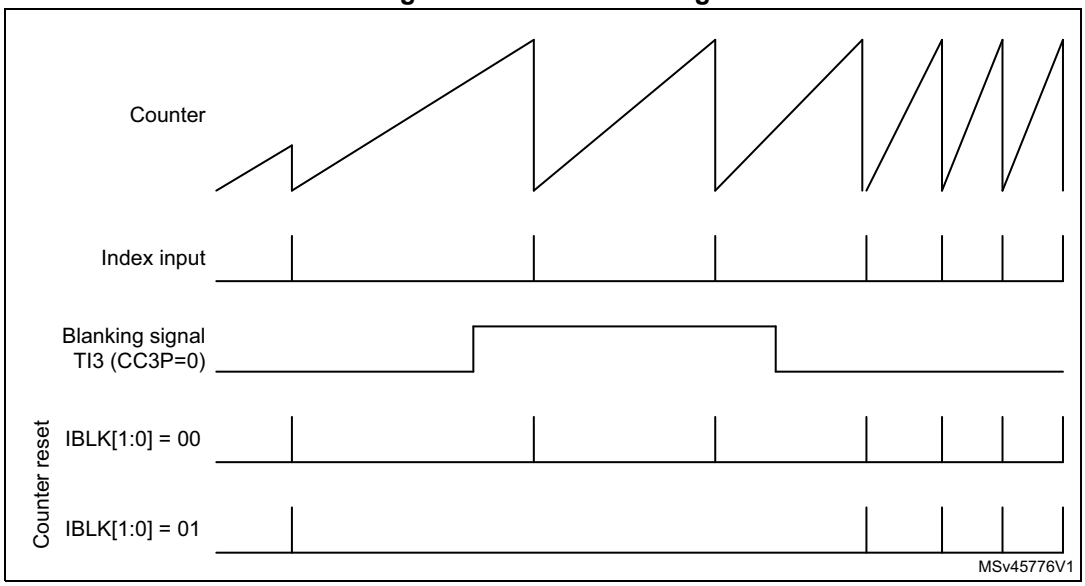
Index blanking

The index event can be blanked using the tim_ti3 or tim_ti4 inputs. During the blanking window, the index events are no longer resetting the counter, as shown on the *Figure 576* below.

This mode is enabled using the IBLK[1:0] bitfield in the TIMx_ECR register, as following:

- IBLK[1:0] = 00: Index signal always active
- IBLK[1:0] = 01: Index signal blanking on tim_ti3 input
- IBLK[1:0] = 10: Index signal blanking on tim_ti4 input

Figure 576. Index blanking



Index management in nonquadrature mode

Figure 577 and *Figure 578* detail how the index is managed in directional clock mode and clock plus direction mode, when the SMS[3:0] bitfield is equal to 1010, 1011, 1100, 1101.

For both of these modes, the index sensitivity is set with the IPOS[0] bit as following:

- IPOS[0] = 0: Index is detected on clock low level
- IPOS[0] = 1: Index is detected on clock high level

The IPOS[1] bit is not-significant.

Figure 577. Index behavior in clock + direction mode, IPOS[0] = 1

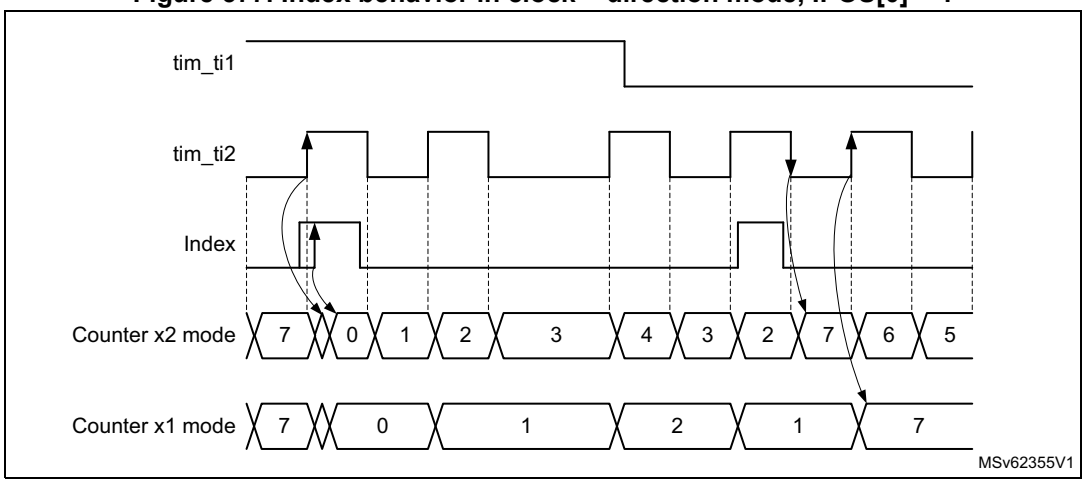
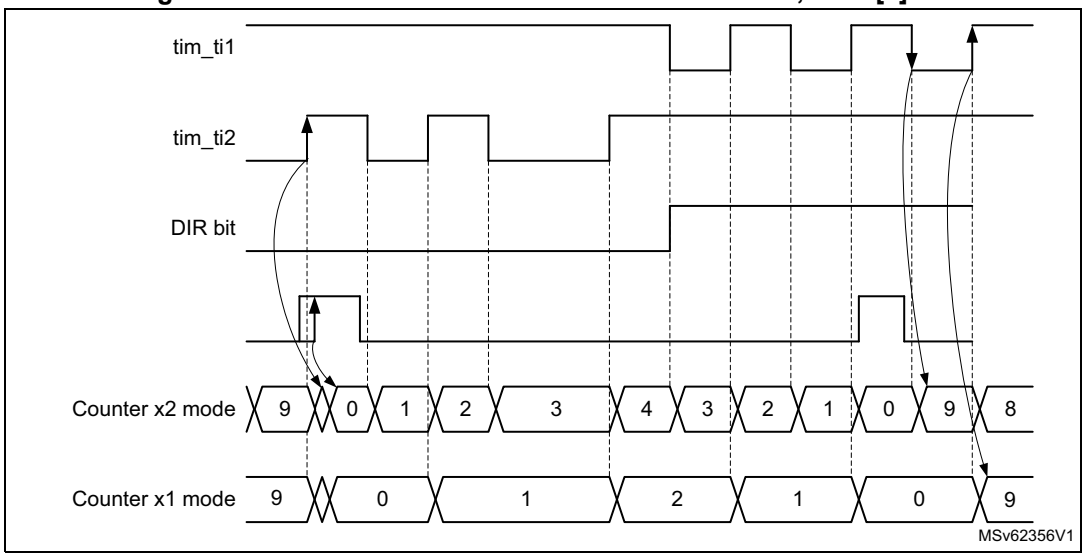


Figure 578. Index behavior in directional clock mode, IPOS[0] = 1

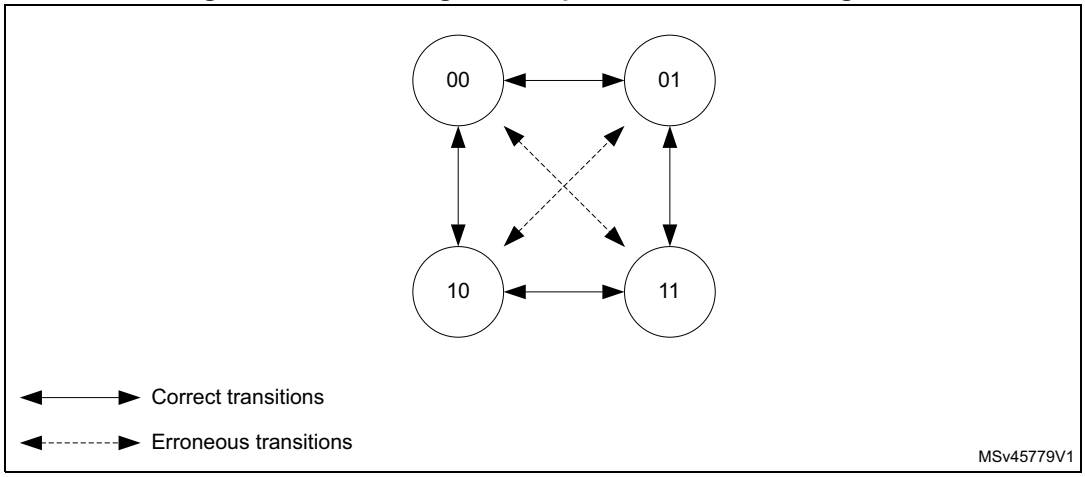


Encoder error management

For encoder configurations where two quadrature signals are available, it is possible to detect transition errors. The reading on the two inputs corresponds to a 2-bit gray code which can be represented as a state diagram, on *Figure 579.*. A single bit is expected to change at once. An erroneous transition sets the TERRF interrupt flag in the TIMx_SR

status register. A transition error interrupt is generated if the TERRIE bit is set in the TIMx_DIER register.

Figure 579. State diagram for quadrature encoded signals



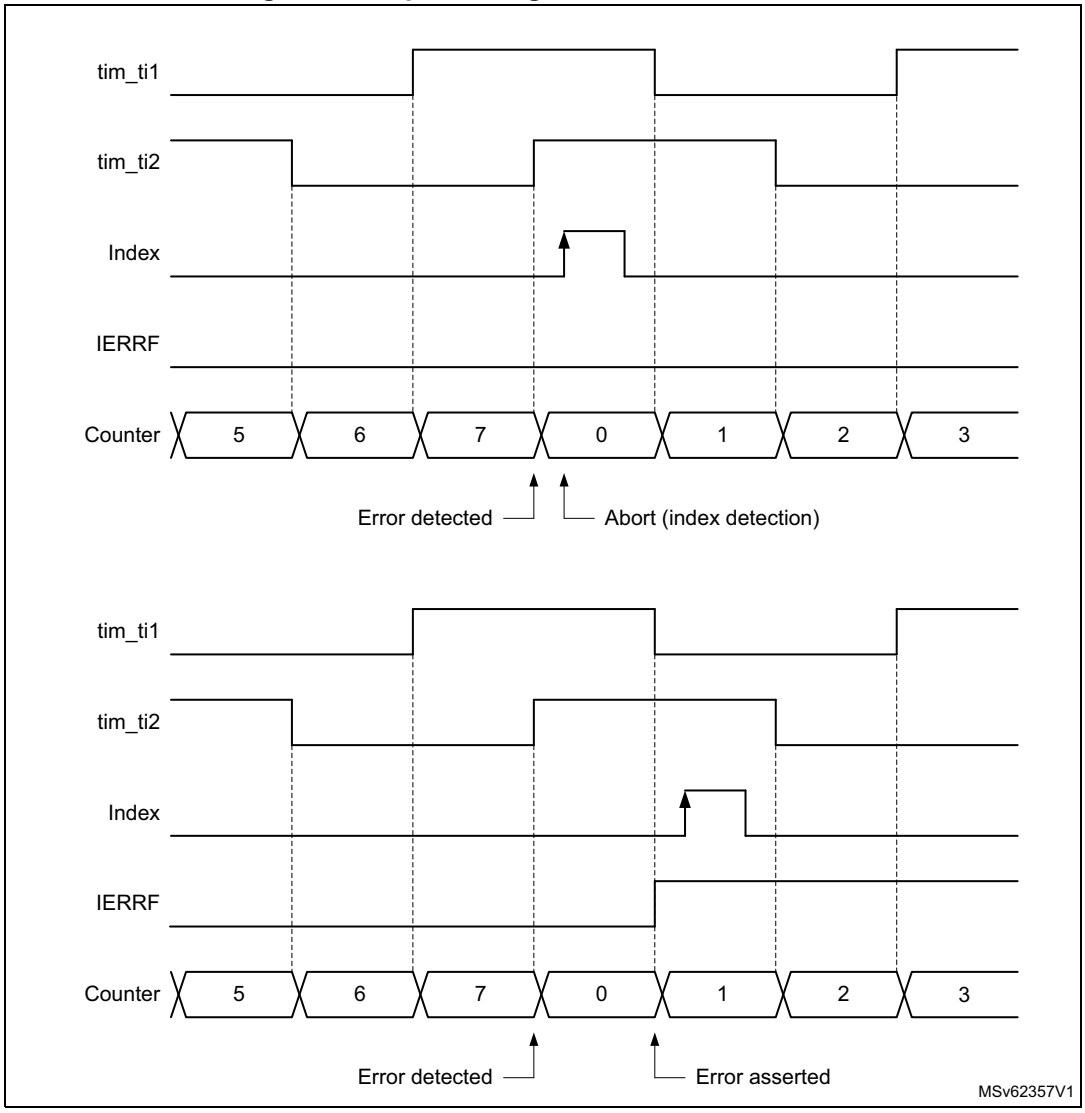
For encoder having an index signal, it is possible to detect abnormal operation resulting in an excess of pulses per revolution. An encoder with N pulses per revolution provides 4xN counts per revolution. The index signal resets the counter every 4xN clock periods.

If the counter value is incremented from TIMx_ARR to 0 or decremented from 0 to TIMxARR value without any index event, this is reported as an index position error.

The overflow threshold is programmed using the TIMx_ARR register. A 1000 lines encoder results in a counter value being between 0 and 3999 (in 4x reading mode). The overflow detection threshold must be programmed by setting TIMx_ARR = 3999 + 1 = 4000.

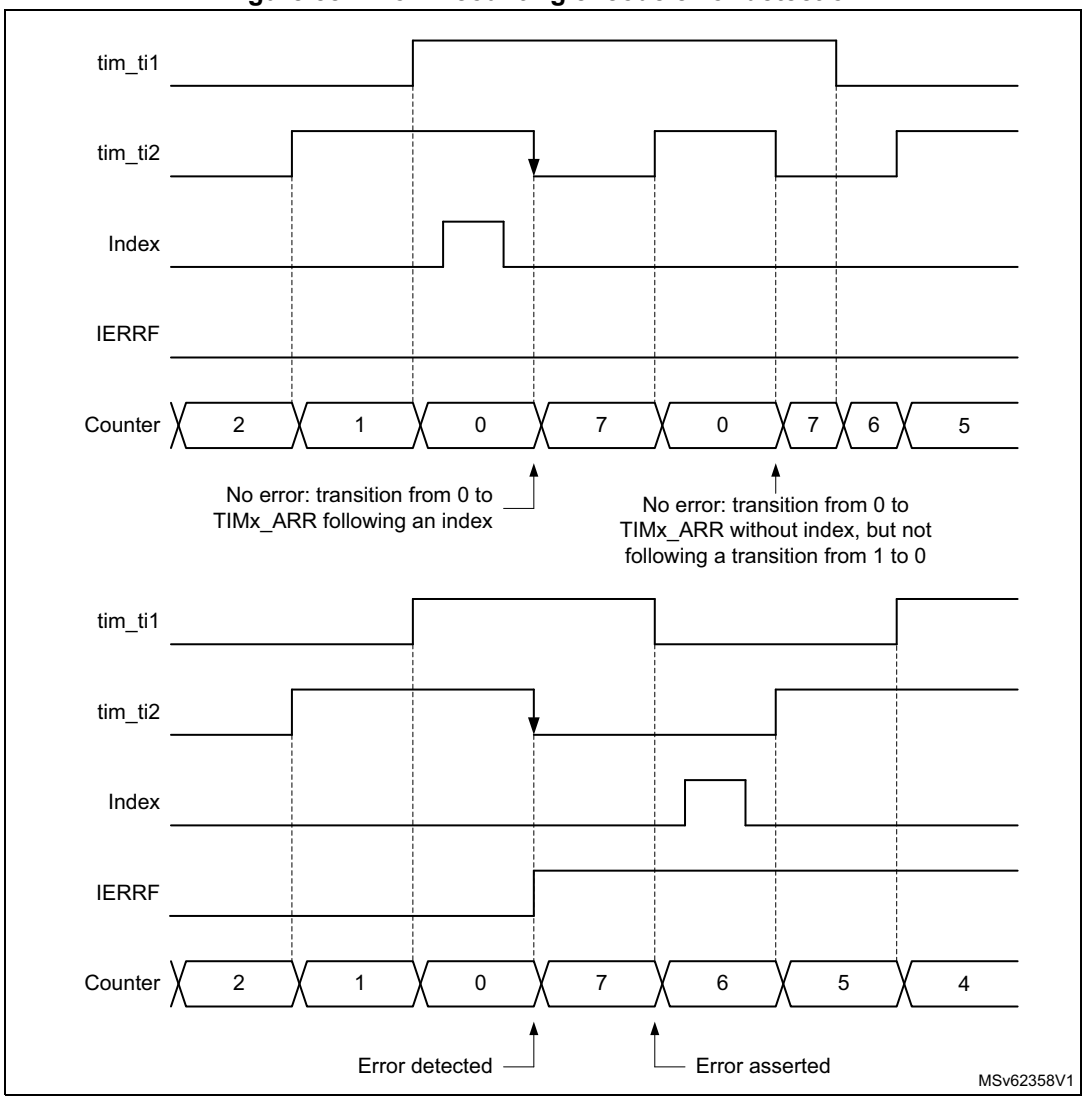
The error assertion is delayed to the transition 0 to 1 when in up-counting. This is cope with narrow index pulses in gated A and B mode, as shown on *Figure 580*.

Figure 580. Up-counting encoder error detection



In down-counting mode, the detection is conditioned by a preliminary transition from 1 to 0. This is to cope with narrow index pulses in gated A and B mode, as shown on *Figure 581*, to avoid any false error detection in case the encoder dithers between TIMx_ARR and 0 immediately after the index detection.

Figure 581. Down-counting encode error detection



An index error sets the IERRF interrupt flag in the TIMx_SR status register. An index error interrupt is generated if the IERRIE bit is set in the TIMx_DIER register.

Functional encoder interrupts

The following interrupts are also available in encoder mode

- Direction change: any change of the counting direction in encoder mode causes the DIR bit in the TIMx_CR1 register to toggle. The direction change sets the DIRF interrupt flag in the TIMx_SR status register. A direction change interrupt is generated if the DIRIE bit is set in the TIMx_DIER register.
- Index event: the index event sets the IDXF interrupt flag in the TIMx_SR status register. An index interrupt is generated if the IDXIE bit is set in the TIMx_DIER register.

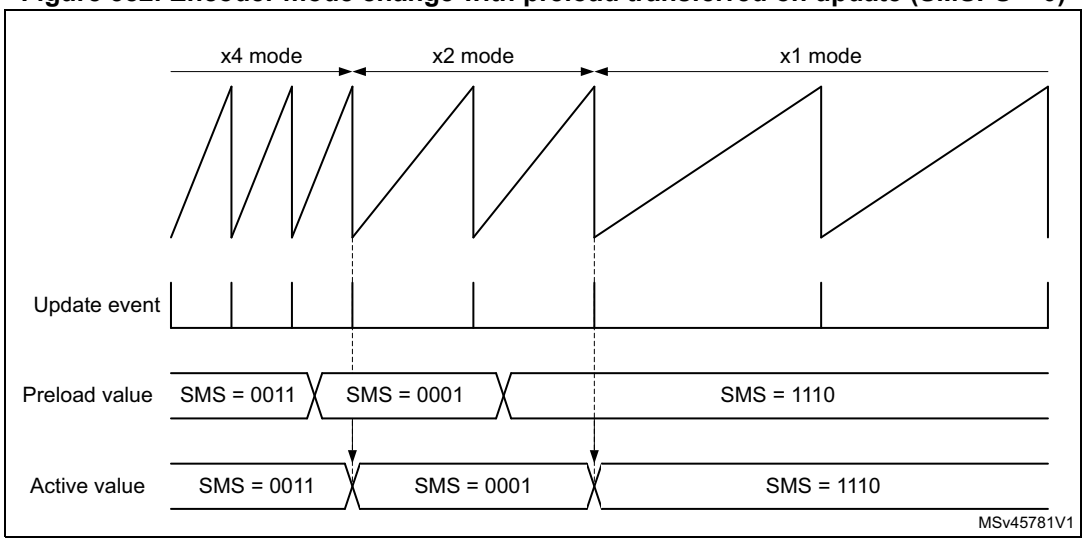
Slave mode selection preload for run-time encoder mode update

It may be necessary to switch from one encoder mode to another during run-time. This is typically done at high-speed to decrease the update interrupt rate, by switching from x4 to x2 to x1 mode, as shown on *Figure 582*.

For this purpose, the SMS[3:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIMx_SMCR register. The trigger for the transfer from SMS[3:0] preload to active value can be selected with the SMSPS bit in the TIMx_SMCR register.

- SMSPS = 0: the transfer is triggered by the update event (UEV) occurring when the counter overflows when upcounting, and underflows when downcounting.
- SMSPS = 1: the transfer is triggered by the index event.

Figure 582. Encoder mode change with preload transferred on update (SMSPS = 0)



Encoder clock output

The encoder mode operating principle is not perfectly suited for high-resolution velocity measurements, at low speed, as it requires a relatively long integration time to have a sufficient number of clock edges and a precise measurement.

At low speed, a better solution is to do an edge-to-edge clock period measurement. This can be achieved using a slave timer. The timer can output the encoder clock information on the tim_trgo output. The slave timer can then perform a period measurement and provide velocity information for each and every encoder clock edge.

This mode is enabled by setting the MMS[3:0] bitfield to 1000, in the TIMx_CR2 register.It is valid for the following SMS[3:0] values: 0001, 0010, 0011, 1010, 1011, 1100, 1101, 1110, 1111. Any other SMS[3:0] code is not allowed and may lead to unexpected behavior.

53.3.26 Direction bit output

Its is possible to output a direction signal out of the timer, on the tim_oc3n and tim_oc4 output signals (copy of the DIR bit in the TIMx_CR1 register). This is achieved by setting the OC3M[3:0] or the OC4M[3:0] bitfield to 1011 in the TIMx_CCMR2 register.

This feature can be used for monitoring the counting direction (or rotation direction) in encoder mode, or to have a signal indicating the up/down phases in center-aligned PWM mode.

53.3.27 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag UIF into the timer counter register's bit 31 (TIMxCNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. In particular cases, it can ease the calculations by avoiding race conditions, caused for instance by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

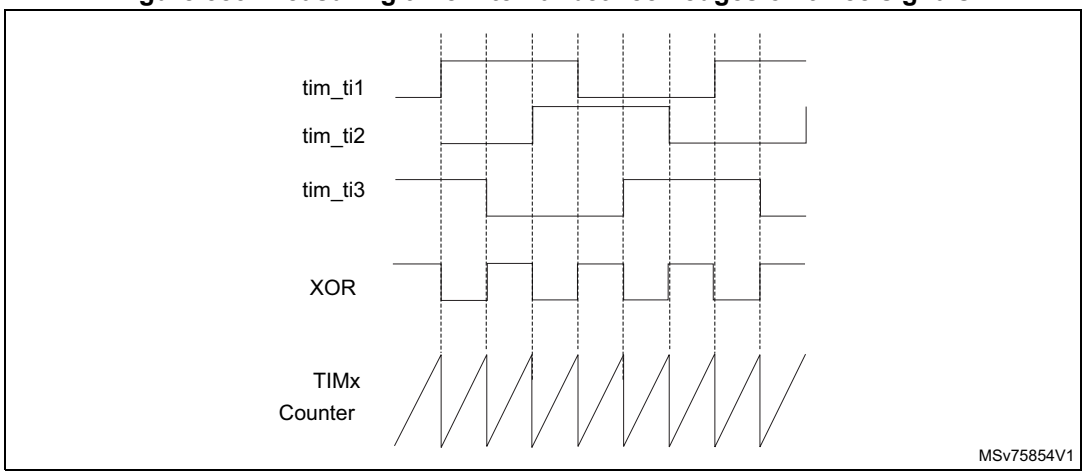
There is no latency between the UIF and UIFCPY flags assertion.

53.3.28 Timer input XOR function

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of an XOR gate, combining the three input pins tim_ti1, tim_ti2 and tim_ti3.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is convenient to measure the interval between edges on two input signals, as per *Figure 583*.

Figure 583. Measuring time interval between edges on three signals



53.3.29 Interfacing with Hall sensors

This is done using the advanced-control timers to generate PWM signals to drive the motor and another timer TIMx referred to as "interfacing timer" in *Figure 584*. The "interfacing timer" captures the three timer input pins (tim_ti1, tim_ti2 and tim_ti3) connected through a XOR to the tim_ti1 input channel (selected by setting the TI1S bit in the TIMx_CR2 register).

The slave mode controller is configured in reset mode; the slave input is tim_ti1f_ed. Thus, each time one of the three inputs toggles, the counter restarts counting from 0. This creates a time base triggered by any change on the Hall inputs.

On the "interfacing timer", capture/compare channel 1 is configured in capture mode, capture signal is tim_trc (See *Figure 527*). The captured value, which corresponds to the time elapsed between two changes on the inputs, gives information about motor speed.

The “interfacing timer” can be used in output mode to generate a pulse which changes the configuration of the channels of the advanced-control timer (by triggering a COM event). The advanced-control timer is used to generate PWM signals to drive the motor. To do this, the interfacing timer channel must be programmed so that a positive pulse is generated after a programmed delay (in output compare or PWM mode). This pulse is sent to the advanced-control timer through the tim_trgo output.

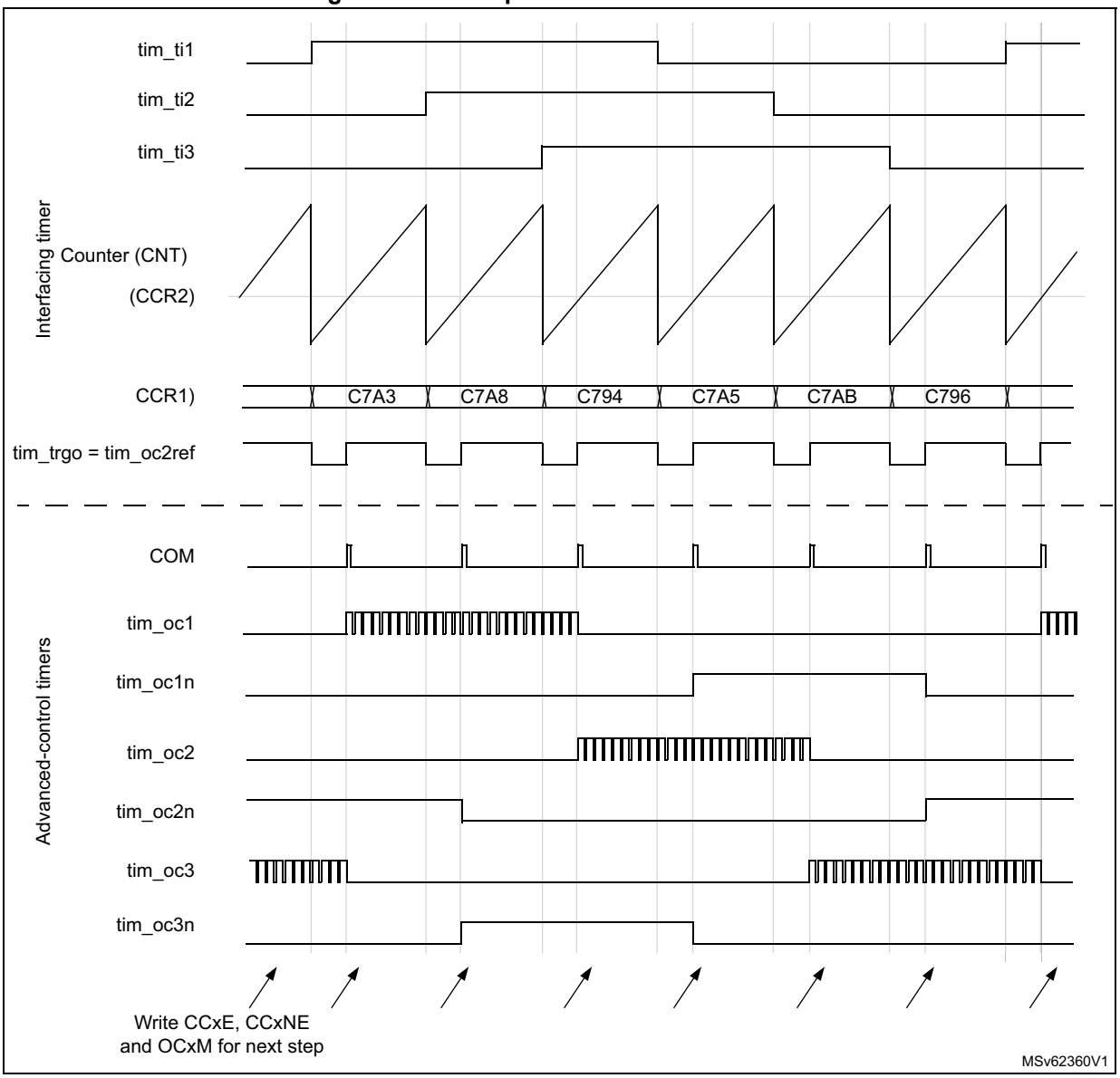
In this example the user wants to change the PWM configuration of the advanced-control timer after a programmed delay each time a change occurs on the Hall inputs connected to one of the TIMx timers.

- Configure three timer inputs ORed to the tim_ti1 input channel by writing the TI1S bit in the TIMx_CR2 register to 1.
- Program the time base: write the TIMx_ARR to the max value (the counter must be cleared by the tim_ti1 change. Set the prescaler to get a maximum counter period longer than the time between two changes on the sensors.
- Program the channel 1 in capture mode (tim_trc selected): write the CC1S bits in the TIMx_CCMR1 register to 01. The digital filter can also be programmed if needed.
- Program the channel 2 in PWM 2 mode with the desired delay: write the OC2M bits to 111 and the CC2S bits to 00 in the TIMx_CCMR1 register.
- Select tim_oc2ref as trigger output on tim_trgo: write the MMS bits in the TIMx_CR2 register to 101.

In the advanced-control timer, the right tim_itrx input must be selected as trigger input, the timer is programmed to generate PWM signals, the capture/compare control signals are preloaded (CCPC = 1 in the TIMx_CR2 register) and the COM event is controlled by the trigger input (CCUS = 1 in the TIMx_CR2 register). The PWM control bits (CCxE, OCxM) are written after a COM event for the next step (this can be done in an interrupt subroutine generated by the rising edge of tim_oc2ref).

Figure 584 describes this example.

Figure 584. Example of Hall sensor interface



53.3.30 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to *Section 54.4.23: Timer synchronization* for details. They can be synchronized in several modes: Reset mode, Gated mode, Trigger mode, Reset + trigger, and gated + reset modes.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

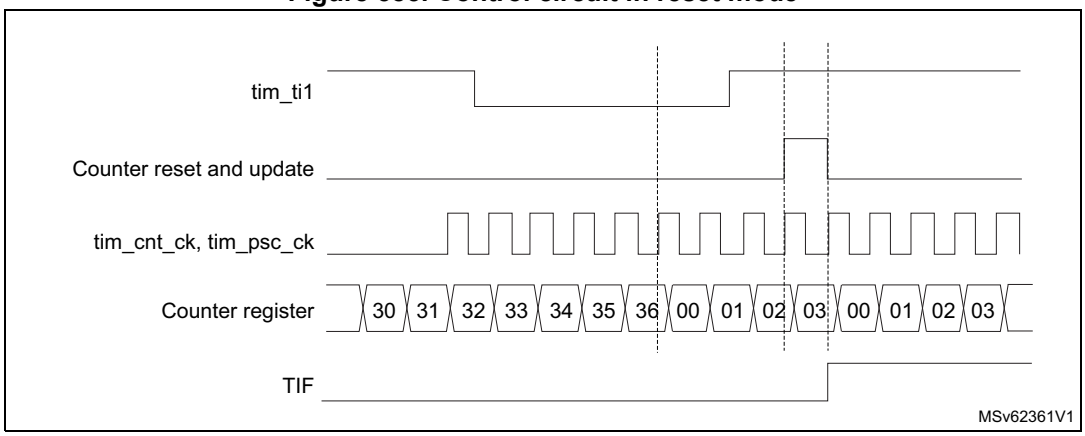
In the following example, the upcounter is cleared in response to a rising edge on tim_ti1 input:

- Configure the channel 1 to detect rising edges on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P = 0 and CC1NP = 0 in TIMx_CCER register to validate the polarity (and detect rising edges only).
- Configure the timer in reset mode by writing SMS = 100 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIMx_SMCR register.
- Start the counter by writing CEN = 1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until tim_ti1 rising edge. When tim_ti1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the autoreload register TIMx_ARR = 0x36. The delay between the rising edge on tim_ti1 and the actual reset of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 585. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

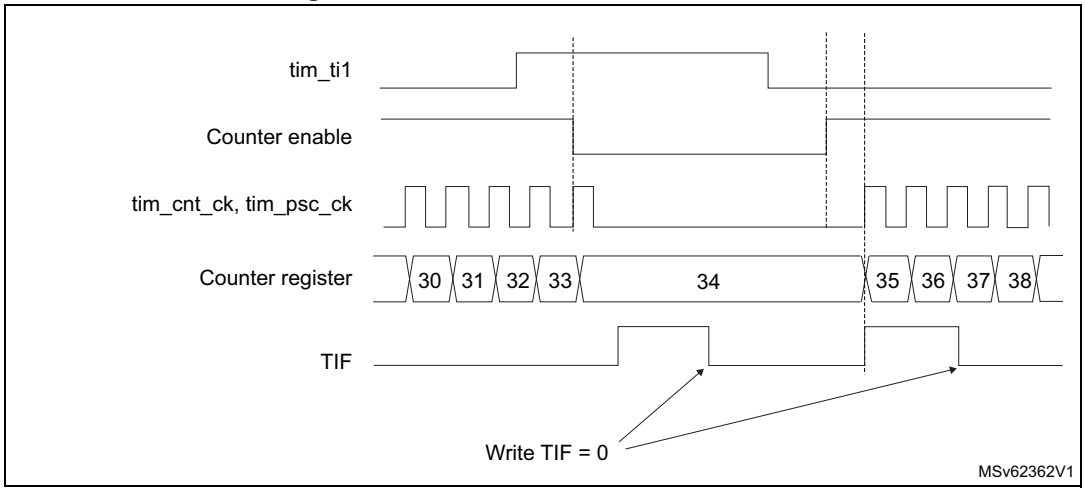
In the following example, the upcounter counts only when tim_ti1 input is low:

- Configure the channel 1 to detect low levels on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in TIMx_CCMR1 register. Write CC1P = 1 and CC1NP = 0 in TIMx_CCER register to validate the polarity (and detect low level only).
- Configure the timer in gated mode by writing SMS = 101 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIMx_SMCR register.
- Enable the counter by writing CEN = 1 in the TIMx_CR1 register (in gated mode, the counter does not start if CEN = 0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as tim_ti1 is low and stops as soon as tim_ti1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on tim_ti1 and the actual stop of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 586. Control circuit in Gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

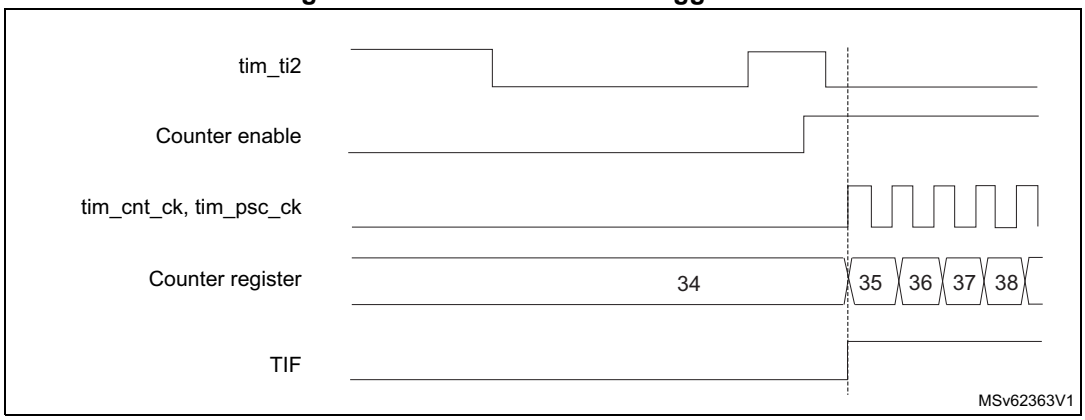
In the following example, the upcounter starts in response to a rising edge on tim_ti2 input:

- Configure the channel 2 to detect rising edges on tim_ti2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC2S bits are configured to select the input capture source only, CC2S = 01 in TIMx_CCMR1 register. Write CC2P = 1 and CC2NP = 0 in TIMx_CCER register to validate the polarity (and detect low level only).
- Configure the timer in trigger mode by writing SMS = 110 in TIMx_SMCR register. Select tim_ti2 as the input source by writing TS = 00110 in TIMx_SMCR register.

When a rising edge occurs on tim_ti2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual start of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 587. Control circuit in trigger mode



Slave mode: Combined reset + trigger mode

In this case, a rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for One-pulse mode.

Slave mode: Combined gated + reset mode

The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

This mode is used to detect out-of-range PWM signal (duty cycle exceeding a maximum expected value).

Slave mode: external clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the tim_etr_in signal is used as external clock input, and another input can be selected as trigger input (in reset mode, gated mode or trigger mode). It is recommended not to select tim_etr_in as tim_trgi through the TS bits of TIMx_SMCR register.

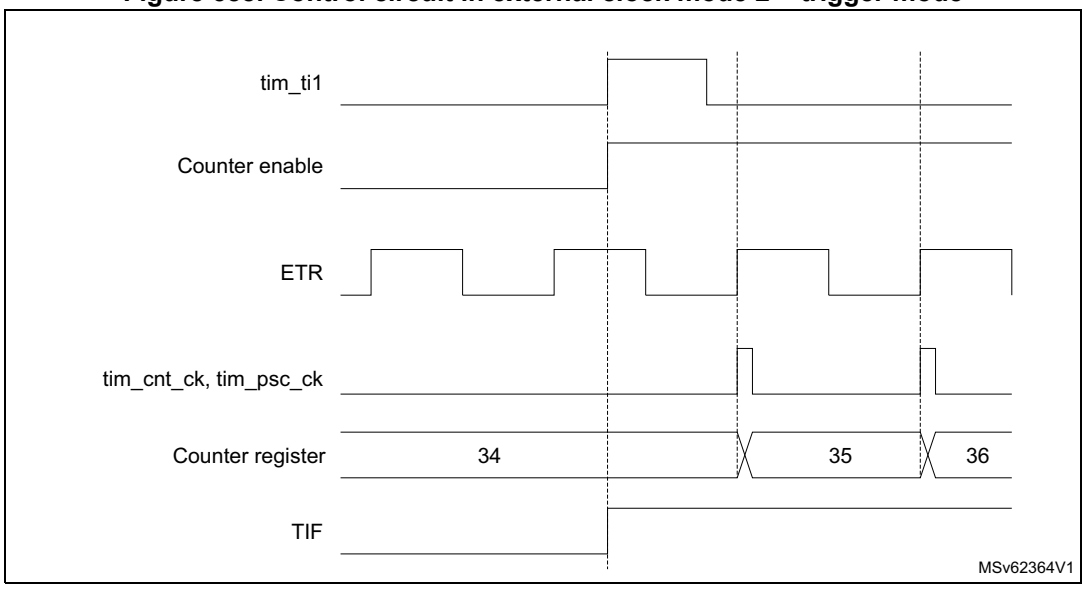
In the following example, the upcounter is incremented at each rising edge of the tim_etr_in signal as soon as a rising edge of tim_ti1 occurs:

1. Configure the external trigger input circuit by programming the TIMx_SMCR register as follows:
   - ETF = 0000: no filter
   - ETPS = 00: prescaler disabled
   - ETP = 0: detection of rising edges on tim_etr_in and ECE = 1 to enable the external clock mode 2.
2. Configure the channel 1 as follows, to detect rising edges on TI:
   - IC1F = 0000: no filter.
   - The capture prescaler is not used for triggering and does not need to be configured.
   - CC1S = 01in TIMx_CCMR1 register to select only the input capture source
   - CC1P = 0 and CC1NP = 0 in TIMx_CCER register to validate the polarity (and detect rising edge only).
3. Configure the timer in trigger mode by writing SMS = 110 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIMx_SMCR register.

A rising edge on tim_ti1 enables the counter and sets the TIF flag. The counter then counts on tim_etr_in rising edges.

The delay between the rising edge of the tim_etr_in signal and the actual reset of the counter is due to the resynchronization circuit on tim_etrp input.

Figure 588. Control circuit in external clock mode 2 + trigger mode



Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo or the tim_trgo2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

53.3.31 ADC triggers

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events. It is also possible to generate a pulse issued by internal edge detectors, such as:

- Rising and falling edges of OC4ref
- Rising edge on OC5ref or falling edge on OC6ref

The triggers are issued on the tim_trgo2 internal line which is redirected to the ADC. There is a total of 16 possible events, which can be selected using the MMS2[3:0] bits in the TIMx_CR2 register.

An example of an application for 3-phase motor drives is given in *Figure 542*.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the tim_trgo or the tim_trgo2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

The clock of the ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the timer.

53.3.32 ADC synchronization

The timer operation can be synchronized to the ADC clock to trigger jitter-free ADC sampling. This function is enabled using the ADSYNC bit in the TIMx_CR2 register.

This feature is useful when the timers and the ADCs are operating with semisynchronous clocks (clocks derived from a same source with integer ratio tim_ker_ck/adc_ker_ck), for instance adc_ker_ck = 75 MHz and tim_ker_ck = 150 MHz or 300 MHz.

ADSYNC must also be set when both peripherals are operating at the same frequency from the same clock source, when jitter-free operation is needed.

ADSYNC must not be set and jitter-free operation is not supported in the following cases:

- when the clock ratio is not an integer (for example adc_ker_ck = 75 MHz and tim_ker_ck = 100 MHz): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum
- when the ADC is operating in asynchronous mode (adc_ker_ck uncorrelated with tim_ker_ck): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum.

When ADSYNC = 1, the timer operation is slightly changed: the counter enable and counter reset events are aligned to the adc_ker_ck ADC clock, to avoid any phase shift due to clocks enable in the RCC.

Jitter-free operation is guaranteed only when one of the two requirements below is met (depending on the selected trigger source).

1. The counter period must be a multiple of the ADC clock period:

$$(\text{TIMx_PSC} + 1) \times (\text{TIMx_ARR} + 1) \times T_{\text{tim_ker_ck}} = n \times T_{\text{adc_ker_ck}}$$

2. The compare value must be a multiple of the ADC clock period:

$$(\text{TIMx_PSC} + 1) \times \text{TIMx_CMPy} \times T_{\text{tim_ker_ck}} = m \times T_{\text{adc_ker_ck}}$$

Note: If none of the two above requirements are met, the trigger is still generated, but the latency is not constant and varies with the timer and ADC clocks phase shift.

Programming guidelines

The ADC synchronization feature must not be modified during run-time, once the counter is enabled and once the ADC has been configured for receiving triggers from the timer.

It is mandatory to follow the procedure below to use the ADC synchronization:

1. Enable the destination ADC clock.
2. Configure the timer and set the ADSYNC bit.
3. Configure the ADC and enable it (using ADSTART and/or JADSTART bits).
4. Start the timer (with the CEN counter enable bit).

53.3.33 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests upon a single event. The main purpose is to be able to reprogram part of the timer multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR register define the DMA base address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register:

Example:

00000: TIMx_CR1

00001: TIMx_CR2

00010: TIMx_SMCR

The DBSS[3:0] bits in the TIMx_DCR register defines the interrupt source that triggers the DMA burst transfers (see *Section 53.6.29: TIMx DMA control register (TIMx_DCR)(x = 1, 8)* for details).

As an example, the timer DMA burst feature is used to update the contents of the CCRx registers (x = 2, 3, 4) upon an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   – DMA channel peripheral address is the DMAR register address.
   – DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers.
   – Number of data to transfer = 3 (see note below).
   – Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bitfields as follows: DBL = 3 transfers, DBA = 0xE and DBSS = 1.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx.
5. Enable the DMA channel.

This example is for the case where every CCRx register to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer must be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5, and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3, and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

53.3.34 TIM1/TIM8 DMA requests

The TIM1/TIM8 can generate a DMA request, as shown in the table below.

Table 497. DMA request

DMA request signal	DMA request	Enable control bit
tim_upd_dma	Update	UDE
tim_cc1_dma	Capture/compare 1	CC1DE
tim_cc2_dma	Capture/compare 2	CC2DE
tim_cc3_dma	Capture/compare 3	CC3DE
tim_cc4_dma	Capture/compare 4	CC4DE
tim_com_dma	Commutation (COM)	COMDE
tim_trgi_dma	Trigger	TDE

53.3.35 Debug mode

When the microcontroller enters debug mode (Cortex®-M55 core halted), the TIMx counter can either continue to work normally or stop, depending on DBG_TIMx_STOP configuration bit in DBG module.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For safety purposes, when the counter is stopped, the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0), typically to force a Hi-Z.

For more details, refer to section Debug support (DBG).

53.4 TIM1/TIM8 low-power modes

Table 498. Effect of low-power modes on TIM1/TIM8

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Standby	The timer is powered-down and must be reinitialized after exiting the Standby mode.

53.5 TIM1/TIM8 interrupts

The TIM1/TIM8 can generate multiple interrupts, as shown in *Table 499*.

Table 499. Interrupt requests

Interrupt acronym		Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
TIM_UPD		Update	UIF	UIE	write 0 in UIF	Yes	No
TIM_CC		Capture/compare 1	CC1IF	CC1IE	write 0 in CC1IF	Yes	No
		Capture/compare 2	CC2IF	CC2IE	write 0 in CC2IF	Yes	No
		Capture/compare 3	CC3IF	CC3IE	write 0 in CC3IF	Yes	No
		Capture/compare 4	CC4IF	CC4IE	write 0 in CC4IF	Yes	No
TIM_TRGI_COM_DIR_IDX	TIM_COM	Commutation (COM)	COMIF	COMIE	write 0 in COMIF	Yes	No
	TIM_TRGI	Trigger	TIF	TIE	write 0 in TIF	Yes	No
	TIM_IDX	Index	IDXF	IDXIE	write 0 in IDXF	Yes	No
	TIM_DIR	Direction	DIRF	DIRIE	write 0 in DIRF	Yes	No
TIM_BRK_TERR_IERR	TIM_BRK	Break	BIF	BIE	write 0 in BIF	Yes	No
		Break2	B2IF		write 0 in B2IF	Yes	No
		System Break	SBIF		write 0 in SBIF	Yes	No
	TIM_IERR	Index Error	IERRF	IERRIE	write 0 in IERRF	Yes	No
	TIM_TERR	Transition Error	TERRF	TERRIE	write 0 in TERRF	Yes	No

53.6 TIM1/TIM8 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

53.6.1 TIMx control register 1 (TIMx_CR1)(x = 1, 8)

Address offset: 0x000

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable

0: Dithering disabled
1: Dithering enabled

Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and the dead-time and sampling clock (t_{DTS})used by the dead-time generators and the digital filters (tim_etr_in, tim_tix),

00: $t_{DTS} = t_{tim_ker_ck}$
01: $t_{DTS} = 2*t_{tim_ker_ck}$
10: $t_{DTS} = 4*t_{tim_ker_ck}$
11: Reserved, do not program this value

Bit 7 **ARPE**: Autoreload preload enable

0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection

00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).
01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIMx_CCMRx register) are set only when the counter is counting down.
10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIMx_CCMRx register) are set only when the counter is counting up.
11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIMx_CCMRx register) are set both when the counter is counting up or down.

Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN = 1)

Bit 4 **DIR**: Direction
0: Counter used as upcounter
1: Counter used as downcounter
Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source
This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:
– Counter overflow/underflow
– Setting the UG bit
– Update generation through the slave mode controller
1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable
This bit is set and cleared by software to enable/disable UEV event generation.
0: UEV enabled. The Update (UEV) event is generated by one of the following events:
– Counter overflow/underflow
– Setting the UG bit
– Update generation through the slave mode controller
Buffered registers are then loaded with their preload values.
1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable
0: Counter disabled
1: Counter enabled
Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

53.6.2 TIMx control register 2 (TIMx_CR2)(x = 1, 8)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	AD SYNC	Res.	Res.	MMS[3]	Res.	MMS2[3:0]				Res.	OIS6	Res.	OIS5
			rw			rw		rw	rw	rw	rw		rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OIS4N	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **ADSYNC**: ADC synchronization

0: The timer operates independently from the ADC

1: The timer operation is synchronized with the ADC clock to provide jitter-free sampling point. This mode can be enabled only with specific ADC / timer clock relationship. Refer to *Section 53.3.32* for requirements.

The ADSYNC must not modified when the counter is enabled (CEN bit is set).

Bits 27:26 Reserved, must be kept at reset value.

Bit 24 Reserved, must be kept at reset value.

Bits 23:20 **MMS2[3:0]**: Master mode selection 2

These bits allow the information to be sent to ADC for synchronization (tim_trgo2) to be selected. The combination is as follows:

0000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (tim_trgo2). If the reset is generated by the trigger input (slave mode controller configured in reset mode), the signal on tim_trgo2 is delayed compared to the actual reset.

0001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (tim_trgo2). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic AND between the CEN control bit and the trigger input when configured in Gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo2, except if the Master/Slave mode is selected (see the MSM bit description in TIMx_SMCR register).

0010: **Update** - the update event is selected as trigger output (tim_trgo2). For instance, a master timer can then be used as a prescaler for a slave timer.

0011: **Compare pulse** - the trigger output sends a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or compare match occurs (tim_trgo2).

0100: **Compare** - tim_oc1refc signal is used as trigger output (tim_trgo2)

0101: **Compare** - tim_oc2refc signal is used as trigger output (tim_trgo2)

0110: **Compare** - tim_oc3refc signal is used as trigger output (tim_trgo2)

0111: **Compare** - tim_oc4refc signal is used as trigger output (tim_trgo2)

1000: **Compare** - tim_oc5refc signal is used as trigger output (tim_trgo2)

1001: **Compare** - tim_oc6refc signal is used as trigger output (tim_trgo2)

1010: **Compare Pulse** - tim_oc4refc rising or falling edges generate pulses on tim_trgo2

1011: **Compare pulse** - tim_oc6refc rising or falling edges generate pulses on tim_trgo2

1100: **Compare pulse** - tim_oc4refc or tim_oc6refc rising edges generate pulses on tim_trgo2

1101: **Compare pulse** - tim_oc4refc rising or tim_oc6refc falling edges generate pulses on tim_trgo2

1110: **Compare pulse** - tim_oc5refc or tim_oc6refc rising edges generate pulses on tim_trgo2

1111: **Compare pulse** - tim_oc5refc rising or tim_oc6refc falling edges generate pulses on tim_trgo2

Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 19 Reserved, must be kept at reset value.

Bit 18 **OIS6**: Output idle state 6 (tim_oc6 output)

Refer to OIS1 bit

Bit 17 Reserved, must be kept at reset value.

Bit 16 **OIS5**: Output idle state 5 (tim_oc5 output)
Refer to OIS1 bit

Bit 15 **OIS4N**: Output idle state 4 (tim_oc4n output)
Refer to OIS1N bit

Bit 14 **OIS4**: Output idle state 4 (tim_oc4 output)
Refer to OIS1 bit

Bit 13 **OIS3N**: Output idle state 3 (tim_oc3n output)
Refer to OIS1N bit

Bit 12 **OIS3**: Output idle state 3 (tim_oc3n output)
Refer to OIS1 bit

Bit 11 **OIS2N**: Output idle state 2 (tim_oc2n output)
Refer to OIS1N bit

Bit 10 **OIS2**: Output idle state 2 (tim_oc2 output)
Refer to OIS1 bit

Bit 9 **OIS1N**: Output idle state 1 (tim_oc1n output)
0: tim_oc1n = 0 after a dead-time when MOE = 0
1: tim_oc1n = 1 after a dead-time when MOE = 0
Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **OIS1**: Output idle state 1 (tim_oc1 output)
0: tim_oc1 = 0 (after a dead-time) when MOE = 0
1: tim_oc1 = 1 (after a dead-time) when MOE = 0
Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 7 **TI1S**: tim_ti1 selection
0: The tim_ti1_in[15:0] multiplexer output is connected to tim_ti1 input
1: tim_ti1_in[15:0], tim_ti2_in[15:0] and tim_ti3_in[15:0] multiplexers outputs are XORed and connected to the tim_ti1 input

Bits 25, 6:4 **MMS[3:0]**: Master mode selection

These bits select the information to be sent in master mode to slave timers for synchronization (tim_trgo). The combination is as follows:

0000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (tim_trgo). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on tim_trgo is delayed compared to the actual reset.

0001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).

0010: **Update** - The update event is selected as trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

0011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred. (tim_trgo).

0100: **Compare** - tim_oc1refc signal is used as trigger output (tim_trgo)

0101: **Compare** - tim_oc2refc signal is used as trigger output (tim_trgo)

0110: **Compare** - tim_oc3refc signal is used as trigger output (tim_trgo)

0111: **Compare** - tim_oc4refc signal is used as trigger output (tim_trgo)

1000: **Encoder Clock output** - The encoder clock signal is used as trigger output (tim_trgo). This code is valid for the following SMS[3:0] values: 0001, 0010, 0011, 1010, 1011, 1100, 1101, 1110, 1111. Any other SMS[3:0] code is not allowed and may lead to unexpected behavior.

Other codes reserved

Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs

1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit only

1: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit or when an rising edge occurs on tim_trgi

Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control

0: CCxE, CCxNE and OCxM bits are not preloaded

1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when a commutation event (COM) occurs (COMG bit set or rising edge detected on tim_trgi, depending on the CCUS bit).

Note: This bit acts only on channels that have a complementary output.

53.6.3 TIMx slave mode control register (TIMx_SMCR)(x = 1, 8)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	SMSPS	SMSPE	Res.	Res.	TS[4:3]		Res.	Res.	Res.	SMS[3]
						rw	rw			rw	rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			Res.	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **SMSPS**: SMS preload source

This bit selects whether the events that triggers the SMS[3:0] bitfield transfer from preload to active

0: The transfer is triggered by the Timer's Update event

1: The transfer is triggered by the Index event

Bit 24 **SMSPE**: SMS preload enable

This bit selects whether the SMS[3:0] bitfield is preloaded

0: SMS[3:0] bitfield is not preloaded

1: SMS[3:0] preload is enabled

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:20 **TS[4:3]**: Trigger selection - bit 4:3

Refer to TS[2:0] description - bits 6:4

Bits 19:17 Reserved, must be kept at reset value.

Bit 15 **ETP**: External trigger polarity

This bit selects whether tim_etr_in or tim_$\overline{\text{etr_in}}$ is used for trigger operations

0: tim_etr_in is non-inverted, active at high level or rising edge.

1: tim_etr_in is inverted, active at low level or falling edge.

Bit 14 **ECE**: External clock enable

This bit enables External clock mode 2.

0: External clock mode 2 disabled

1: External clock mode 2 enabled. The counter is clocked by any active edge on the tim_etrf signal.

Note: Setting the ECE bit has the same effect as selecting external clock mode 1 with tim_trgi connected to tim_etrf (SMS = 111 and TS = 00111).

It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, tim_trgi must not be connected to tim_etrf in this case (TS bits must not be 00111).

If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is tim_etrf.

Bits 13:12 **ETPS[1:0]**: External trigger prescaler

External trigger signal tim_etrp frequency must be at most 1/4 of TIMxCLK frequency. A prescaler can be enabled to reduce tim_etrp frequency. It is useful when inputting fast external clocks on tim_etr_in.

00: Prescaler OFF
01: tim_etr_in frequency divided by 2
10: tim_etr_in frequency divided by 4
11: tim_etr_in frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bitfield then defines the frequency used to sample tim_etrp signal and the length of the digital filter applied to tim_etrp. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Bit 7 **MSM:** Master/slave mode

0: No action
1: The effect of an event on the trigger input (tim_trgi) is delayed to allow a perfect synchronization between the current timer and its slaves (through tim_trgo). It is useful if we want to synchronize several timers on a single external event.

Bits 6:4 **TS[2:0]:** Trigger selection

This bitfield is combined with TS[4:3] bits.

This bitfield selects the trigger input to be used to synchronize the counter.

00000: Internal Trigger 0 (tim_itr0)
00001: Internal Trigger 1 (tim_itr1)
00010: Internal Trigger 2 (tim_itr2)
00011: Internal Trigger 3 (tim_itr3)
00100: tim_ti1 Edge Detector (tim_ti1f_ed)
00101: Filtered Timer Input 1 (tim_ti1fp1)
00110: Filtered Timer Input 2 (tim_ti2fp2)
00111: External Trigger input (tim_etrf)
01000: Internal Trigger 4 (tim_itr4)
01001: Internal Trigger 5 (tim_itr5)
01010: Internal Trigger 6 (tim_itr6)
01011: Internal Trigger 7 (tim_itr7)
01100: Internal Trigger 8 (tim_itr8)
01101: Internal Trigger 9 (tim_itr9)
01110: Internal Trigger 10 (tim_itr10)
01111: Internal trigger 11 (tim_itr11)
10000: Internal trigger 12 (tim_itr12)
10001: Internal trigger 13 (tim_itr13)
10010: Internal trigger 14 (tim_itr14)
10011: Internal trigger 15 (tim_itr15)
Others: Reserved

See for more details on tim_itrx meaning for each Timer.

Note: These bits must be changed only when they are not used (for example when SMS = 000) to avoid wrong edge detections at the transition.

Bit 3 Reserved, must be kept at reset value.

Bits 16, 2:0 **SMS[3:0]:** Slave mode selection

When external signals are selected the active edge of the trigger signal (tim_trgi) is linked to the polarity selected on the external input (refer to ETP bit in TIMx_SMCR for tim_etr_in and CCxP/CCxNP bits in TIMx_CCER register for tim_ti1fp1 and tim_ti2fp2).

0000: Slave mode disabled - if CEN = 1 then the prescaler is clocked directly by the internal clock.

0001: Quadrature encoder mode 1, x2 mode- Counter counts up/down on tim_ti1fp1 edge depending on tim_ti2fp2 level.

0010: Quadrature encoder mode 2, x2 mode - Counter counts up/down on tim_ti2fp2 edge depending on tim_ti1fp1 level.

0011: Quadrature encoder mode 3, x4 mode - Counter counts up/down on both tim_ti1fp1 and tim_ti2fp2 edges depending on the level of the other input.

0100: Reset mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter and generates an update of the registers.

0101: Gated mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

0110: Trigger mode - The counter starts at a rising edge of the trigger tim_trgi (but it is not reset). Only the start of the counter is controlled.

0111: External Clock mode 1 - Rising edges of the selected trigger (tim_trgi) clock the counter.

1000: Combined reset + trigger mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers and starts the counter.

1001: Combined gated + reset mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

1010: Encoder mode: Clock plus direction, x2 mode.

1011: Encoder mode: Clock plus direction, x1 mode, tim_ti2fp2 edge sensitivity is set by CC2P

1100: Encoder mode: Directional Clock, x2 mode.

1101: Encoder mode: Directional Clock, x1 mode, tim_ti1fp1 and tim_ti2fp2 edge sensitivity is set by CC1P and CC2P.

1110: Quadrature encoder mode: x1 mode, counting on tim_ti1fp1 edges only, edge sensitivity is set by CC1P.

1111: Quadrature encoder mode: x1 mode, counting on tim_ti2fp2 edges only, edge sensitivity is set by CC2P.

Note: The gated mode must not be used if tim_ti1f_ed is selected as the trigger input (TS = 00100). Indeed, tim_ti1f_ed outputs 1 pulse for each transition on TI1F, whereas the gated mode checks the level of the trigger signal.

Note: The clock of the slave peripherals (timer, ADC, ...) receiving the tim_trgo or the tim_trgo2 signals must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

53.6.4 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 1, 8)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRIE	IERRIE	DIRIE	IDXIE	Res.	Res.	Res.	Res.
								rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TERRIE**: Transition error interrupt enable
0: Transition error interrupt disabled
1: Transition error interrupt enabled

Bit 22 **IERRIE**: Index error interrupt enable
0: Index error interrupt disabled
1: Index error interrupt enabled

Bit 21 **DIRIE**: Direction change interrupt enable
0: Direction Change interrupt disabled
1: Direction Change interrupt enabled

Bit 20 **IDXIE**: Index interrupt enable
0: Index interrupt disabled
1: Index Change interrupt enabled

Bits 19:15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable
0: Trigger DMA request disabled
1: Trigger DMA request enabled

Bit 13 **COMDE**: COM DMA request enable
0: COM DMA request disabled
1: COM DMA request enabled

Bit 12 **CC4DE**: Capture/compare 4 DMA request enable
0: CC4 DMA request disabled
1: CC4 DMA request enabled

Bit 11 **CC3DE**: Capture/compare 3 DMA request enable
0: CC3 DMA request disabled
1: CC3 DMA request enabled

Bit 10 **CC2DE**: Capture/compare 2 DMA request enable
0: CC2 DMA request disabled
1: CC2 DMA request enabled

Bit 9 **CC1DE**: Capture/compare 1 DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable
0: Break interrupt disabled
1: Break interrupt enabled

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled
1: Trigger interrupt enabled

Bit 5 **COMIE**: COM interrupt enable
0: COM interrupt disabled
1: COM interrupt enabled

Bit 4 **CC4IE**: Capture/compare 4 interrupt enable
0: CC4 interrupt disabled
1: CC4 interrupt enabled

Bit 3 **CC3IE**: Capture/compare 3 interrupt enable
0: CC3 interrupt disabled
1: CC3 interrupt enabled

Bit 2 **CC2IE**: Capture/compare 2 interrupt enable
0: CC2 interrupt disabled
1: CC2 interrupt enabled

Bit 1 **CC1IE**: Capture/compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

53.6.5 TIMx status register (TIMx_SR)(x = 1, 8)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRF	IERRF	DIRF	IDXF	Res.	Res.	CC6IF	CC5IF
								rc_w0	rc_w0	rc_w0	rc_w0			rc_w0	rc_w0
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	SBIF	CC4OF	CC3OF	CC2OF	CC1OF	B2IF	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TERRF**: Transition error interrupt flag
This flag is set by hardware when a transition error is detected in encoder mode. It is cleared by software by writing it to 0.
0: No encoder transition error has been detected.
1: An encoder transition error has been detected

Bit 22 **IERRF**: Index error interrupt flag
This flag is set by hardware when an index error is detected. It is cleared by software by writing it to 0.
0: No index error has been detected.
1: An index error has been detected

Bit 21 **DIRF**: Direction change interrupt flag
This flag is set by hardware when the direction changes in encoder mode (DIR bit value in TIMx_CR is changing). It is cleared by software by writing it to 0.
0: No direction change
1: Direction change

Bit 20 **IDXF**: Index interrupt flag
This flag is set by hardware when an index event is detected. It is cleared by software by writing it to 0.
0: No index event occurred.
1: An index event has occurred

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CC6IF**: Compare 6 interrupt flag
Refer to CC1IF description
Note: Channel 6 can only be configured as output.

Bit 16 **CC5IF**: Compare 5 interrupt flag
Refer to CC1IF description
Note: Channel 5 can only be configured as output.

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **SBIF**: System break interrupt flag
This flag is set by hardware as soon as the system break input goes active. It can be cleared by software if the system break input is not active.
This flag must be reset to re-start PWM operation.
0: No break event occurred.
1: An active level has been detected on the system break input. An interrupt is generated if BIE = 1 in the TIMx_DIER register.

Bit 12 **CC4OF**: Capture/compare 4 overcapture flag
Refer to CC1OF description

Bit 11 **CC3OF**: Capture/compare 3 overcapture flag
Refer to CC1OF description

Bit 10 **CC2OF**: Capture/compare 2 overcapture flag
Refer to CC1OF description

Bit 9 **CC1OF**: Capture/compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.

0: No overcapture has been detected.

1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bit 8 **B2IF**: Break 2 interrupt flag

This flag is set by hardware as soon as the break 2 input goes active. It can be cleared by software if the break 2 input is not active.

0: No break event occurred.

1: An active level has been detected on the break 2 input. An interrupt is generated if BIE = 1 in the TIMx_DIER register.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.

0: No break event occurred.

1: An active level has been detected on the break input. An interrupt is generated if BIE = 1 in the TIMx_DIER register.

Bit 6 **TIF**: Trigger interrupt flag

This flag is set by hardware on the TRG trigger event (active edge detected on tim_trgi input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.

0: No trigger event occurred.

1: Trigger interrupt pending.

Bit 5 **COMIF**: COM interrupt flag

This flag is set by hardware on COM event (when capture/compare Control bits - CCxE, CCxNE, OCxM - have been updated). It is cleared by software.

0: No COM event occurred.

1: COM interrupt pending.

Bit 4 **CC4IF**: Capture/compare 4 interrupt flag

Refer to CC1IF description

Bit 3 **CC3IF**: Capture/compare 3 interrupt flag

Refer to CC1IF description

Bit 2 **CC2IF**: Capture/compare 2 interrupt flag
Refer to CC1IF description

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag
This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).
0: No compare match / No input capture occurred
1: A compare match or an input capture occurred
If channel CC1 is configured as output: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in downcounting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.
If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0 **UIF**: Update interrupt flag
This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
–At overflow or underflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS = 0 in the TIMx_CR1 register.
–When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS = 0 and UDIS = 0 in the TIMx_CR1 register.
–When CNT is reinitialized by a trigger event (refer to *Section 53.6.3: TIMx slave mode control register (TIMx_SMCR)(x = 1, 8)*), if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

53.6.6 TIMx event generation register (TIMx_EGR)(x = 1, 8)

Address offset: 0x014

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	B2G	BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
							w	w	w	w	w	w	w	w	w

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **B2G**: Break 2 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A break 2 event is generated. MOE bit is cleared and B2IF flag is set. Related interrupt can occur if enabled.

Bit 7 **BG**: Break generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 **TG**: Trigger generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 **COMG**: Capture/compare control update generation
This bit can be set by software, it is automatically cleared by hardware
0: No action
1: CCxE, CCxNE and OCxM bits update (providing CCPC bit is set)
Note: This bit acts only on channels having a complementary output.

Bit 4 **CC4G**: Capture/compare 4 generation
Refer to CC1G description

Bit 3 **CC3G**: Capture/compare 3 generation
Refer to CC1G description

Bit 2 **CC2G**: Capture/compare 2 generation
Refer to CC1G description

Bit 1 **CC1G**: Capture/compare 1 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.
If channel CC1 is configured as input:
The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR = 0 (upcounting), else it takes the autoreload value (TIMx_ARR) if DIR = 1 (downcounting).

53.6.7 TIMx capture/compare mode register 1 (TIMx_CCMR1) (x = 1, 8)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode:

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2

10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1

11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bitfield defines the frequency used to sample tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}

0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2

0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4

0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8

0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6

0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8

0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6

0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8

1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6

1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8

1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5

1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6

1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8

1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5

1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6

1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on CC1 input (tim_ic1). The prescaler is reset as soon as CC1E = 0 (TIMx_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input

01: capture is done once every 2 events

10: capture is done once every 4 events

11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/compare 1 Selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1

10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2

11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

53.6.8 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 1, 8)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2 CE	OC2M[2:0]			OC2 PE	OC2 FE	CC2S[1:0]		OC1 CE	OC1M[2:0]			OC1 PE	OC1 FE	CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode:

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC2CE:** Output compare 2 clear enable

Bits 24, 14:12 **OC2M[3:0]**: Output compare 2 mode

Bit 11 **OC2PE**: Output compare 2 preload enable

Bit 10 **OC2FE**: Output compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2

10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1

11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bit 7 **OC1CE:** Output compare 1 clear enable

0: tim_oc1ref is not affected by the tim_ocref_clr_int signal

1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr_int signal (tim_ocref_clr input or tim_etrf input)

Bits 16, 6:4 **OC1M[3:0]**: Output compare 1 mode

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 and tim_oc1n are derived. tim_oc1ref is active high whereas tim_oc1 and tim_oc1n active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs.This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the ouput keeps the state (active or inactive) it had before entering the frozen state.

0001: Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - tim_oc1ref toggles when TIMx_CNT = TIMx_CCR1.

0100: Force inactive level - tim_oc1ref is forced low.

0101: Force active level - tim_oc1ref is forced high.

0110: PWM mode 1 - In upcounting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive. In downcounting, channel 1 is inactive (tim_oc1ref = 0) as long as TIMx_CNT>TIMx_CCR1 else active (tim_oc1ref = 1).

0111: PWM mode 2 - In upcounting, channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active. In downcounting, channel 1 is active as long as TIMx_CNT>TIMx_CCR1 else inactive.

1000: Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Reserved,

1011: Reserved,

1100: Combined PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc is the logical OR between tim_oc1ref and tim_oc2ref.

1101: Combined PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc is the logical AND between tim_oc1ref and tim_oc2ref.

1110: Asymmetric PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc outputs tim_oc1ref when the counter is counting up, tim_oc2ref when it is counting down.

1111: Asymmetric PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc outputs tim_oc1ref when the counter is counting up, tim_oc2ref when it is counting down.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output).

Note: In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.

Note: On channels having a complementary output, this bitfield is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the OC1M active bits take the new value from the preloaded bits only when a COM event is generated.

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output).

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1

10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2

11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

53.6.9 TIMx capture/compare mode register 2 (TIMx_CCMR2) (x = 1, 8)

Address offset: 0x01C

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 3 in input capture mode and channel 4 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC4F[3:0]				IC4PSC[1:0]		CC4S[1:0]		IC3F[3:0]				IC3PSC[1:0]		CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC4F[3:0]**: Input capture 4 filter

Bits 11:10 **IC4PSC[1:0]**: Input capture 4 prescaler

Bits 9:8 **CC4S[1:0]**: Capture/compare 4 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti4
10: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti3
11: CC4 channel is configured as input, tim_ic4 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).

Bits 7:4 **IC3F[3:0]**: Input capture 3 filter

Bits 3:2 **IC3PSC[1:0]**: Input capture 3 prescaler

Bits 1:0 **CC3S[1:0]**: Capture/compare 3 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti3
10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4
11: CC3 channel is configured as input, tim_ic3 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).

53.6.10 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)(x = 1, 8)

Address offset: 0x01C

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 3 in input capture mode and channel 4 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M[3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4 CE	OC4M[2:0]			OC4 PE	OC4 FE	CC4S[1:0]		OC3 CE	OC3M[2:0]			OC3 PE	OC3 FE	CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC4CE**: Output compare 4 clear enable

Bits 24, 14:12 **OC4M[3:0]**: Output compare 4 mode
Refer to OC3M[3:0] bit description

Bit 11 **OC4PE**: Output compare 4 preload enable

Bit 10 **OC4FE**: Output compare 4 fast enable

Bits 9:8 **CC4S[1:0]**: Capture/compare 4 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti4
10: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti3
11: CC4 channel is configured as input, tim_ic4 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).

Bit 7 **OC3CE:** Output compare 3 clear enable

Bits 16, 6:4 **OC3M[3:0]**: Output compare 3 mode

These bits define the behavior of the output reference signal tim_oc3ref from which tim_oc3 and tim_oc3n are derived. tim_oc3ref is active high whereas tim_oc3 and tim_oc3n active level depends on CC3P and CC3NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR3 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base).

0001: Set channel 3 to active level on match. tim_oc3ref signal is forced high when the counter TIMx_CNT matches the capture/compare register 3 (TIMx_CCR3).

0010: Set channel 3 to inactive level on match. tim_oc3ref signal is forced low when the counter TIMx_CNT matches the capture/compare register 3 (TIMx_CCR3).

0011: Toggle - tim_oc3ref toggles when TIMx_CNT = TIMx_CCR3.

0100: Force inactive level - tim_oc3ref is forced low.

0101: Force active level - tim_oc3ref is forced high.

0110: PWM mode 1 - In upcounting, channel 3 is active as long as TIMx_CNT<TIMx_CCR3 else inactive. In downcounting, channel 3 is inactive (tim_oc3ref = 0) as long as TIMx_CNT>TIMx_CCR3 else active (tim_oc3ref = 1).

0111: PWM mode 2 - In upcounting, channel 3 is inactive as long as TIMx_CNT<TIMx_CCR3 else active. In downcounting, channel 3 is active as long as TIMx_CNT>TIMx_CCR3 else inactive.

1000: Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Pulse on compare: a pulse is generated on tim_oc3ref upon CCR3 match event, as per PWPRSC[2:0] and PW[7:0] bitfields programming in TIMxECR.

1011: Direction output. The tim_oc3ref signal is overridden by a copy of the DIR bit.

1100: Combined PWM mode 1 - tim_oc3ref has the same behavior as in PWM mode 1. tim_oc3refc is the logical OR between tim_oc3ref and tim_oc4ref.

1101: Combined PWM mode 2 - tim_oc3ref has the same behavior as in PWM mode 2. tim_oc3refc is the logical AND between tim_oc3ref and tim_oc4ref.

1110: Asymmetric PWM mode 1 - tim_oc3ref has the same behavior as in PWM mode 1. tim_oc3refc outputs tim_oc3ref when the counter is counting up, tim_oc4ref when it is counting down.

1111: Asymmetric PWM mode 2 - tim_oc3ref has the same behavior as in PWM mode 2. tim_oc3refc outputs tim_oc3ref when the counter is counting up, tim_oc4ref when it is counting down.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output).

Note: In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

On channels having a complementary output, this bitfield is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the OC3M active bits take the new value from the preloaded bits only when a COM event is generated.

Bit 3 **OC3PE**: Output compare 3 preload enable

Bit 2 **OC3FE**: Output compare 3 fast enable

Bits 1:0 **CC3S[1:0]**: Capture/compare 3 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC3 channel is configured as output

01: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti3

10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4

11: CC3 channel is configured as input, tim_ic3 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).

53.6.11 TIMx capture/compare enable register (TIMx_CCER)(x = 1, 8)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC6P	CC6E	Res.	Res.	CC5P	CC5E
										rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	CC4NE	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **CC6P**: Capture/compare 6 output polarity
Refer to CC1P description

Bit 20 **CC6E**: Capture/compare 6 output enable
Refer to CC1E description

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CC5P**: Capture/compare 5 output polarity
Refer to CC1P description

Bit 16 **CC5E**: Capture/compare 5 output enable
Refer to CC1E description

Bit 15 **CC4NP**: Capture/compare 4 complementary output polarity
Refer to CC1NP description

Bit 14 **CC4NE**: Capture/compare 4 complementary output enable
Refer to CC1NE description

Bit 13 **CC4P**: Capture/compare 4 output polarity
Refer to CC1P description

Bit 12 **CC4E**: Capture/compare 4 output enable
Refer to CC1E description

Bit 11 **CC3NP**: Capture/compare 3 complementary output polarity
Refer to CC1NP description

Bit 10 **CC3NE**: Capture/compare 3 complementary output enable
Refer to CC1NE description

Bit 9 **CC3P**: Capture/compare 3 output polarity
Refer to CC1P description

Bit 8 **CC3E**: Capture/compare 3 output enable
Refer to CC1E description

Bit 7 **CC2NP**: Capture/compare 2 complementary output polarity
Refer to CC1NP description

Bit 6 **CC2NE**: Capture/compare 2 complementary output enable
Refer to CC1NE description

Bit 5 **CC2P**: Capture/compare 2 output polarity
Refer to CC1P description

Bit 4 **CC2E**: Capture/compare 2 output enable
Refer to CC1E description

Bit 3 **CC1NP**: Capture/compare 1 complementary output polarity
CC1 channel configured as output:
0: tim_oc1n active high.
1: tim_oc1n active low.
CC1 channel configured as input:
This bit is used in conjunction with CC1P to define the polarity of tim_ti1fp1 and tim_ti2fp1.
Refer to CC1P description.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (channel configured as output).

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 2 **CC1NE**: Capture/compare 1 complementary output enable
0: Off - tim_oc1n is not active. tim_oc1n level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.
1: On - tim_oc1n signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NE active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 1 **CC1P**: Capture/compare 1 output polarity

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP = 0, CC1P = 0:non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).

CC1NP = 0, CC1P = 1:inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).

CC1NP = 1, CC1P = 1: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

CC1NP = 1, CC1P = 0:the configuration is reserved, it must not be used.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0 **CC1E**: Capture/compare 1 output enable

0: Capture mode disabled / OC1 is not active (see below)

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

When CC1 channel is configured as output, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to *Table 500* for details.

Note: On channels having a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1E active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Table 500. Output control bits for complementary tim_ocx and tim_ocxn channels with break feature

Control bits					Output states[1]	
MOE bit	OSSI bit	OSSR bit	CCxE bit	CCxNE bit	tim_ocx output state	tim_ocxn output state
1	X	X	0	0	Output disabled (not driven by the timer: Hi-Z) tim_ocx = 0, tim_ocxn = 0	
		0	0	1	Output disabled (not driven by the timer: Hi-Z) tim_ocx = 0	tim_ocxref + Polarity tim_ocxn = tim_ocxref xor CCxNP
		0	1	0	tim_ocxref + Polarity tim_ocx = tim_ocxref xor CCxP	Output Disabled (not driven by the timer: Hi-Z) tim_ocxn = 0
		X	1	1	OCREF + Polarity + dead-time	Complementary to OCREF (not OCREF) + Polarity + dead-time
		1	0	1	Off-State (output enabled with inactive state) tim_ocx = CCxP	tim_ocxref + Polarity tim_ocxn = tim_ocxref x or CCxNP
		1	1	0	tim_ocxref + Polarity tim_ocx = tim_ocxref xor CCxP	Off-State (output enabled with inactive state) tim_ocxn = CCxNP
0	0	X	X	X	Output disabled (not driven by the timer: Hi-Z).	
	1		0	0		
			0	1	Off-State (output enabled with inactive state) Asynchronously: tim_ocx = CCxP, tim_ocxn = CCxNP (if tim_brk or tim_brk2 is triggered). Then (this is valid only if tim_brk is triggered), if the clock is present: tim_ocx = OISx and tim_ocxn = OISxN after a dead-time, assuming that OISx and OISxN do not correspond to OCX and tim_ocxn both in active state (may cause a short circuit when driving switches in half-bridge configuration). *Note: tim_brk2 can only be used if OSSI = OSSR = 1.*	
			1	0		
			1	1		

1. When both outputs of a channel are not used (control taken over by GPIO), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary tim_ocx and tim_ocxn channels depends on the tim_ocx and tim_ocxn channel state and the GPIO registers.

53.6.12 TIMx counter (TIMx_CNT)(x = 1, 8)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF copy

This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in the TIMxCR1 is reset, bit 31 is reserved and read at 0.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

Non-dithering mode (DITHEN = 0)

The register holds the counter value.

Dithering mode (DITHEN = 1)

The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

53.6.13 TIMx prescaler (TIMx_PSC)(x = 1, 8)

Address offset: 0x028

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency ($f_{tim_cnt_ck}$) is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

53.6.14 TIMx autoreload register (TIMx_ARR)(x = 1, 8)

Address offset: 0x02C

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Autoreload value

ARR is the value to be loaded in the actual autoreload register.

Refer to the *Section 53.3.3: Time-base unit* for more details about ARR update and behavior.

The counter is blocked while the autoreload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the autoreload value.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

53.6.15 TIMx repetition counter register (TIMx_RCR)(x = 1, 8)

Address offset: 0x030

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **REP[15:0]**: Repetition counter reload value

This bitfield defines the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable. It also defines the update interrupt generation rate, if this interrupt is enable.

When the repetition down-counter reaches zero, an update event is generated and it restarts counting from REP value. As the repetition counter is reloaded with REP value only at the repetition update event UEV, any write to the TIMx_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to:

– the number of PWM periods in edge-aligned mode

– the number of half PWM period in center-aligned mode.

53.6.16 TIMx capture/compare register 1 (TIMx_CCR1)(x = 1, 8)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR1[19:0]**: Capture/compare 1 value

If channel CC1 is configured as output: CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.

If channel CC1 is configured as input: CR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIMx_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

53.6.17 TIMx capture/compare register 2 (TIMx_CCR2)(x = 1, 8)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR2[19:0]**: Capture/compare 2 value

If channel CC2 is configured as output: CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc2 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR2[19:4]. The CCR2[3:0] bitfield contains the dithered part.

If channel CC2 is configured as input: CCR2 is the counter value transferred by the last input capture 2 event (tim_ic2). The TIMx_CCR2 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR2[19:4]. The CCR2[3:0] bits are reset.

53.6.18 TIMx capture/compare register 3 (TIMx_CCR3)(x = 1, 8)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR3[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR3[19:0]**: Capture/compare value

If channel CC3 is configured as output: CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc3 output.
Non-dithering mode (DITHEN = 0)
The register holds the compare value in CCR3[15:0]. The CCR3[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in CCR3[19:4]. The CCR3[3:0] bitfield contains the dithered part.
If channel CC3 is configured as input: CCR3 is the counter value transferred by the last input capture 3 event (tim_ic3). The TIMx_CCR3 register is read-only and cannot be programmed.
Non-dithering mode (DITHEN = 0)
The register holds the capture value in CCR3[15:0]. The CCR3[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the capture in CCR3[19:4]. The CCR3[3:0] bits are reset.

53.6.19 TIMx capture/compare register 4 (TIMx_CCR4)(x = 1, 8)

Address offset: 0x040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR4[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR4[19:0]**: Capture/compare value

If channel CC4 is configured as output: CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on tim_oc4 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR4[15:0]. The CCR4[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR4[19:4]. The CCR4[3:0] bitfield contains the dithered part.

If channel CC4 is configured as input: CCR4 is the counter value transferred by the last input capture 4 event (tim_ic4). The TIMx_CCR4 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR4[15:0]. The CCR4[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR4[19:4]. The CCR4[3:0] bits are reset.

53.6.20 TIMx break and dead-time register (TIMx_BDTR)(x = 1, 8)

Address offset: 0x044

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	BK2BID	BKBID	BK2 DSRM	BK DSRM	BK2P	BK2E	BK2F[3:0]				BKF[3:0]			
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the bits BKBID/BK2BID/BK2P, BK2E, BK2F[3:0], BKF[3:0], AOE, BKP, BKE, OSSI, OSSR, and DTG[7:0] can be write-locked depending on the LOCK configuration, it can be necessary to configure all of them during the first write access to the TIMx_BDTR register.

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **BK2BID**: Break2 bidirectional
Refer to BKBID description

Bit 28 **BKBID**: Break bidirectional
0: Break input tim_brk in input mode
1: Break input tim_brk in bidirectional mode
In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.
Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).
Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 27 **BK2DSRM**: Break2 disarm
Refer to BKDSRM description

Bit 26 **BKDSRM**: Break disarm
0: Break input tim_brk is armed
1: Break input tim_brk is disarmed
This bit is cleared by hardware when no break source is active.
The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled it until it is reset by hardware, indicating that the fault condition has disappeared.
Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 25 **BK2P**: Break 2 polarity
0: Break input tim_brk2 is active low
1: Break input tim_brk2 is active high
Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).
Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 24 **BK2E**: Break 2 enable
This bit enables the complete break 2 protection, see *Figure 547: Break and Break2 circuitry overview*).
0: Break2 function disabled
1: Break2 function enabled
Note: The BRKIN2 must only be used with OSSR = OSSI = 1.
Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).
Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bits 23:20 **BK2F[3:0]**: Break 2 filter

This bitfield defines the frequency used to sample tim_brk2 input and the length of the digital filter applied to tim_brk2. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, tim_brk2 acts asynchronously
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 19:16 **BKF[3:0]**: Break filter

This bitfield defines the frequency used to sample tim_brk input and the length of the digital filter applied to tim_brk. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, tim_brk acts asynchronously
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as one of the break inputs is active (tim_brk or tim_brk2). It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: In response to a break 2 event. OC and OCN outputs are disabled
In response to a break event or if MOE is written to 0: OC and OCN outputs are disabled or forced to idle state depending on the OSSI bit.

1: OC and OCN outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIMx_CCER register).
See OC/OCN enable description for more details (*Section 53.6.11: TIMx capture/compare enable register (TIMx_CCER)(x = 1, 8)*).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if none of the break inputs tim_brk and tim_brk2 is active)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input tim_brk is active low

1: Break input tim_brk is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable

This bit enables the complete break protection (including all sources connected to tim_sys_brk and BKIN sources, as per *Figure 547: Break and Break2 circuitry overview*).

0: Break function disabled

1: Break function enabled

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE = 1 on channels having a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See OC/OCN enable description for more details (*Section 53.6.11: TIMx capture/compare enable register (TIMx_CCER)(x = 1, 8)*).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO logic, which forces a Hi-Z state).

1: When inactive, OC/OCN outputs are enabled with their inactive level as soon as CCxE = 1 or CCxNE = 1 (the output is still controlled by the timer).

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **OSSI**: Off-state selection for idle mode

This bit is used when MOE = 0 due to a break event or by a software write, on channels configured as outputs.
See OC/OCN enable description for more details (*Section 53.6.11: TIMx capture/compare enable register (TIMx_CCER)(x = 1, 8)*).

0: When inactive, OC/OCN outputs are disabled (the timer releases the output control which is taken over by the GPIO logic and which imposes a Hi-Z state).

1: When inactive, OC/OCN outputs are first forced with their inactive level then forced to their idle level after the deadtime. The timer maintains its control over the output.

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected.

01: LOCK Level 1 = DTG bits in TIMx_BDTR register, OISx and OISxN bits in TIMx_CR2 register and BKBID/BK2BID/BKE/BKP/AOE bits in TIMx_BDTR register can no longer be written.

10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.

11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

Note: The LOCK bits can be written only once after the reset. Once the TIMx_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bitfield defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.
DTG[7:5] = 0xx => DT = DTG[7:0]x t_{dtg} with $t_{dtg} = t_{DTS}$.
DTG[7:5] = 10x => DT = (64+DTG[5:0])xt_{dtg} with $T_{dtg} = 2xt_{DTS}$.
DTG[7:5] = 110 => DT = (32+DTG[4:0])xt_{dtg} with $T_{dtg} = 8xt_{DTS}$.
DTG[7:5] = 111 => DT = (32+DTG[4:0])xt_{dtg} with $T_{dtg} = 16xt_{DTS}$.
Example if T_{DTS} = 125 ns (8 MHz), dead-time possible values are:
0 to 15875 ns by 125 ns steps,
16 µs to 31750 ns by 250 ns steps,
32 µs to 63 µs by 1 µs steps,
64 µs to 126 µs by 2 µs steps

Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

53.6.21 TIMx capture/compare register 5 (TIMx_CCR5)(x = 1, 8)

Address offset: 0x048

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GC5C3	GC5C2	GC5C1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR5[19:16]			
rw	rw	rw										rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR5[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **GC5C3**: Group channel 5 and channel 3
Distortion on channel 3 output:
0: No effect of tim_oc5ref on tim_oc3refc
1: tim_oc3refc is the logical AND of tim_oc3ref and tim_oc5ref
This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR2).
Note: it is also possible to apply this distortion on combined PWM signals.

Bit 30 **GC5C2**: Group channel 5 and channel 2
Distortion on channel 2 output:
0: No effect of tim_oc5ref on tim_oc2refc
1: tim_oc2refc is the logical AND of tim_oc2ref and tim_oc5ref
This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR1).
Note: it is also possible to apply this distortion on combined PWM signals.

Bit 29 **GC5C1**: Group channel 5 and channel 1
Distortion on channel 1 output:
0: No effect of oc5ref on oc1refc
1: oc1refc is the logical AND of oc1ref and oc5ref
This bit can either have immediate effect or be preloaded and taken into account after an update event (if preload feature is selected in TIMxCCMR1).
Note: it is also possible to apply this distortion on combined PWM signals.

Bits 28:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR5[19:0]**: Capture/compare 5 value
CCR5 is the value to be loaded in the actual capture/compare 5 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC5PE). Else the preload value is copied in the active capture/compare 5 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc5 output.
Non-dithering mode (DITHEN = 0)
The register holds the compare value in CCR5[15:0]. The CCR5[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in CCR5[19:4]. The CCR5[3:0] bitfield contains the dithered part.

53.6.22 TIMx capture/compare register 6 (TIMx_CCR6)(x = 1, 8)

Address offset: 0x04C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR6[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR6[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR6[19:0]**: Capture/compare 6 value

CCR6 is the value to be loaded in the actual capture/compare 6 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC6PE). Else the preload value is copied in the active capture/compare 6 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc6 output.
Non-dithering mode (DITHEN = 0)
The register holds the compare value in CCR6[15:0]. The CCR6[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in CCR6[19:4]. The CCR6[3:0] bitfield contains the dithered part.

53.6.23 TIMx capture/compare mode register 3 (TIMx_CCMR3) (x = 1, 8)

Address offset: 0x050

Reset value: 0x0000 0000

Refer to the above CCMR1 register description. Channels 5 and 6 can only be configured in output.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC6M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC5M[3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC6 CE	OC6M[2:0]			OC6 PE	OC6FE	Res.	Res.	OC5 CE	OC5M[2:0]			OC5PE	OC5FE	Res.	Res.
rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw		

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC6CE**: Output compare 6 clear enable

Bits 24, 14:12 **OC6M[3:0]**: Output compare 6 mode

Bit 11 **OC6PE**: Output compare 6 preload enable

Bit 10 **OC6FE**: Output compare 6 fast enable

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **OC5CE:** Output compare 5 clear enable

Bits 16, 6:4 **OC5M[3:0]**: Output compare 5 mode

Bit 3 **OC5PE**: Output compare 5 preload enable

Bit 2 **OC5FE**: Output compare 5 fast enable

Bits 1:0 Reserved, must be kept at reset value.

53.6.24 TIMx timer deadtime register 2 (TIMx_DTR2)(x = 1, 8)

Address offset: 0x054

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPE	DTAE
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTGF[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **DTPE**: Deadtime preload enable

0: Deadtime value is not preloaded

1: Deadtime value preload is enabled

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 16 **DTAE**: Deadtime asymmetric enable

0: Deadtime on rising and falling edges are identical, and defined with DTG[7:0] register

1: Deadtime on rising edge is defined with DTG[7:0] register and deadtime on falling edge is defined with DTGF[7:0] bits.

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **DTGF[7:0]**: Dead-time falling edge generator setup

This bitfield defines the duration of the dead-time inserted between the complementary outputs, on the falling edge.

DTGF[7:5] = 0xx => DTF = DTGF[7:0]x t_{dtg} with $t_{dtg} = t_{DTS}$.

DTGF[7:5] = 10x => DTF = (64+DTGF[5:0])xt_{dtg} with $T_{dtg} = 2xt_{DTS}$.

DTGF[7:5] = 110 => DTF = (32+DTGF[4:0])xt_{dtg} with $T_{dtg} = 8xt_{DTS}$.

DTGF[7:5] = 111 => DTF = (32+DTGF[4:0])xt_{dtg} with $T_{dtg} = 16xt_{DTS}$.

Example if T_{DTS} = 125 ns (8 MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

53.6.25 TIMx timer encoder control register (TIMx_ECR)(x = 1, 8)

Address offset: 0x058

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	PWPRSC[2:0]			PW[7:0]							
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPOS[1:0]		FIDX	IBLK[1:0]		IDIR[1:0]		IE
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:24 **PWPRSC[2:0]**: Pulse width prescaler

This bitfield sets the clock prescaler for the pulse generator, as following:

$t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim_ker_ck}$

Bits 23:16 **PW[7:0]**: Pulse width

This bitfield defines the pulse duration, as following:

$t_{PW} = PW[7:0] \times t_{PWG}$

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:6 **IPOS[1:0]**: Index positioning

In quadrature encoder mode (SMS[3:0] = 0001, 0010, 0011, 1110, 1111), this bit indicates in which AB input configuration the Index event resets the counter.

00: Index resets the counter when AB = 00
01: Index resets the counter when AB = 01
10: Index resets the counter when AB = 10
11: Index resets the counter when AB = 11

In directional clock mode or clock plus direction mode (SMS[3:0] = 1010, 1011, 1100, 1101), these bits indicates on which level the Index event resets the counter. In bidirectional clock mode, this applies for both clock inputs.

x0: Index resets the counter when clock is 0
x1: Index resets the counter when clock is 1

Note: IPOS[1] bit is not significant

Bit 5 **FIDX**: First index

This bit indicates if the first index only is taken into account

0: Index is always active
1: the first Index only resets the counter

Bits 4:3 **IBLK[1:0]**: Index blanking
This bit indicates if the Index event is conditioned by the tim_ti3 or tim_ti4 input
00: Index always active
01: Index disabled when tim_ti3 input is active, as per CC3P bitfield
10: Index disabled when tim_ti4 input is active, as per CC4P bitfield
11: Reserved

Bits 2:1 **IDIR[1:0]**: Index direction
This bit indicates in which direction the Index event resets the counter.
00: Index resets the counter whatever the direction
01: Index resets the counter when up-counting only
10: Index resets the counter when down-counting only
11: Reserved

Bit 0 **IE**: Index enable
This bit indicates if the Index event resets the counter.
0: Index disabled
1: Index enabled

53.6.26 TIMx timer input selection register (TIMx_TISEL)(x = 1, 8)

Address offset: 0x05C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **TI4SEL[3:0]**: Selects tim_ti4[15:0] input
0000: tim_ti4_in0: TIMx_CH4
0001: tim_ti4_in1
...
1111: tim_ti4_in15
Refer to *Section 53.3.2: TIM1/TIM8 pins and internal signals* for interconnects list.

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **TI3SEL[3:0]**: Selects tim_ti3[15:0] input
0000: tim_ti3_in0: TIMx_CH2
0001: tim_ti3_in1
...
1111: tim_ti3_in15
Refer to *Section 53.3.2: TIM1/TIM8 pins and internal signals* for interconnects list.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: Selects tim_ti2[15:0] input
0000: tim_ti2_in0: TIMx_CH2
0001: tim_ti2_in1
...
1111: tim_ti2_in15
Refer to *Section 53.3.2: TIM1/TIM8 pins and internal signals* for interconnects list.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: Selects tim_ti1[15:0] input
0000: tim_ti1_in0: TIMx_CH1
0001: tim_ti1_in1
...
1111: tim_ti1_in15
Refer to *Section 53.3.2: TIM1/TIM8 pins and internal signals* for interconnects list.

53.6.27 TIMx alternate function option register 1 (TIMx_AF1)(x = 1, 8)

Address offset: 0x060

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL[3:2]	
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETRSEL[1:0]		BK CMP4P	BK CMP3P	BK CMP2P	BK CMP1P	BKINP	BK CMP8E	BK CMP7E	BK CMP6E	BK CMP5E	BK CMP4E	BK CMP3E	BK CMP2E	BK CMP1E	BKINE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:14 **ETRSEL[3:0]**: etr_in source selection
These bits select the etr_in input source.
0000: tim_etr0: TIMx_ETR input
0001: tim_etr1
...
1111: tim_etr15
Refer to *Section 53.3.2: TIM1/TIM8 pins and internal signals* for product specific implementation.

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 13 **BKCMP4P**: tim_brk_cmp4 input polarity
This bit selects the tim_brk_cmp4 input sensitivity. It must be programmed together with the BKP polarity bit.
0: tim_brk_cmp4 input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)
1: tim_brk_cmp4 input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 12 **BKCMP3P**: tim_brk_cmp3 input polarity

This bit selects the tim_brk_cmp3 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp3 input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)

1: tim_brk_cmp3 input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 11 **BKCMP2P**: tim_brk_cmp2 input polarity

This bit selects the tim_brk_cmp2 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp2 input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)

1: tim_brk_cmp2 input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **BKCMP1P**: tim_brk_cmp1 input polarity

This bit selects the tim_brk_cmp1 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp1 input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)

1: tim_brk_cmp1 input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 9 **BKINP**: TIMx_BKIN input polarity

This bit selects the TIMx_BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: TIMx_BKIN input polarity is not inverted (active low if BKP = 0, active high if BKP = 1)

1: TIMx_BKIN input polarity is inverted (active high if BKP = 0, active low if BKP = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BKCMP8E**: tim_brk_cmp8 enable

This bit enables the tim_brk_cmp8 for the timer's tim_brk input. tim_brk_cmp8 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp8 input disabled

1: tim_brk_cmp8 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 7 **BKCMP7E**: tim_brk_cmp7 enable

This bit enables the tim_brk_cmp7 for the timer's tim_brk input. tim_brk_cmp7 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp7 input disabled

1: tim_brk_cmp7 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 6 **BKCMP6E**: tim_brk_cmp6 enable

This bit enables the tim_brk_cmp6 for the timer's tim_brk input. tim_brk_cmp6 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp6 input disabled

1: tim_brk_cmp6 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 5 **BKCMP5E**: tim_brk_cmp5 enable

This bit enables the tim_brk_cmp5 for the timer's tim_brk input. tim_brk_cmp5 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp5 input disabled

1: tim_brk_cmp5 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 4 **BKCMP4E**: tim_brk_cmp4 enable

This bit enables the tim_brk_cmp4 for the timer's tim_brk input. tim_brk_cmp4 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp4 input disabled

1: tim_brk_cmp4 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 3 **BKCMP3E**: tim_brk_cmp3 enable

This bit enables the tim_brk_cmp3 for the timer's tim_brk input. tim_brk_cmp3 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp3 input disabled

1: tim_brk_cmp3 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 2 **BKCMP2E**: tim_brk_cmp2 enable

This bit enables the tim_brk_cmp2 for the timer's tim_brk input. tim_brk_cmp2 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp2 input disabled

1: tim_brk_cmp2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 1 **BKCMP1E**: tim_brk_cmp1 enable

This bit enables the tim_brk_cmp1 for the timer's tim_brk input. tim_brk_cmp1 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp1 input disabled

1: tim_brk_cmp1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **BKINE**: TIMx_BKIN input enable

This bit enables the TIMx_BKIN alternate function input for the timer's tim_brk input. TIMx_BKIN input is 'ORed' with the other tim_brk sources.

0: TIMx_BKIN input disabled

1: TIMx_BKIN input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Refer to Section 53.3.2: TIM1/TIM8 pins and internal signals for product specific implementation.

53.6.28 TIMx alternate function register 2 (TIMx_AF2)(x = 1, 8)

Address offset: 0x064

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL[2:0]		
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	BK2CMP4P	BK2CMP3P	BK2CMP2P	BK2CMP1P	BK2INP	BK2CMP8E	BK2CMP7E	BK2CMP6E	BK2CMP5E	BK2CMP4E	BK2CMP3E	BK2CMP2E	BK2CMP1E	BK2INE
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **OCRSEL[2:0]**: ocref_clr source selection

These bits select the ocref_clr input source.

000: tim_ocref_clr0

001: tim_ocref_clr1

...

111: tim_ocref_clr7

Refer to *Section 53.3.2: TIM1/TIM8 pins and internal signals* for product specific information.

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **BK2CMP4P**: tim_brk2_cmp4 input polarity

This bit selects the tim_brk2_cmp4 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: tim_brk2_cmp4 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: tim_brk2_cmp4 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 12 **BK2CMP3P**: tim_brk2_cmp3 input polarity

This bit selects the tim_brk2_cmp3 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: tim_brk2_cmp3 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: tim_brk2_cmp3 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 11 **BK2CMP2P**: tim_brk2_cmp2 input polarity

This bit selects the tim_brk2_cmp2 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: tim_brk2_cmp2 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: tim_brk2_cmp2 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **BK2CMP1P**: tim_brk2_cmp1 input polarity

This bit selects the tim_brk2_cmp1 input sensitivity. It must be programmed together with the BK2P polarity bit.

0: tim_brk2_cmp1 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: tim_brk2_cmp1 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 9 **BK2INP**: TIMx_BKIN2 input polarity

This bit selects the TIMx_BKIN2 alternate function input sensitivity. It must be programmed together with the BK2P polarity bit.

0: TIMx_BKIN2 input polarity is not inverted (active low if BK2P = 0, active high if BK2P = 1)

1: TIMx_BKIN2 input polarity is inverted (active high if BK2P = 0, active low if BK2P = 1)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BK2CMP8E**: tim_brk2_cmp8 enable

This bit enables the tim_brk2_cmp8 for the timer's tim_brk2 input. tim_brk2_cmp8 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp8 input disabled

1: tim_brk2_cmp8 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 7 **BK2CMP7E**: tim_brk2_cmp7 enable

This bit enables the tim_brk2_cmp7 for the timer's tim_brk2 input. tim_brk2_cmp7 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp7 input disabled

1: tim_brk2_cmp7 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 6 **BK2CMP6E**: tim_brk2_cmp6 enable

This bit enables the tim_brk2_cmp6 for the timer's tim_brk2 input. tim_brk2_cmp6 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp6 input disabled

1: tim_brk2_cmp6 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 5 **BK2CMP5E**: tim_brk2_cmp5 enable

This bit enables the tim_brk2_cmp5 for the timer's tim_brk2 input. tim_brk2_cmp5 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp5 input disabled

1: tim_brk2_cmp5 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 4 **BK2CMP4E**: tim_brk2_cmp4 enable

This bit enables the tim_brk2_cmp4 for the timer's tim_brk2 input. tim_brk2_cmp4 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp4 input disabled

1: tim_brk2_cmp4 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 3 **BK2CMP3E**: tim_brk2_cmp3 enable

This bit enables the tim_brk2_cmp3 for the timer's tim_brk2 input. tim_brk2_cmp3 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp3 input disabled

1: tim_brk2_cmp3 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 2 **BK2CMP2E**: tim_brk2_cmp2 enable

This bit enables the tim_brk2_cmp2 for the timer's tim_brk2 input. tim_brk2_cmp2 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp2 input disabled

1: tim_brk2_cmp2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 1 **BK2CMP1E**: tim_brk2_cmp1 enable

This bit enables the tim_brk2_cmp1 for the timer's tim_brk2 input. tim_brk2_cmp1 output is 'ORed' with the other tim_brk2 sources.

0: tim_brk2_cmp1 input disabled

1: tim_brk2_cmp1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **BK2INE**: TIMx_BKIN2 input enable

This bit enables the TIMx_BKIN2 alternate function input for the timer's tim_brk2 input. TIMx_BKIN2 input is 'ORed' with the other tim_brk2 sources.

0: TIMx_BKIN2 input disabled

1: TIMx_BKIN2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Note: Refer to Section 53.3.2: TIM1/TIM8 pins and internal signals for product specific implementation.

53.6.29 TIMx DMA control register (TIMx_DCR)(x = 1, 8)

Address offset: 0x3DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **DBSS[3:0]**: DMA burst source selection

This bitfield defines the interrupt source that triggers the DMA burst transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address).

0000: Reserved
0001: Update
0010: CC1
0011: CC2
0100: CC3
0101: CC4
0110: COM
0111: Trigger
Others: reserved

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit vector defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).

00000: 1 transfer
00001: 2 transfers
00010: 3 transfers
...
11010: 26 transfers

Example: Let us consider the following transfer: DBL = 7 bytes & DBA = TIM2_CR1.

–If DBL = 7 bytes and DBA = TIM2_CR1 represents the address of the byte to be transferred, the address of the transfer is given by the following equation:

(TIMx_CR1 address) + DBA + (DMA index), where DMA index = DBL

In this example, 7 bytes are added to (TIMx_CR1 address) + DBA, which gives us the address from/to which the data are copied. In this case, the transfer is done to 7 registers starting from the following address: (TIMx_CR1 address) + DBA

According to the configuration of the DMA Data Size, several cases may occur:

–If the DMA Data Size is configured in half-words, 16-bit data are transferred to each of the 7 registers.

–If the DMA Data Size is configured in bytes, the data are also transferred to 7 registers: the first register contains the first MSB byte, the second register, the first LSB byte and so on. So with the transfer Timer, one also has to specify the size of data transferred by DMA.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bits vector defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:

00000: TIMx_CR1
00001: TIMx_CR2
00010: TIMx_SMCR
...

53.6.30 TIMx DMA address for full transfer (TIMx_DMAR)(x = 1, 8)

Address offset: 0x3E0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMAB[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address (TIMx_CR1 address) + (DBA + DMA index) x 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

53.6.31 TIMx register map

TIMx registers are mapped as 16-bit addressable registers as described in the table below:

Table 501. TIMx register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	TIMx_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DITHEN	UIFREMA	Res.	CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value																				0	0		0	0	0	0	0	0	0	0	0	0
0x004	TIMx_CR2	Res.	Res.	Res.	ADSYNC	Res.	Res.	MMS[3]	Res.	MMS2[3:0]				Res.	OIS6	Res.	OIS5	OIS4N	OIS4	OIS3N	OIS3	OIS2N	OIS2	OIS1N	OIS1	TI1S	MMS [2:0]			CCDS	CCUS	Res.	CCPC
	Reset value				0			0		0	0	0	0		0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
0x008	TIMx_SMCR	Res.	Res.	Res.	Res.	Res.	Res.	SMSPS	SMSPE	Res.	Res.	TS [4:3]		Res.	Res.	Res.	SMS[3]	ETP	ECE	ETP S [1:0]		ETF[3:0]			MSM	TS[2:0]			Res.	SMS[2:0]			
	Reset value							0	0			0	0				0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
0x00C	TIMx_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRIE	IERRIE	DIRIE	IDXIE	Res.	Res.	Res.	Res.	Res.	TDE	COMDE	CC4DE	CC3DE	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value									0	0	0	0						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	TIMx_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRF	IERRF	DIRF	IDXF	Res.	Res.	CC6IF	CC5IF	Res.	Res.	SBIF	CC4OF	CC3OF	CC2OF	CC1OF	B2IF	BIF	TIF	COMIF	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value									0	0	0	0			0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	TIMx_EGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	B2G	BG	TG	COMG	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																								0	0	0	0	0	0	0	0	0

Table 501. TIMx register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x018	TIMx_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2 PSC [1:0]		CC2 S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1 S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR1 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2 S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1 S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	TIMx_CCMR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M[3]	OC4CE	OC4M [2:0]			OC4PE	OC4FE	CC4 S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3 S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC4F[3:0]				IC4 PSC [1:0]		CC4 S [1:0]		IC3F[3:0]				IC3 PSC [1:0]		CC3 S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	TIMx_CCER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC6P	CC6E	Res.	Res.	CC5P	CC5E	CC4NP	CC4NE	CC4P	CC4E	CC3NP	CC3NE	CC3P	CC3E	CC2NP	CC2NE	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
	Reset value											0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	TIMx_CNT	UIFCPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	TIMx_PSC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	TIMx_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:0]																			
	Reset value													0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x030	TIMx_RCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	TIMx_CCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	TIMx_CCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	TIMx_CCR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR3[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x040	TIMx_CCR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR4[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	TIMx_BDTR	Res.	Res.	BK2BID	BKBID	BK2DSRM	BKDSRM	BK2P	BK2E	BK2F[3:0]				BKF[3:0]				MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]							
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 501. TIMx register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x048	TIMx_CCR5	GC5C3	GC5C2	GC5C1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR5[19:0]																			
	Reset value	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	TIMx_CCR6	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR6[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	TIMx_CCMR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC6M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC5M[3]	OC6CE	OC6M [2:0]			OC6PE	OC6FE	Res.	Res.	OC5CE	OC5M [2:0]			OC5PE	OC5FE	Res.	Res.
	Reset value								0								0	0	0	0	0	0	0			0	0	0	0	0	0		
0x054	TIMx_DTR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPE	DTAE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTGF[7:0]							
	Reset value															0	0									0	0	0	0	0	0	0	0
0x058	TIMx_ECR	Res.	Res.	Res.	Res.	Res.	PWPRSC[2:0]			PW[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPOS [1:0]		FIDX	IBLK [1:0]		IDIR [1:0]		IE
	Reset value						0	0	0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0
0x05C	TIMx_TISEL	Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]				Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value					0	0	0	0					0	0	0	0					0	0	0	0					0	0	0	0
0x060	TIMx_AF1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res ETRSEL [3:0]				BKCMP4P	BKCMP3P	BKCMP2P	BKCMP1P	BKINP	BKCMP8E	BKCMP7E	BKCMP6E	BKCMP5E	BKCMP4E	BKCMP3E	BKCMP2E	BKCMP1E	BKINE
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x064	TIMx_AF2	Res.	Res.	Res	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL [2:0]			Res.	Res.	BK2CMP4P	BK2CMP3P	BK2CMP2P	BK2CMP1P	BK2INP	BK2CMP8E	BK2CMP7E	BK2CMP6E	BK2CMP5E	BK2CMP4E	BK2CMP3E	BK2CMP2E	BK2CMP1E	BK2INE
	Reset value														0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x068.. 0x3D8	Reserved	Res.																															
0x3DC	TIMx_DCR	Res.	Res.	Res	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]				Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value													0	0	0	0				0	0	0	0	0				0	0	0	0	0
0x3E0	TIMx_DMAR	DMAB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

54 General-purpose timers (TIM2/TIM3/TIM4/TIM5)

54.1 TIM2/TIM3/TIM4/TIM5 introduction

The general-purpose timers consist of a 16-bit or 32-bit autoreload counter driven by a programmable prescaler.

They can be used for a variety of purposes, including measuring the pulse lengths of input signals (*input capture*) or generating output waveforms (*output compare and PWM*).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 54.4.23: Timer synchronization*.

54.2 TIM2/TIM3/TIM4/TIM5 main features

General-purpose TIMx timer features include:

- 16-bit or 32-bit up, down, up/down autoreload counter.
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535.
- Up to four independent channels for:
  - Input capture.
  - Output compare.
  - PWM generation (edge- and center-aligned modes).
  - One-pulse mode output.
- Synchronization circuit to control the timer with external signals and to interconnect several timers.
- Interrupt/DMA generation on the following events:
  - Update: counter overflow/underflow, counter initialization (by software or internal/external trigger).
  - Trigger event (counter start, stop, initialization, or count by internal/external trigger).
  - Input capture.
  - Output compare.
- Supports incremental (quadrature) encoder and hall-sensor circuitry for positioning purposes.
- Trigger input for external clock or cycle-by-cycle current management.
- ADC synchronization for jitter-free sampling points.

54.3 TIM2/TIM3/TIM4/TIM5 implementation

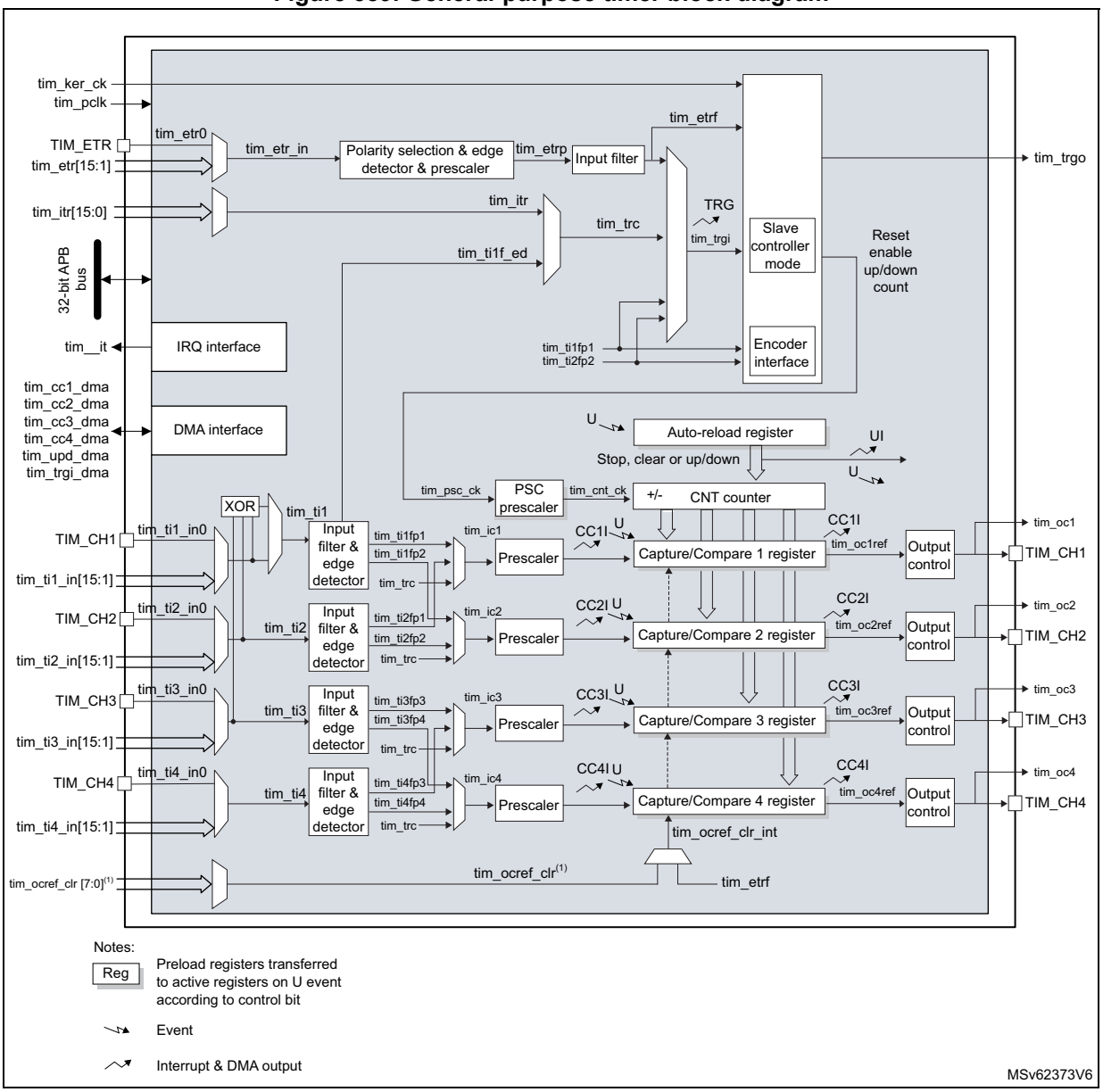
Table 502. STM32N6x5/x7xx general purpose timers

Timer instance	TIM2	TIM3	TIM4	TIM5
Resolution	32-bit	16-bit	32-bit	32-bit
OCREF clear selection	No	No	No	No
Sources	tim_etrf	tim_etrf	tim_etrf	tim_etrf

54.4 TIM2/TIM3/TIM4/TIM5 functional description

54.4.1 Block diagram

Figure 589. General-purpose timer block diagram



1. This feature is not available on all timers, refer to *Section 54.3: TIM2/TIM3/TIM4/TIM5 implementation*.

54.4.2 TIM2/TIM3/TIM4/TIM5 pins and internal signals

Table 503 and *Table 504* in this section summarize the TIM inputs and outputs.

Table 503. TIM input/output pins

Pin name	Signal type	Description
TIM_CH1 TIM_CH2 TIM_CH3 TIM_CH4	Input/Output	Timer multi-purpose channels. Each channel be used for capture, compare, or PWM. TIM_CH1 and TIM_CH2 can also be used as external clock (below 1/4 of the tim_ker_ck clock) , external trigger and quadrature encoder inputs. TIM_CH1, TIM_CH2 and TIM_CH3 can be used to interface with digital hall effect sensors.
TIM_ETR	Input	External trigger input. This input can be used as external trigger or as external clock source. This input can receive a clock with a frequency higher than the tim_ker_ck if the tim_etr_in prescaler is used.

Table 504. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_ti1_in[15:0] tim_ti2_in[15:0] tim_ti3_in[15:0] tim_ti4_in[15:0]	Input	Internal timer inputs bus. The tim_ti1_in[15:0] and tim_ti2_in[15:0] inputs can be used for capture or as external clock (below 1/4 of the tim_ker_ck clock) and for quadrature encoder signals.
tim_etr[15:0]	Input	External trigger internal input bus. These inputs can be used as trigger, external clock or for hardware cycle-by-cycle pulse width control. These inputs can receive clock with a frequency higher than the tim_ker_ck if the tim_etr_in prescaler is used.
tim_itr[15:0]	Input	Internal trigger input bus. These inputs can be used for the slave mode controller or as a input clock (below 1/4 of the tim_ker_ck clock).
tim_trgo	Output	Internal trigger output. This trigger can trigger other on-chip peripherals.
tim_ocref_clr[7:0]	Input	Timer tim_ocref_clr input bus. These inputs can be used to clear the tim_ocxref signals, typically for hardware cycle-by-cycle pulse width control.
tim_pclk	Input	Timer APB clock.
tim_ker_ck	Input	Timer kernel clock

Table 504. TIM internal input/output signals (continued)

Internal signal name	Signal type	Description
tim_it	Output	Global Timer interrupt, gathering capture/compare, update and break trigger requests.
tim_cc1_dma tim_cc2_dma tim_cc3_dma tim_cc4_dma	Output	Timer capture/compare [4:1] dma requests.
tim_upd_dma	Output	Timer update dma request.
tim_trgi_dma	Output	Timer trigger dma request.

Tables below list the sources connected to the tim_ti[4:1] input multiplexers.

Table 505. Interconnect to the tim_ti1 input multiplexer

tim_ti1 inputs	Sources			
	TIM2	TIM3	TIM4	TIM5
tim_ti1_in0	TIM2_CH1	TIM3_CH1	TIM4_CH1	TIM5_CH1
tim_ti1_in1	eth_ptp_pps_out	eth_ptp_pps_out	Reserved	Reserved
tim_ti1_in2	Reserved	fdcan_rtp	Reserved	fdcan_rtp
tim_ti1_in3		fdcan_tmp		fdcan_tmp
tim_ti1_in4		fdcan_soc		Reserved
tim_ti1_in[15:2]	Reserved			

Table 506. Interconnect to the tim_ti2 input multiplexer

tim_ti2 inputs	Sources			
	TIM2	TIM3	TIM4	TIM5
tim_ti2_in0	TIM2_CH2	TIM3_CH2	TIM4_CH2	TIM5_CH2
tim_ti2_in[15:1]	Reserved			

Table 507. Interconnect to the tim_ti3 input multiplexer

tim_ti3 inputs	Sources			
	TIM2	TIM3	TIM4	TIM5
tim_ti3_in0	TIM2_CH3	TIM3_CH3	TIM4_CH3	TIM5_CH3
tim_ti3_in[15:1]	Reserved			

Table 508. Interconnect to the tim_ti4 input multiplexer

tim_ti4 inputs	Sources			
	TIM2	TIM3	TIM4	TIM5
tim_ti4_in0	TIM2_CH4	TIM3_CH4	TIM4_CH4	TIM5_CH4
tim_ti4_in[15:1]	Reserved			

The table below lists the internal sources connected to the tim_etr input multiplexer.

Table 509. TIMx internal trigger connection

TIMx	TIM2	TIM3	TIM4	TIM5
tim_itr0	tim1_trgo	tim1_trgo	tim1_trgo	tim1_trgo
tim_itr1	Reserved	tim2_trgo	tim2_trgo	tim2_trgo
tim_itr2	tim3_trgo	Reserved	tim3_trgo	tim3_trgo
tim_itr3	tim4_trgo	tim4_trgo	Reserved	tim4_trgo
tim_itr4	tim5_trgo	tim5_trgo	tim5_trgo	Reserved
tim_itr5	tim8_trgo	tim8_trgo	tim8_trgo	tim8_trgo
tim_itr6	tim9_trgo	tim9_trgo	tim9_trgo	tim9_trgo
tim_itr7	tim12_trgo	tim12_trgo	tim12_trgo	tim12_trgo
tim_itr8	tim13_oc1	tim13_oc1	tim13_oc1	tim13_oc1
tim_itr9	tim14_oc1	tim14_oc1	tim14_oc1	tim14_oc1
tim_itr10	tim15_trgo	tim15_trgo	tim15_trgo	tim15_trgo
tim_itr11	tim16_oc1	tim16_oc1	tim16_oc1	tim16_oc1
tim_itr12	tim17_oc1	tim17_oc1	tim17_oc1	tim17_oc1
tim_itr13	Reserved			fdcan_soc
tim_itr[15:14]	Reserved			

The table below lists the internal sources connected to the tim_etr input multiplexer.

Table 510. Interconnect to the tim_etr input multiplexer

Timer external trigger input signal	Timer external trigger signal assignment			
	TIM2	TIM3	TIM4	TIM5
tim_etr0	TIM2_ETR	TIM3_ETR	TIM4_ETR	TIM5_ETR
tim_etr1	DCMIPP_HSYNC	DCMIPP_HSYNC	DCMIPP_HSYNC	sai2_fs_a
tim_etr2	LCD_HSYNC	LCD_HSYNC	LCD_HSYNC	sai2_fs_b
tim_etr3	Reserved	Reserved	Reserved	DCMIPP_HSYNC
tim_etr4	sai2_fs_a			LTDC_HSYNC
tim_etr5	sai2_fs_b			Reserved

Table 510. Interconnect to the tim_etr input multiplexer (continued)

Timer external trigger input signal	Timer external trigger signal assignment			
	TIM2	TIM3	TIM4	TIM5
tim_etr6	GFXTIM_TE	GFXTIM_TE	GFXTIM_TE	GFXTIM_TE
tim_etr7	DCMIPP_VSYNC	DCMIPP_VSYNC	DCMIPP_VSYNC	DCMIPP_VSYNC
tim_etr8	LCD_VSYNC	LCD_VSYNC	LCD_VSYNC	LCD_VSYNC
tim_etr9	Reserved	tim2_etr0	tim2_etr0	tim2_etr0
tim_etr10	tim3_etr0	Reserved	tim3_etr0	tim3_etr0
tim_etr11	tim4_etr0	tim4_etr0	Reserved	tim4_etr0
tim_etr12	tim5_etr0	tim5_etr0	tim5_etr0	Reserved
tim_etr13	eth_ptp_pps_out	eth_ptp_pps_out	Reserved	Reserved
tim_etr14	otg1_sof	Reserved		otg1_sof
tim_etr15	otg2_sof			otg2_sof

54.4.3 Time-base unit

The main block of the programmable timer is a 16-bit/32-bit counter with its related autoreload register. The counter can count up, down or both up and down. The counter clock can be divided by a prescaler.

The counter, the autoreload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Autoreload register (TIMx_ARR).

The autoreload register is preloaded. Writing to or reading from the autoreload register accesses the preload register. The content of the preload register is transferred into the shadow register permanently or at each update event (UEV), depending on the autoreload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow (or underflow when down-counting) and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the actual counter enable signal CNT_EN is set one clock cycle after CEN.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit/32-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 590 and *Figure 591* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 590. Counter timing diagram with prescaler division change from 1 to 2

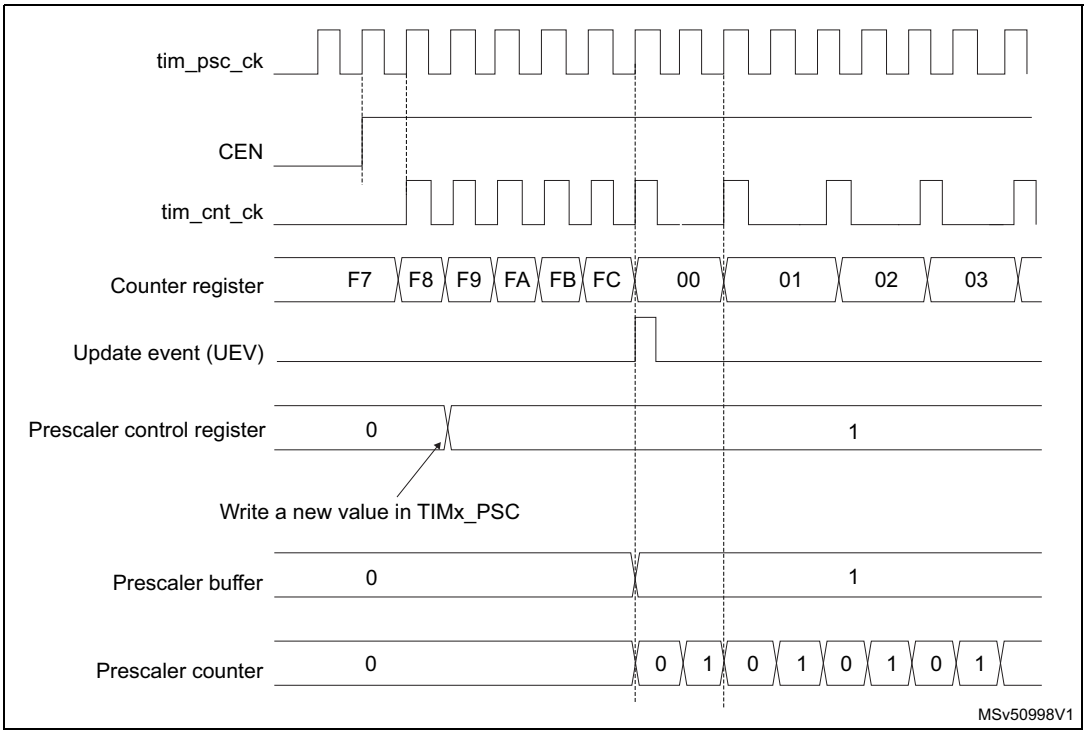
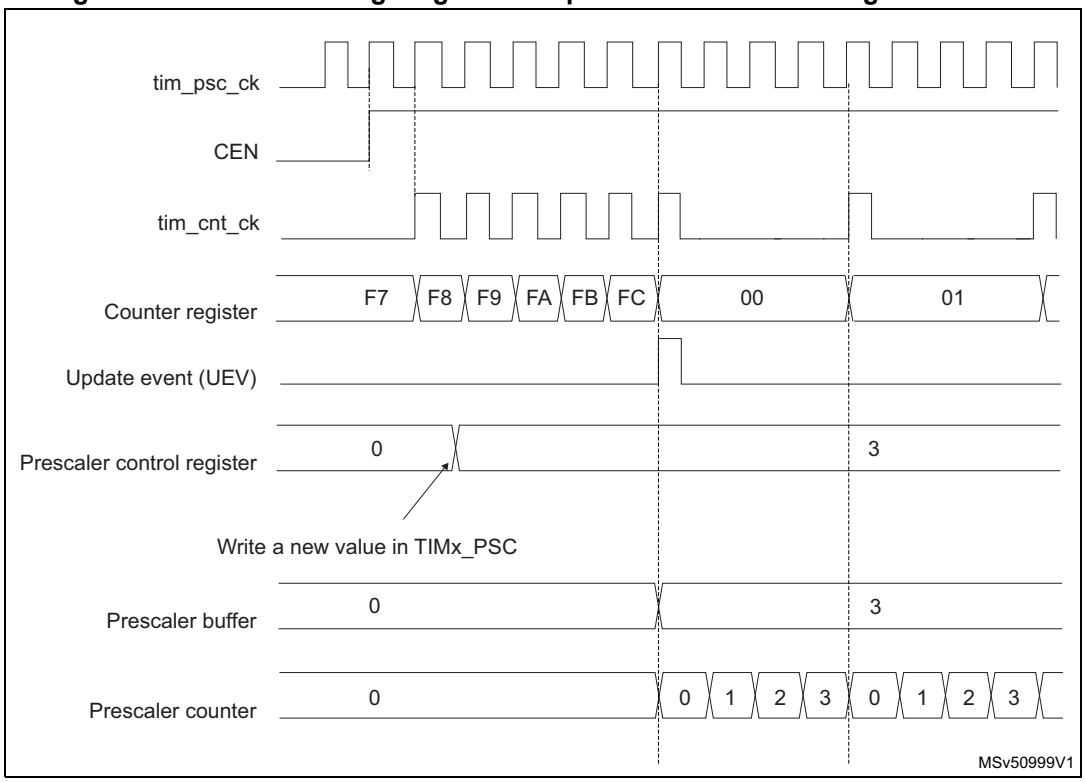


Figure 591. Counter timing diagram with prescaler division change from 1 to 4



54.4.4 Counter modes

Up-counting mode

In up-counting mode, the counter counts from 0 to the autoreload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

An update event can be generated at each counter overflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller).

The UEV event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The autoreload shadow register is updated with the preload value (TIMx_ARR).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.

Figure 592. Counter timing diagram, internal clock divided by 1

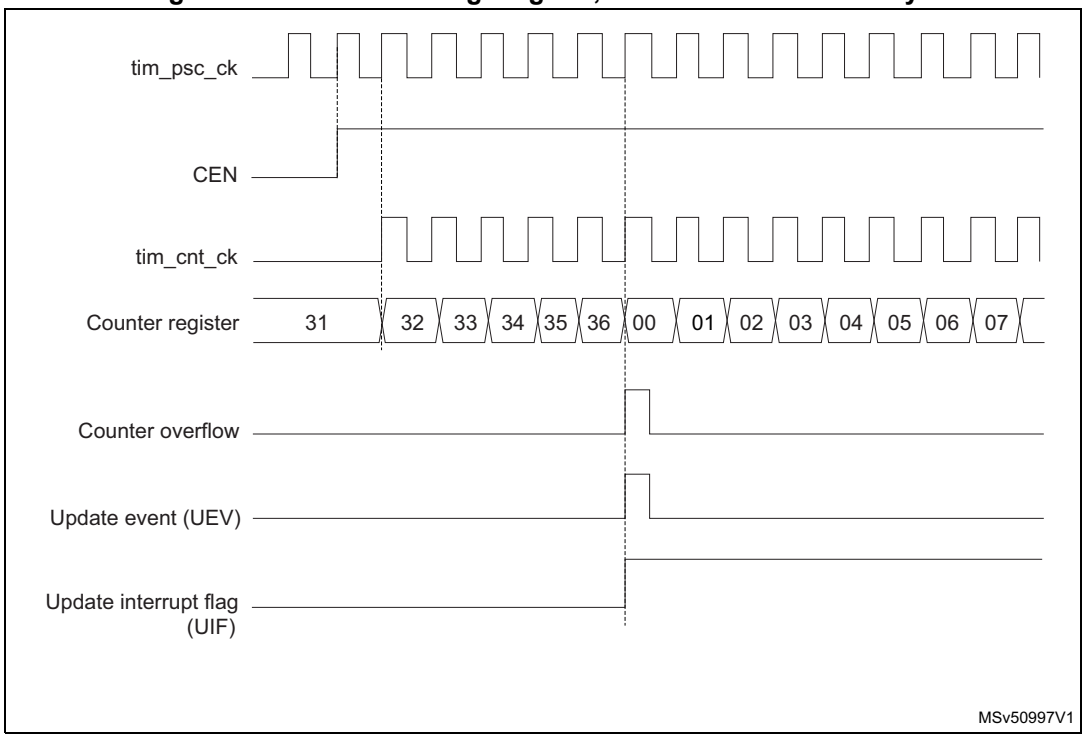


Figure 593. Counter timing diagram, internal clock divided by 2

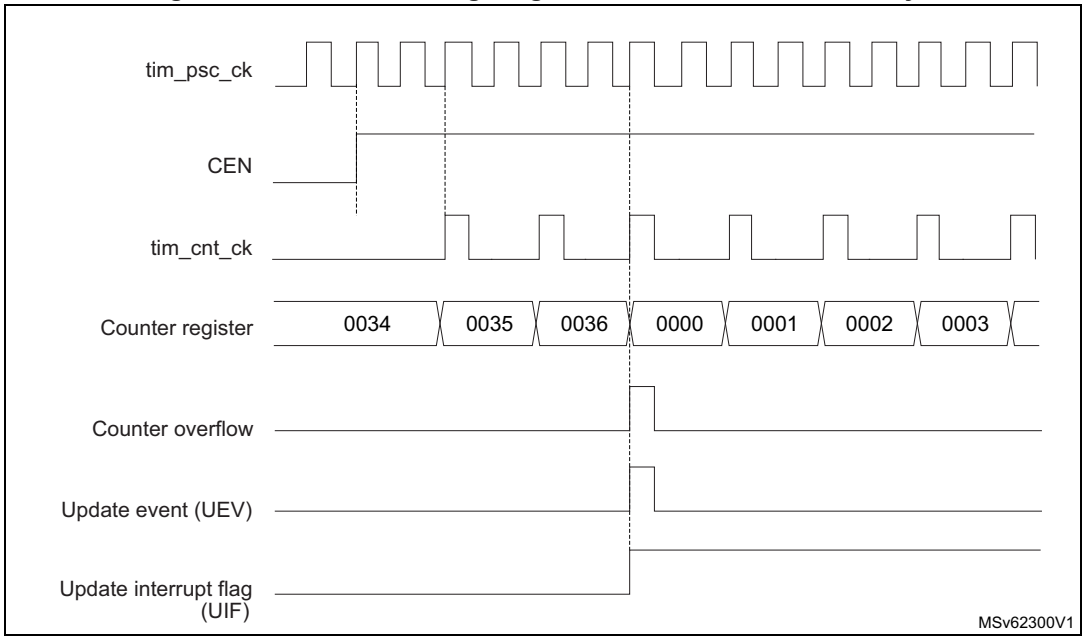


Figure 594. Counter timing diagram, internal clock divided by 4

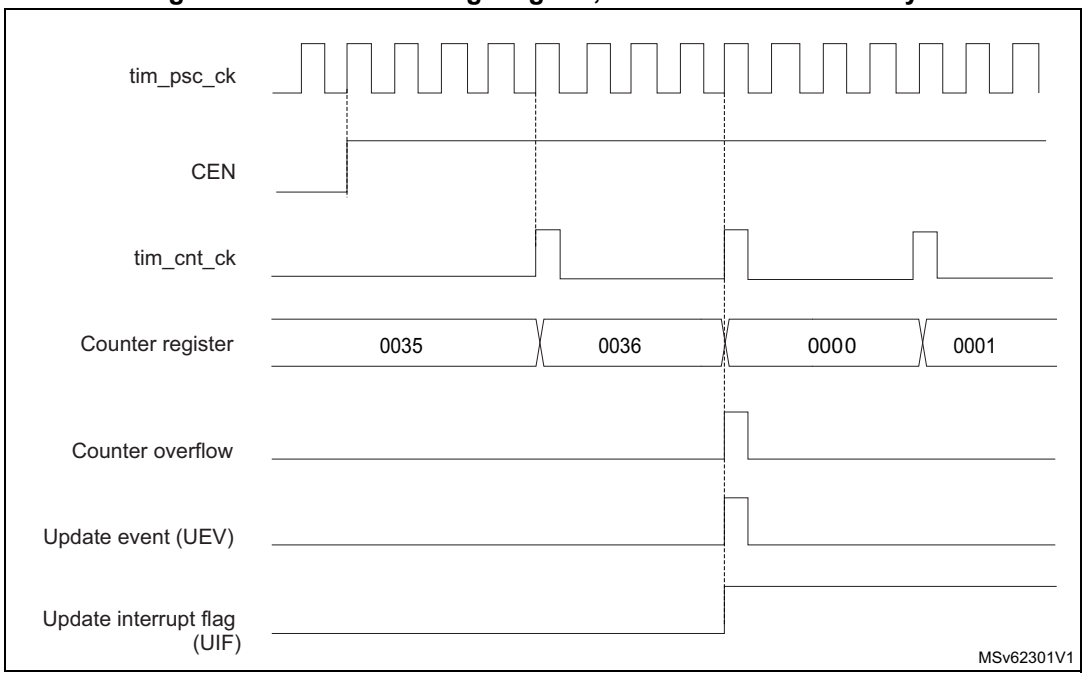


Figure 595. Counter timing diagram, internal clock divided by N

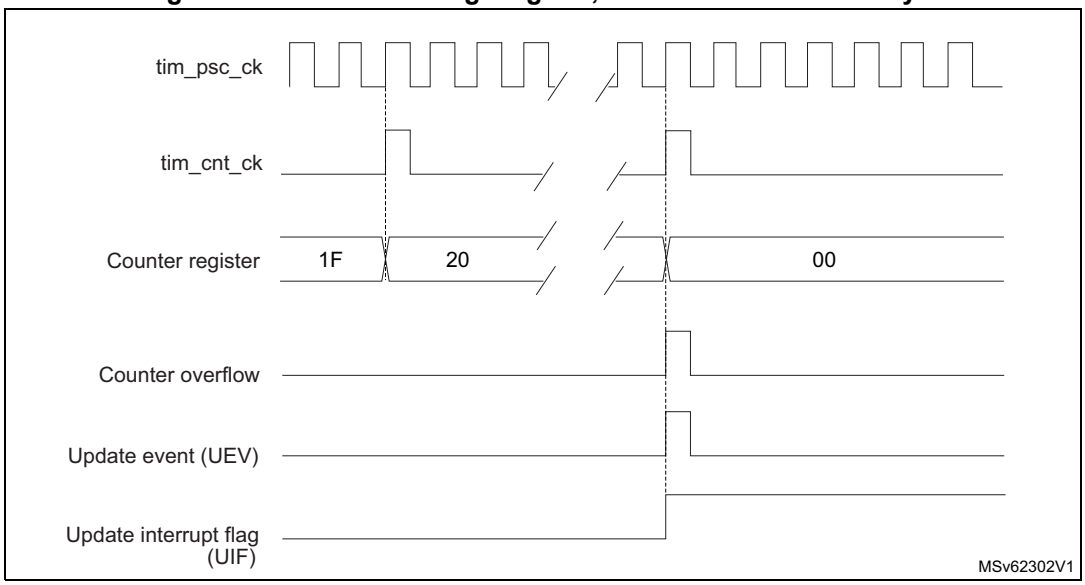


Figure 596. Counter timing diagram, Update event when ARPE = 0 (TIMx_ARR not preloaded)

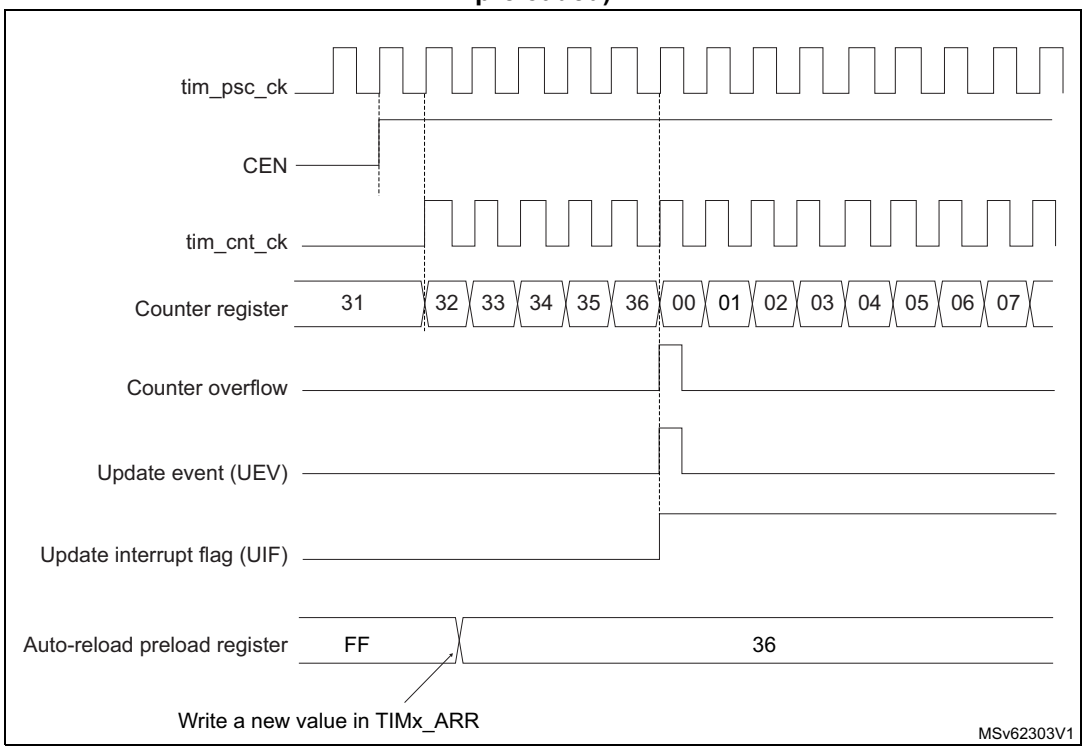
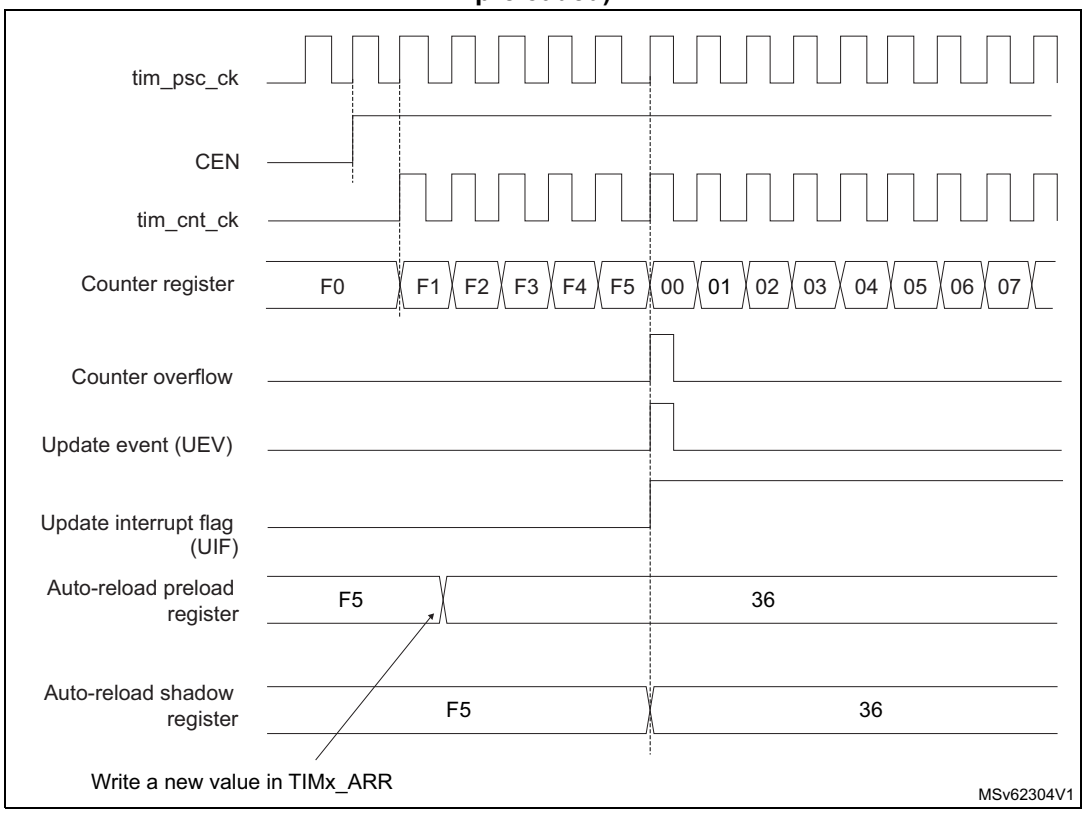


Figure 597. Counter timing diagram, Update event when ARPE = 1 (TIMx_ARR preloaded)



Down-counting mode

In down-counting mode, the counter counts from the autoreload value (content of the TIMx_ARR register) down to 0, then restarts from the autoreload value and generates a counter underflow event.

An update event can be generated at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller)

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until UDIS bit has been written to 0. However, the counter restarts from the current autoreload value, whereas the counter of the prescaler restarts from 0 (but the prescale rate does not change).

In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The autoreload active register is updated with the preload value (content of the TIMx_ARR register). Note that the autoreload is updated before the counter is reloaded, so that the next period is the expected one.

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.

Figure 598. Counter timing diagram, internal clock divided by 1

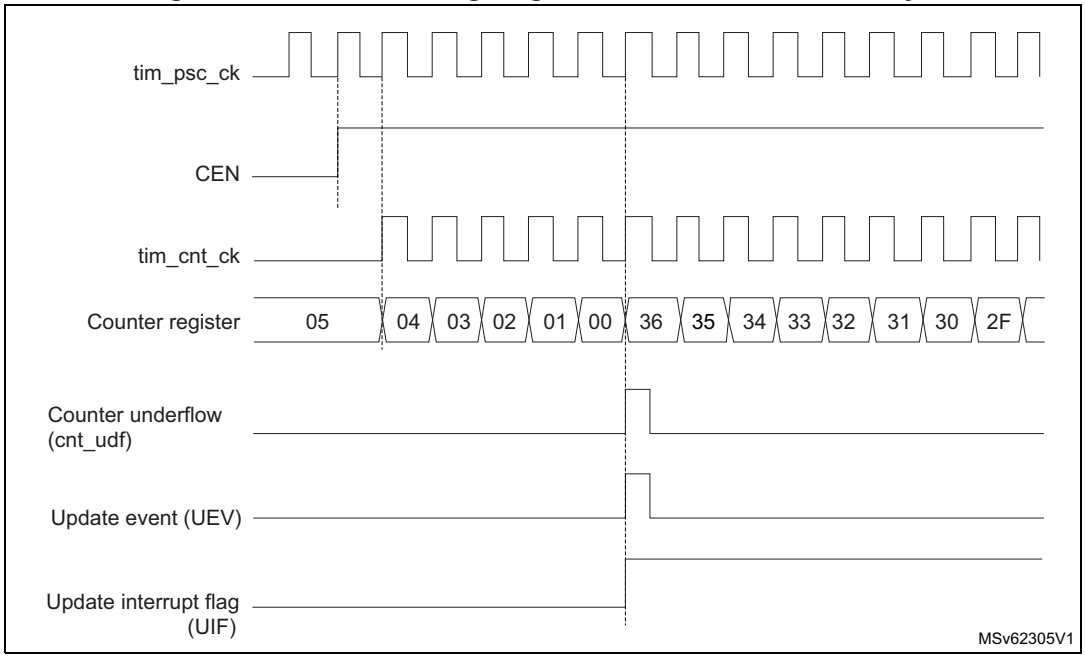


Figure 599. Counter timing diagram, internal clock divided by 2

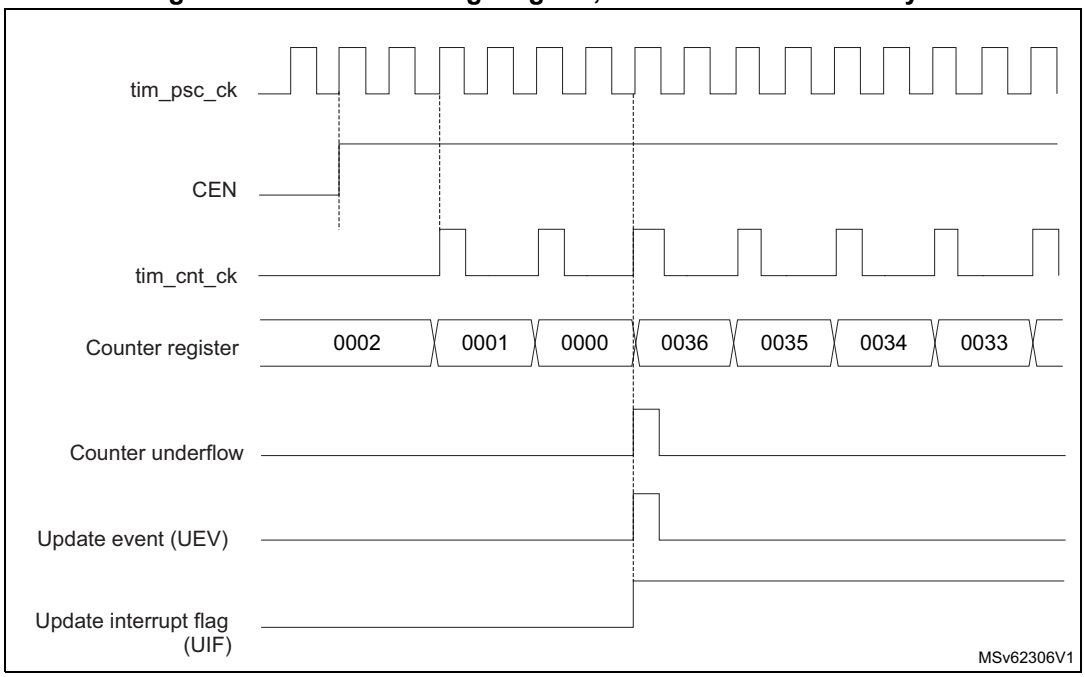


Figure 600. Counter timing diagram, internal clock divided by 4

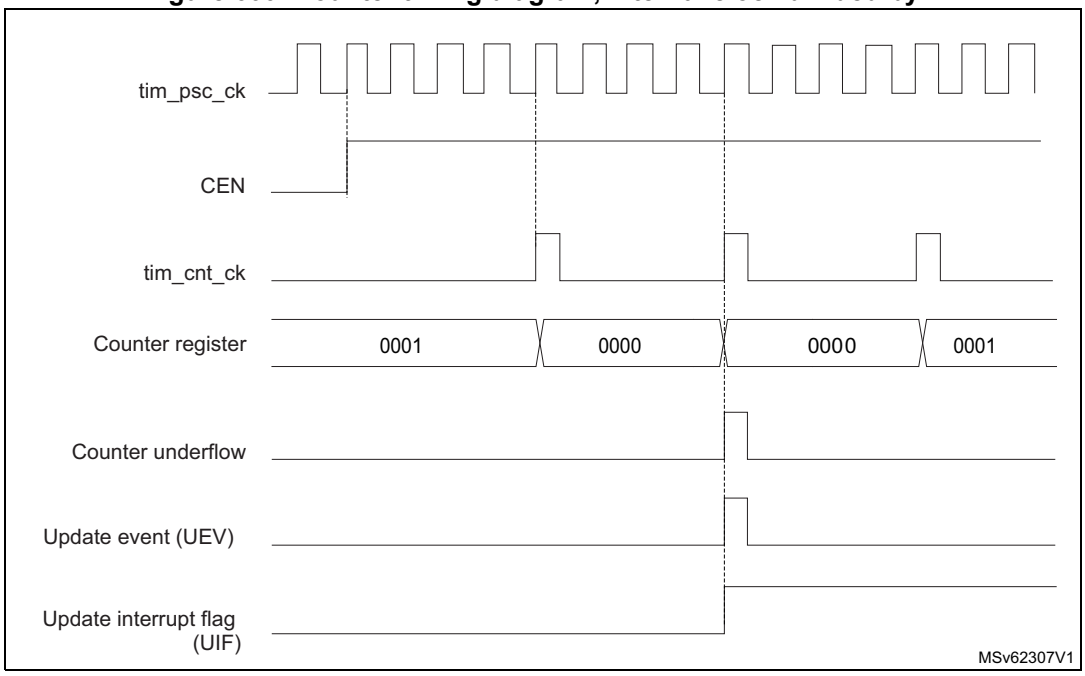


Figure 601. Counter timing diagram, internal clock divided by N

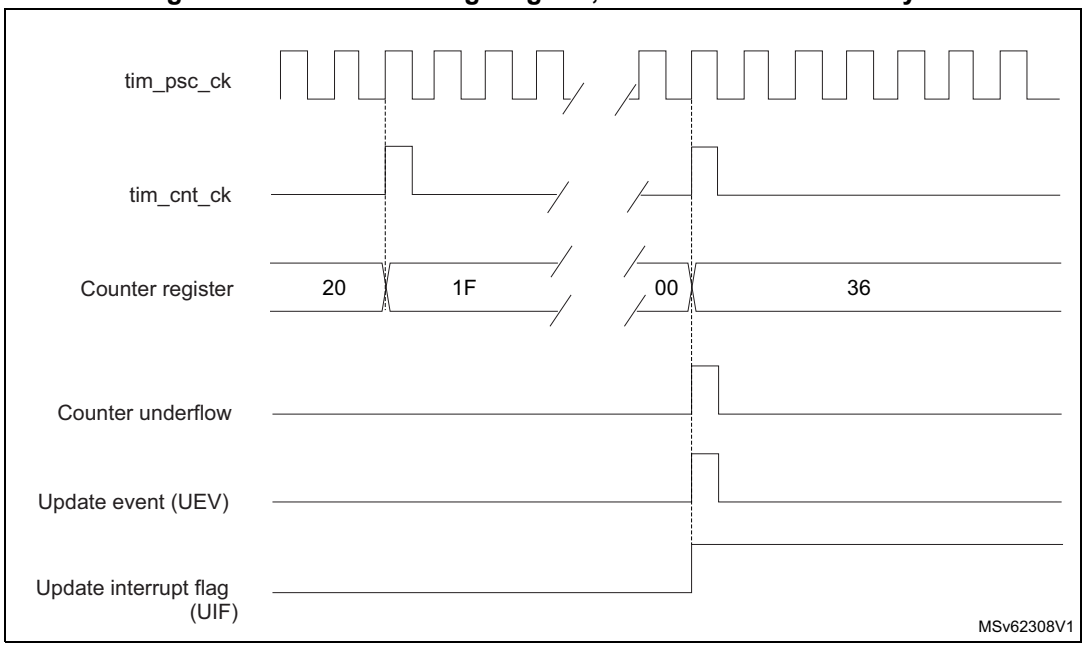
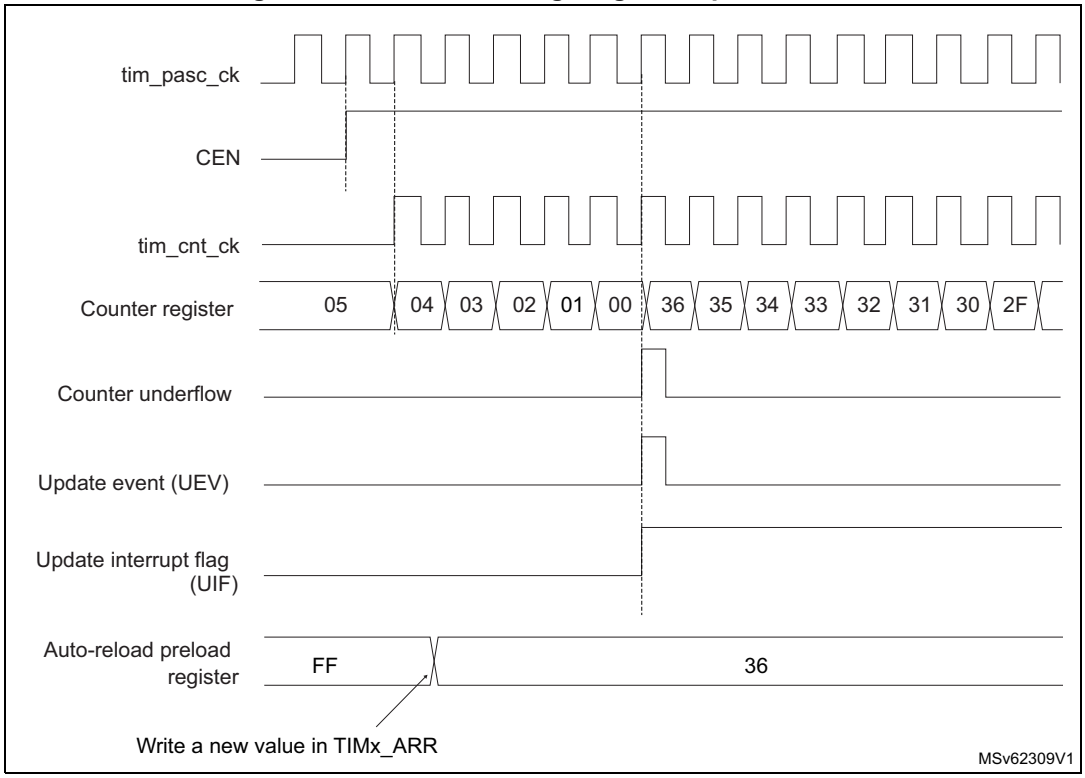


Figure 602. Counter timing diagram, Update event



Center-aligned mode (up/down-counting)

In center-aligned mode, the counter counts from 0 to the autoreload value (content of the TIMx_ARR register) – 1, generates a counter overflow event, then counts from the

autoreload value down to 1 and generates a counter underflow event. Then it restarts counting from 0.

Center-aligned mode is active when the CMS bits in TIMx_CR1 register are not equal to 00. The output compare interrupt flag of channels configured in output is set when: the counter counts down (Center aligned mode 1, CMS = 01), the counter counts up (Center aligned mode 2, CMS = 10) the counter counts up and down (Center aligned mode 3, CMS = 11).

In this mode, the direction bit (DIR from TIMx_CR1 register) cannot be written. It is updated by hardware and gives the current direction of the counter.

The update event can be generated at each counter overflow and at each counter underflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event. In this case, the counter restarts counting from 0, as well as the counter of the prescaler.

The UEV update event can be disabled by software by setting the UDIS bit in TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter continues counting up and down, based on the current autoreload value.

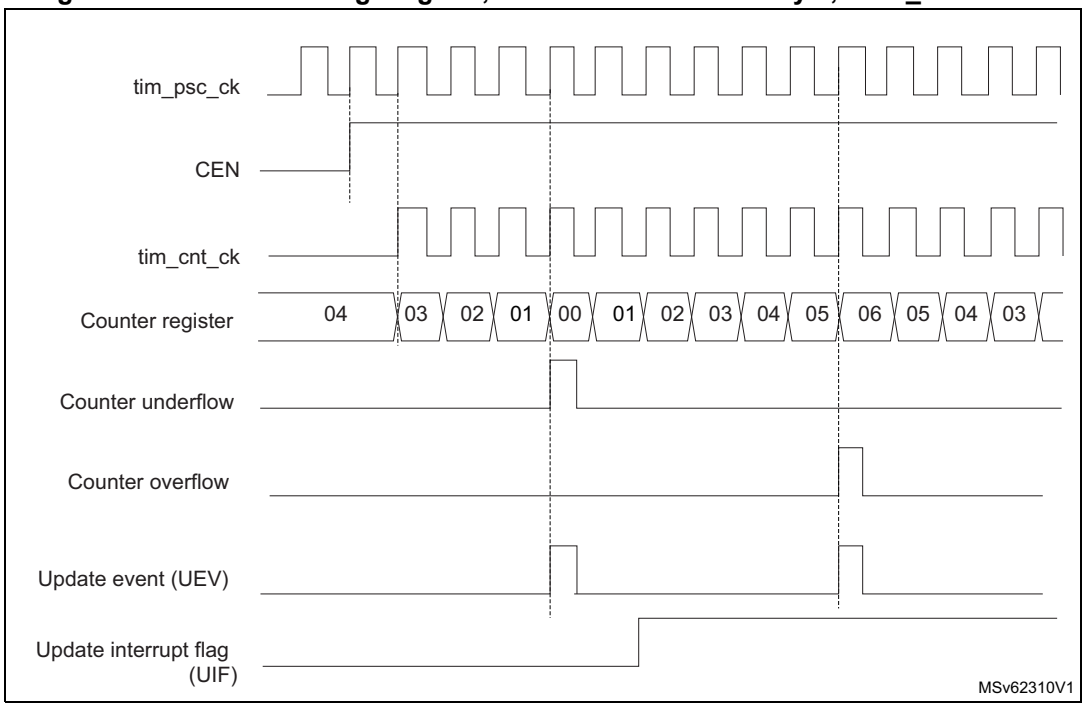
In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).
- The autoreload active register is updated with the preload value (content of the TIMx_ARR register). Note that if the update source is a counter overflow, the autoreload is updated before the counter is reloaded, so that the next period is the expected one (the counter is loaded with the new value).

The following figures show some examples of the counter behavior for different clock frequencies.

Figure 603. Counter timing diagram, internal clock divided by 1, TIMx_ARR = 0x6



1. Here, center-aligned mode 1 is used (for more details refer to *Section 54.5.1: TIMx control register 1 (TIMx_CR1)(x = 2 to 5)*).

Figure 604. Counter timing diagram, internal clock divided by 2

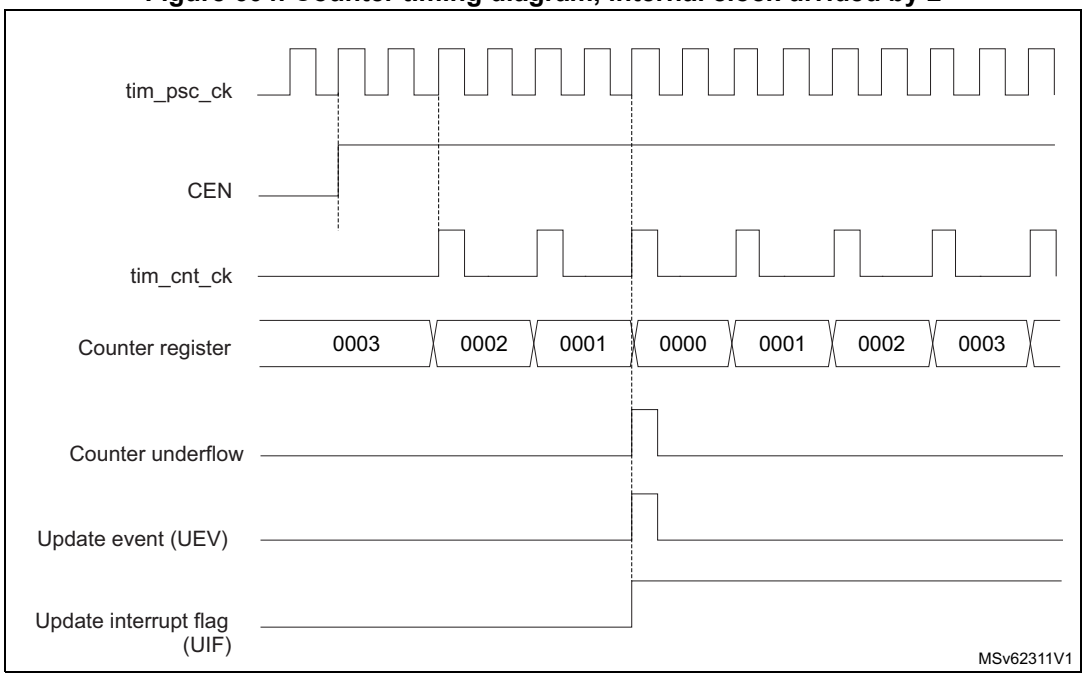
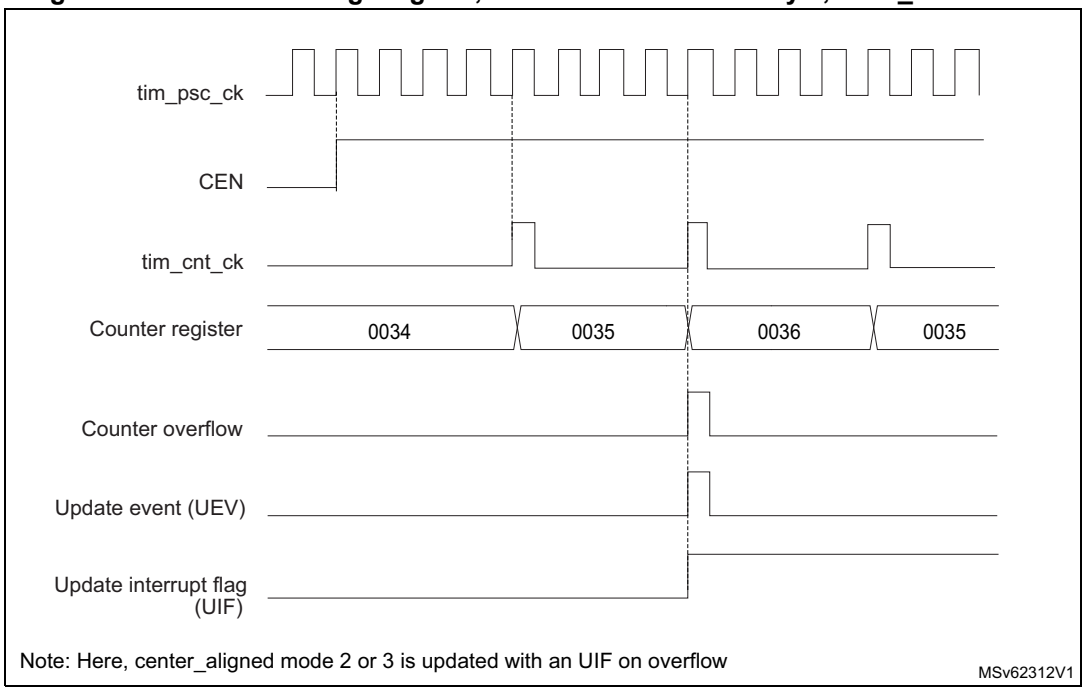


Figure 605. Counter timing diagram, internal clock divided by 4, TIMx_ARR = 0x36



1. Center-aligned mode 2 or 3 is used with a UIF on overflow.

Figure 606. Counter timing diagram, internal clock divided by N

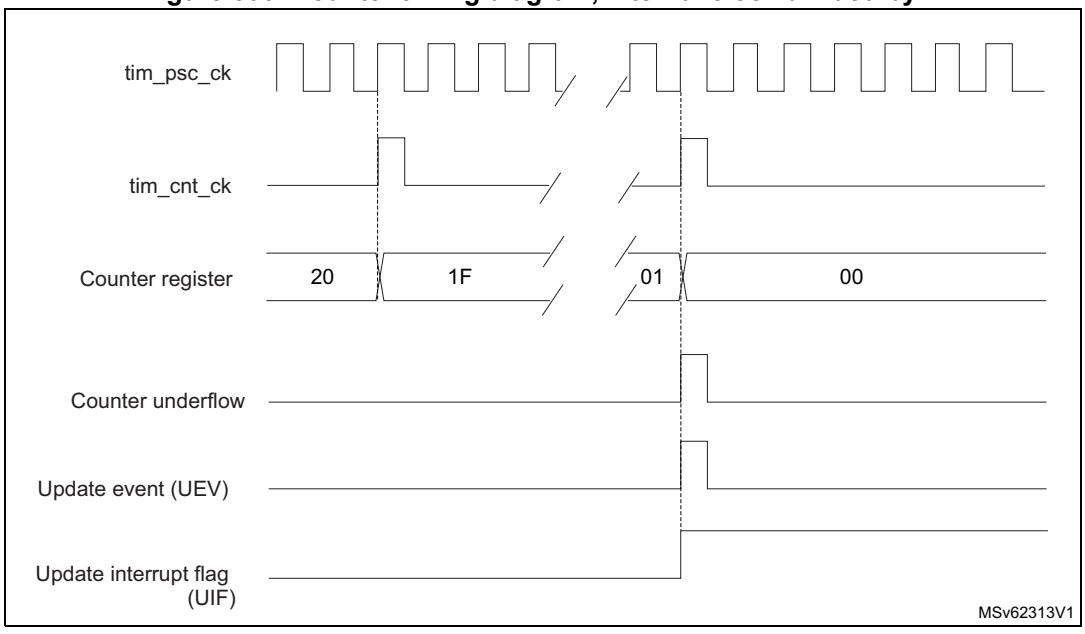


Figure 607. Counter timing diagram, Update event with ARPE = 1 (counter underflow)

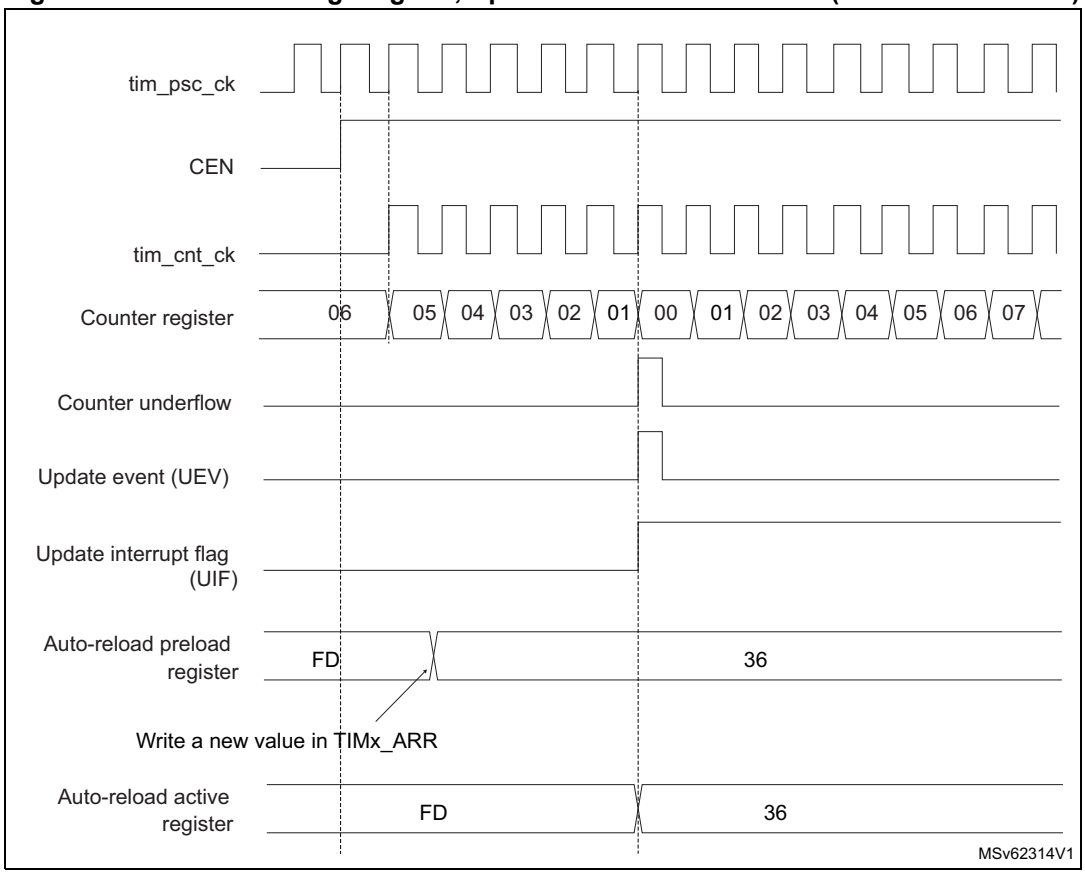
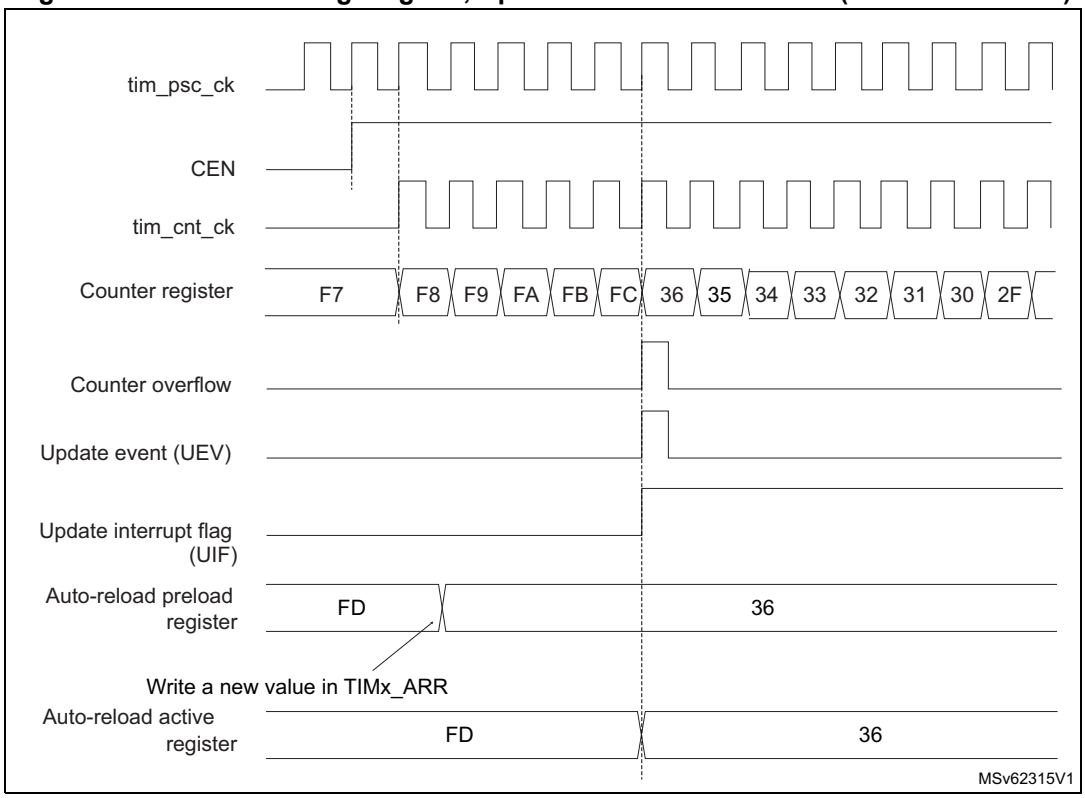


Figure 608. Counter timing diagram, Update event with ARPE = 1 (counter overflow)



54.4.5 Clock selection

The counter clock can be provided by the following clock sources:

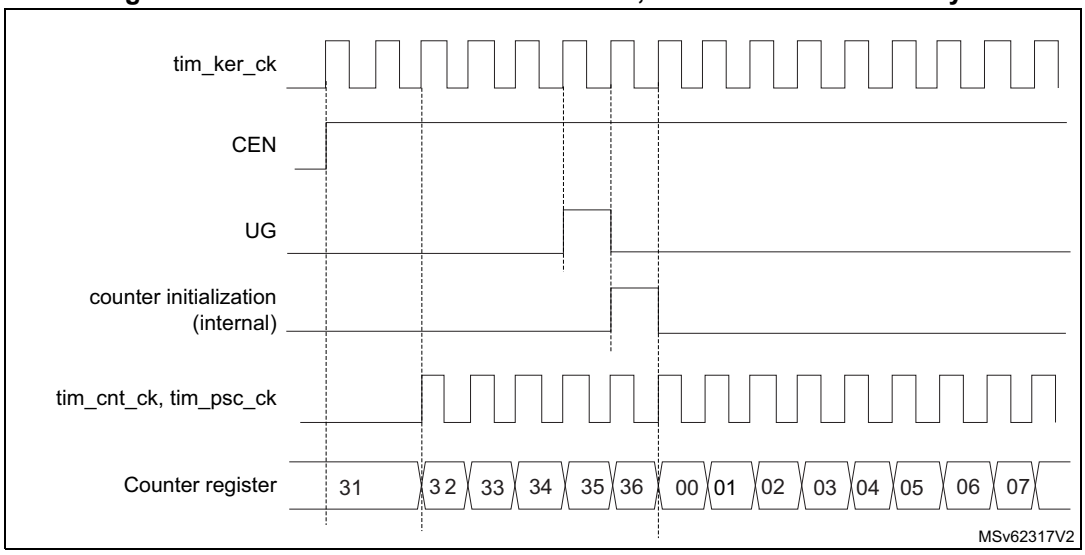
- Internal clock (tim_ker_ck).
- External clock mode1: external input pin (tim_ti1 or tim_ti2).
- External clock mode2: external trigger input (tim_etr_in).
- Internal trigger inputs (tim_itr): using one timer as prescaler for another timer, for example, timer 1 can be configured to act as a prescaler for timer 2. Refer to *Using one timer as prescaler for another timer* for more details.

Internal clock source (tim_ker_ck)

If the slave mode controller is disabled (SMS = 000 in the TIMx_SMCR register), then the CEN, DIR (in the TIMx_CR1 register), and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 609 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

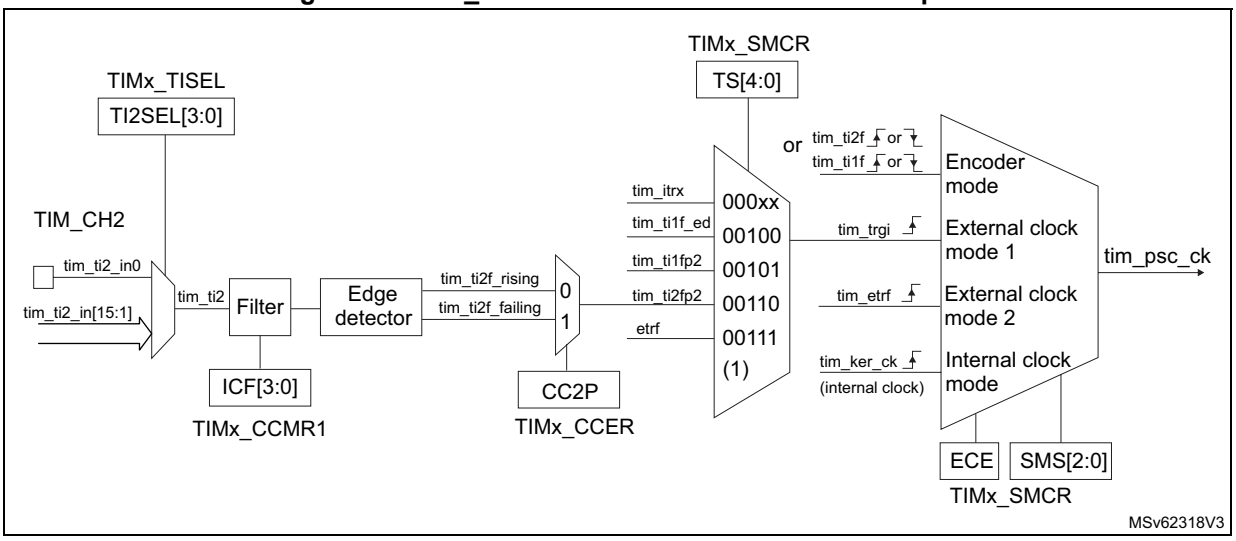
Figure 609. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1

This mode is selected when SMS = 111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 610. tim_ti2 external clock connection example



1. Codes ranging from 01000 to 11111: tim_itr[15:0].

For example, to configure the upcounter to count in response to a rising edge on the tim_ti2 input, use the following procedure:

1. Select the proper tim_ti2_in[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Configure channel 2 to detect rising edges on the tim_ti2 input by writing CC2S= 01 in the TIMx_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F = 0000).

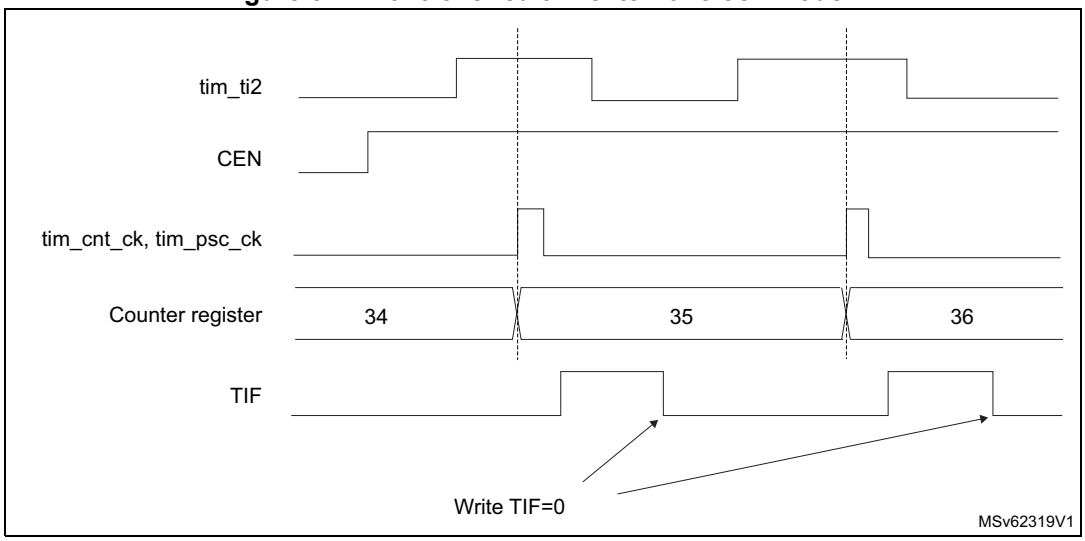
Note: The capture prescaler is not used for triggering, so it does not need to be configured.

4. Select rising edge polarity by writing CC2P = 0 and CC2NP = 0 in the TIMx_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS = 111 in the TIMx_SMCR register.
6. Select tim_ti2 as the input source by writing TS = 00110 in the TIMx_SMCR register.
7. Enable the counter by writing CEN = 1 in the TIMx_CR1 register.

When a rising edge occurs on tim_ti2, the counter counts once and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual clock of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 611. Control circuit in external clock mode 1



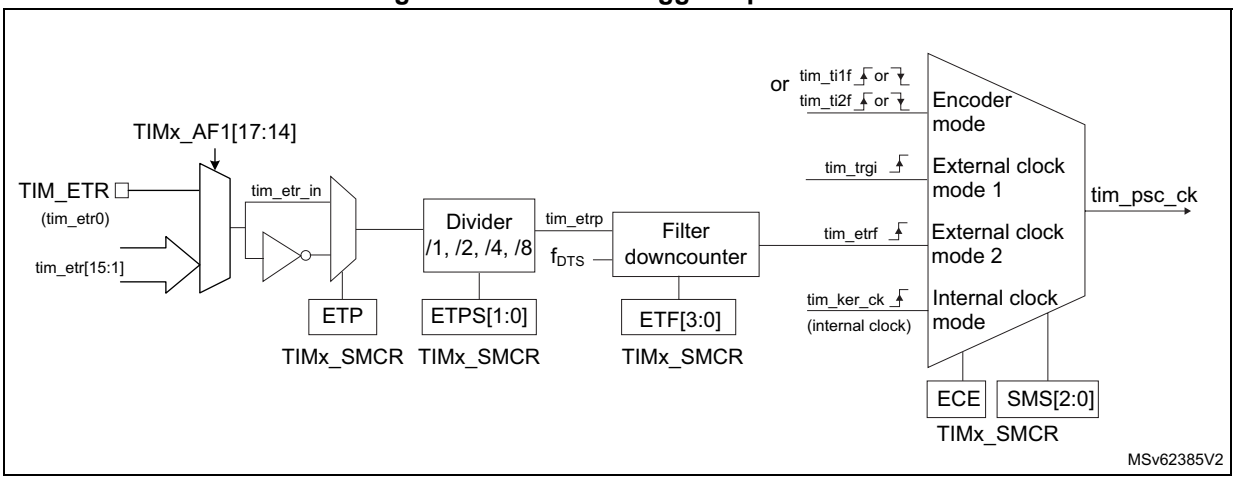
External clock source mode 2

This mode is selected by writing ECE = 1 in the TIMx_SMCR register.

The counter can count at each rising or falling edge on the external trigger input tim_etr_in.

Figure 612 gives an overview of the external trigger input block.

Figure 612. External trigger input block



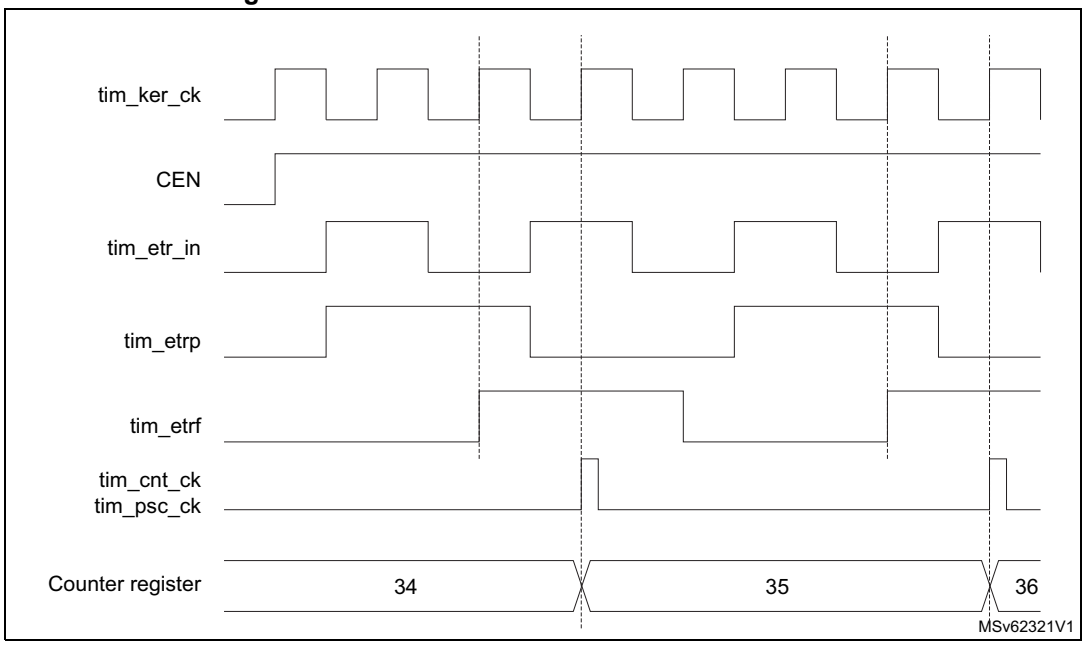
For example, to configure the upcounter to count each two rising edges on tim_etr_in, use the following procedure:

1. Select the proper tim_etr_in source (internal or external) with the ETRSEL[3:0] bits in the TIMx_AF1 register.
2. As no filter is needed in this example, write ETF[3:0] = 0000 in the TIMx_SMCR register.
3. Set the prescaler by writing ETPS[1:0] = 01 in the TIMx_SMCR register.
4. Select rising edge detection on the tim_etr_in by writing ETP = 0 in the TIMx_SMCR register.
5. Enable external clock mode 2 by writing ECE = 1 in the TIMx_SMCR register.
6. Enable the counter by writing CEN = 1 in the TIMx_CR1 register.

The counter counts once each two tim_etr_in rising edges.

The delay between the rising edge on tim_etr_in and the actual clock of the counter is due to the resynchronization circuit on the tim_etrp signal. As a consequence, the maximum frequency that can be correctly captured by the counter is at most ¼ of tim_ker_ck frequency. When the ETRP signal is faster, the user must apply a division of the external signal by a proper ETPS prescaler setting.

Figure 613. Control circuit in external clock mode 2



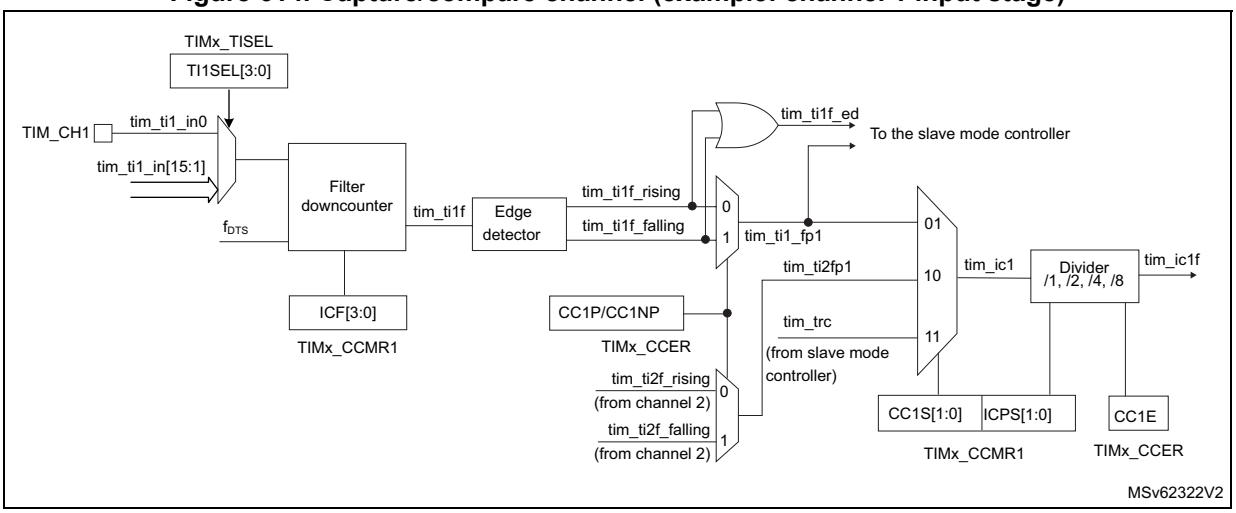
54.4.6 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

The following figure gives an overview of one Capture/Compare channel.

The input stage samples the corresponding tim_tix input to generate a filtered signal tim_tixf. Then, an edge detector with polarity selection generates a signal (tim_tixfpy) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 614. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: tim_ocxref (active high). The polarity acts at the end of the chain.

Figure 615. Capture/compare channel 1 main circuit

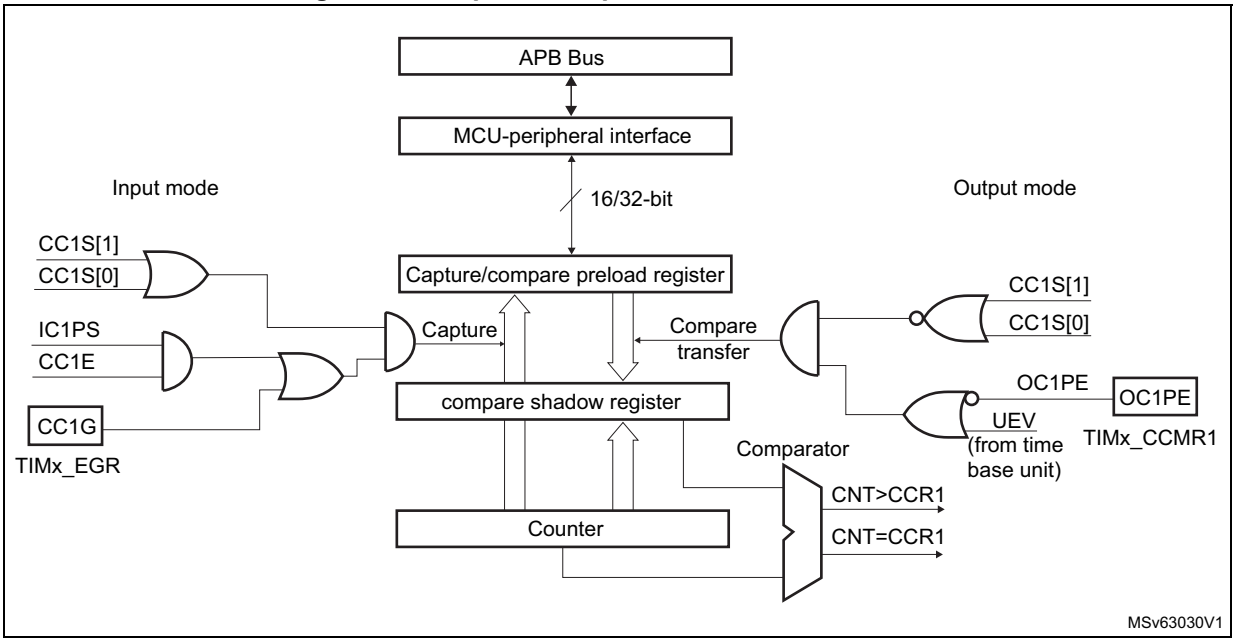
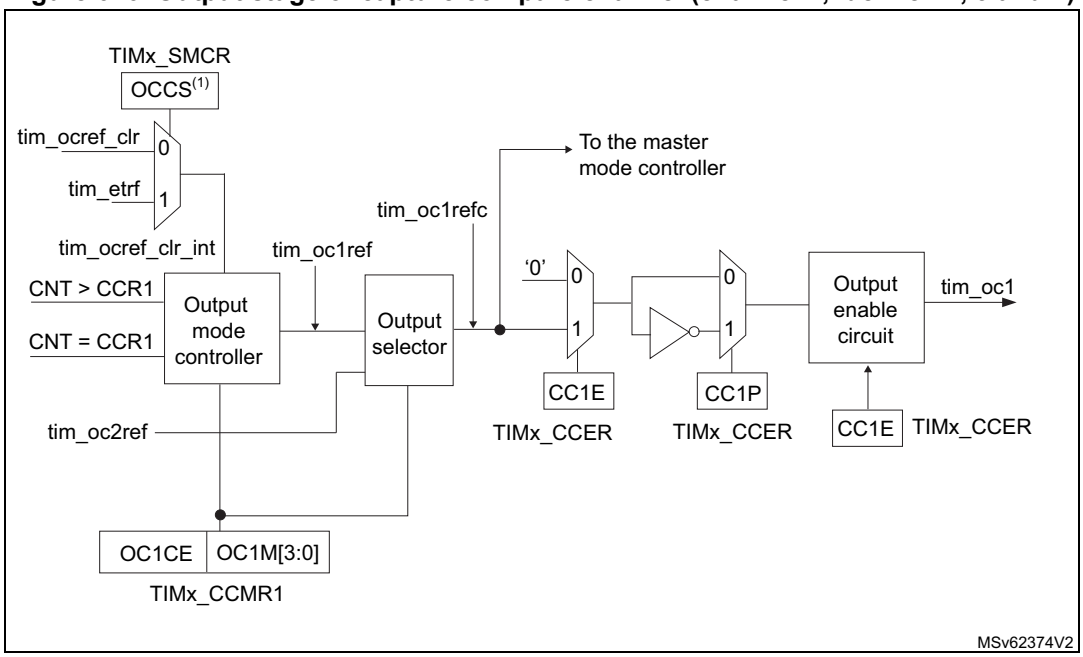


Figure 616. Output stage of capture/compare channel (channel 1, idem ch.2, 3 and 4)



1. Available on some instances only. If not available, tim_etrf is directly connected to tim_ocref_clr_int.

The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

54.4.7 Input capture mode

In input capture mode, the capture/compare registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the overcapture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to 0 or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx_CCR1 when tim_ti1 input rises. To do this, use the following procedure:

1. Select the proper tim_tix_in[15:0] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input: TIMx_CCR1 must be linked to the tim_ti1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input and the TIMx_CCR1 register becomes read-only.
3. Program the needed input filter duration in relation with the signal connected to the timer (when the input is one of the tim_tix (ICxF bits in the TIMx_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at most five internal clock cycles. We must program a filter duration longer than these five clock cycles. We can validate a transition on tim_ti1 when eight consecutive samples with the new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.
4. Select the edge of the active transition on the tim_ti1 channel by writing the CC1P and CC1NP bits to 000 in the TIMx_CCER register (rising edge in this case).
5. Program the input prescaler. In this example, the capture is to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIMx_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which may happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

54.4.8 PWM input mode

This mode is used to measure both the period and the duty cycle of a PWM signal connected to single tim_tix input:

- The TIMx_CCR1 register holds the period value (interval between two consecutive rising edges).
- The TIMx_CCR2 register holds the pulse width (interval between two consecutive rising and falling edges).

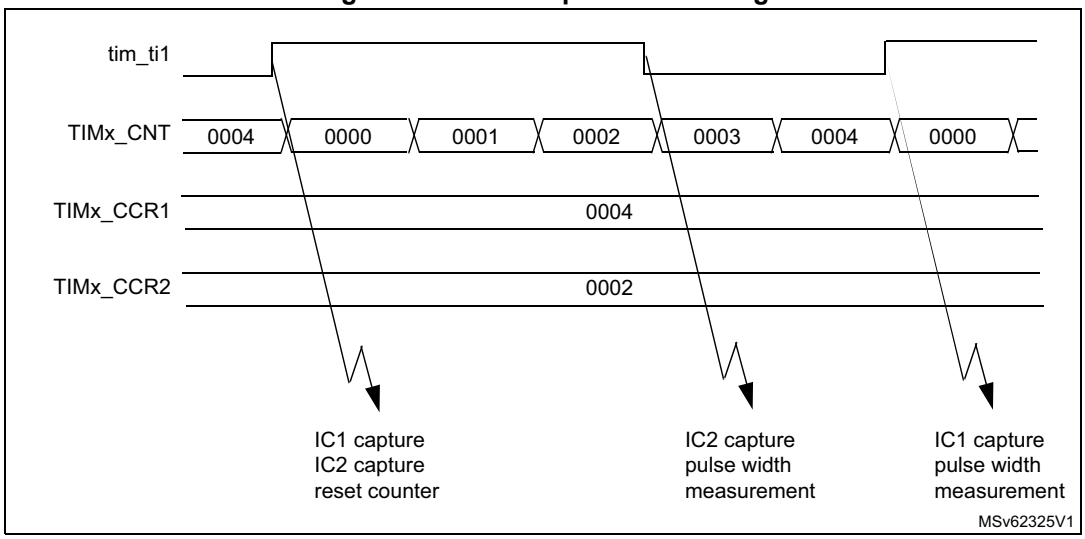
This mode is a particular case of input capture mode. The set-up procedure is similar with the following differences:

- Two ICx signals are mapped on the same tim_tix input.
- These two ICx signals are active on edges with opposite polarity.
- One of the two TIxFP signals is selected as trigger input and the slave mode controller is configured in reset mode.

The period and the pulse width of a PWM signal applied on tim_ti1 can be measured using the following procedure:

1. Select the proper tim_tix_in[15:0] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (tim_ti1 selected).
3. Select the active polarity for tim_ti1fp1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P to 0 and the CC1NP bit to 0 (active on rising edge).
4. Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (tim_ti1 selected).
5. Select the active polarity for tim_ti1fp2 (used for capture in TIMx_CCR2): write the CC2P bit to 1 and the CC2NP bit to 0 (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIMx_SMCR register (tim_ti1fp1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to 1 in the TIMx_CCER register.

Figure 617. PWM input mode timing



1. The PWM input mode can be used only with the TIMx_CH1/TIMx_CH2 signals due to the fact that only tim_ti1fp1 and tim_ti2fp2 are connected to the slave mode controller.

54.4.9 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (tim_ocxref and then tim_ocx) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (tim_ocxref/tim_ocx) to its active level, the user just needs to write 101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus tim_ocxref is forced high (tim_ocxref is always active high) and tim_ocx get opposite value to CCxP polarity bit.

For example: CCxP = 0 (tim_ocx active high) => tim_ocx is forced to high level.

tim_ocxref signal can be forced low by writing the OCxM bits to 100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the Output Compare mode section.

54.4.10 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM = 000), be set

active (OCxM = 001), be set inactive (OCxM = 010) or can toggle (OCxM = 011) on match.

- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

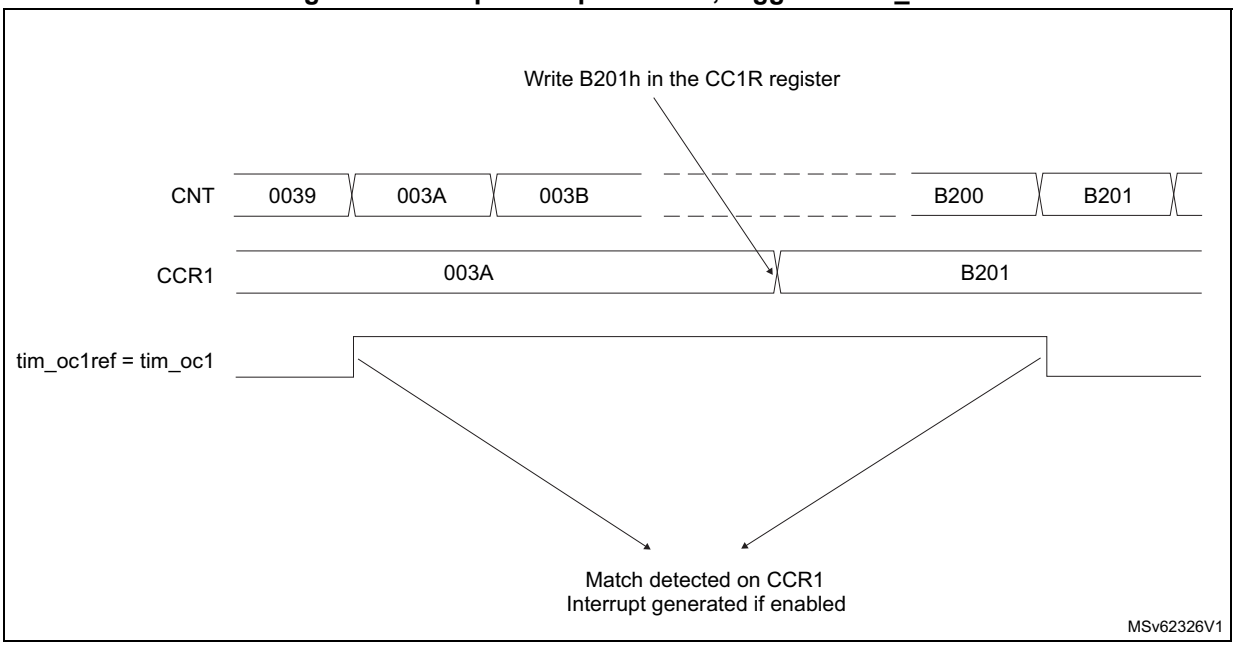
In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE and/or CCxDE bits if an interrupt and/or a DMA request is to be generated.
4. Select the output mode. For example:
   a) Write OCxM = 0011 to toggle tim_ocx output pin when CNT matches CCRx.
   b) Write OCxPE = 0 to disable preload register.
   c) Write CCxP = 0 to select active high polarity.
   d) Write CCxE = 1 to enable the output.
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE = 0, else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 618*.

Figure 618. Output compare mode, toggle on tim_oc1



54.4.11 PWM mode

Pulse width modulation mode is used to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per tim_ocx output) by writing 110 (PWM mode 1) or 111 (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the autoreload preload register (in up-counting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

tim_ocx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. tim_ocx output is enabled by the CCxE bit in the TIMx_CCER register. Refer to the TIMx_CCERx register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter). The tim_ocref_clr can be cleared by an external event through the tim_etr_in or the tim_oceref_clr signals. In this case the tim_ocref_clr signal is asserted only:

- After a compare match event.
- When the output compare mode (OCxM bits in TIMx_CCMRx register) switches from the “frozen” configuration (no comparison, OCxM = 000) to one of the PWM modes (OCxM = 110 or 111). This forces the PWM by software while the timer is running.

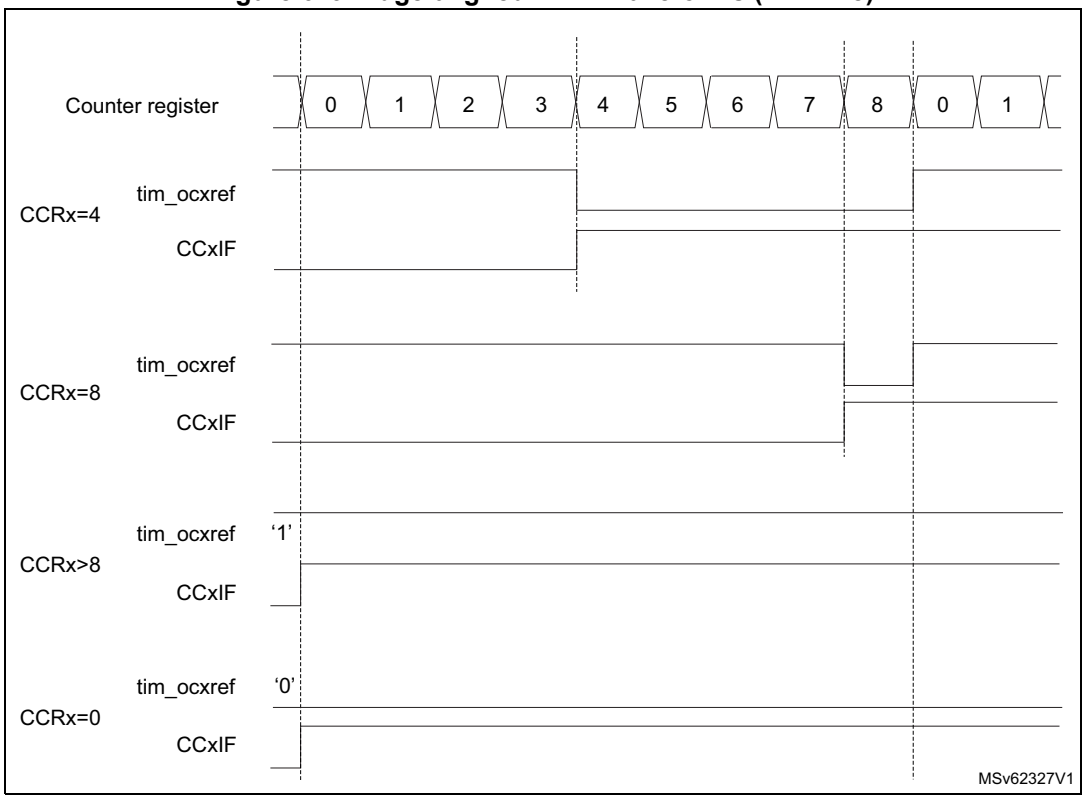
The timer is able to generate PWM in edge-aligned mode or center-aligned mode depending on the CMS bits in the TIMx_CR1 register.

PWM edge-aligned mode

- Up-counting configuration
- Up-counting is active when the DIR bit in the TIMx_CR1 register is low. Refer to *Up-counting mode*.

  In the following example, we consider PWM mode 1. The reference PWM signal tim_ocxref is high as long as TIMx_CNT <TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the autoreload value (in TIMx_ARR) then tim_ocxref is held at 1. If the compare value is 0 then tim_ocxref is held at 0. *Figure 619* shows some edge-aligned PWM waveforms in an example where TIMx_ARR = 8.

Figure 619. Edge-aligned PWM waveforms (ARR = 8)



Down-counting configuration

- Down-counting is active when DIR bit in TIMx_CR1 register is high. Refer to *Down-counting mode*.

  In PWM mode 1, the reference signal tim_ocxref is low as long as TIMx_CNT>TIMx_CCRx else it becomes high. If the compare value in TIMx_CCRx is greater than the autoreload value in TIMx_ARR, then tim_ocxref is held at 100%. PWM is not possible in this mode.

PWM center-aligned mode

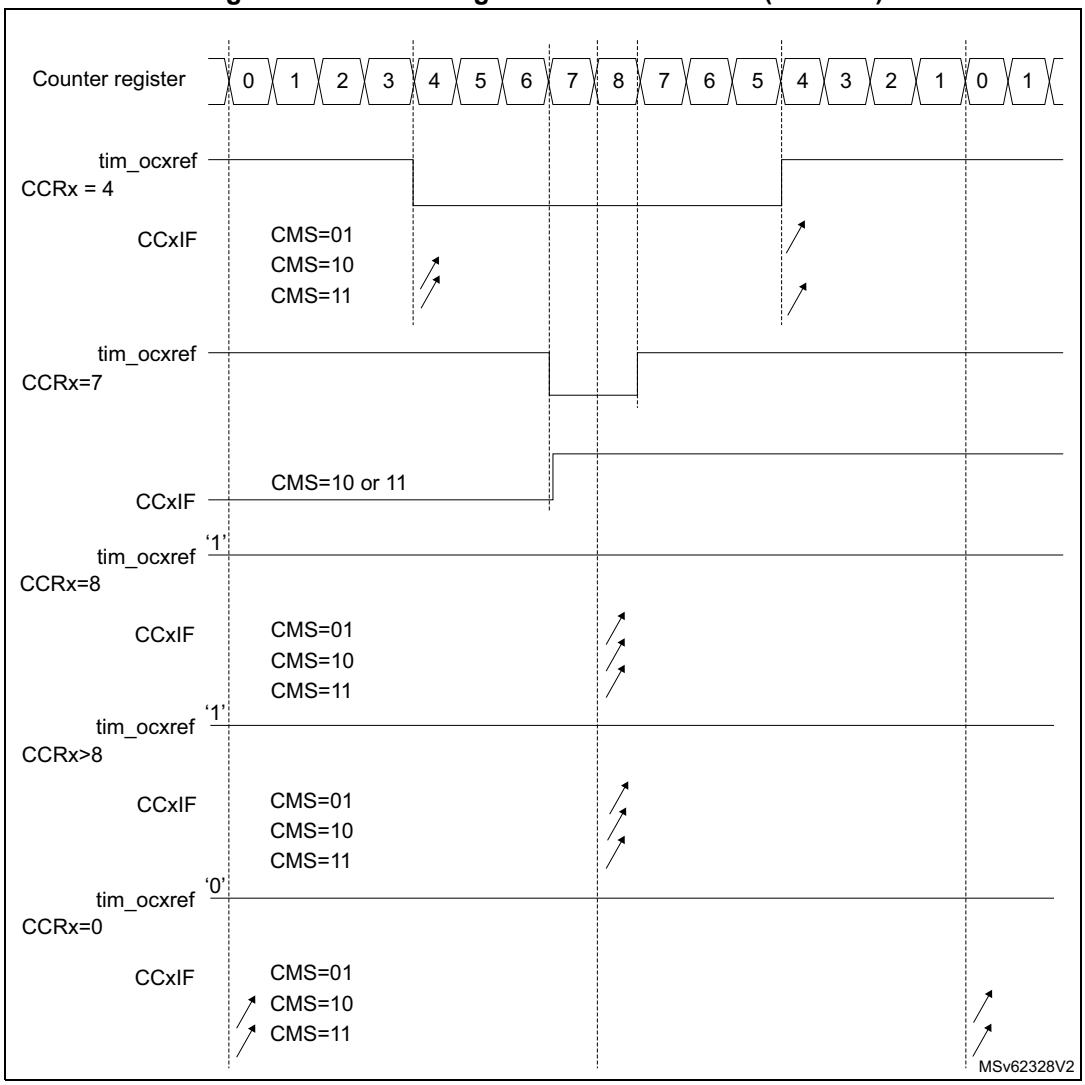
Center-aligned mode is active when the CMS bits in TIMx_CR1 register are different from 00 (all the remaining configurations having the same effect on the tim_ocxref/tim_ocx signals). The compare flag is set when the counter counts up, when it counts down or both when it counts up and down depending on the CMS bits configuration. The direction bit

(DIR) in the TIMx_CR1 register is updated by hardware and must not be changed by software. Refer to *Center-aligned mode (up/down-counting)*.

Figure 620 shows some center-aligned PWM waveforms in an example where:

- TIMx_ARR = 8.
- PWM mode is the PWM mode 1.
- The flag is set when the counter counts down corresponding to the center-aligned mode 1 selected for CMS = 01 in TIMx_CR1 register.

Figure 620. Center-aligned PWM waveforms (ARR = 8)



Hints on using center-aligned mode:

- When starting in center-aligned mode, the current up-down configuration is used. It means that the counter counts up or down depending on the value written in the DIR bit

in the TIMx_CR1 register. Moreover, the DIR and CMS bits must not be changed at the same time by the software.

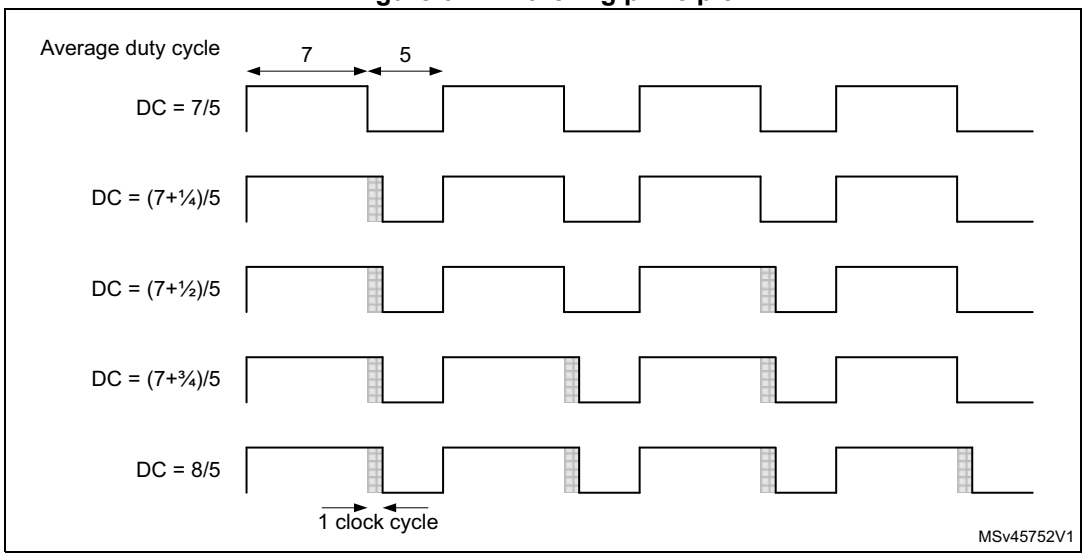
- Writing to the counter while running in center-aligned mode is not recommended as it can lead to unexpected results. In particular:
  - The direction is not updated if a value greater than the autoreload value is written in the counter (TIMx_CNT>TIMx_ARR). For example, if the counter was counting up, it continues to count up.
  - The direction is updated if 0 or the TIMx_ARR value is written in the counter but no update event UEV is generated.
- The safest way to use center-aligned mode is to generate an update by software (setting the UG bit in the TIMx_EGR register) just before starting the counter and not to write the counter while it is running.

Dithering mode

The PWM mode effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIMx_CR1 register. This applies to both the CCR (for duty cycle resolution increase) and ARR (for PWM frequency resolution increase).

The operating principle is to have the actual CCR (or ARR) value slightly changed (adding or not one timer clock period) over 16 consecutive PWM periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average duty cycle or PWM period. *Figure 621* presents the dithering principle applied to four consecutive PWM cycles.

Figure 621. Dithering principle



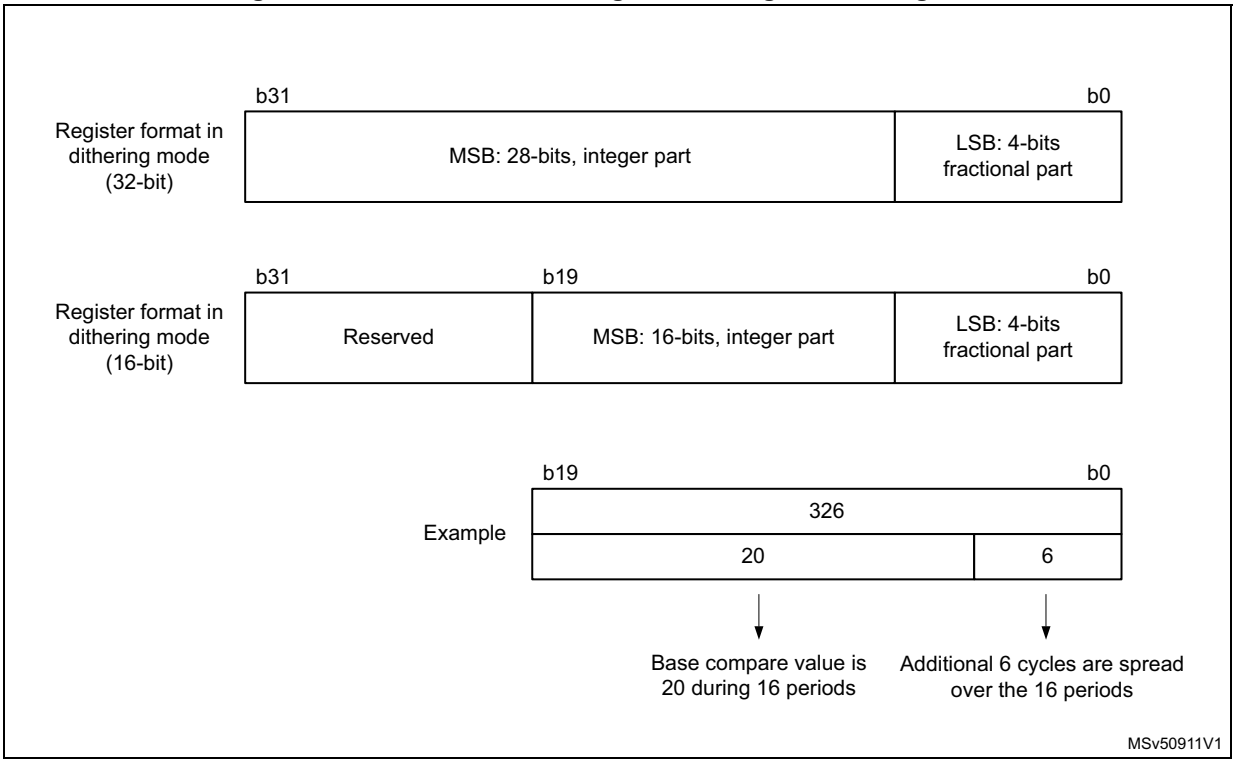
When the dithering mode is enabled, the register coding is changed as following (see *Figure 622* for example):

- The four LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted by four places and are coding for the base value. In 16-bit mode, the 16-bit format is maintained.

Note: The following sequence must be followed when resetting the DITHEN bit:

1. *CEN and ARPE bits must be reset.*
2. *The DITHEN bit must be reset.*
3. *The CCIF flags must be cleared.*
4. *The CEN bit can be set (eventually with ARPE = 1).*

Figure 622. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode (16-bit timer): } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

$$\text{Dithering mode (32-bit timer): } F_{pwmMin} = \frac{F_{Tim}}{268435454 + \frac{15}{16}}$$

Note: For 16-bit timers, the maximum TIMx_ARR and TIMx_CCRy values are limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part). For 32-bit timers, the maximum TIMx_ARR and TIMx_CCRy values are limited to

0xFFFFFFEF in dithering mode (corresponds to 264435454 for the integer part and 15 for the dithered part).

As shown on *Figure 623* and *Figure 624*, the dithering mode is used to increase the PWM resolution.

Figure 623. PWM resolution vs frequency (16-bit mode)

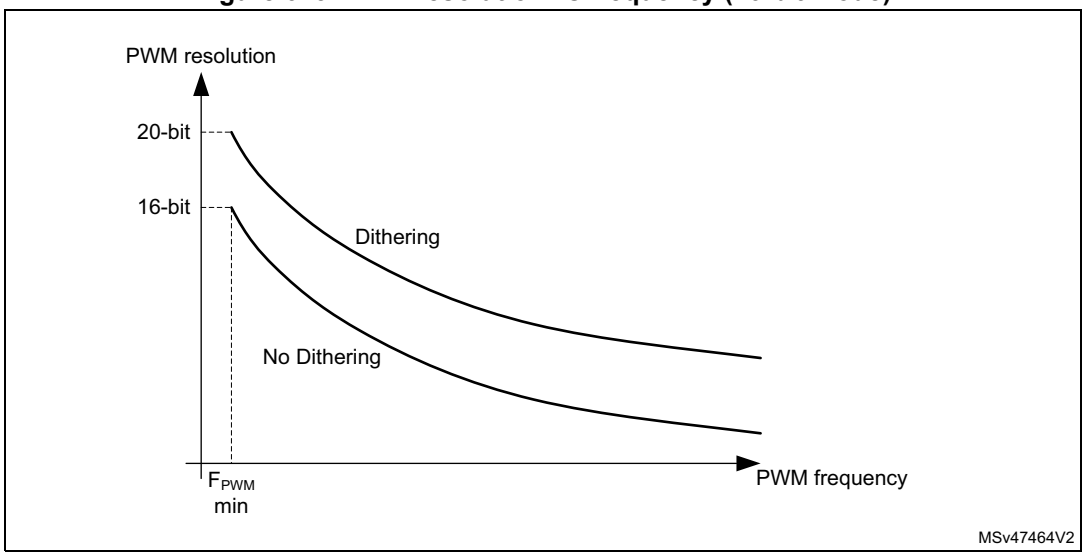
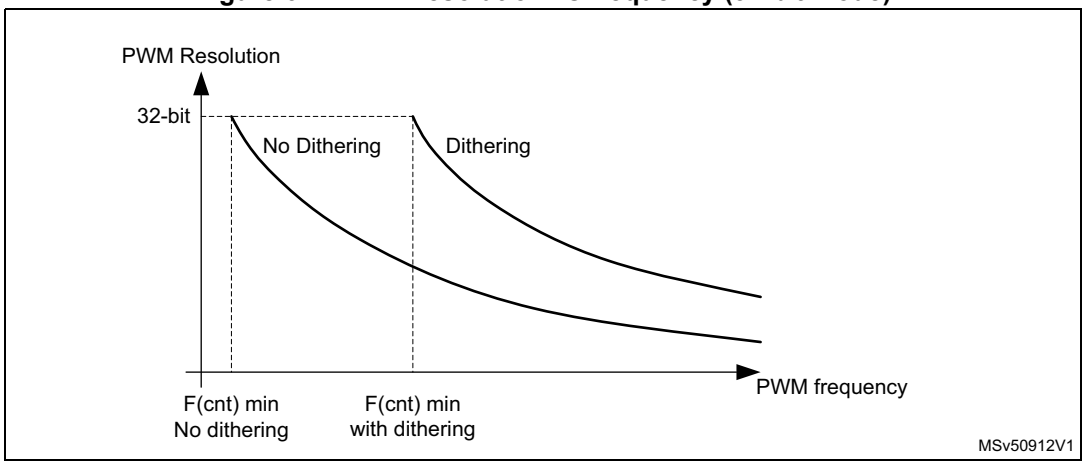
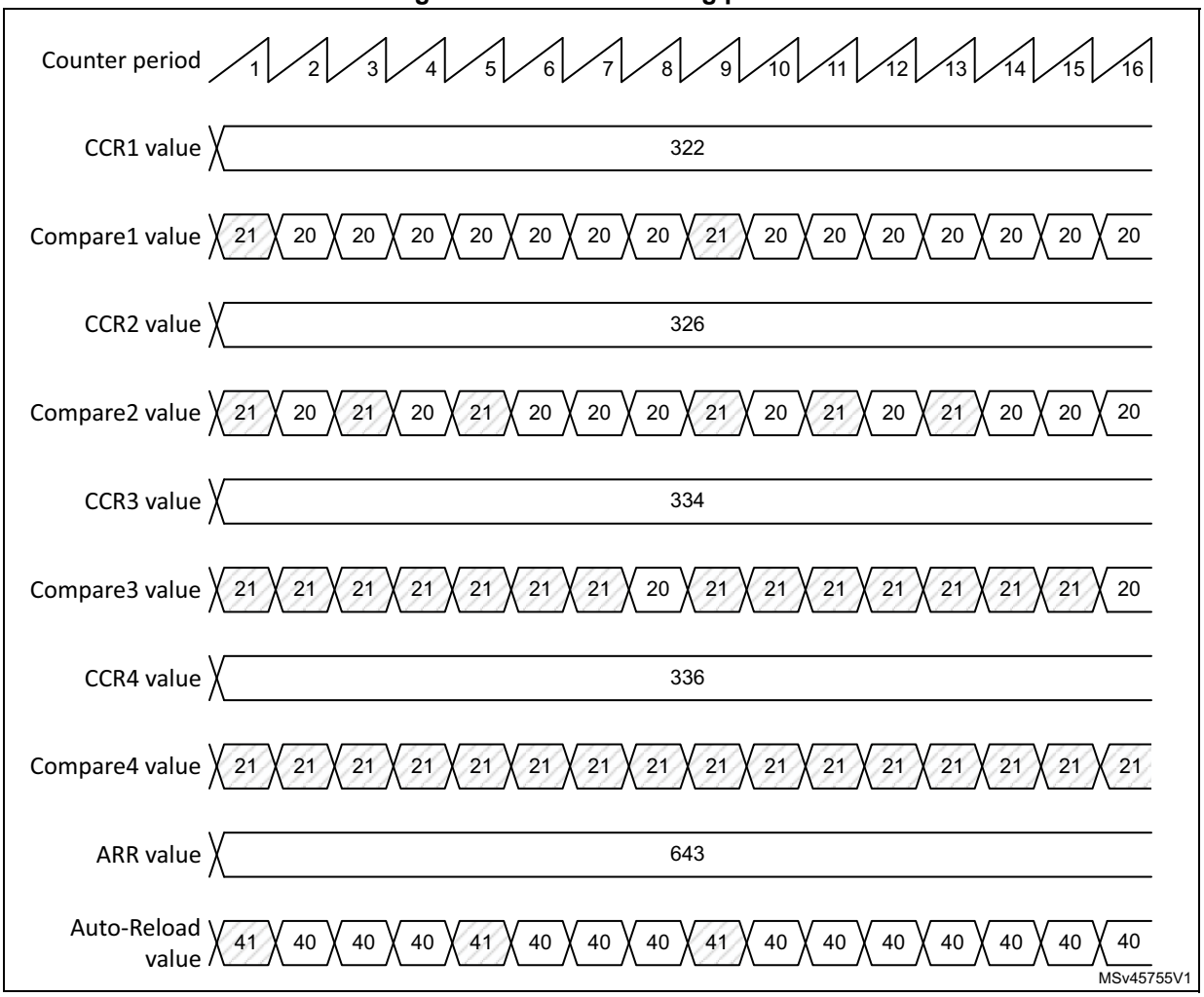


Figure 624. PWM resolution vs frequency (32-bit mode)



The duty cycle and/or period changes are spread over 16 consecutive periods, as described in *Figure 625*.

Figure 625. PWM dithering pattern



The autoreload and compare values increments are spread following specific patterns described in *Table 511*. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 511. CCR and ARR register change dithering pattern

LSB value	PWM period															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

The dithering mode is also available in center-aligned PWM mode (CMS bits in TIMx_CR1 register are not equal to 00). In this case, the dithering pattern is applied over eight consecutive PWM periods, considering the up and down-counting phases as shown in *Figure 626*.

Figure 626. Dithering effect on duty cycle in center-aligned PWM mode

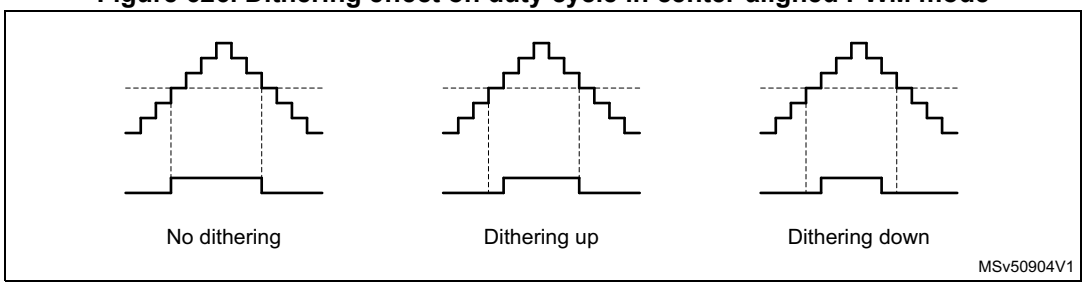


Table 512 shows how the dithering pattern is added in center-aligned PWM mode.

Table 512. CCR register change dithering pattern in center-aligned PWM mode

LSB value	PWM period															
	1		2		3		4		5		6		7		8	
	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn	Up	Dn
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

54.4.12 Asymmetric PWM mode

Asymmetric mode allows two center-aligned PWM signals to be generated with a programmable phase shift. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and the phase-shift are determined by a pair of TIMx_CCRx registers. One register controls the PWM during up-counting, the second during down-counting, so that PWM is adjusted every half PWM cycle:

- tim_oc1refc (or tim_oc2refc) is controlled by TIMx_CCR1 and TIMx_CCR2.
- tim_oc3refc (or tim_oc4refc) is controlled by TIMx_CCR3 and TIMx_CCR4.

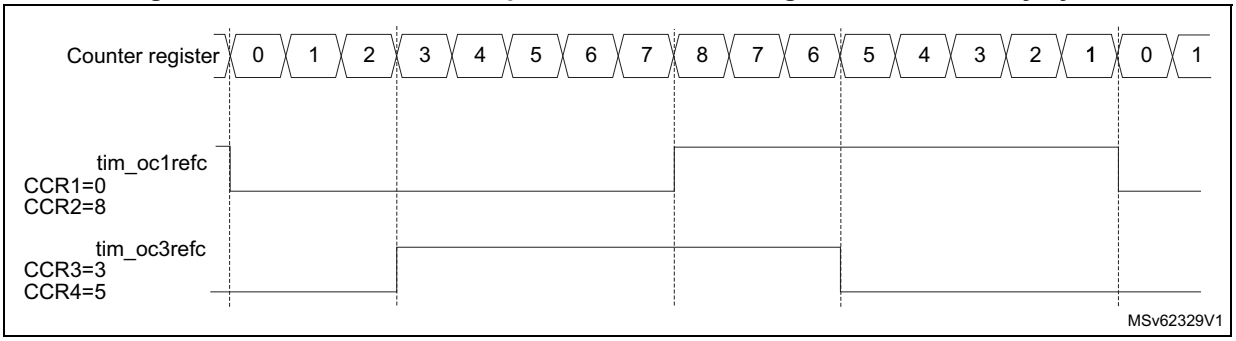
Asymmetric PWM mode can be selected independently on two channels (one tim_ocx output per pair of CCR registers) by writing 1110 (Asymmetric PWM mode 1) or 1111 (Asymmetric PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

Note: The OCxM[3:0] bitfield is split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

When a given channel is used as asymmetric PWM channel, its secondary channel can also be used. For instance, if an tim_oc1refc signal is generated on channel 1 (Asymmetric PWM mode 1), it is possible to output either the tim_oc2ref signal on channel 2, or an tim_oc2refc signal resulting from asymmetric PWM mode 2.

Figure 627 shows an example of signals that can be generated using asymmetric PWM mode (channels 1 to 4 are configured in asymmetric PWM mode 2).

Figure 627. Generation of two phase-shifted PWM signals with 50% duty cycle



54.4.13 Combined PWM mode

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, tim_ocxrefc, are made of an OR or AND logical combination of two reference PWMs:

- tim_oc1refc (or tim_oc2refc) is controlled by TIMx_CCR1 and TIMx_CCR2
- tim_oc3refc (or tim_oc4refc) is controlled by TIMx_CCR3 and TIMx_CCR4

Combined PWM mode can be selected independently on two channels (one tim_ocx output per pair of CCR registers) by writing 1100 (Combined PWM mode 1) or 1101 (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

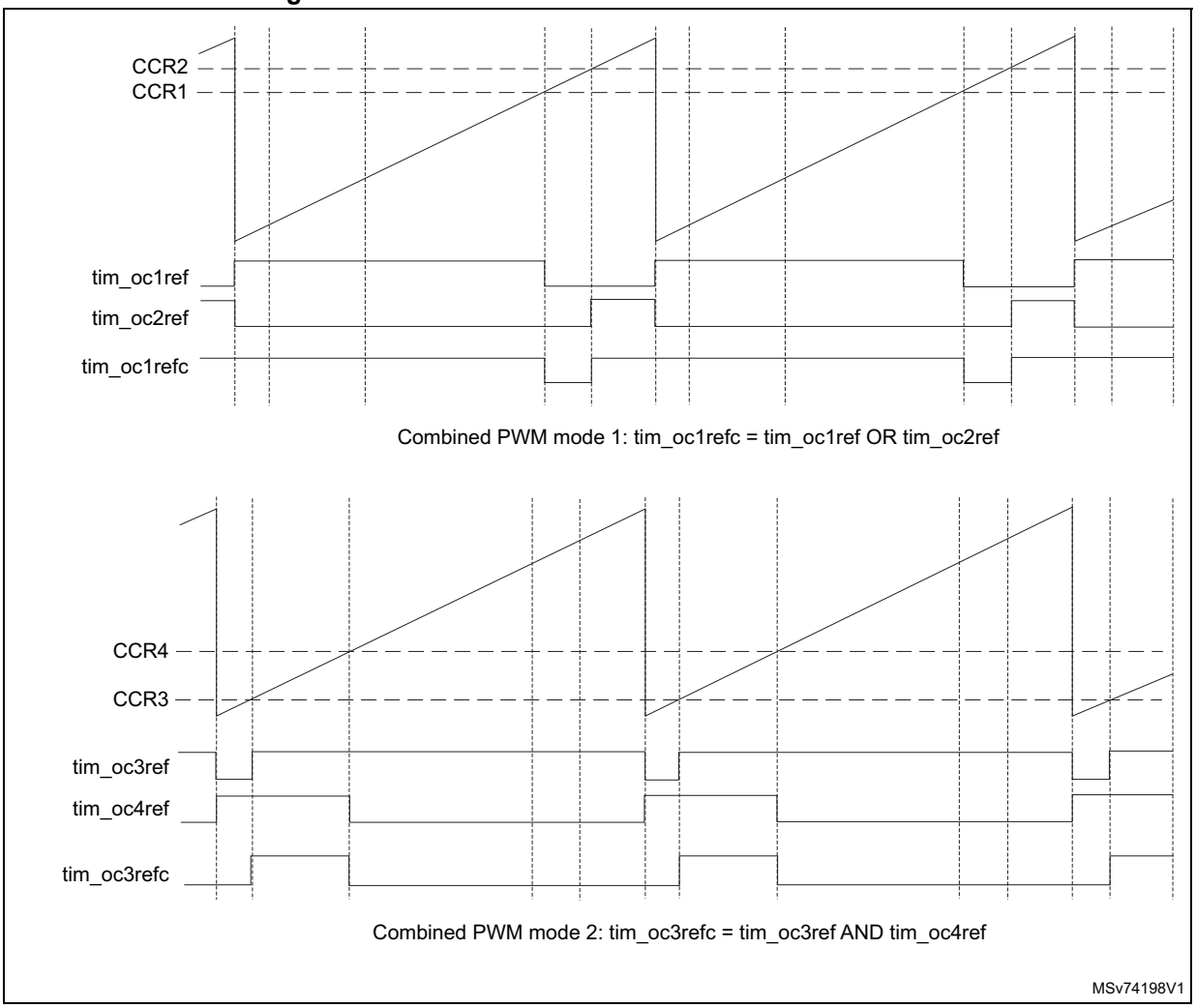
When a given channel is used as combined PWM channel, its secondary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

Note: The OCxM[3:0] bitfield is split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

Figure 628 shows an example of signals that can be generated using combined PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 1.
- Channel 2 is configured in PWM mode 2.
- Channel 3 is configured in Combined PWM mode 2.
- Channel 4 is configured in PWM mode 1.

Figure 628. Combined PWM mode on channels 1 and 3



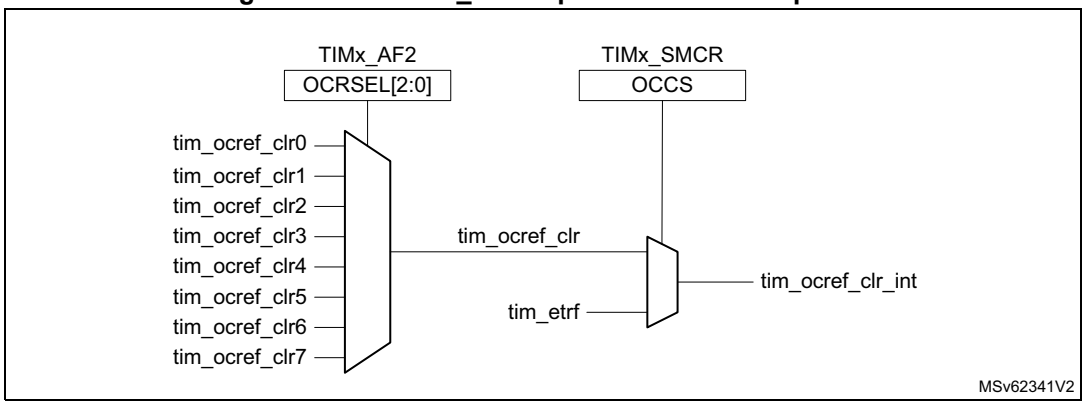
54.4.14 Clearing the tim_ocxref signal on an external event

The tim_ocxref signal of a given channel can be cleared when a high level is applied on the tim_ocref_clr_int input (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1). tim_ocxref remains low until the next transition to the active state, on the following PWM cycle. This function can only be used in Output compare and PWM modes. It does not work in Forced mode.

The tim_ocref_clr_int source depends on the OCREF clear selection feature implementation, refer to *Section 54.3: TIM2/TIM3/TIM4/TIM5 implementation*.

If the OCREF clear selection feature is implemented, the tim_ocref_clr_int can be selected between the tim_ocref_clr input and the tim_etrf input (tim_etr_in after the filter) by configuring the OCCS bit in the TIMx_SMCR register. The tim_ocref_clr input can be selected among several tim_ocref_clr[7:0] inputs, using the OCRSEL[2:0] bitfield in the TIMx_AF2 register, as shown in *Figure 629*.

Figure 629. OCREF_CLR input selection multiplexer



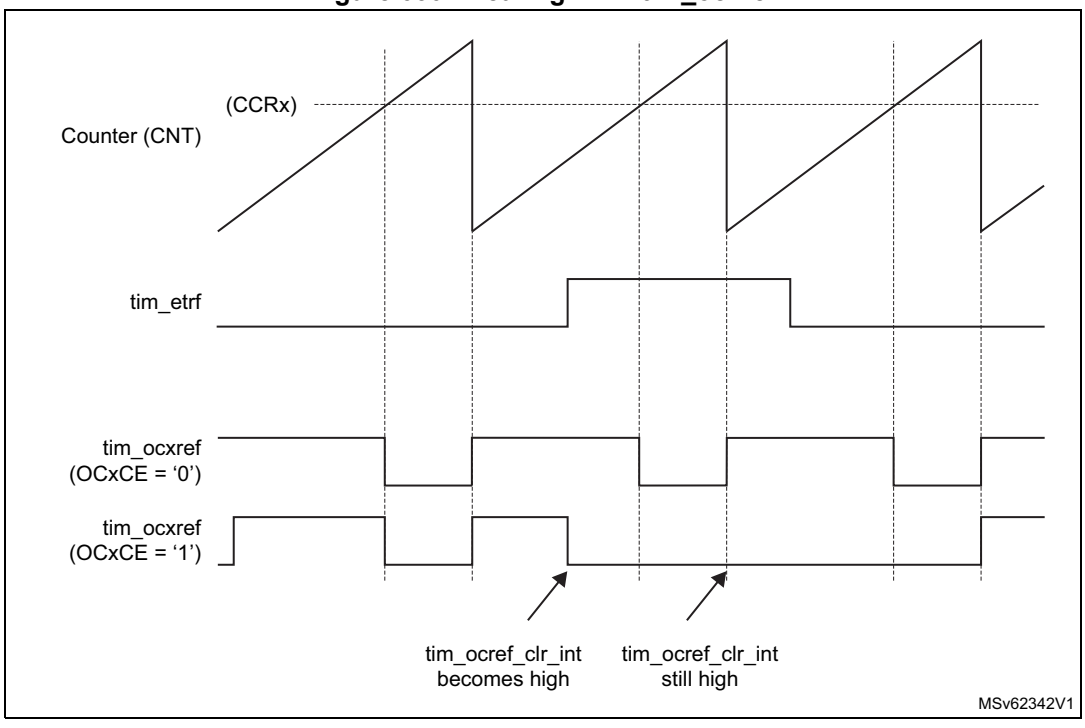
If the OCREF clear selection feature is not implemented, the tim_ocref_clr_int input is directly connected to the tim_etrf input.

For example, the tim_ocref_clr_int signal can be connected to the output of a comparator to be used for current handling. In this case, tim_etr_in must be configured as follows:

1. The external trigger prescaler must be kept off: bits ETPS[1:0] in the TIMx_SMCR register are cleared to 00.
2. The external clock mode 2 must be disabled: bit ECE in the TIMx_SMCR register is cleared to 0.
3. The external trigger polarity (ETP) and the external trigger filter (ETF) can be configured according to the application's needs.

Figure 630 shows the behavior of the tim_ocxref signal when the tim_etrf input becomes high, for both values of the OCxCE enable bit. In this example, the timer TIMx is programmed in PWM mode.

Figure 630. Clearing TIMx tim_ocxref



Note: In case of a PWM with a 100% duty cycle (if CCRx>ARR), tim_ocxref is enabled again at the next counter overflow.

54.4.15 One-pulse mode

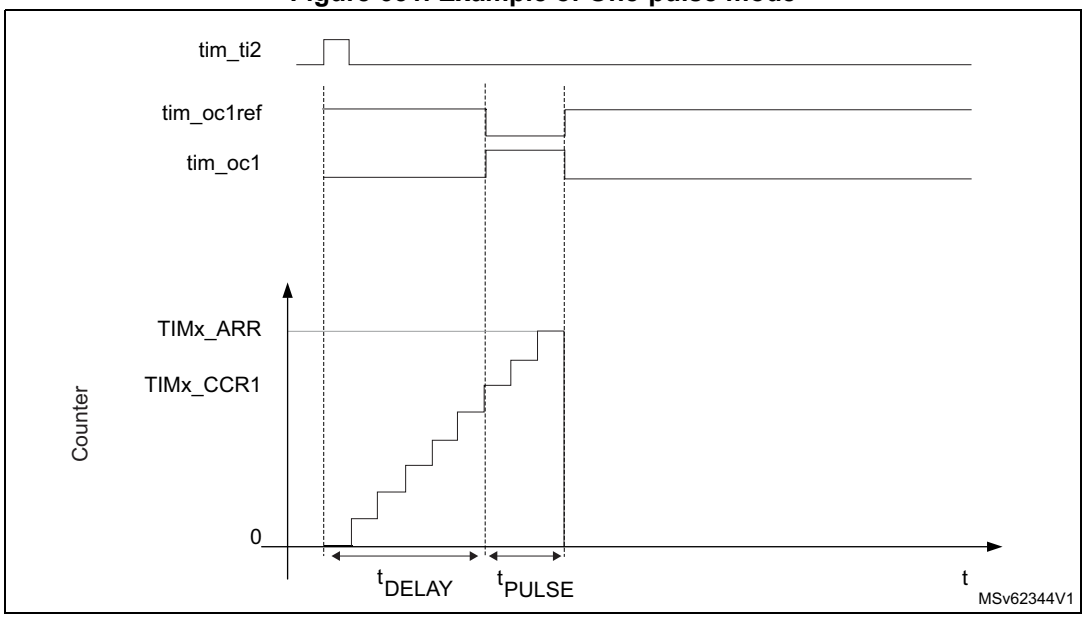
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

CNT<CCRx ≤ ARR (in particular, 0<CCRx).

Figure 631. Example of One-pulse mode



For example if the user wants to generate a positive pulse on tim_oc1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the tim_ti2 input pin.

Use tim_ti2fp2 as trigger 1:

1. Select the proper tim_ti2_in[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Map tim_ti2fp2 on tim_ti2 by writing CC2S = 01 in the TIMx_CCMR1 register.
3. tim_ti2fp2 must detect a rising edge, write CC2P = 0 and CC2NP = 0 in the TIMx_CCER register.
4. Configure tim_ti2fp2 as trigger for the slave mode controller (tim_trgi) by writing TS = 00110 in the TIMx_SMCR register.
5. tim_ti2fp2 is used to start the counter by writing SMS to 110 in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the autoreload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Suppose the user wants to build a waveform with a transition from 0 to 1 when a compare match occurs and a transition from 1 to 0 when the counter reaches the autoreload value. To do this PWM mode 2 must be enabled by writing OC1M = 111 in the TIMx_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE = 1 in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the autoreload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on tim_ti2. CC1P is written to 0 in this example.

In this example, the DIR and CMS bits in the TIMx_CR1 register must be low.

Since only one pulse (Single mode) is needed, a one must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the autoreload value back to 0). When OPM bit in the TIMx_CR1 register is set to 0, so the Repetitive mode is selected.

Particular case: tim_ocx fast enable:

In One-pulse mode, the edge detection on tim_tix input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min we can get.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then tim_ocxref (and tim_ocx) is forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

54.4.16 Retriggerable one-pulse mode

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with non-retriggerable one-pulse mode described in *Section 54.4.15*:

- The pulse starts as soon as the trigger occurs (no programmable delay).
- The pulse is extended if a new trigger occurs before the previous one is completed.

The timer must be in Slave mode, with the bits SMS[3:0] = 1000 (Combined Reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to 1000 or 1001 for Retriggerable OPM mode 1 or 2.

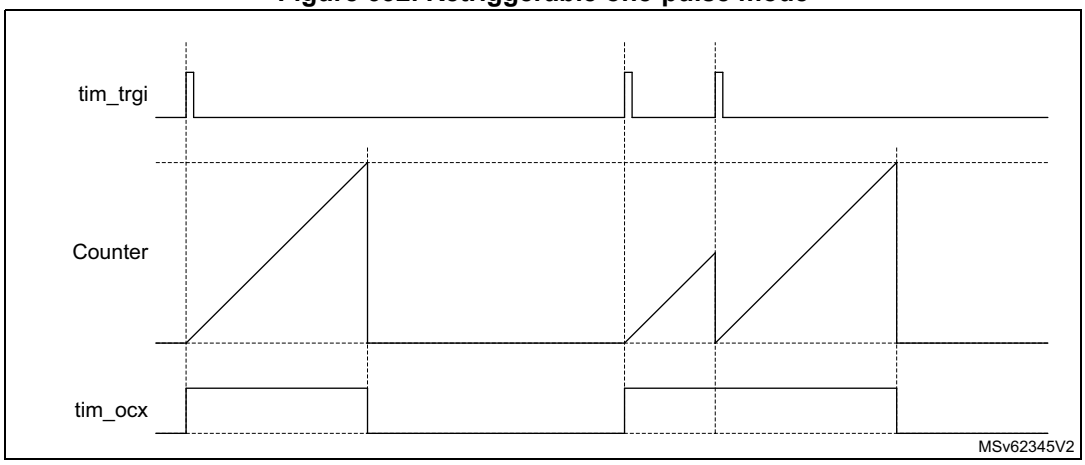
If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in down-counting mode CCRx must be above or equal to ARR.

Note: In Retriggerable one-pulse mode, the CCxIF flag is not significant.

The OCxM[3:0] and SMS[3:0] bitfields are split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.

Figure 632. Retriggerable one-pulse mode



54.4.17 Pulse on compare mode

A pulse can be generated upon compare match event. A signal with a programmable pulse width generated when the counter value equals a given compare value, for debugging or synchronization purposes.

This mode is available for any slave mode selection, including encoder modes, in edge and center aligned counting modes. It is solely available for channel 3 and channel 4. The pulse generator is unique and is shared by the two channels, as shown on *Figure 633*.

Figure 633. Pulse generator circuitry

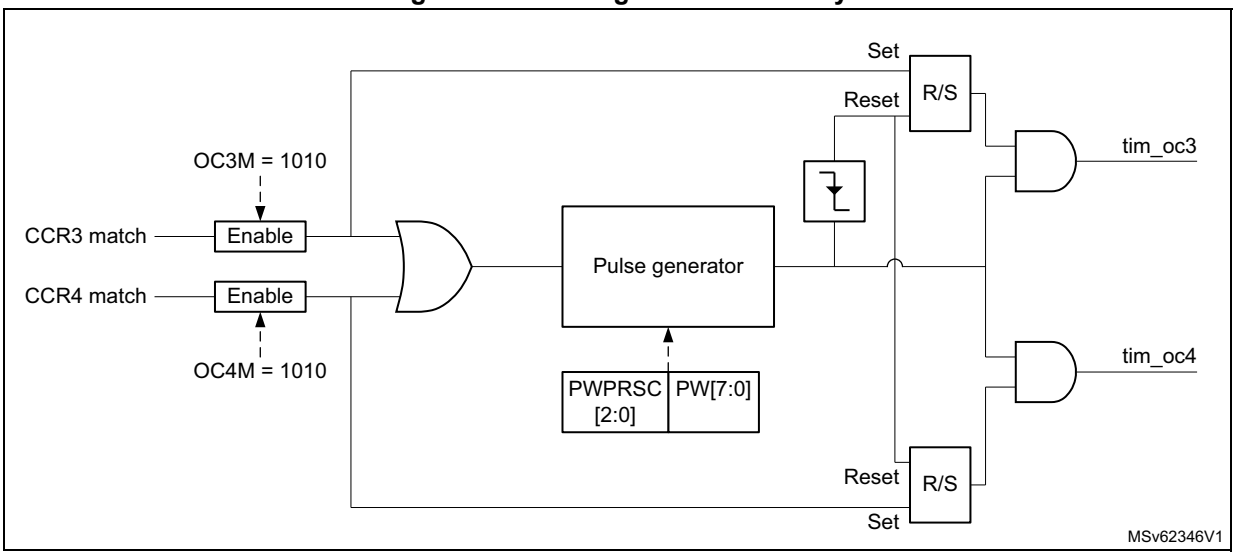
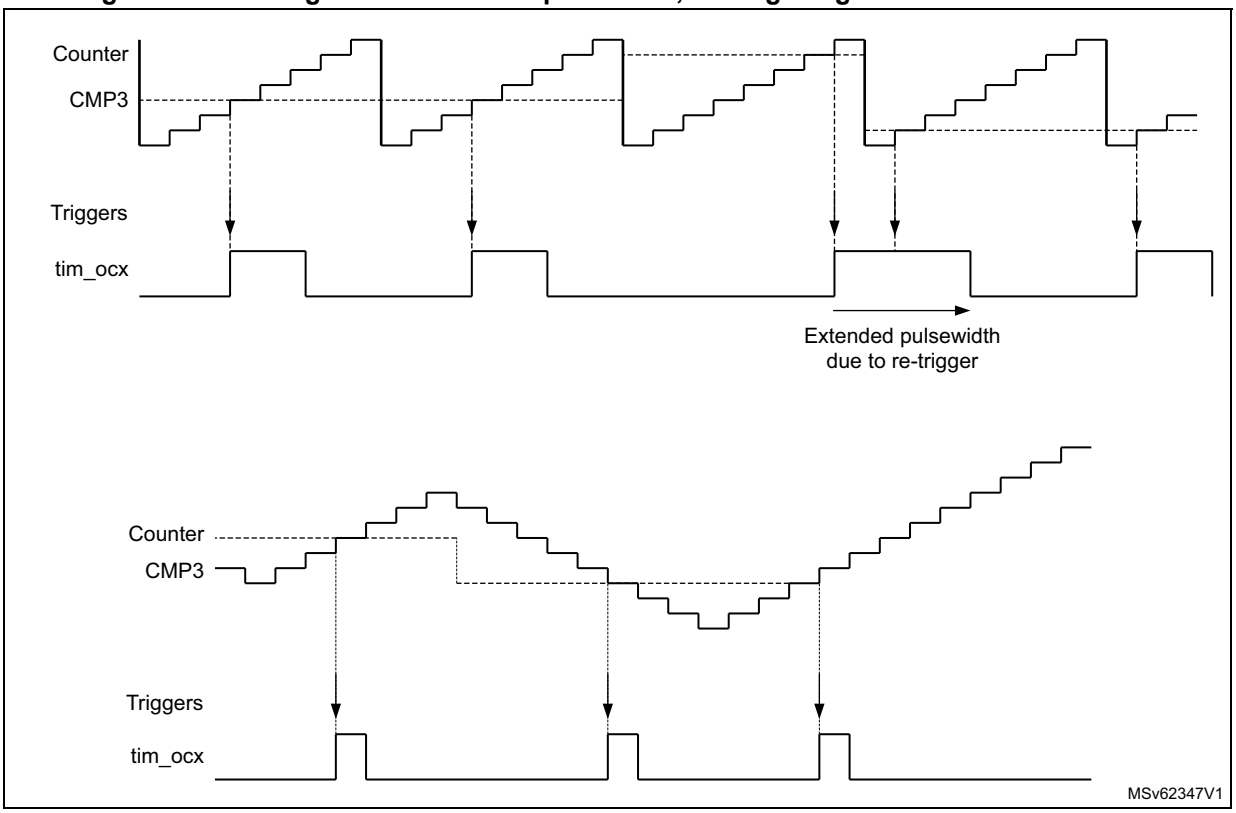


Figure 634 shows how the pulse is generated for edge-aligned and encoder operating modes.

Figure 634. Pulse generation on compare event, for edge-aligned and encoder modes



This output compare mode is selected using the OC3M[3:0] and OC4M[3:0] bitfields in TIMx_CCMR2 register.

The pulse width is programmed using the PW[7:0] bitfield in the register, using a specific clock prescaled according to PWPRSC[2:0] bits, as follows:

$t_{PW} = PW[7:0] \times t_{PWG}$

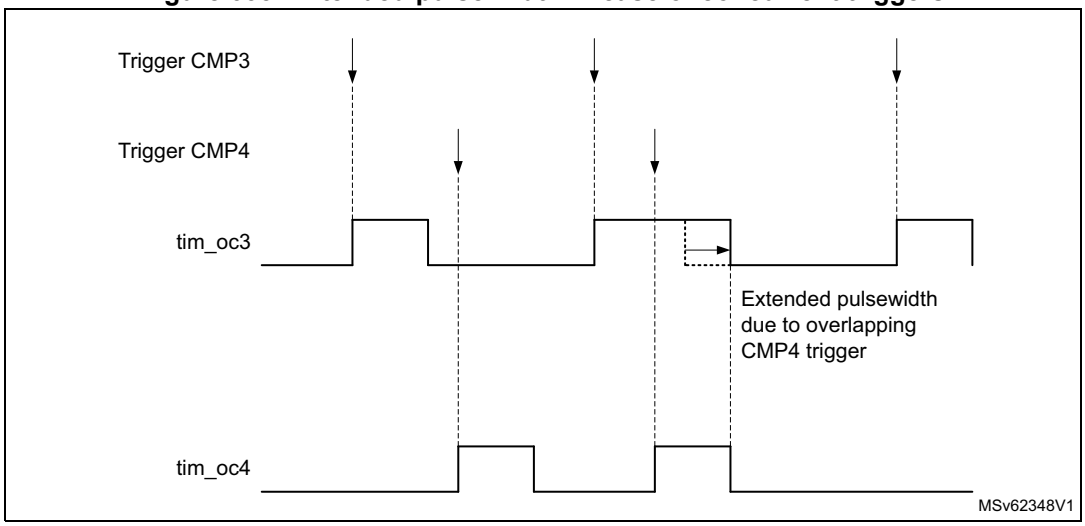
where $t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim_ker_ck}$.

gives the resolution and maximum values depending on the prescaler value.

The pulse is retriggerable: a new trigger while the pulse is ongoing, causes the pulse to be extended.

Note: If the two channels are enabled simultaneously, the pulses are issued independently as long as the trigger on one channel is not overlapping the pulse generated on the concurrent output. On the opposite, if the two triggers are overlapping, the pulse width related to the first arriving trigger is extended (because of the retrigger), while the pulse width of the last arriving trigger is correct (as shown on Figure 635).

Figure 635. Extended pulse width in case of concurrent triggers



54.4.18 Encoder interface mode

Quadrature encoder

To select Encoder interface mode write SMS = 0001 in the TIMx_SMCR register if the counter is counting on tim_ti1 edges only, SMS = 0010 if it is counting on tim_ti2 edges only and SMS = 0011 if it is counting on both tim_ti1 and tim_ti2 edges.

Select the tim_ti1 and tim_ti2 polarity by programming the CC1P and CC2P bits in the TIMx_CCER register. CC1NP and CC2NP must be kept cleared. When needed, the input filter can be programmed as well.

The two inputs tim_ti1 and tim_ti2 are used to interface to an incremental encoder. Refer to *Table 513*. The counter is clocked by each valid transition on tim_ti1fp1 or tim_ti2fp2 (tim_ti1 and tim_ti2 after input filter and polarity selection, tim_ti1fp1 = tim_ti1 if not filtered and not inverted, tim_ti2fp2 = tim_ti2 if not filtered and not inverted) assuming that it is enabled (CEN bit in TIMx_CR1 register written to 1). The sequence of transitions of the two inputs is evaluated and generates count pulses as well as the direction signal. Depending on the sequence the counter counts up or down, the DIR bit in the TIMx_CR1 register is modified by hardware accordingly. The DIR bit is calculated at each transition on any input (tim_ti1 or

tim_ti2), whatever the counter is counting on tim_ti1 only, tim_ti2 only or both tim_ti1 and tim_ti2.

Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the autoreload value in the TIMx_ARR register (0 to ARR or ARR down to 0 depending on the direction). So the TIMx_ARR must be configured before starting. In the same way, the capture, compare, prescaler, trigger output features continue to work as normal. Encoder mode and External clock mode 2 are not compatible and must not be selected together.

In this mode, the counter is modified automatically following the speed and the direction of the quadrature encoder and its content, therefore, always represents the encoder's position. The count direction corresponds to the rotation direction of the connected sensor. The table summarizes the possible combinations, assuming tim_ti1 and tim_ti2 do not switch at the same time.

Table 513. Counting direction versus encoder signals(CC1P = CC2P = 0)

Active edge	SMS[3:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
Counting on tim_ti1 only x1 mode	1110	High	Down	Up	No count	No count
		Low	No count	No count	No count	No count
Counting on tim_ti2 only x1 mode	1111	High	No count	No count	Up	Down
		Low	No count	No count	No count	No count
Counting on tim_ti1 only x2 mode	0001	High	Down	Up	No count	No count
		Low	Up	Down	No count	Down
Counting on tim_ti2 only x2 mode	0010	High	No count	No count	Up	Down
		Low	No count	No count	Down	Up
Counting on tim_ti1 and tim_ti2 x4 mode	0011	High	Down	Up	Up	Down
		Low	Up	Down	Down	Up

A quadrature encoder can be connected directly to the MCU without external interface logic. However, comparators are normally be used to convert the encoder's differential outputs to digital signals. This greatly increases noise immunity. The third encoder output which indicates the mechanical zero position, can be connected to the external trigger input and trigger a counter reset.

Figure 636 gives an example of counter operation, showing count signal generation and direction control. It also shows how input jitter is compensated where both edges are

selected. This might occur if the sensor is positioned near to one of the switching points. For this example we assume that the configuration is the following:

- CC1S = 01 (TIMx_CCMR1 register, tim_ti1fp1 mapped on tim_ti1).
- CC2S = 01 (TIMx_CCMR1 register, tim_ti2fp2 mapped on tim_ti2).
- CC1P and CC1NP = 0 (TIMx_CCER register, tim_ti1fp1 noninverted, tim_ti1fp1 = tim_ti1).
- CC2P and CC2NP = 0 (TIMx_CCER register, tim_ti2fp2 noninverted, tim_ti2fp2 = tim_ti2).
- SMS = 0011 (TIMx_SMCR register, both inputs are active on both rising and falling edges).
- CEN = 1 (TIMx_CR1 register, counter is enabled).

Figure 636. Example of counter operation in encoder interface mode

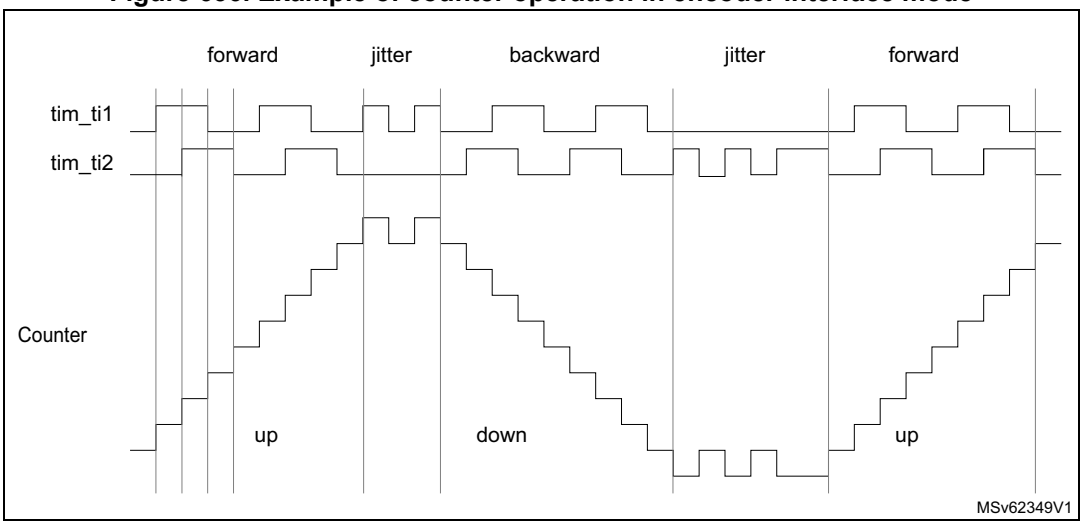


Figure 637 gives an example of counter behavior when tim_ti1fp1 polarity is inverted (same configuration as above except CC1P = 1).

Figure 637. Example of encoder interface mode with tim_ti1fp1 polarity inverted

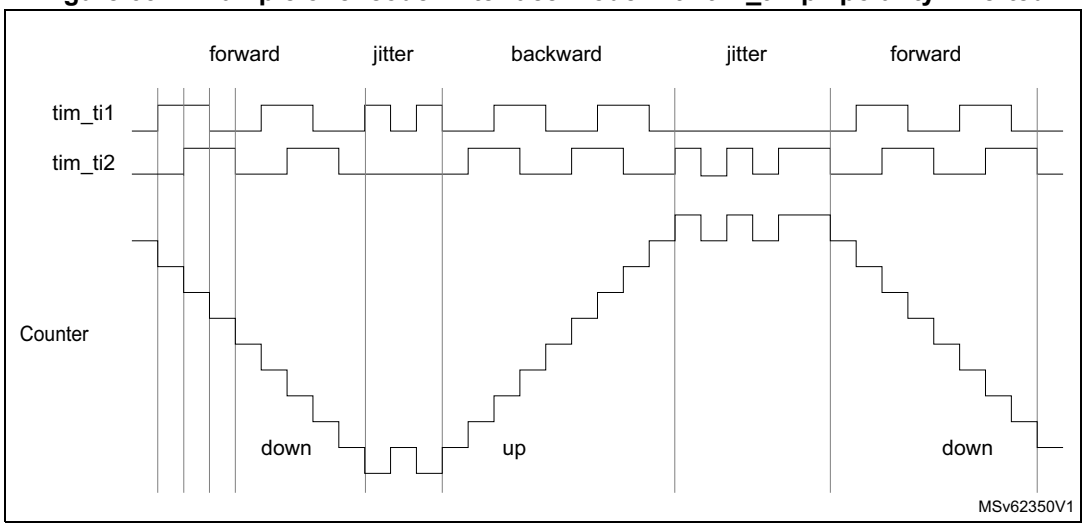
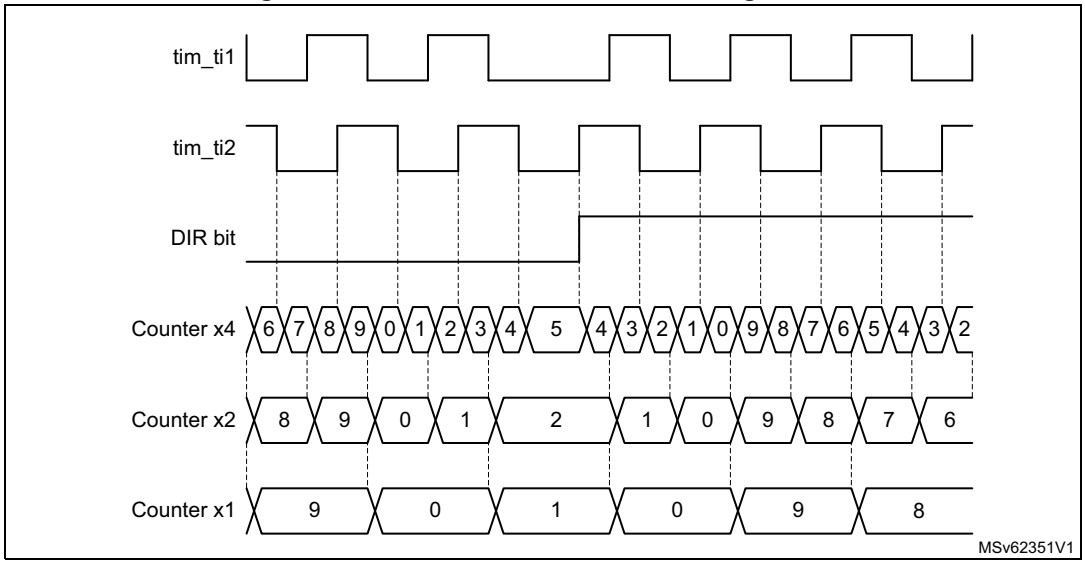


Figure 638 shows the timer counter value during a speed reversal, for various counting modes.

Figure 638. Quadrature encoder counting modes



The timer, when configured in Encoder Interface mode provides information on the sensor's current position. Dynamic information can be obtained (speed, acceleration, deceleration) by measuring the period between two encoder events using a second timer configured in capture mode. The output of the encoder which indicates the mechanical zero can be used for this purpose. Depending on the time between two events, the counter can also be read at regular times. This can be done by latching the counter value into a third input capture register if available (then the capture signal must be periodic and can be generated by another timer). When available, it is also possible to read its value through a DMA request.

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag (UIF) into the timer counter register's bit 31 (TIMx_CNT[31]). This allows both the counter value and a potential roll-over condition signaled by the UIFCPY flag to be read in an atomic way. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

Clock plus direction encoder mode

In addition to the quadrature encoder mode, the timer offers support for other types of encoders.

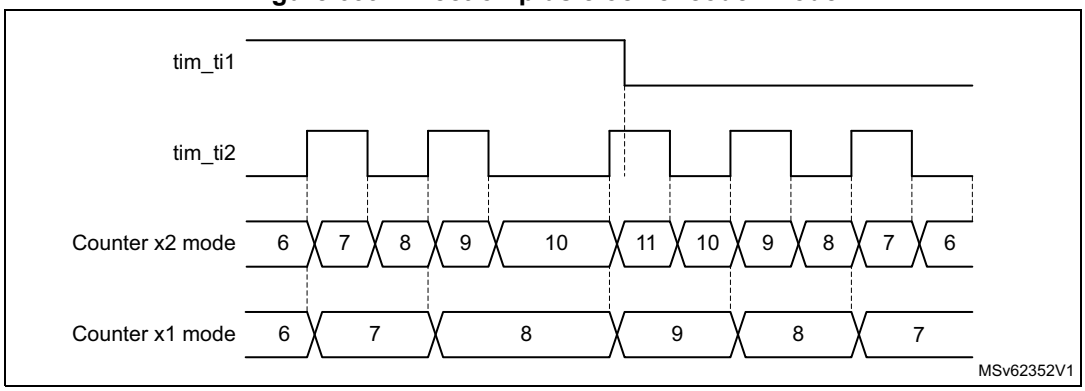
In the "clock plus direction" mode shown on *Figure 639*, the clock is provided on a single line, on tim_ti2, while the direction is forced using the tim_ti1 input.

This mode is enabled with the SMS[3:0] bitfield in the TIMx_SMCR register, as following:

- 1010: x2 mode, the counter is updated on both rising and falling edges of the clock.
- 1011: x1 mode, the counter is updated on a single clock edge, as per CC2P bit value: CC2P = 0 corresponds to rising edge sensitivity and CC2P = 1 corresponds to falling edge sensitivity.

The polarity of the direction signal on tim_ti1 is set with the CC1P bit: 0 corresponds to positive polarity (up-counting when tim_ti1 is high and down-counting when tim_ti1 is low) and CC1P = 1 corresponds to negative polarity (up-counting when tim_ti1 is low).

Figure 639. Direction plus clock encoder mode



Directional clock encoder mode

In the “directional clock” mode on *Figure 640*, the clocks are provided on two lines, with a single one at once, depending on the direction, so as to have one up-counting clock line and one down-counting clock line.

This mode is enabled with the SMS[3:0] bitfield in the TIMx_SMCR register, as following:

- 1100: x2 mode, the counter is updated on both rising and falling edges of any of the two clock lines. The CC1P and CC2P bits are coding for the clock idle state. CCxP = 0 corresponds to high-level idle state (refer to *Figure 640*) and CCxP = 1 corresponds to low-level idle state (refer to *Figure 641*).
- 1101: x1 mode, the counter is updated on a single clock edge, as per CC1P and CC2P bit value. CCxP = 0 corresponds to falling edge sensitivity and high-level idle state (refer to *Figure 640*), CCxP = 1 corresponds to rising edge sensitivity and low-level idle state (refer to *Figure 641*).

Figure 640. Directional clock encoder mode (CC1P = CC2P = 0)

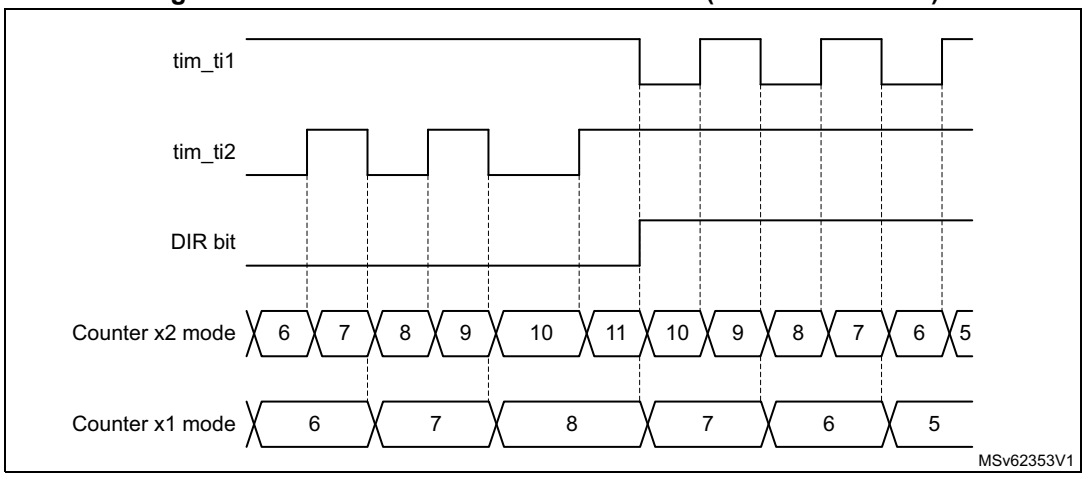


Figure 641. Directional clock encoder mode (CC1P = CC2P = 1)

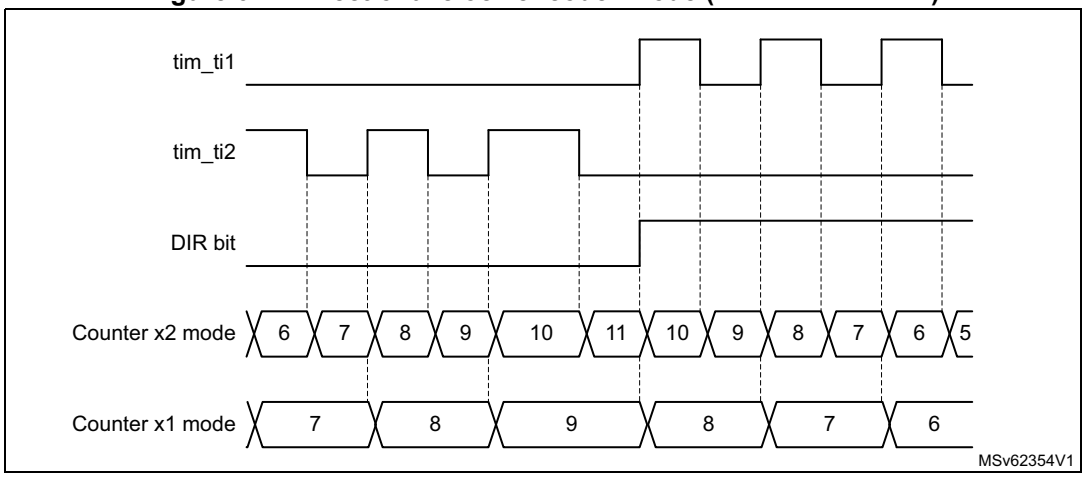


Table 514 details how the directional clock mode operates, for any input transition.

Table 514. Counting direction versus encoder signals and polarity settings

Directional clock mode	SMS[3:0]	Level on opposite signal (tim_ti1fp1 for tim_ti2, tim_ti2fp2 for tim_ti1)	tim_ti1fp1 signal		tim_ti2fp2 signal	
			Rising	Falling	Rising	Falling
x2 mode CCxP = 0	1100	High	Down	Down	Up	Up
		Low	No count	No count	No count	No count
x2 mode CCxP = 1	1100	High	No count	No count	No count	No count
		Low	Down	Down	Up	Up
x1 mode CCxP = 0	1101	High	No count	Down	No count	Up
		Low	No count	No count	No count	No count
x1 mode CCxP = 1	1101	High	No count	No count	No count	No count
		Low	Down	No count	Up	No count

Index input

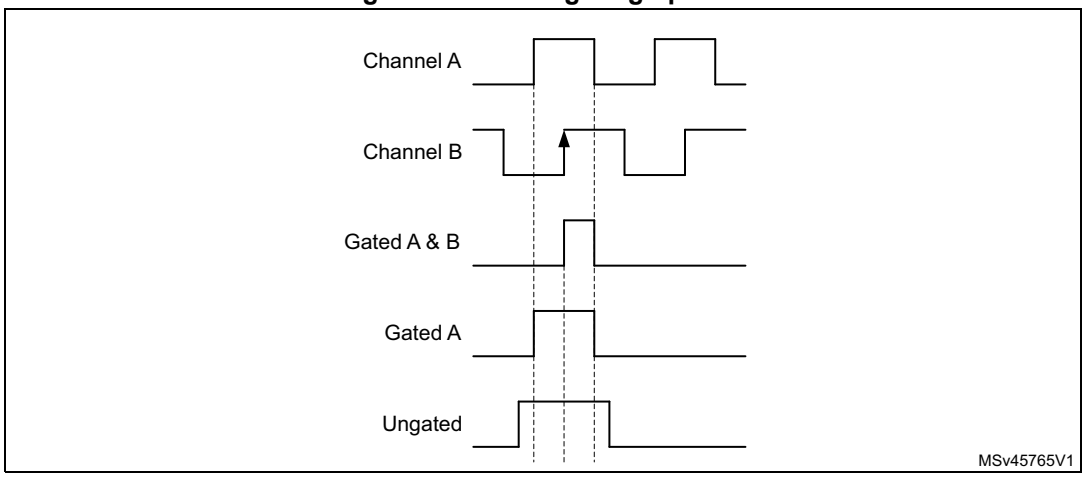
The counter can be reset by an index signal coming from the encoder, indicating an absolute reference position. The index signal must be connected to the tim_etr_in input. It can be filtered using the digital input filter.

The index functionality is enabled with the IE bit in the TIMx_ECR register. The IE bit must be set only in encoder mode, when the SMS[3:0] bitfield has the following values: 0001, 0010, 011, 1010, 1011, 1100, 1101, 1110, 1111.

Available encoders are proposed with several options for index pulse conditioning, as per *Figure 642*:

- Gated with A and B: the pulse width is 1/4 of one channel period, aligned with both A and B edges.
- Gated with A (or gated with B): the pulse width is 1/2 of one channel period, aligned with the two edges on channel A (resp. channel B).
- Ungated: the pulse width is up to one channel period, without any alignment to the edges.

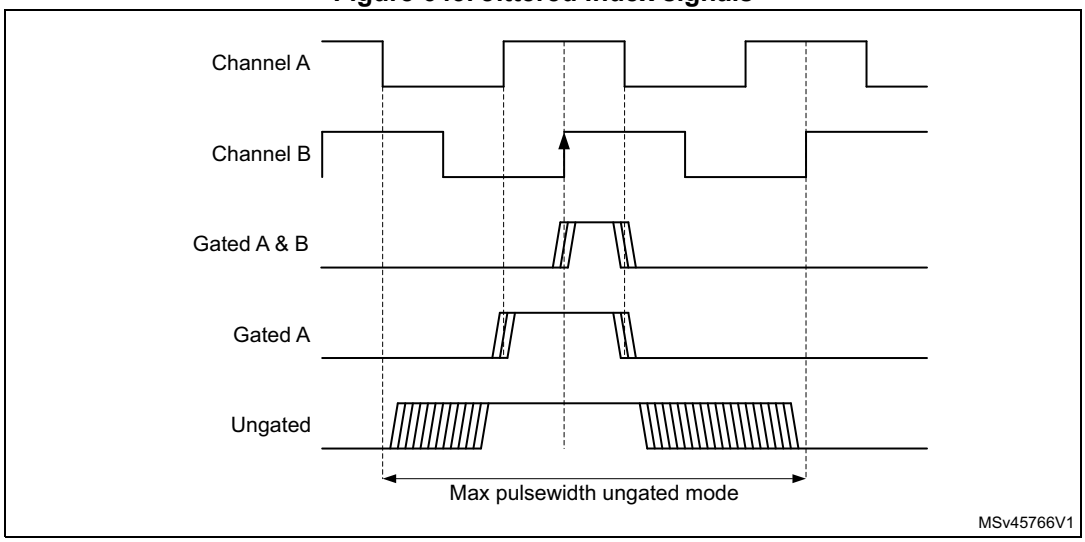
Figure 642. Index gating options



The circuitry tolerates jitter on index signal, whatever the gating mode, as shown on *Figure 643*.

In ungated mode, the signal must be strictly below two encoder periods. If the pulse width is greater or equal to two encoder period, the counter is reset multiple times.

Figure 643. Jittered Index signals



The timer supports the three gating options identically, without any specific programming needed. It is only necessary to define on which encoder state (for example channel A and channel B state combination) the index must be synchronized, using the IPOS[1:0] bitfield in the TIMx_ECR register.

The index detection event acts differently depending on counting direction to ensure symmetrical operation during speed reversal:

- The counter is reset during up-counting (DIR bit = 0).
- The counter is set to TIMx_ARR when down-counting.

This allows the index to be generated on the very same mechanical angular position whatever the counting direction. *Figure 644* shows at which position is the index generated, for a simplistic example (an encoder providing four edges par mechanical rotation).

Figure 644. Index generation for IPOS[1:0] = 11

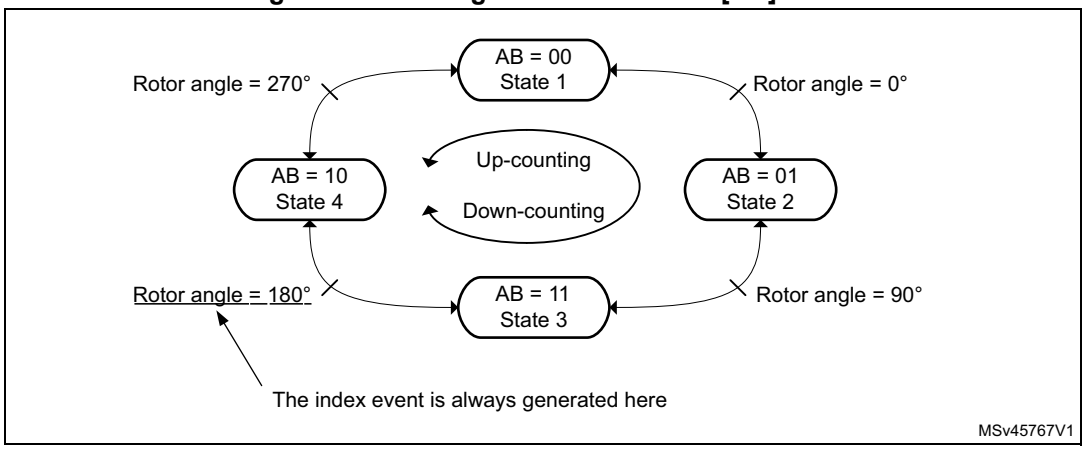


Figure 645 presents waveforms and corresponding values for IPOS[1:0] = 11. It shows that the instant at which the counter value is forced is automatically adjusted depending on the counting direction:

- Counter set to 0 when encoder state is 11 (ChA = 1, ChB = 1), when up-counting (DIR bit = 0).
- Counter set to TIMx_ARR when exiting the 11 state, when down-counting (DIR bit = 1).

An interrupt can be issued upon index detection event.

The arrows are indicating on which transition is the index event interrupt generated.

Figure 645. Counter reading with index gated on channel A (IPOS[1:0] = 11)

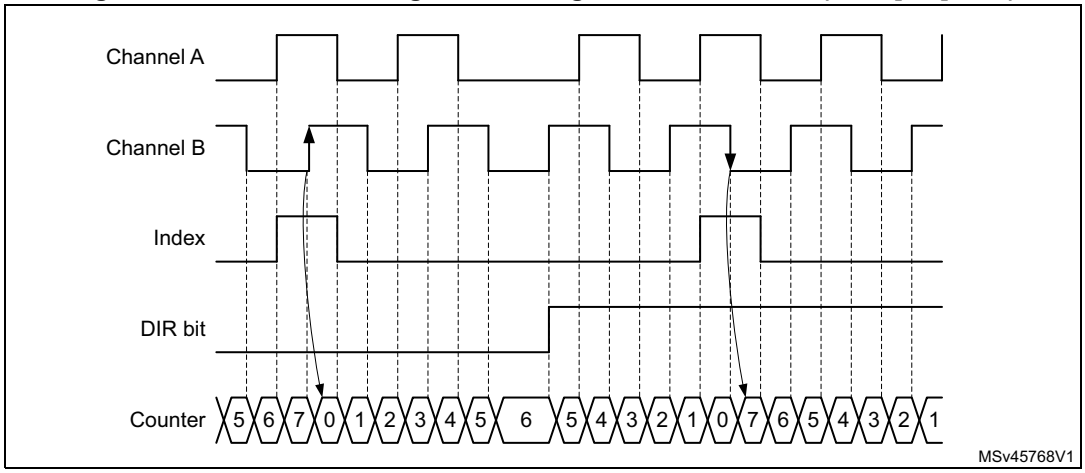


Figure 646 presents waveforms and corresponding values for the ungated mode. The arrows are indicating on which transition is the index event generated.

Figure 646. Counter reading with index ungated (IPOS[1:0] = 00)

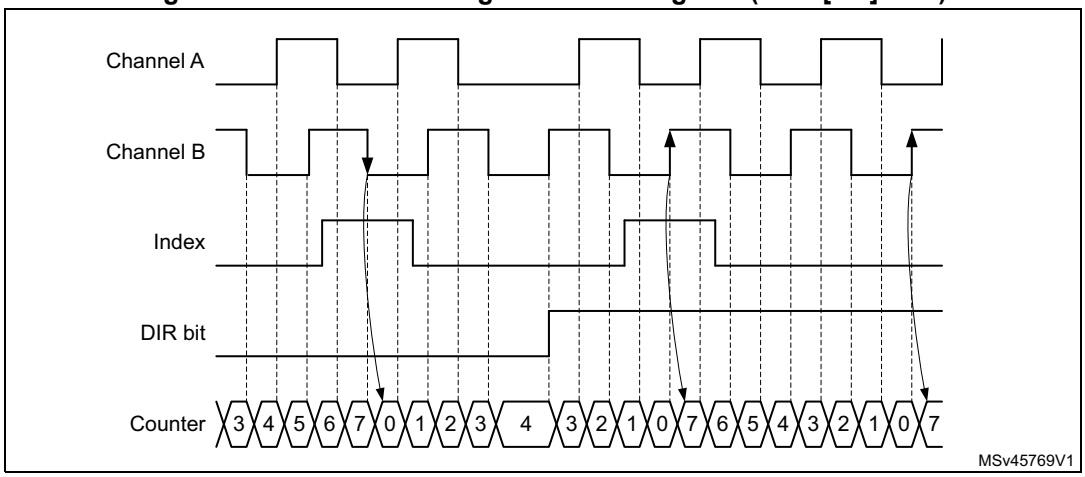


Figure 647 shows how the gated on A & B mode is handled, for various pulse alignment scenarios. The arrows are indicating on which transition is the index event generated.

Figure 647. Counter reading with index gated on channel A and B

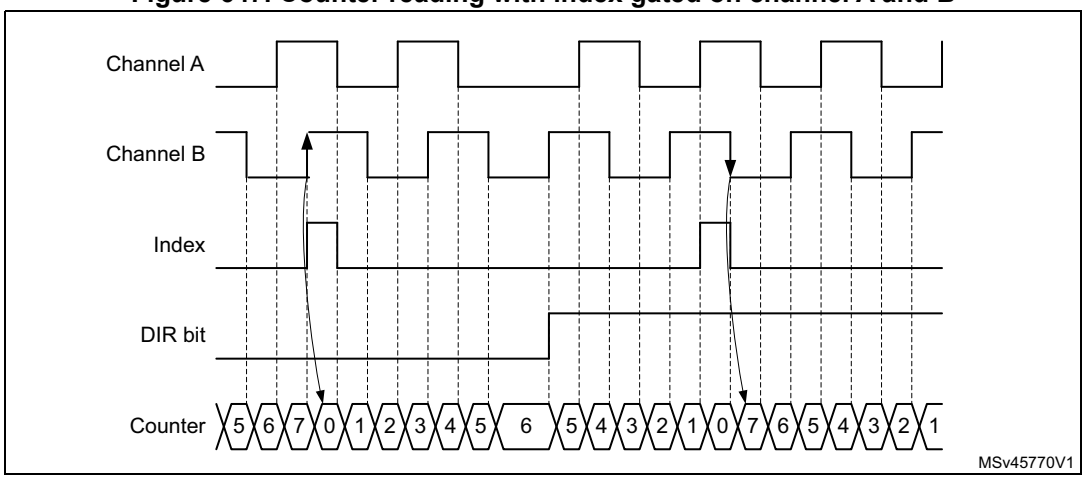


Figure 648 and *Figure 649* detail the case where the subsequent index pulse may be narrower than one quarter of the encoder clock period.

Figure 648. Encoder mode behavior in case of narrow index pulse (IPOS[1:0] = 11)

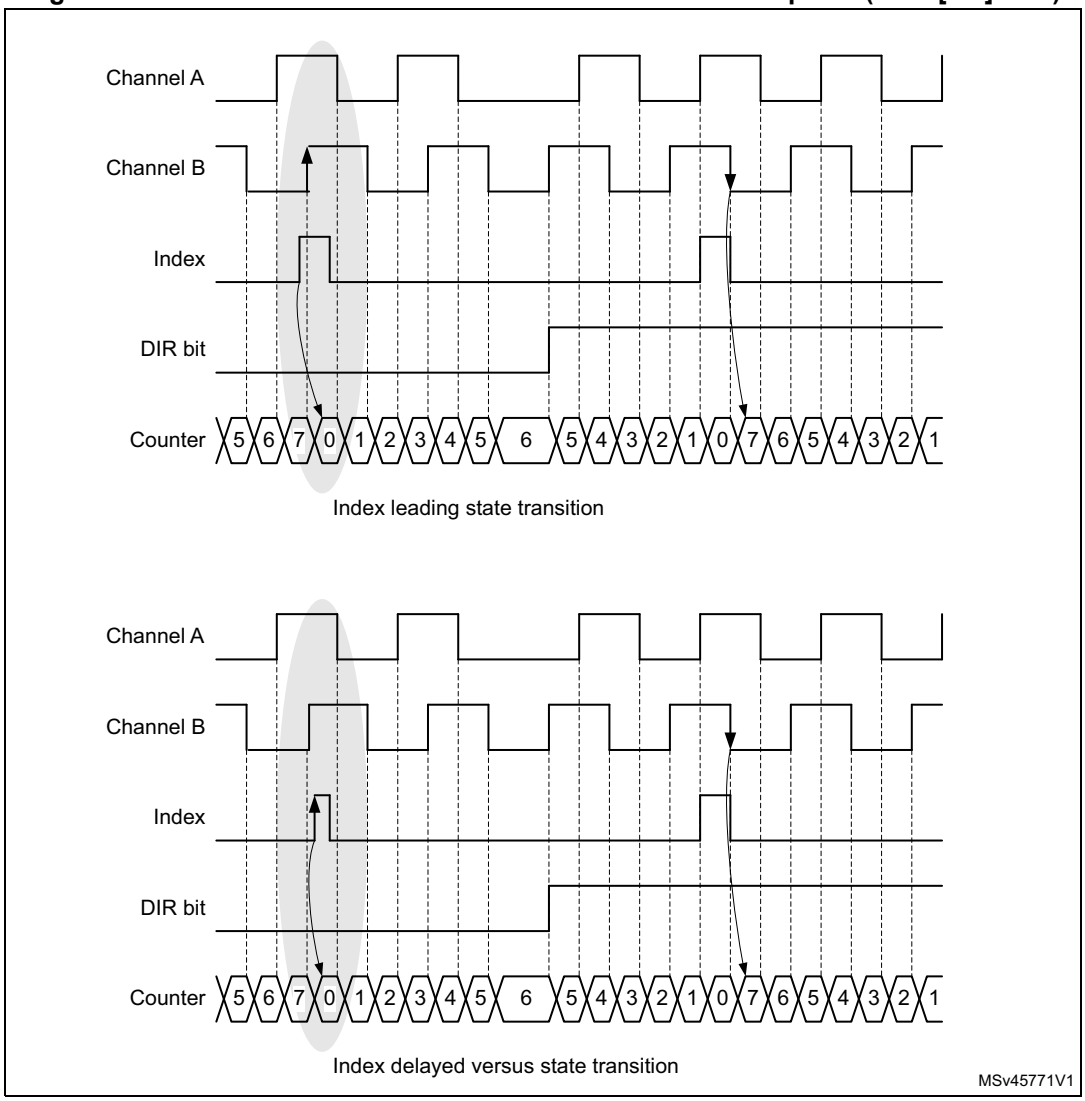


Figure 649. Counter reset Narrow index pulse (closer view, ARR = 0x07)

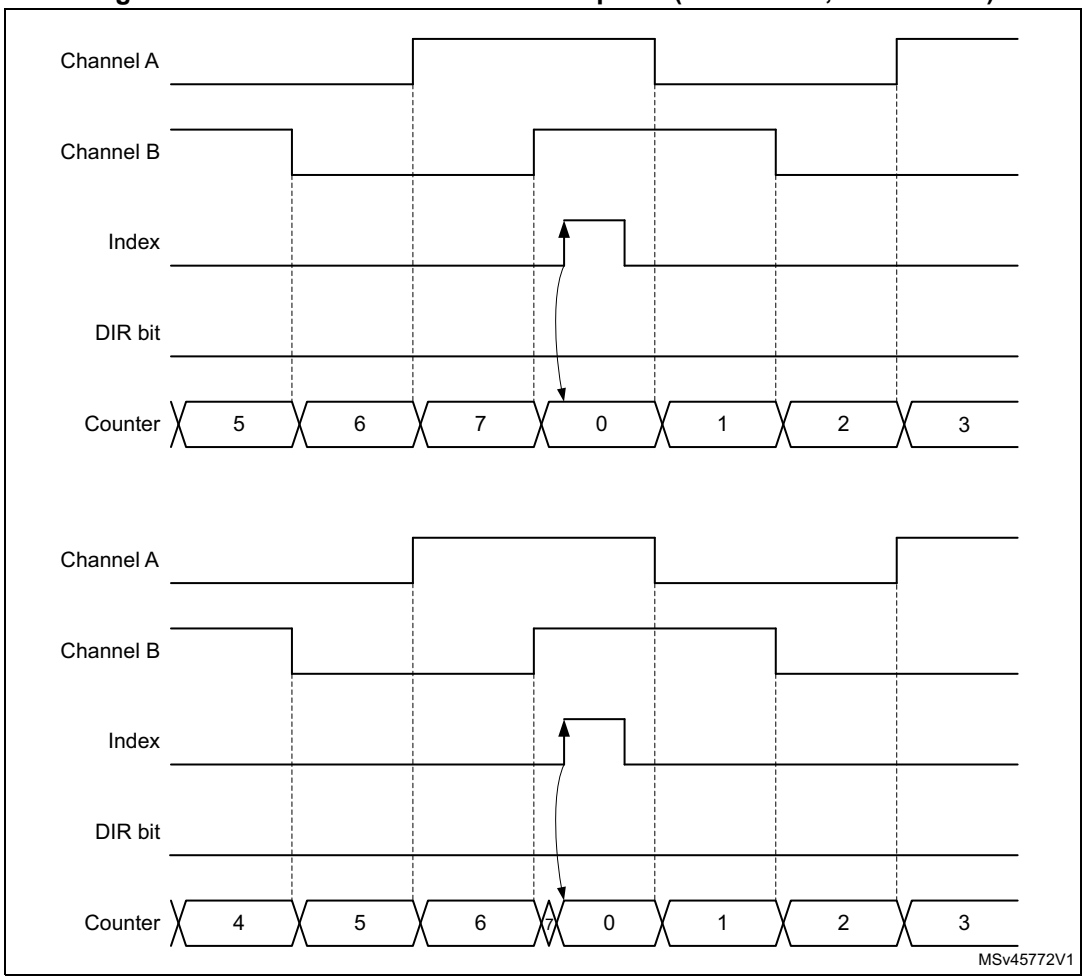
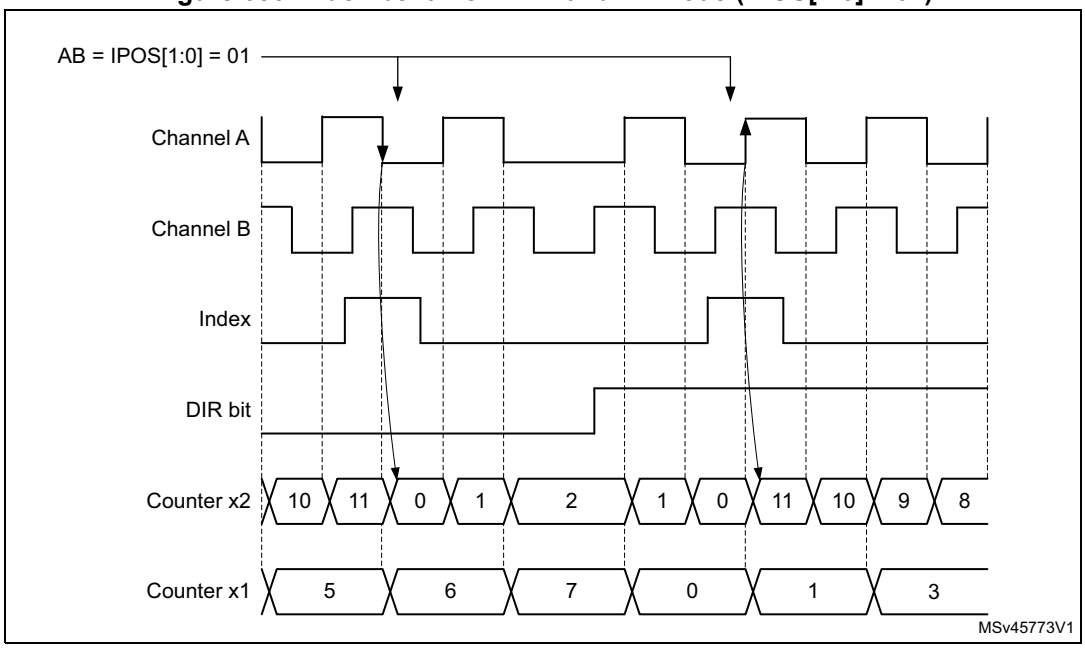


Figure 650 shows how the index is managed in x1 and x2 modes.

Figure 650. Index behavior in x1 and x2 mode (IPOS[1:0] = 01)

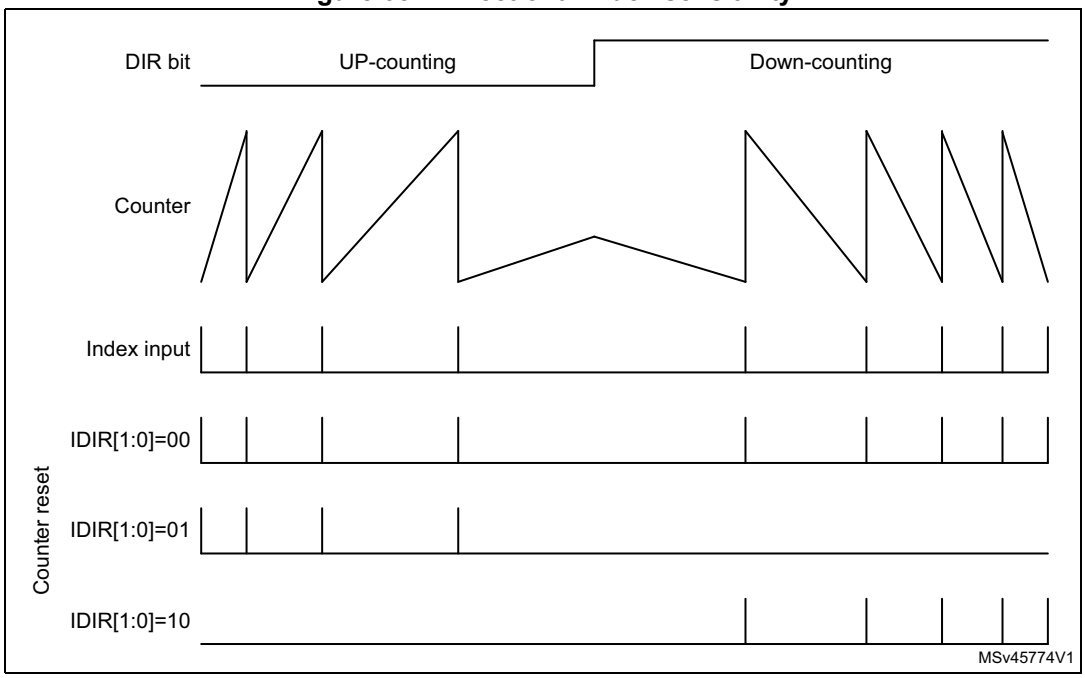


Directional index sensitivity

The IDIR[1:0] bitfield in the TIMx_ECR register allows the index to be active only in a selected counting direction.

Figure 651 shows the relationship between index and counter reset events, depending on IDIR[1:0] value.

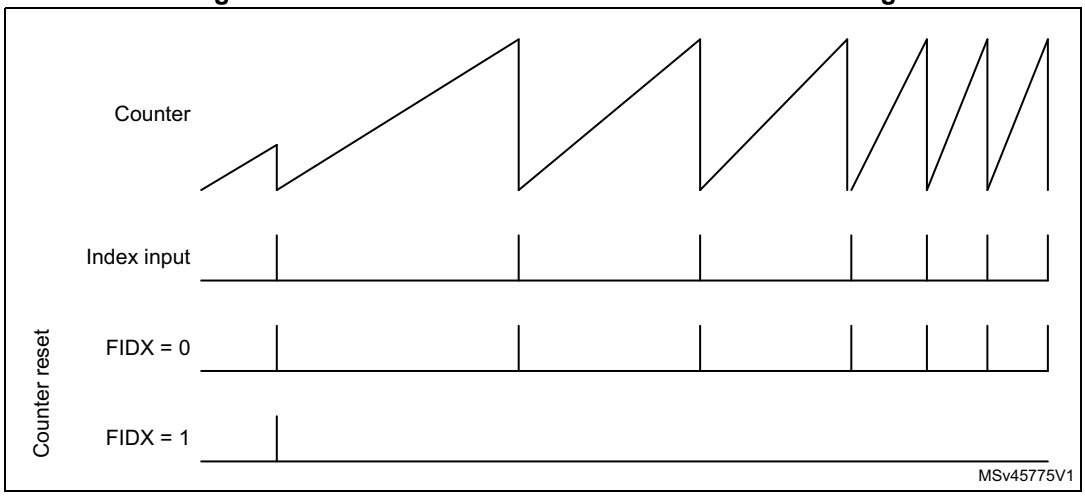
Figure 651. Directional index sensitivity



Special first index event management

The FIDX bit in the TIMx_ECR register allows the index to be taken only once, as shown on *Figure 652*. Once the first index has arrived, any subsequent index is ignored. If needed, the circuitry can be rearmed by writing the FIDX bit to 0 and setting it again to 1.

Figure 652. Counter reset as function of FIDX bit setting



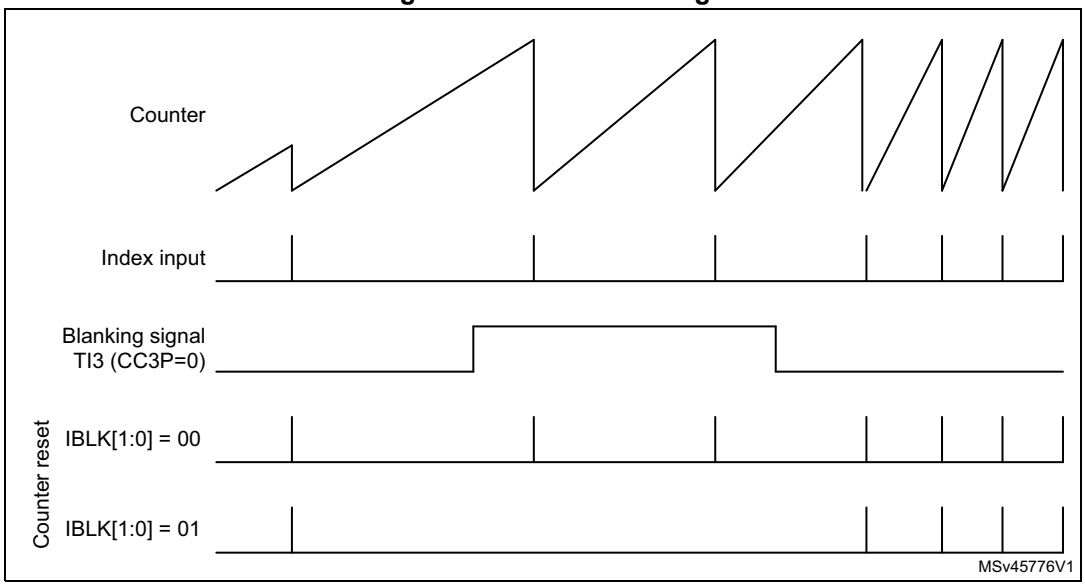
Index blanking

The index event can be blanked using the tim_ti3 or tim_ti4 inputs. During the blanking window, the index events are no longer resetting the counter, as shown on *Figure 653*.

This mode is enabled using the IBLK[1:0] bitfield in the TIMx_ECR register, as following:

- IBLK[1:0] = 00: Index signal always active.
- IBLK[1:0] = 01: Index signal blanking on tim_ti3 input.
- IBLK[1:0] = 10: Index signal blanking on tim_ti4 input.

Figure 653. Index blanking



Index management in nonquadrature mode

Figure 654 and *Figure 655* detail how the index is managed in directional clock mode and clock plus direction mode, when the SMS[3:0] bitfield is equal to 1010, 1011, 1100, 1101.

For both of these modes, the index sensitivity is set with the IPOS[0] bit as following:

- IPOS[0] = 0: Index is detected on clock low level.
- IPOS[0] = 1: Index is detected on clock high level.

The IPOS[1] bit is not-significant.

Figure 654. Index behavior in clock + direction mode, IPOS[0] = 1

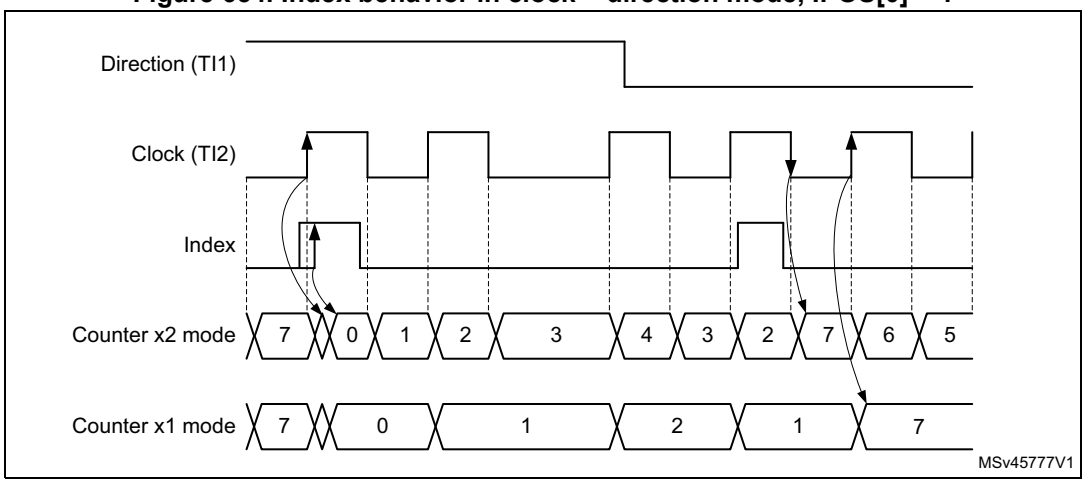
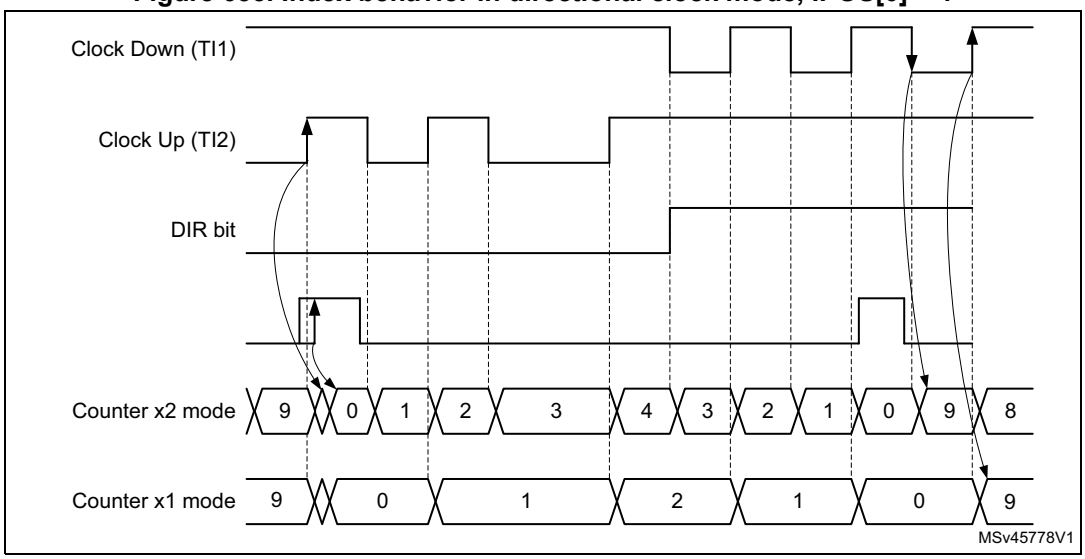


Figure 655. Index behavior in directional clock mode, IPOS[0] = 1

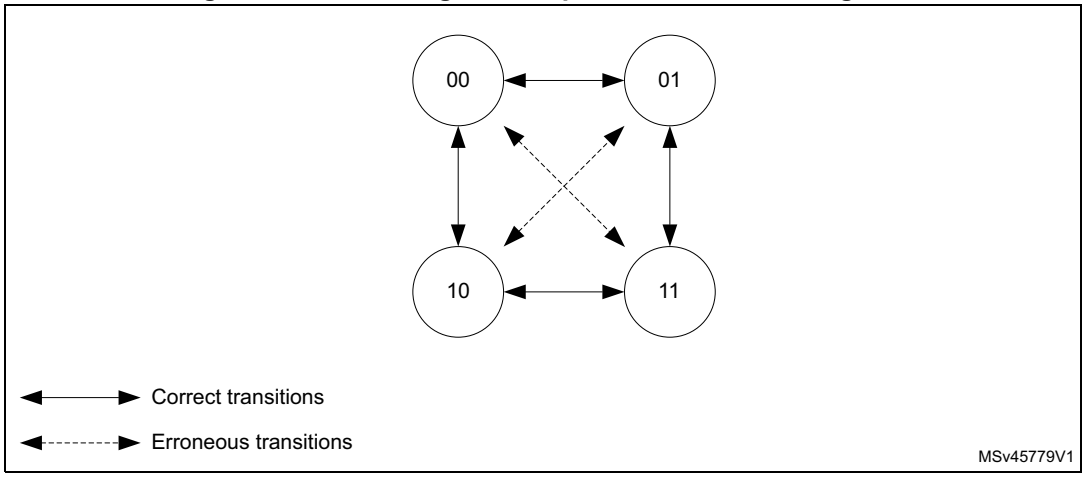


Encoder error management

For encoder configurations where two quadrature signals are available, it is possible to detect transition errors. The reading on the two inputs corresponds to a 2-bit gray code which can be represented as a state diagram, on *Figure 656*. A single bit is expected to change at once. An erroneous transition sets the TERRF interrupt flag in the TIMx_SR

status register. A transition error interrupt is generated if the TERRIE bit is set in the TIMx_DIER register.

Figure 656. State diagram for quadrature encoded signals



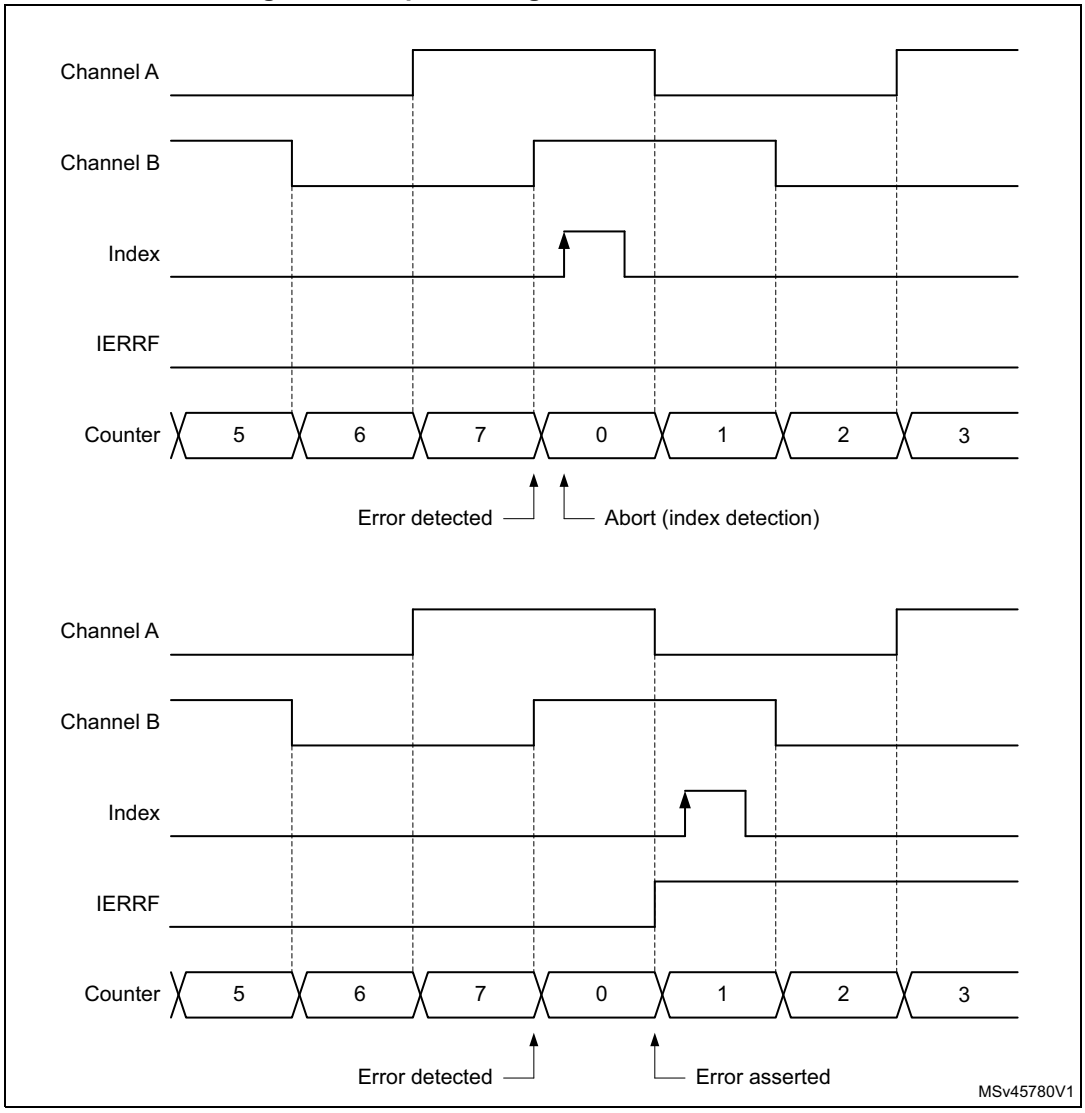
For encoder having an index signal, it is possible to detect abnormal operation resulting in an excess of pulses per revolution. An encoder with N pulses per revolution provides 4xN counts per revolution. The index signal resets the counter every 4xN clock periods.

If the counter value is incremented from TIMx_ARR to 0 or decremented from 0 to TIMx_ARR value without any index event, this is reported as an index position error.

The overflow threshold is programmed using the TIMx_ARR register. A 1000 lines encoder results in a counter value being between 0 and 3999 (in 4x reading mode). The overflow detection threshold must be programmed by setting TIMx_ARR = 3999 + 1 = 4000.

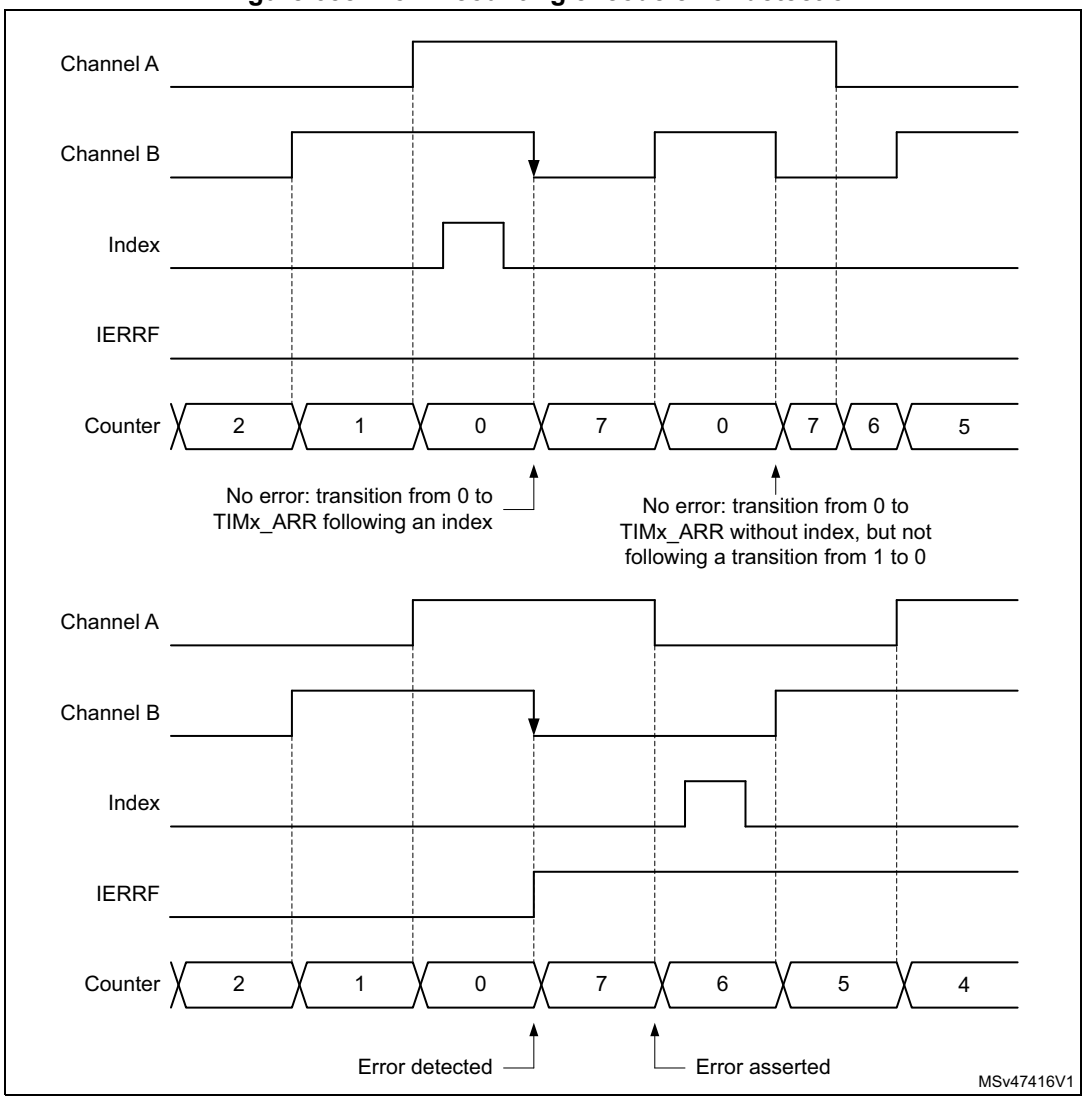
The error assertion is delayed to the transition 0 to 1 when in up-counting. This is to cope with narrow index pulses in gated A and B mode, as shown on *Figure 657*.

Figure 657. Up-counting encoder error detection



In down-counting mode, the detection is conditioned by a preliminary transition from 1 to 0. This is to cope with narrow index pulses in gated A and B mode, as shown on *Figure 658*, to avoid any false error detection in case the encoder dithers between TIMx_ARR and 0 immediately after the index detection.

Figure 658. Down-counting encode error detection



An index error sets the IERRF interrupt flag in the TIMx_SR status register. An index error interrupt is generated if the IERRIE bit is set in the TIMx_DIER register.

Functional encoder interrupts

The following interrupts are also available in encoder mode

- Direction change: any change of the counting direction in encoder mode causes the DIR bit in the TIMx_CR1 register to toggle. The direction change sets the DIRF interrupt flag in the TIMx_SR status register. A direction change interrupt is generated if the DIRIE bit is set in the TIMx_DIER register.
- Index event: the index event sets the IDXF interrupt flag in the TIMx_SR status register. An index interrupt is generated if the IDXIE bit is set in the TIMx_DIER register.

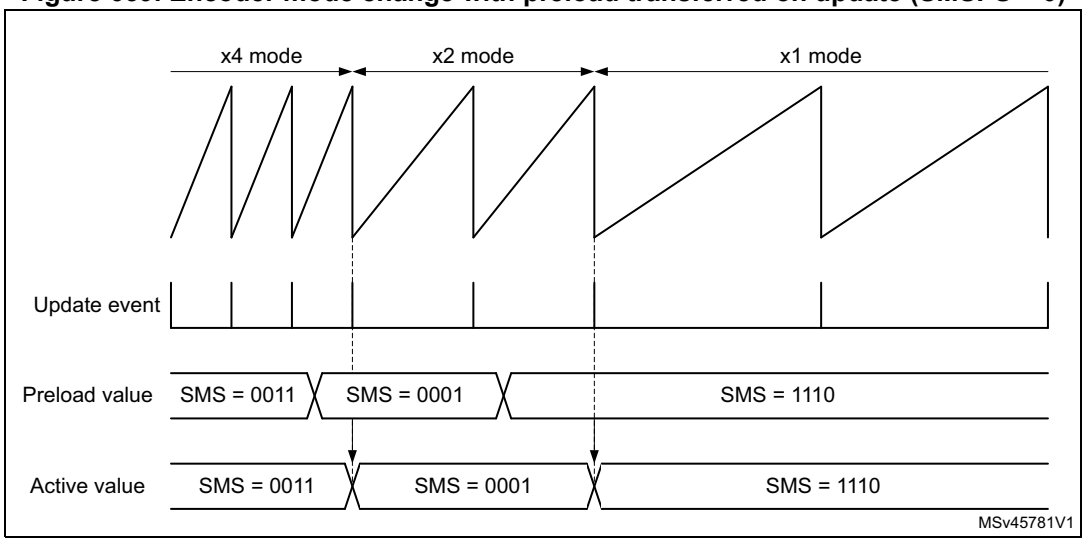
Slave mode selection preload for run-time encoder mode update

It can be necessary to switch from one encoder mode to another during run-time. This is typically done at high-speed to decrease the update interrupt rate, by switching from x4 to x2 to x1 mode, as shown on *Figure 659*.

For this purpose, the SMS[3:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIMx_SMCR register. The trigger for the transfer from SMS[3:0] preload to active value can be selected with the SMSPS bit in the TIMx_SMCR register.

- SMSPS = 0: the transfer is triggered by the update event (UEV) occurring when the counter overflows when up-counting, and underflows when down-counting.
- SMSPS = 1: the transfer is triggered by the index event.

Figure 659. Encoder mode change with preload transferred on update (SMSPS = 0)



Encoder clock output

The encoder mode operating principle is not perfectly suited for high-resolution velocity measurements, at low speed, as it requires a relatively long integration time to have a sufficient number of clock edges and a precise measurement.

At low speed, a better solution is to do an edge-to-edge clock period measurement. This can be achieved using a slave timer. The timer can output the encoder clock information on the tim_trgo output. The slave timer can then perform a period measurement and provide velocity information for each and every encoder clock edge.

This mode is enabled by setting the MMS[3:0] bitfield to 1000, in the TIMx_CR2 register. It is valid for the following SMS[3:0] values: 0001, 0010, 0011, 1010, 1011, 1100, 1101, 1110, 1111. Any other SMS[3:0] code is not allowed and may lead to unexpected behavior.

54.4.19 Direction bit output

It is possible to output a direction signal out of the timer, on the tim_oc3 and tim_oc4 output signals (copy of the DIR bit in the TIMx_CR1 register). This is achieved by setting the OC3M[3:0] or the OC4M[3:0] bitfield to 1011 in the TIMx_CCMR2 register.

This feature can be used for monitoring the counting direction (or rotation direction) in encoder mode, or to have a signal indicating the up/down phases in center-aligned PWM mode.

54.4.20 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag (UIF) into bit 31 of the timer counter register's bit 31 (TIMx_CNT[31]). This is used to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. It eases the calculation of angular speed by avoiding race conditions caused, for instance, by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the UIF and UIFCPY flag assertions.

In 32-bit timer implementations, when the IUFREMAP bit is set, bit 31 of the counter is overwritten by the UIFCPY flag upon read access (the counter's most significant bit is only accessible in write mode).

54.4.21 Timer input XOR function

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of an XOR gate, combining the three input pins tim_ti1, tim_ti2 and tim_ti3.

The XOR output can be used with all the timer input functions such as trigger or input capture.

An example of this feature used to interface Hall sensors is given in *Section 53.3.29: Interfacing with Hall sensors*.

54.4.22 Timers and external trigger synchronization

The TIMx timers can be synchronized with an external trigger in several modes: Reset mode, Gated mode, Trigger mode, Reset + trigger and gated + reset modes.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

In the following example, the upcounter is cleared in response to a rising edge on tim_ti1 input:

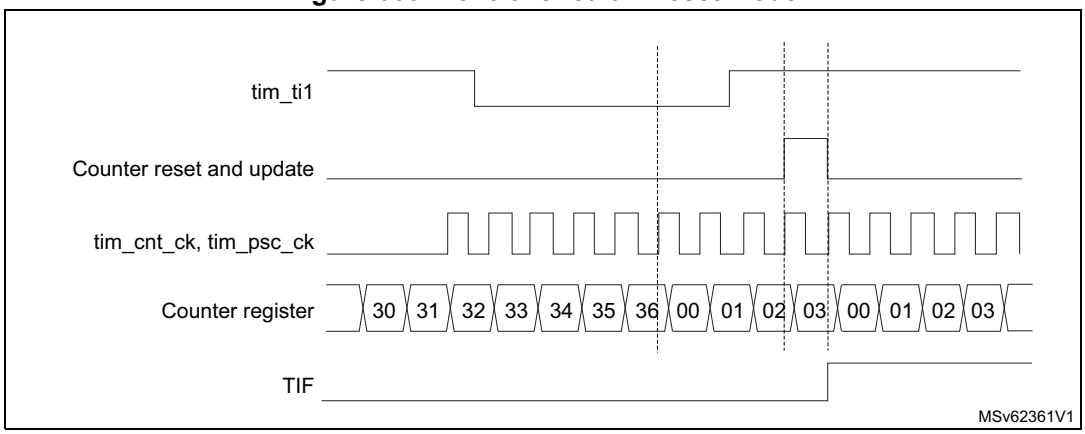
1. Configure the channel 1 to detect rising edges on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P = 0 and CC1NP = 0 in TIMx_CCER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS = 100 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIMx_SMCR register.
3. Start the counter by writing CEN = 1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until tim_ti1 rising edge. When tim_ti1 rises, the counter is cleared and restarts from 0. In the meantime, the

trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the autoreload register TIMx_ARR = 0x36. The delay between the rising edge on tim_ti1 and the actual reset of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 660. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

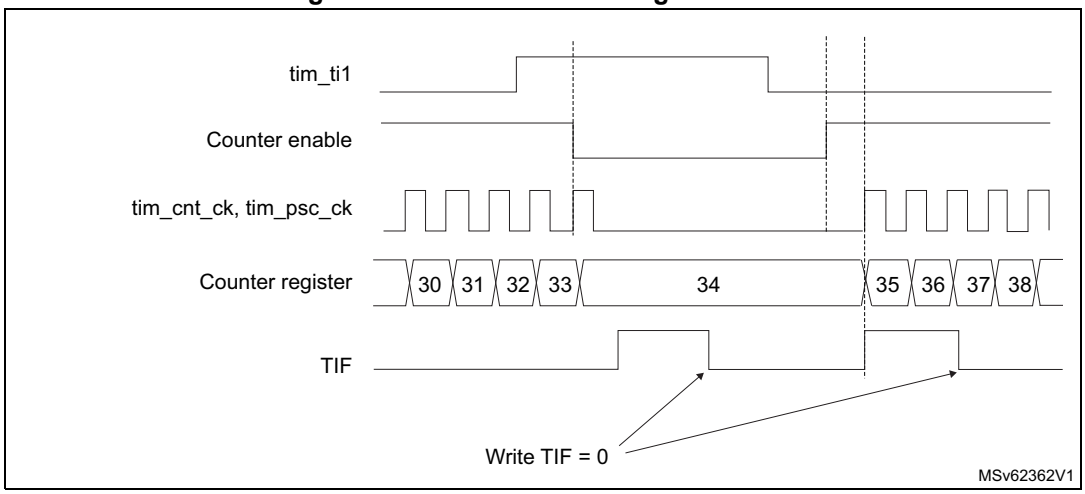
In the following example, the upcounter counts only when tim_ti1 input is low:

1. Configure the channel 1 to detect low levels on tim_ti1. Configure the input filter duration (in this example, we do not need any filter, so we keep IC1F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in TIMx_CCMR1 register. Write CC1P = 1 and CC1NP = 0 in TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in gated mode by writing SMS = 101 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIMx_SMCR register.
3. Enable the counter by writing CEN = 1 in the TIMx_CR1 register (in gated mode, the counter does not start if CEN = 0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as tim_ti1 is low and stops as soon as tim_ti1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on tim_ti1 and the actual stop of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 661. Control circuit in gated mode



Note: The configuration "CCxP = CCxNP = 1" (detection of both rising and falling edges) does not have any effect in gated mode because gated mode acts on a level and not on an edge.

Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

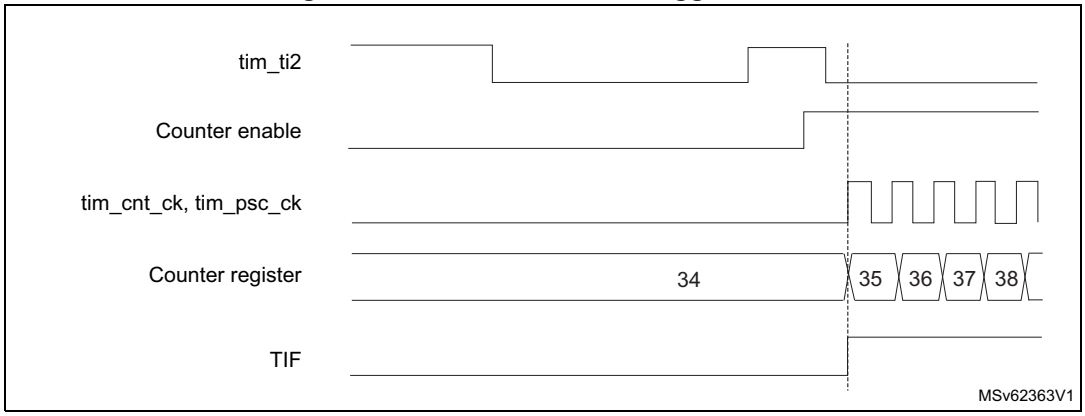
In the following example, the upcounter starts in response to a rising edge on tim_ti2 input:

1. Configure the channel 2 to detect rising edges on tim_ti2. Configure the input filter duration (in this example, we do not need any filter, so we keep IC2F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. CC2S bits are selecting the input capture source only, CC2S = 01 in TIMx_CCMR1 register. Write CC2P = 1 and CC2NP = 0 in TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in trigger mode by writing SMS = 110 in TIMx_SMCR register. Select tim_ti2 as the input source by writing TS = 00110 in TIMx_SMCR register.

When a rising edge occurs on tim_ti2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual start of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 662. Control circuit in trigger mode



Slave mode selection preload for run-time encoder mode update

The SMS[3:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIMx_SMCR register. The trigger for the transfer from SMS[3:0] preload to active value is the update event (UEV) occurring when the counter overflows.

Slave mode – combined reset + trigger mode

In this case, a rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

Slave mode – combined gated + reset mode

The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset as soon as the trigger becomes low. Both start and stop of the counter are controlled.

This mode is used to detect out-of-range PWM signal (duty cycle exceeding a maximum expected value).

Slave mode – external clock mode 2 + trigger mode

The external clock mode 2 can be used in addition to another slave mode (except external clock mode 1 and encoder mode). In this case, the tim_etr_in signal is used as external clock input, and another input can be selected as trigger input when operating in reset mode, gated mode, or trigger mode. It is recommended not to select tim_etr_in as tim_trgi through the TS bits of TIMx_SMCR register.

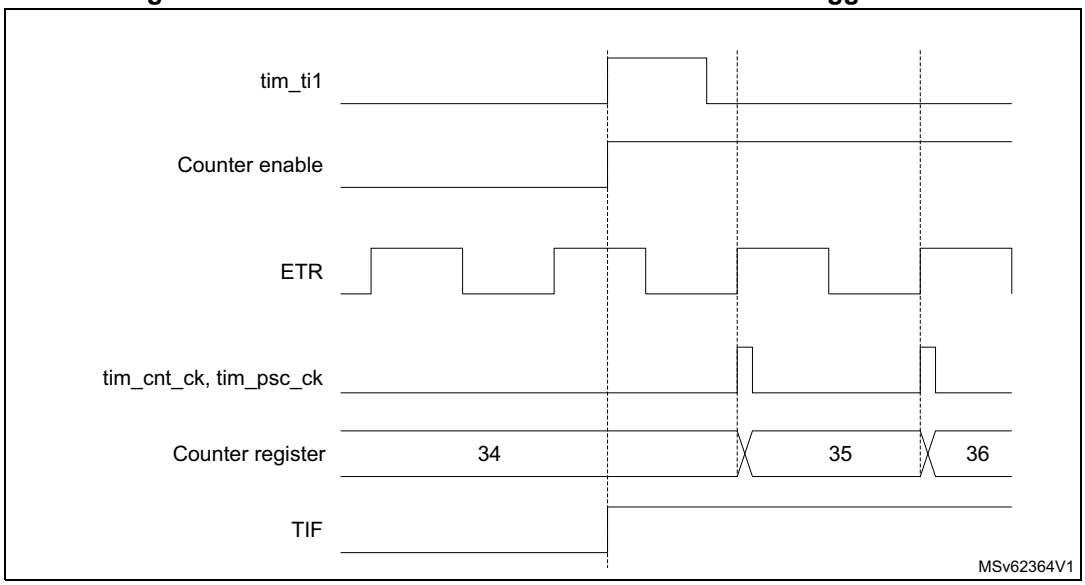
In the following example, the upcounter is incremented at each rising edge of the tim_etr_in signal as soon as a rising edge of tim_ti1 occurs:

1. Configure the external trigger input circuit by programming the TIMx_SMCR register as follows:
   - ETF = 0000: no filter.
   - ETPS = 00: prescaler disabled.
   - ETP = 0: detection of rising edges on tim_etr_in and ECE = 1 to enable the external clock mode 2.
2. Configure the channel 1 as follows, to detect rising edges on TI:
   - IC1F = 0000: no filter.
   - The capture prescaler is not used for triggering and does not need to be configured.
   - CC1S = 01in TIMx_CCMR1 register to select only the input capture source.
   - CC1P = 0 and CC1NP = 0 in TIMx_CCER register to validate the polarity (and detect rising edge only).
3. Configure the timer in trigger mode by writing SMS = 110 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIMx_SMCR register.

A rising edge on tim_ti1 enables the counter and sets the TIF flag. The counter then counts on tim_etr_in rising edges.

The delay between the rising edge of the tim_etr_in signal and the actual reset of the counter is due to the resynchronization circuit on tim_etrp input.

Figure 663. Control circuit in external clock mode 2 + trigger mode



54.4.23 Timer synchronization

The TIMx timers are linked together internally for timer synchronization or chaining. When one timer is configured in Master mode, it can reset, start, stop, or clock the counter of another timer configured in Slave mode.

Figure 664 and *Figure 665* show examples of master/slave timer connections.

Figure 664. Master/Slave timer example

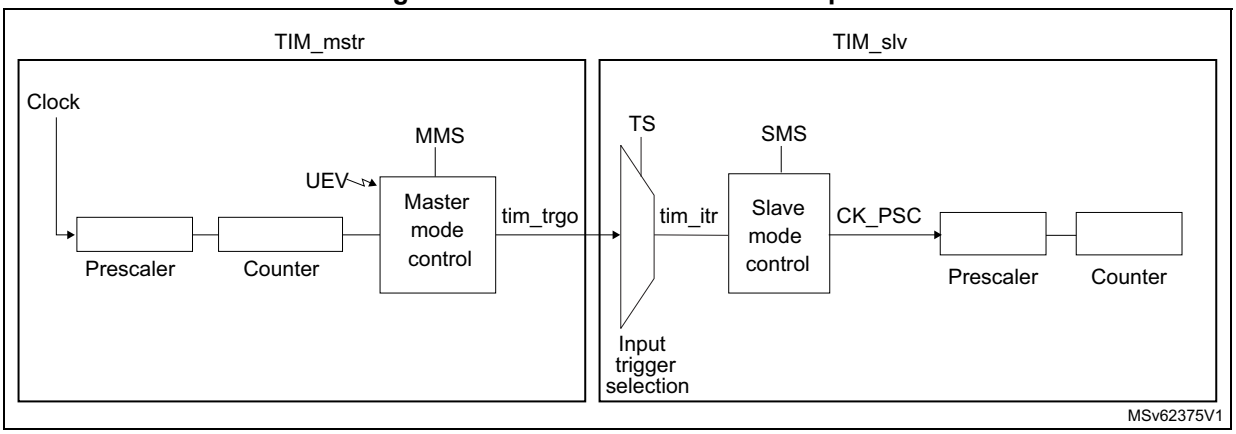
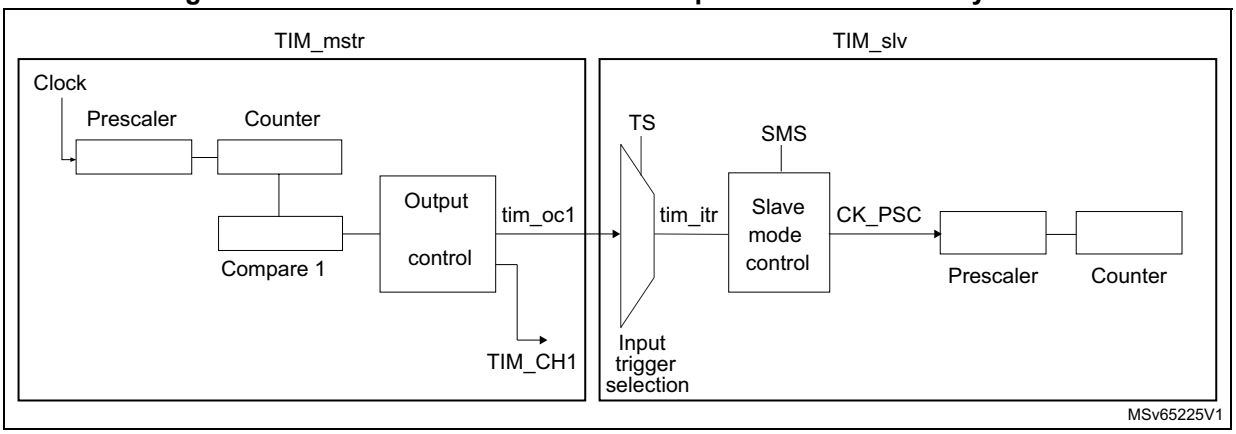


Figure 665. Master/slave connection example with 1 channel only timers



Note: The timers with one channel only (see Figure 665) do not feature a master mode. However, the tim_oc1 output signal can serve as trigger for slave timer (see TIMx internal trigger connection table in Section 54.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals).
The tim_oc1 signal pulse width must be programmed to be at least two clock cycles of the destination timer, to make sure the slave timer detects the trigger.
For instance, if the destination timer tim_ker_ck clock is four times slower than the source timer, the OC1 pulse width must be eight clock cycles.

Using one timer as prescaler for another timer

For example, TIM_mstr can be configured to act as a prescaler for TIM_slv. Refer to *Figure 664*. To do this:

1. Configure TIM_mstr in master mode so that it outputs a periodic trigger signal on each update event UEV. If MMS = 010 is written in the TIM_mstr_CR2 register, a rising edge is output on tim_trgo each time an update event is generated.
2. To connect the tim_trgo output of TIM_mstr to TIM_slv, TIM_slv must be configured in slave mode using ITR2 as internal trigger. This is selected through the TS bits in the TIM_slv_SMCR register (writing TS = 00010).
3. Then the slave mode controller must be put in external clock mode 1 (write SMS = 111 in the TIM_slv_SMCR register). This causes TIM_slv to be clocked by the rising edge of the periodic TIM_mstr trigger signal (which correspond to the TIM_mstr counter overflow).
4. Finally both timers must be enabled by setting their respective CEN bits (TIMx_CR1 register).

Note: If tim_ocx is selected on TIM_mstr as the trigger output (MMS = 1xx), its rising edge is used to clock the counter of TIM_slv.

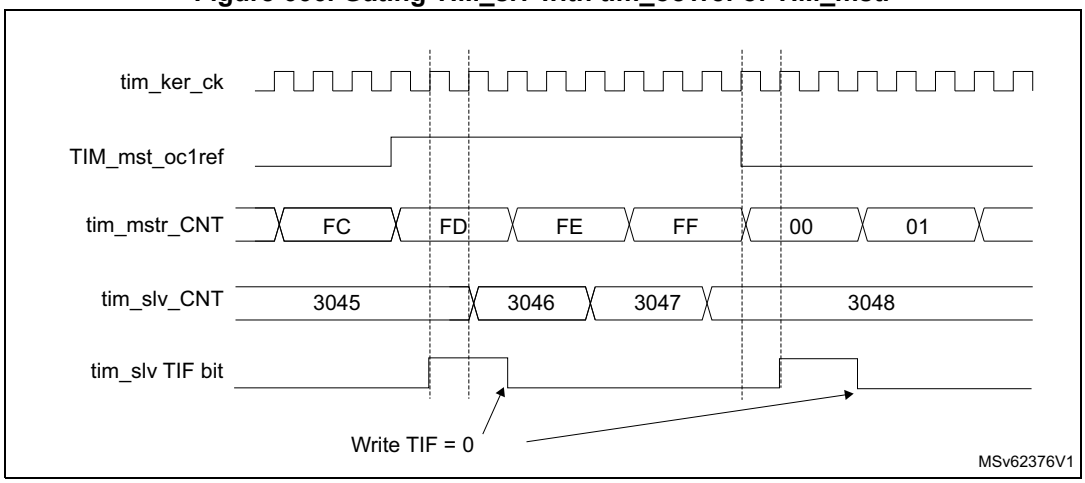
Using one timer to enable another timer

In this example, we control the enable of TIM_slv with the output compare 1 of TIM_mstr. Refer to *Figure 664* for connections. TIM_slv counts on the divided internal clock only when tim_oc1ref of TIM_mstr is high. Both counter clock frequencies are divided by 3 by the prescaler compared to tim_ker_ck ($f_{tim_cnt_ck} = f_{tim_ker_ck}/3$).

1. Configure TIM_mstr master mode to send its output compare 1 reference (tim_oc1ref) signal as trigger output (MMS = 100 in the TIM_mstr_CR2 register).
2. Configure the TIM_mstr tim_oc1ref waveform (TIM_mstr_CCMR1 register).
3. Configure TIM_slv to get the input trigger from TIM_mstr (TS = 00010 in the TIM_slv_SMCR register).
4. Configure TIM_slv in gated mode (SMS = 101 in TIM_slv_SMCR register).
5. Enable TIM_slv by writing 1 in the CEN bit (TIM_slv_CR1 register).
6. Start TIM_mstr by writing 1 in the CEN bit (TIM_mstr_CR1 register).

Note: The slave timer counter clock is not synchronized with the master timer counter clock, this mode only affects the TIM_slv counter enable signal.

Figure 666. Gating TIM_slv with tim_oc1ref of TIM_mstr

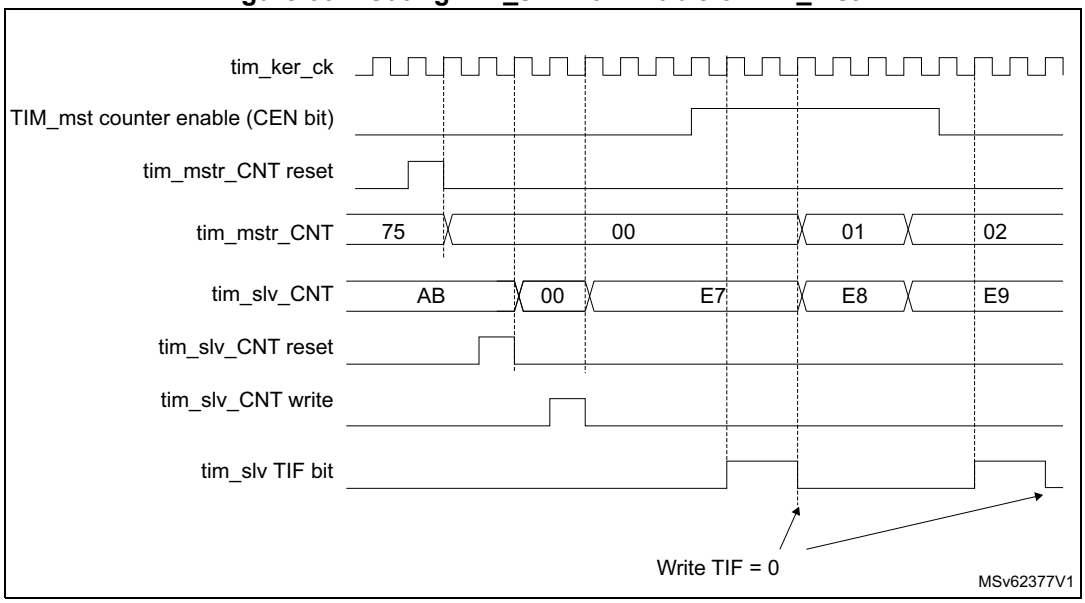


In the example in *Figure 666*, the TIM_slv counter and prescaler are not initialized before being started. So they start counting from their current value. It is possible to start from a given value by resetting both timers before starting TIM_mstr. Then any value can be written in the timer counters. The timers can easily be reset by software using the UG bit in the TIMx_EGR registers.

In the next example (refer to *Figure 667*), we synchronize TIM_mstr and TIM_slv. TIM_mstr is the master and starts from 0. TIM_slv is the slave and starts from 0xE7. The prescaler ratio is the same for both timers. TIM_slv stops when TIM_mstr is disabled by writing 0 to the CEN bit in the TIM_mstr_CR1 register:

1. Configure TIM_mstr master mode to send its output compare 1 reference (tim_oc1ref) signal as trigger output (MMS = 100 in the TIM_mstr_CR2 register).
2. Configure the TIM_mstr tim_oc1ref waveform (TIM_mstr_CCMR1 register).
3. Configure TIM_slv to get the input trigger from TIM_mstr (TS = 00010 in the TIM_slv_SMCR register).
4. Configure TIM_slv in gated mode (SMS = 101 in TIM_slv_SMCR register).
5. Reset TIM_mstr by writing 1 in UG bit (TIM_mstr_EGR register).
6. Reset TIM_slv by writing 1 in UG bit (TIM_slv_EGR register).
7. Initialize TIM_slv to 0xE7 by writing 0xE7 in the TIM_slv counter (TIM_slv_CNT).
8. Enable TIM_slv by writing 1 in the CEN bit (TIM_slv_CR1 register).
9. Start TIM_mstr by writing 1 in the CEN bit (TIM_mstr_CR1 register).
10. Stop TIM_mstr by writing 0 in the CEN bit (TIM_mstr_CR1 register).

Figure 667. Gating TIM_slv with Enable of TIM_mstr

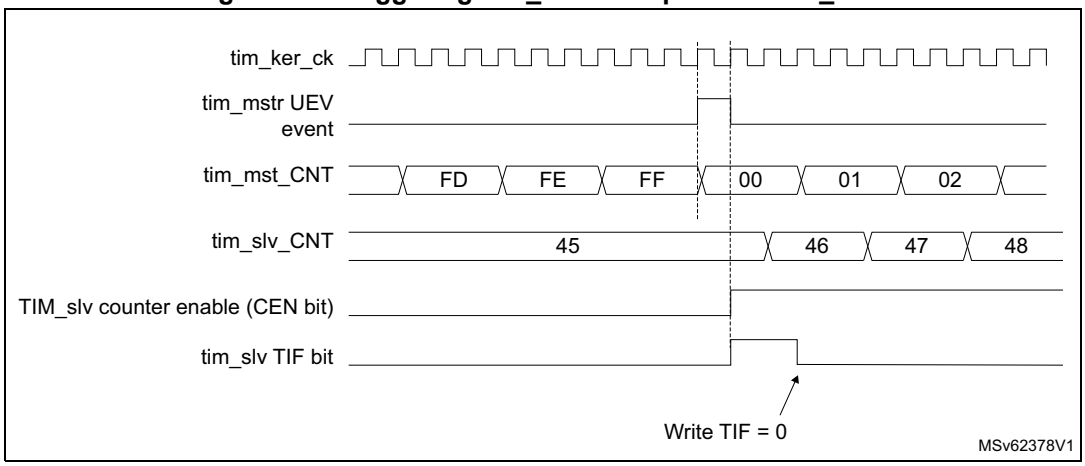


Using one timer to start another timer

In this example, we set the enable of TIM_slv with the update event of TIM_mstr. Refer to *Figure 664* for connections. TIM_slv starts counting from its current value (which can be nonzero) on the divided internal clock as soon as the update event is generated by TIM_mstr. When TIM_slv receives the trigger signal its CEN bit is automatically set and the counter counts until we write 0 to the CEN bit in the TIM_slv_CR1 register. Both counter clock frequencies are divided by 3 by the prescaler compared to tim_ker_ck ($f_{tim_cnt_ck} = f_{tim_ker_ck}/3$).

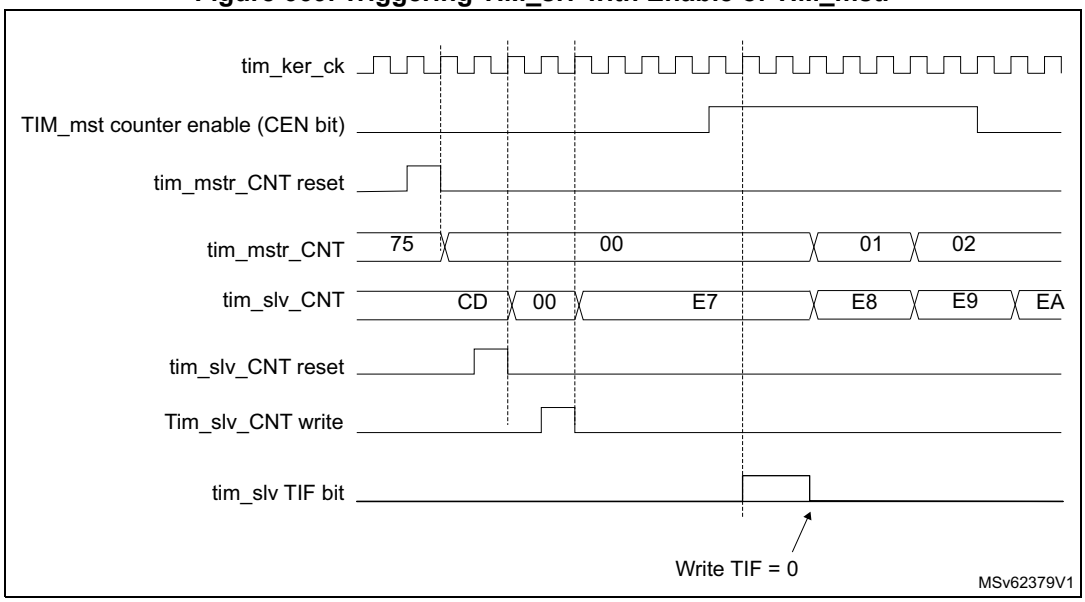
1. Configure TIM_mstr master mode to send its update event (UEV) as trigger output (MMS = 010 in the TIM_mstr_CR2 register).
2. Configure the TIM_mstr period (TIM_mstr_ARR registers).
3. Configure TIM_slv to get the input trigger from TIM_mstr (TS = 00010 in the TIM_slv_SMCR register).
4. Configure TIM_slv in trigger mode (SMS = 110 in TIM_slv_SMCR register).
5. Start TIM_mstr by writing 1 in the CEN bit (TIM_mstr_CR1 register).

Figure 668. Triggering TIM_slv with update of TIM_mstr



As in the previous example, both counters can be initialized before starting counting. *Figure 669* shows the behavior with the same configuration as in *Figure 668* but in trigger mode (SMS = 110 in the TIM_slv_SMCR register) instead of gated mode.

Figure 669. Triggering TIM_slv with Enable of TIM_mstr



Starting two timers synchronously in response to an external trigger

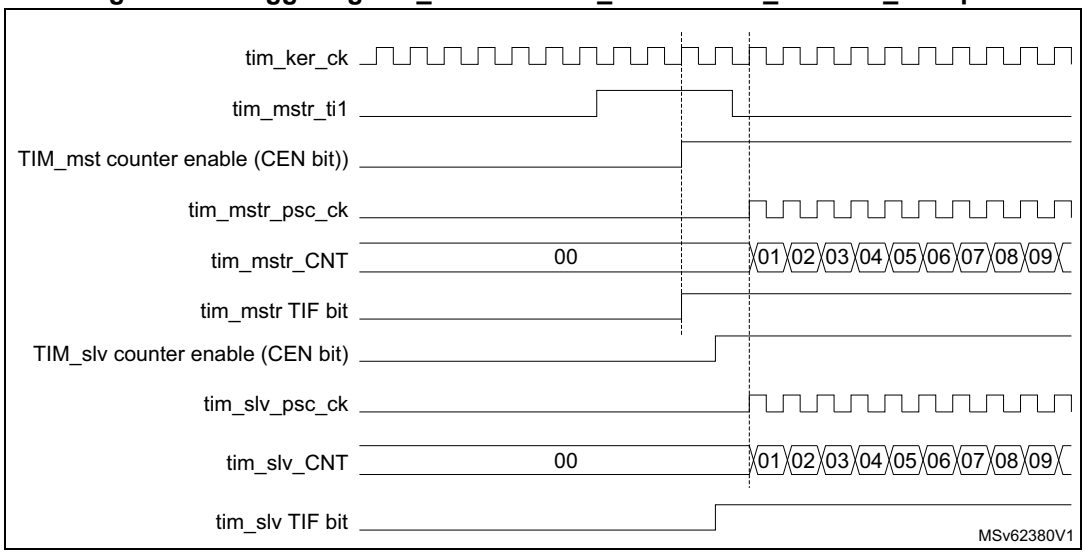
In this example, we set the enable of TIM_mstr when its tim_ti1 input rises, and the enable of TIM_slv with the enable of TIM_mstr. Refer to *Figure 664* for connections. To ensure the counters are aligned, TIM_mstr must be configured in Master/Slave mode (slave with respect to tim_ti1, master with respect to TIM_slv):

1. Configure TIM_mstr master mode to send its enable as trigger output (MMS = 001 in the TIM_mstr_CR2 register).
2. Configure TIM_mstr slave mode to get the input trigger from tim_ti1 (TS = 00100 in the TIM_mstr_SMCR register).
3. Configure TIM_mstr in trigger mode (SMS = 110 in the TIM_mstr_SMCR register).
4. Configure the TIM_mstr in Master/Slave mode by writing MSM = 1 (TIM_mstr_SMCR register).
5. Configure TIM_slv to get the input trigger from TIM_mstr (TS = 00000 in the TIM_slv_SMCR register).
6. Configure TIM_slv in trigger mode (SMS = 110 in the TIM_slv_SMCR register).

When a rising edge occurs on tim_ti1 (TIM_mstr), both counters start counting synchronously on the internal clock and both TIF flags are set.

Note: In this example both timers are initialized before starting (by setting their respective UG bits). Both counters starts from 0, but an offset can easily be inserted between them by writing any of the counter registers (TIMx_CNT). One can see that the master/slave mode inserts a delay between CNT_EN and CK_PSC on TIM_mstr.

Figure 670. Triggering TIM_mstr and TIM_slv with TIM_mstr tim_ti1 input



Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

54.4.24 ADC triggers

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events.

Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

54.4.25 ADC synchronization

The timer operation can be synchronized to the ADC clock to trigger jitter-free ADC sampling. This function is enabled using the ADSYNC bit in the TIMx_CR2 register.

This feature is useful when the timers and the ADCs are operating with semisynchronous clocks (clocks derived from a same source with integer ratio tim_ker_ck/adc_ker_ck), for instance adc_ker_ck = 75 MHz and tim_ker_ck = 150 MHz or 300 MHz.

ADSYNC must also be set when both peripherals are operating at the same frequency from the same clock source, when jitter-free operation is needed.

ADSYNC must not be set and jitter-free operation is not supported in the following cases:

- When the clock ratio is not an integer (for example adc_ker_ck = 75 MHz and tim_ker_ck = 100 MHz): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum.
- When the ADC is operating in asynchronous mode (adc_ker_ck uncorrelated with tim_ker_ck): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum.

When ADSYNC = 1, the timer operation is slightly changed: the counter enable and counter reset events are aligned to the adc_ker_ck ADC clock, to avoid any phase shift due to clocks enable in the RCC.

Jitter-free operation is guaranteed only when one of the two requirements below is met (depending on the selected trigger source):

1. The counter period must be a multiple of the ADC clock period.

$$(\text{TIMx_PSC} + 1) \times (\text{TIMx_ARR} + 1) \times T_{\text{tim_ker_ck}} = n \times T_{\text{adc_ker_ck}}$$

2. The compare value must be a multiple of the ADC clock period.

$$(\text{TIMx_PSC} + 1) \times \text{TIMx_CMPy} \times T_{\text{tim_ker_ck}} = m \times T_{\text{adc_ker_ck}}$$

Note: If none of the two above requirements is met, the trigger is still generated, but the latency is not constant and varies with the timer and ADC clocks phase shift.

Programming guidelines

The ADC synchronization feature must not be modified during run-time, once the counter is enabled and once the ADC has been configured for receiving triggers from the timer.

It is mandatory to follow the procedure below to use the ADC synchronization:

1. Enable the destination ADC clock.
2. Configure the timer and set the ADSYNC bit.
3. Configure the ADC and enable it (using ADSTART and/or JADSTART bits).
4. Start the timer (with the CEN counter enable bit).

54.4.26 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests upon a single event. The main purpose is to be able to reprogram part of the timer multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR registers define the DMA base address for DMA transfers (when read/write accesses are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register:

Example:

00000: TIMx_CR1

00001: TIMx_CR2

00010: TIMx_SMCR

The DBSS[3:0] bits in the TIMx_DCR register defines the interrupt source that triggers the DMA burst transfers (see *Section 54.5.23: TIMx DMA control register (TIMx_DCR)(x = 2 to 5)* for details).

As an example, the timer DMA burst feature is used to update the contents of the CCRx registers (x = 2, 3, 4) upon an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   - DMA channel peripheral address is the DMAR register address.
   - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into CCRx registers.
   - Number of data to transfer = 3 (See note below).
   - Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bitfields as follows: DBL = 3 transfers, DBA = 0xE and DBSS = 1.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx.
5. Enable the DMA channel.

This example is for the case where every CCRx register has to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer must be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5, and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3, and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

54.4.27 TIM2/TIM3/TIM4/TIM5 DMA requests

The TIM2/TIM3/TIM4/TIM5 can generate a DMA requests, as shown in *Table 515*.

Table 515. DMA request

DMA request signal	DMA request	Enable control bit
tim_upd_dma	Update	UDE
tim_cc1_dma	Capture/compare 1	CC1DE
tim_cc2_dma	Capture/compare 2	CC2DE
tim_cc3_dma	Capture/compare 3	CC3DE
tim_cc4_dma	Capture/compare 4	CC4DE
tim_trgi_dma	Trigger	TDE

Note: Some timer's DMA requests may not be connected to the DMA controller. Refer to the DMA section(s) for more details.

54.4.28 Debug mode

When the microcontroller enters debug mode (Cortex®-M55 core halted), the TIMx counter can either continue to work normally or stops.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For more details, refer to section Debug support (DBG).

54.4.29 TIM2/TIM3/TIM4/TIM5 low-power modes

Table 516. Effect of low-power modes on TIM2/TIM3/TIM4/TIM5

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Standby	The timer is powered-down and must be reinitialized after exiting the Standby mode.

54.4.30 TIM2/TIM3/TIM4/TIM5 interrupts

The TIM2/TIM3/TIM4/TIM5 can generate multiple interrupts, as shown in *Table 517*.

Table 517. Interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
TIM_UP	Update	UIF	UIE	write 0 in UIF	Yes	No
TIM_CC	Capture/compare 1	CC1IF	CC1IE	write 0 in CC1IF	Yes	No
	Capture/compare 2	CC2IF	CC2IE	write 0 in CC2IF	Yes	No
	Capture/compare 3	CC3IF	CC3IE	write 0 in CC3IF	Yes	No
	Capture/compare 4	CC4IF	CC4IE	write 0 in CC4IF	Yes	No
TIM_TRG	Trigger	TIF	TIE	write 0 in TIF	Yes	No
TIM_DIR_IDX	Index	IDXF	IDXIE	write 0 in IDXF	Yes	No
	Direction	DIRF	DIRIE	write 0 in DIRF	Yes	No
TIM_IERR	Index Error	IERRF	IERRIE	write 0 in IERRF	Yes	No
TIM_TER	Transition Error	TERRF	TERRIE	write 0 in TERRF	Yes	No

54.5 TIM2/TIM3/TIM4/TIM5 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

54.5.1 TIMx control register 1 (TIMx_CR1)(x = 2 to 5)

Address offset: 0x000

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	CMS[1:0]		DIR	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering Enable

0: Dithering disabled

1: Dithering enabled

Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.

1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and sampling clock used by the digital filters (tim_etr_in, tim_tix),

00: $t_{DTS} = t_{tim_ker_ck}$

01: $t_{DTS} = 2 \times t_{tim_ker_ck}$

10: $t_{DTS} = 4 \times t_{tim_ker_ck}$

11: Reserved

Bit 7 **ARPE**: Autoreload preload enable

0: TIMx_ARR register is not buffered

1: TIMx_ARR register is buffered

Bits 6:5 **CMS[1:0]**: Center-aligned mode selection

00: Edge-aligned mode. The counter counts up or down depending on the direction bit (DIR).

01: Center-aligned mode 1. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIMx_CCMRx register) are set only when the counter is counting down.

10: Center-aligned mode 2. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIMx_CCMRx register) are set only when the counter is counting up.

11: Center-aligned mode 3. The counter counts up and down alternatively. Output compare interrupt flags of channels configured in output (CCxS = 00 in TIMx_CCMRx register) are set both when the counter is counting up or down.

Note: It is not allowed to switch from edge-aligned mode to center-aligned mode as long as the counter is enabled (CEN = 1)

Bit 4 **DIR**: Direction
0: Counter used as upcounter
1: Counter used as downcounter
Note: This bit is read only when the timer is configured in Center-aligned mode or Encoder mode.

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source
This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:
- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable
This bit is set and cleared by software to enable/disable UEV event generation.
0: UEV enabled. The Update (UEV) event is generated by one of the following events:
- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.
1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable
0: Counter disabled
1: Counter enabled
Note: External clock, gated mode and encoder mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.
CEN is cleared automatically in one-pulse mode, when an update event occurs.

54.5.2 TIMx control register 2 (TIMx_CR2)(x = 2 to 5)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	AD SYNC	Res.	Res.	MMS[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			rw			rw									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			CCDS	Res.	Res.	Res.
								rw	rw	rw	rw	rw			

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **ADSYNC**: ADC synchronization

0: The timer operates independently from the ADC

1: The timer operation is synchronized with the ADC clock to provide jitter-free sampling point. This mode can be enabled only with specific ADC / timer clock relationship. Refer to *Section 54.4.25* for requirements.

The ADSYNC must not modified when the counter is enabled (CEN bit is set).

Bits 27:26 Reserved, must be kept at reset value.

Bits 24:8 Reserved, must be kept at reset value.

Bit 7 **TI1S**: tim_ti1 selection

0: The tim_ti1_in[15:0] multiplexer output is to tim_ti1 input

1: The tim_ti1_in[15:0], tim_ti2_in[15:0] and tim_ti3_in[15:0] multiplexers outputs are XORed and connected to the tim_ti1 input. See also *Section 53.3.29: Interfacing with Hall sensors*.

Bits 25, 6, 5, 4 **MMS[3:0]**: Master mode selection

These bits are used to select the information to be sent in master mode to slave timers for synchronization (tim_trgo). The combination is as follows:

0000: **Reset** - the UG bit from the TIMx_EGR register is used as trigger output (tim_trgo). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on tim_trgo is delayed compared to the actual reset.

0001: **Enable** - the Counter enable signal, CNT_EN, is used as trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode.
When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).

0010: **Update** - The update event is selected as trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

0011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred (tim_trgo).

0100: **Compare** - tim_oc1refc signal is used as trigger output (tim_trgo)

0101: **Compare** - tim_oc2refc signal is used as trigger output (tim_trgo)

0110: **Compare** - tim_oc3refc signal is used as trigger output (tim_trgo)

0111: **Compare** - tim_oc4refc signal is used as trigger output (tim_trgo)

1000: **Encoder clock output** - The encoder clock signal is used as trigger output (tim_trgo). This code is valid for the following SMS[3:0] values: 0001, 0010, 0011, 1010, 1011, 1100, 1101, 1110, 1111. Any other SMS[3:0] code is not allowed and may lead to unexpected behavior.

Others: Reserved

Note: The clock of the slave timer or ADC must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs

1: CCx DMA requests sent when update event occurs

Bits 2:0 Reserved, must be kept at reset value.

54.5.3 TIMx slave mode control register (TIMx_SMCR)(x = 2 to 5)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	SMSPS	SMSPE	Res.	Res.	TS[4:3]		Res.	Res.	Res.	SMS[3]
						rw	rw			rw	rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETP	ECE	ETPS[1:0]		ETF[3:0]				MSM	TS[2:0]			OCCS	SMS[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **SMSPS**: SMS preload source

This bit selects whether the events that triggers the SMS[3:0] bitfield transfer from preload to active

0: The transfer is triggered by the Timer's Update event

1: The transfer is triggered by the Index event

Bit 24 **SMSPE**: SMS preload enable

This bit selects whether the SMS[3:0] bitfield is preloaded

0: SMS[3:0] bitfield is not preloaded

1: SMS[3:0] preload is enabled

Bits 23:22 Reserved, must be kept at reset value.

Bits 19:17 Reserved, must be kept at reset value.

Bit 15 **ETP**: External trigger polarity

This bit selects whether tim_etr_in or $\overline{\text{tim_etr_in}}$ is used for trigger operations

0: tim_etr_in is non-inverted, active at high level or rising edge

1: tim_etr_in is inverted, active at low level or falling edge

Bit 14 **ECE**: External clock enable

This bit enables External clock mode 2.

0: External clock mode 2 disabled

1: External clock mode 2 enabled. The counter is clocked by any active edge on the tim_etrf signal.

Note: Setting the ECE bit has the same effect as selecting external clock mode 1 with tim_trgi connected to tim_etrf (SMS = 111 and TS = 00111).
It is possible to simultaneously use external clock mode 2 with the following slave modes: reset mode, gated mode and trigger mode. Nevertheless, tim_trgi must not be connected to tim_etrf in this case (TS bits must not be 00111).
If external clock mode 1 and external clock mode 2 are enabled at the same time, the external clock input is tim_etrf.

Bits 13:12 **ETPS[1:0]**: External trigger prescaler

External trigger signal tim_etrp frequency must be at most 1/4 of tim_ker_ck frequency. A prescaler can be enabled to reduce tim_etrp frequency. It is useful when inputting fast external clocks on tim_etr_in.

00: Prescaler OFF

01: tim_etrp frequency divided by 2

10: tim_etrp frequency divided by 4

11: tim_etrp frequency divided by 8

Bits 11:8 **ETF[3:0]**: External trigger filter

This bitfield then defines the frequency used to sample tim_etrp signal and the length of the digital filter applied to tim_etrp. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Bit 7 **MSM**: Master/Slave mode

0: No action
1: The effect of an event on the trigger input (tim_trgi) is delayed to allow a perfect synchronization between the current timer and its slaves (through tim_trgo). It is useful if we want to synchronize several timers on a single external event.

Bits 21, 20, 6, 5, 4 **TS[4:0]**: Trigger selection

This bitfield selects the trigger input to be used to synchronize the counter.

00000: Internal trigger 0 (tim_itr0)
00001: Internal trigger 1 (tim_itr1)
00010: Internal trigger 2 (tim_itr2)
00011: Internal trigger 3 (tim_itr3)
00100: tim_ti1 edge detector (tim_ti1f_ed)
00101: Filtered timer input 1 (tim_ti1fp1)
00110: Filtered timer input 2 (tim_ti2fp2)
00111: External trigger input (tim_etrf)
01000: Internal trigger 4 (tim_itr4)
01001: Internal trigger 5 (tim_itr5)
01010: Internal trigger 6 (tim_itr6)
01011: Internal trigger 7 (tim_itr7)
01100: Internal trigger 8 (tim_itr8)
01101: Internal trigger 9 (tim_itr9)
01110: Internal trigger 10 (tim_itr10)
01111: Internal trigger 11 (tim_itr11)
10000: Internal trigger 12 (tim_itr12)
10001: Internal trigger 13 (tim_itr13)
10010: Internal trigger 14 (tim_itr14)
10011: Internal trigger 15 (tim_itr15)
Others: Reserved

See *Section 54.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation details.

Note: These bits must be changed only when they are not used (for example when SMS = 000) to avoid wrong edge detections at the transition.

Bit 3 **OCCS**: OCREF clear selection

This bit is used to select the OCREF clear source
0: tim_ocref_clr_int is connected to the tim_ocref_clr input
1: tim_ocref_clr_int is connected to tim_etrf

Note: If the OCREF clear selection feature is not supported, this bit is reserved and forced by hardware to 0. Section 54.3: TIM2/TIM3/TIM4/TIM5 implementation.

Bits 16, 2, 1, 0 **SMS[3:0]**: Slave mode selection

When external signals are selected the active edge of the trigger signal (tim_trgi) is linked to the polarity selected on the external input (refer to ETP bit in TIMx_SMCR for tim_etr_in and CCxP/CCxNP bits in TIMx_CCER register for tim_ti1fp1 and tim_ti2fp2).

0000: Slave mode disabled - if CEN = 1 then the prescaler is clocked directly by the internal clock.

0001: Encoder mode 1 - Counter counts up/down on tim_ti1fp1 edge depending on tim_ti2fp2 level.

0010: Encoder mode 2 - Counter counts up/down on tim_ti2fp2 edge depending on tim_ti1fp1 level.

0011: Encoder mode 3 - Counter counts up/down on both tim_ti1fp1 and tim_ti2fp2 edges depending on the level of the other input.

0100: Reset mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter and generates an update of the registers.

0101: Gated mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

0110: Trigger mode - The counter starts at a rising edge of the trigger tim_trgi (but it is not reset). Only the start of the counter is controlled.

0111: External clock mode 1 - Rising edges of the selected trigger (tim_trgi) clock the counter.

1000: Combined reset + trigger mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers and starts the counter.

1001: Combined gated + reset mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

1010: Encoder mode: Clock plus direction, x2 mode.

1011: Encoder mode: Clock plus direction, x1 mode, tim_ti2fp2 edge sensitivity is set by CC2P.

1100: Encoder mode: Directional clock, x2 mode.

1101: Encoder mode: Directional clock, x1 mode, tim_ti1fp1 and tim_ti2fp2 edge sensitivity is set by CC1P and CC2P.

1110: Quadrature encoder mode: x1 mode, counting on tim_ti1fp1 edges only, edge sensitivity is set by CC1P.

1111: Quadrature encoder mode: x1 mode, counting on tim_ti2fp2 edges only, edge sensitivity is set by CC2P.

Note: The gated mode must not be used if tim_ti1f_ed is selected as the trigger input (TS = 00100). Indeed, tim_ti1f_ed outputs 1 pulse for each transition on tim_ti1f, whereas the gated mode checks the level of the trigger signal.

Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

54.5.4 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 2 to 5)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRIE	IERRIE	DIRIE	IDXIE	Res.	Res.	Res.	Res.
								rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	Res.	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res.	TIE	Res.	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	rw		rw	rw	rw	rw	rw		rw		rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TERRIE**: Transition error interrupt enable
0: Transition error interrupt disabled
1: Transition error interrupt enabled

Bit 22 **IERRIE**: Index error interrupt enable
0: Index error interrupt disabled
1: Index error interrupt enabled

Bit 21 **DIRIE**: Direction change interrupt enable
0: Direction change interrupt disabled
1: Direction change interrupt enabled

Bit 20 **IDXIE**: Index interrupt enable
0: Index interrupt disabled
1: Index interrupt enabled

Bits 19:15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable
0: Trigger DMA request disabled.
1: Trigger DMA request enabled.

Bit 13 Reserved, must be kept at reset value.

Bit 12 **CC4DE**: Capture/Compare 4 DMA request enable
0: CC4 DMA request disabled.
1: CC4 DMA request enabled.

Bit 11 **CC3DE**: Capture/Compare 3 DMA request enable
0: CC3 DMA request disabled.
1: CC3 DMA request enabled.

Bit 10 **CC2DE**: Capture/Compare 2 DMA request enable
0: CC2 DMA request disabled.
1: CC2 DMA request enabled.

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable
0: CC1 DMA request disabled.
1: CC1 DMA request enabled.

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled.
1: Update DMA request enabled.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled.
1: Trigger interrupt enabled.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4IE**: Capture/Compare 4 interrupt enable
0: CC4 interrupt disabled.
1: CC4 interrupt enabled.

Bit 3 **CC3IE**: Capture/Compare 3 interrupt enable
0: CC3 interrupt disabled.
1: CC3 interrupt enabled.

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable
0: CC2 interrupt disabled.
1: CC2 interrupt enabled.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled.
1: CC1 interrupt enabled.

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled.
1: Update interrupt enabled.

54.5.5 TIMx status register (TIMx_SR)(x = 2 to 5)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRF	IERRF	DIRF	IDXF	Res.	Res.	Res.	Res.
								rc_w0	rc_w0	rc_w0	rc_w0				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	CC4OF	CC3OF	CC2OF	CC1OF	Res.	Res.	TIF	Res.	CC4IF	CC3IF	CC2IF	CC1IF	UIF
			rc_w0	rc_w0	rc_w0	rc_w0			rc_w0		rc_w0	rc_w0	rc_w0	rc_w0	rc_w0

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TERRF**: Transition error interrupt flag
This flag is set by hardware when a transition error is detected in encoder mode. It is cleared by software by writing it to 0.
0: No encoder transition error has been detected.
1: An encoder transition error has been detected

Bit 22 **IERRF**: Index error interrupt flag
This flag is set by hardware when an index error is detected. It is cleared by software by writing it to 0.
0: No index error has been detected.
1: An index error has been detected

Bit 21 **DIRF**: Direction change interrupt flag
This flag is set by hardware when the direction changes in encoder mode (DIR bit value in TIMx_CR is changing). It is cleared by software by writing it to 0.
0: No direction change
1: Direction change

Bit 20 **IDXF**: Index interrupt flag
This flag is set by hardware when an index event is detected. It is cleared by software by writing it to 0.
0: No index event occurred.
1: An index event has occurred

Bits 19:13 Reserved, must be kept at reset value.

Bit 12 **CC4OF**: Capture/Compare 4 overcapture flag
refer to CC1OF description

Bit 11 **CC3OF**: Capture/Compare 3 overcapture flag
refer to CC1OF description

Bit 10 **CC2OF**: Capture/compare 2 overcapture flag
refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.
0: No overcapture has been detected.
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **TIF**: Trigger interrupt flag
This flag is set by hardware on the TRG trigger event (active edge detected on tim_trgi input) when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.
0: No trigger event occurred.
1: Trigger interrupt pending.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4IF**: Capture/Compare 4 interrupt flag
Refer to CC1IF description

Bit 3 **CC3IF**: Capture/Compare 3 interrupt flag
Refer to CC1IF description

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag
Refer to CC1IF description

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag
This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).
0: No compare match / No input capture occurred
1: A compare match or an input capture occurred
If channel CC1 is configured as output: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are three possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.
If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0 **UIF**: Update interrupt flag
This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
At overflow or underflow and if UDIS = 0 in the TIMx_CR1 register.
When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS = 0 and UDIS = 0 in the TIMx_CR1 register.
When CNT is reinitialized by a trigger event (refer to the synchro control register description), if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

54.5.6 TIMx event generation register (TIMx_EGR)(x = 2 to 5)

Address offset: 0x014

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
									w		w	w	w	w	w

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **TG**: Trigger generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: The TIF flag is set in TIMx_SR register. Related interrupt or DMA transfer can occur if enabled.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **CC4G**: Capture/compare 4 generation
Refer to CC1G description

Bit 3 **CC3G**: Capture/compare 3 generation
Refer to CC1G description

Bit 2 **CC2G**: Capture/compare 2 generation
Refer to CC1G description

Bit 1 **CC1G**: Capture/compare 1 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.
If channel CC1 is configured as input:
The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Re-initialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared if the center-aligned mode is selected or if DIR = 0 (up-counting), else it takes the autoreload value (TIMx_ARR) if DIR = 1 (down-counting).

54.5.7 TIMx capture/compare mode register 1 (TIMx_CCMR1)(x = 2 to 5)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output.

01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2.

10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1.

11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bitfield defines the frequency used to sample tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}

0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2

0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4

0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8

0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6

0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8

0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6

0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8

1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6

1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8

1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5

1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6

1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8

1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5

1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6

1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on CC1 input (tim_ic1). The prescaler is reset as soon as CC1E = 0 (TIMx_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input

01: capture is done once every 2 events

10: capture is done once every 4 events

11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1

10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2

11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

54.5.8 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 2 to 5)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M [3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2CE	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC2CE**: Output compare 2 clear enable

Bits 24, 14:12 **OC2M[3:0]**: Output compare 2 mode
refer to OC1M description on bits 6:4

Bit 11 **OC2PE**: Output compare 2 preload enable

Bit 10 **OC2FE**: Output compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2
10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1
11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)
Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bit 7 **OC1CE**: Output compare 1 clear enable
0: tim_oc1ref is not affected by the tim_ocref_clr_int input
1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr_int input

Bits 16, 6:4 **OC1M[3:0]**: Output compare 1 mode

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 is derived. tim_oc1ref is active high whereas tim_oc1 active level depends on CC1P bit.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the ouput keeps the state (active or inactive) it had before entering the frozen state.

0001: Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - tim_oc1ref toggles when TIMx_CNT = TIMx_CCR1.

0100: Force inactive level - tim_oc1ref is forced low.

0101: Force active level - tim_oc1ref is forced high.

0110: PWM mode 1 - In up-counting, channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive. In down-counting, channel 1 is inactive (tim_oc1ref = 0) as long as TIMx_CNT>TIMx_CCR1 else active (tim_oc1ref = 1).

0111: PWM mode 2 - In up-counting, channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active. In down-counting, channel 1 is active as long as TIMx_CNT>TIMx_CCR1 else inactive.

1000: Retriggerable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retriggerable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Reserved.

1011: Reserved.

1100: Combined PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc is the logical OR between tim_oc1ref and tim_oc2ref.

1101: Combined PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc is the logical AND between tim_oc1ref and tim_oc2ref.

1110: Asymmetric PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc outputs tim_oc1ref when the counter is counting up, tim_oc2ref when it is counting down.

1111: Asymmetric PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc outputs tim_oc1ref when the counter is counting up, tim_oc2ref when it is counting down.

Note: In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.
1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.
0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.
1: An active edge on the trigger input acts like a compare match on CC1 output. Then, OC is set to the compare level independently from the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to three clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output.
01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1.
10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2.
11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

54.5.9 TIMx capture/compare mode register 2 (TIMx_CCMR2)(x = 2 to 5)

Address offset: 0x01C

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC4F[3:0]				IC4PSC[1:0]		CC4S[1:0]		IC3F[3:0]				IC3PSC[1:0]		CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC4F[3:0]**: Input capture 4 filter

Bits 11:10 **IC4PSC[1:0]**: Input capture 4 prescaler

Bits 9:8 **CC4S[1:0]**: Capture/Compare 4 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC4 channel is configured as output
01: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti4
10: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti3
11: CC4 channel is configured as input, tim_ic4 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).

Bits 7:4 **IC3F[3:0]**: Input capture 3 filter

Bits 3:2 **IC3PSC[1:0]**: Input capture 3 prescaler

Bits 1:0 **CC3S[1:0]**: Capture/Compare 3 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti3
10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4
11: CC3 channel is configured as input, tim_ic3 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)
Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).

54.5.10 TIMx capture/compare mode register 2 [alternate] (TIMx_CCMR2)(x = 2 to 5)

Address offset: 0x01C

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M [3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M [3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC4CE	OC4M[2:0]			OC4PE	OC4FE	CC4S[1:0]		OC3CE	OC3M[2:0]			OC3PE	OC3FE	CC3S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC4CE**: Output compare 4 clear enable

Bits 24, 14:12 **OC4M[3:0]**: Output compare 4 mode
Refer to OC3M[3:0]

Bit 11 **OC4PE**: Output compare 4 preload enable

Bit 10 **OC4FE**: Output compare 4 fast enable

Bits 9:8 **CC4S[1:0]**: Capture/Compare 4 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC4 channel is configured as output

01: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti4

10: CC4 channel is configured as input, tim_ic4 is mapped on tim_ti3

11: CC4 channel is configured as input, tim_ic4 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC4S bits are writable only when the channel is OFF (CC4E = 0 in TIMx_CCER).

Bit 7 **OC3CE**: Output compare 3 clear enable

Bits 16, 6:4 **OC3M[3:0]**: Output compare 3 mode

These bits define the behavior of the output reference signal tim_oc3ref from which tim_oc3 and tim_oc3n are derived. tim_oc3ref is active high whereas tim_oc3 and tim_oc3n active level depends on CC3P and CC3NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR3 and the counter TIMx_CNT has no effect on the outputs.(this mode is used to generate a timing base).

0001: Set channel 3 to active level on match. tim_oc3ref signal is forced high when the counter TIMx_CNT matches the capture/compare register 3 (TIMx_CCR3).

0010: Set channel 3 to inactive level on match. tim_oc3ref signal is forced low when the counter TIMx_CNT matches the capture/compare register 3 (TIMx_CCR3).

0011: Toggle - tim_oc3ref toggles when TIMx_CNT = TIMx_CCR3.

0100: Force inactive level - tim_oc3ref is forced low.

0101: Force active level - tim_oc3ref is forced high.

0110: PWM mode 1 - In up-counting, channel 3 is active as long as TIMx_CNT<TIMx_CCR3 else inactive. In down-counting, channel 3 is inactive (tim_oc3ref = 0) as long as TIMx_CNT>TIMx_CCR3 else active (tim_oc3ref = 1).

0111: PWM mode 2 - In up-counting, channel 3 is inactive as long as TIMx_CNT<TIMx_CCR3 else active. In down-counting, channel 3 is active as long as TIMx_CNT>TIMx_CCR3 else inactive.

1000: Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Pulse on compare: a pulse is generated on tim_oc3ref upon CCR3 match event, as per PWPRSC[2:0] and PW[7:0] bitfields programming in TIMx_ECR.

1011: Direction output. The tim_oc3ref signal is overridden by a copy of the DIR bit.

1100: Combined PWM mode 1 - tim_oc3ref has the same behavior as in PWM mode 1. tim_oc3refc is the logical OR between tim_oc3ref and tim_oc4ref.

1101: Combined PWM mode 2 - tim_oc3ref has the same behavior as in PWM mode 2. tim_oc3refc is the logical AND between tim_oc3ref and tim_oc4ref.

1110: Asymmetric PWM mode 1 - tim_oc3ref has the same behavior as in PWM mode 1. tim_oc3refc outputs tim_oc3ref when the counter is counting up, tim_oc4ref when it is counting down.

1111: Asymmetric PWM mode 2 - tim_oc3ref has the same behavior as in PWM mode 2. tim_oc3refc outputs tim_oc3ref when the counter is counting up, tim_oc4ref when it is counting down.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output).

Note: In PWM mode, the OCREF level changes only when the result of the comparison changes or when the output compare mode switches from “frozen” mode to “PWM” mode.

On channels having a complementary output, this bitfield is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the OC3M active bits take the new value from the preloaded bits only when a COM event is generated.

Bit 3 **OC3PE**: Output compare 3 preload enable

Bit 2 **OC3FE**: Output compare 3 fast enable

Bits 1:0 **CC3S[1:0]**: Capture/Compare 3 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC3 channel is configured as output
01: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti3
10: CC3 channel is configured as input, tim_ic3 is mapped on tim_ti4
11: CC3 channel is configured as input, tim_ic3 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: CC3S bits are writable only when the channel is OFF (CC3E = 0 in TIMx_CCER).

54.5.11 TIMx capture/compare enable register (TIMx_CCER)(x = 2 to 5)

Address offset: 0x020

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CC4NP	Res.	CC4P	CC4E	CC3NP	Res.	CC3P	CC3E	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
rw		rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw

Bit 15 **CC4NP**: *Capture/Compare 4 output Polarity.*
Refer to CC1NP description

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC4P**: *Capture/Compare 4 output Polarity.*
Refer to CC1P description

Bit 12 **CC4E**: *Capture/Compare 4 output enable.*
refer to CC1E description

Bit 11 **CC3NP**: *Capture/Compare 3 output Polarity.*
Refer to CC1NP description

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC3P**: *Capture/Compare 3 output Polarity.*
Refer to CC1P description

Bit 8 **CC3E**: *Capture/Compare 3 output enable.*
Refer to CC1E description

Bit 7 **CC2NP**: *Capture/Compare 2 output Polarity.*
Refer to CC1NP description

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CC2P**: *Capture/Compare 2 output Polarity.*
refer to CC1P description

Bit 4 **CC2E**: *Capture/Compare 2 output enable.*
Refer to CC1E description

Bit 3 **CC1NP**: *Capture/Compare 1 output Polarity.*

CC1 channel configured as output: CC1NP must be kept cleared in this case.

CC1 channel configured as input: This bit is used in conjunction with CC1P to define tim_ti1fp1/tim_ti2fp1 polarity. refer to CC1P description.

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CC1P**: *Capture/Compare 1 output Polarity.*

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP = 0, CC1P = 0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode or encoder mode).

CC1NP = 0, CC1P = 1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode or encoder mode).

CC1NP = 1, CC1P = 1: non-inverted/both edges. The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode). This configuration must not be used in encoder mode.

CC1NP = 1, CC1P = 0: this configuration is reserved, it must not be used.

Bit 0 **CC1E**: *Capture/Compare 1 output enable.*

0: Capture mode disabled / OC1 is not active

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

Table 518. Output control bit for standard tim_ocx channels

CCxE bit	tim_ocx output state
0	Output disabled (not driven by the timer: Hi-Z)
1	Output enabled (tim_ocx = tim_ocxref + Polarity)

Note: The state of the external IO pins connected to the standard tim_ocx channels depends only on the GPIO registers when CCxE = 0.

54.5.12 TIMx counter (TIMx_CNT)(x = 2 to 5)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIFCPY_CNT[31]	CNT[30:16]														
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY_CNT[31]**: Value depends on IUFREMAP in TIMx_CR1.
If UIFREMAP = 0
CNT[31]: Most significant bit of counter value
If UIFREMAP = 1
UIFCPY: UIF Copy
This bit is a read-only copy of the UIF bit of the TIMx_ISR register

Bits 30:0 **CNT[30:0]**: Least significant part of counter value
Non-dithering mode (DITHEN = 0)
The register holds the counter value.
Dithering mode (DITHEN = 1)
The register holds the non-dithered part in CNT[30:0]. The fractional part is not available.

54.5.13 TIMx prescaler (TIMx_PSC)(x = 2 to 5)

Address offset: 0x028

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value
The counter clock frequency tim_cnt_ck is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

54.5.14 TIMx autoreload register (TIMx_ARR)(x = 2 to 5)

Address offset: 0x02C

Reset value: 0xFFFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ARR[31:0]**: Autoreload value

ARR is the value to be loaded in the actual autoreload register.

Refer to the *Section 54.4.3: Time-base unit* for more details about ARR update and behavior.

The counter is blocked while the autoreload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the autoreload value.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[31:4]. The ARR[3:0] bitfield contains the dithered part.

54.5.15 TIMx capture/compare register 1 (TIMx_CCR1)(x = 2 to 5)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR1[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CCR1[31:0]**: Capture/compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[31:4]. The CCR1[3:0] bitfield contains the dithered part.

If channel CC1 is configured as input:

CCR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIMx_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[31:0]. The CCR1[3:0] bits are reset.

54.5.16 TIMx capture/compare register 2 (TIMx_CCR2)(x = 2 to 5)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR2[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CCR2[31:0]**: Capture/compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc2 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR2[31:4]. The CCR2[3:0] bitfield contains the dithered part.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 2 event (tim_ic2). The TIMx_CCR2 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR2[31:0]. The CCR2[3:0] bits are reset.

54.5.17 TIMx capture/compare register 3 (TIMx_CCR3)(x = 2 to 5)

Address offset: 0x03C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR3[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR3[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CCR3[31:0]**: Capture/compare 3 value

If channel CC3 is configured as output:

CCR3 is the value to be loaded in the actual capture/compare 3 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR3 register (bit OC3PE). Else the preload value is copied in the active capture/compare 3 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc3 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR3[31:4]. The CCR3[3:0] bitfield contains the dithered part.

If channel CC3 is configured as input:

CCR3 is the counter value transferred by the last input capture 3 event (tim_ic3). The TIMx_CCR3 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR3[31:0]. The CCR3[3:0] bits are reset.

54.5.18 TIMx capture/compare register 4 (TIMx_CCR4)(x = 2 to 5)

Address offset: 0x040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCR4[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR4[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CCR4[31:0]**: Capture/compare 4 value

If channel CC4 is configured as output:

CCR4 is the value to be loaded in the actual capture/compare 4 register (preload value). It is loaded permanently if the preload feature is not selected in the TIMx_CCMR4 register (bit OC4PE). Else the preload value is copied in the active capture/compare 4 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc4 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR4[31:4]. The CCR4[3:0] bitfield contains the dithered part.

If channel CC4 is configured as input:

CCR4 is the counter value transferred by the last input capture 4 event (tim_ic4). The TIMx_CCR4 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR4[31:0]. The CCR4[3:0] bits are reset.

54.5.19 TIMx timer encoder control register (TIMx_ECR)(x = 2 to 5)

Address offset: 0x058

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	PWPRSC[2:0]			PW[7:0]							
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPOS[1:0]		FIDX	IBLK[1:0]		IDIR[1:0]		IE
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bits 26:24 **PWPRSC[2:0]**: Pulse width prescaler

This bitfield sets the clock prescaler for the pulse generator, as following:

$t_{PWG} = (2^{(PWPRSC[2:0])}) \times t_{tim_ker_ck}$

Bits 23:16 **PW[7:0]**: Pulse width

This bitfield defines the pulse duration, as following:

$t_{PW} = PW[7:0] \times t_{PWG}$

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:6 **IPOS[1:0]**: Index positioning

In quadrature encoder mode (SMS[3:0] = 0001, 0010, 0011, 1110, 1111), this bit indicates in which AB input configuration the Index event resets the counter.

00: Index resets the counter when AB = 00
01: Index resets the counter when AB = 01
10: Index resets the counter when AB = 10
11: Index resets the counter when AB = 11

In directional clock mode or clock plus direction mode (SMS[3:0] = 1010, 1011, 1100, 1101), these bits indicates on which level the Index event resets the counter. In bidirectional clock mode, this applies for both clock inputs.

x0: Index resets the counter when clock is 0
x1: Index resets the counter when clock is 1

Note: IPOS[1] bit is not significant

Bit 5 **FIDX**: First index

This bit indicates if the first index only is taken into account

0: Index is always active
1: the first Index only resets the counter

Bits 4:3 **IBLK[1:0]**: Index blanking

This bit indicates if the Index event is conditioned by the tim_ti3 input

00: Index always active
01: Index disabled hen tim_ti3 input is active, as per CC3P bitfield
10: Index disabled when tim_ti4 input is active, as per CC4P bitfield
11: Reserved

Bits 2:1 **IDIR[1:0]**: Index direction

This bit indicates in which direction the Index event resets the counter.

00: Index resets the counter whatever the direction
01: Index resets the counter when up-counting only
10: Index resets the counter when down-counting only
11: Reserved

Bit 0 **IE**: Index enable

This bit indicates if the Index event resets the counter.

0: Index disabled
1: Index enabled

54.5.20 TIMx timer input selection register (TIMx_TISEL)(x = 2 to 5)

Address offset: 0x05C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **TI4SEL[3:0]**: Selects tim_ti4[15:0] input
0000: tim_ti4_in0: TIMx_CH4
0001: tim_ti4_in1
...
1111: tim_ti4_in15
Refer to *Section 54.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **TI3SEL[3:0]**: Selects tim_ti3[15:0] input
0000: tim_ti3_in0: TIMx_CH3
0001: tim_ti3_in1
...
1111: tim_ti3_in15
Refer to *Section 54.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: Selects tim_ti2[15:0] input
0000: tim_ti2_in0: TIMx_CH2
0001: tim_ti2_in1
...
1111: tim_ti2_in15
Refer to *Section 54.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: Selects tim_ti1[15:0] input
0000: tim_ti1_in0: TIMx_CH1
0001: tim_ti1_in1
...
1111: tim_ti1_in15
Refer to *Section 54.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

54.5.21 TIMx alternate function register 1 (TIMx_AF1)(x = 2 to 5)

Address offset: 0x060

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL[3:2]	
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETRSEL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw														

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:14 **ETRSEL[3:0]**: etr_in source selection
These bits select the etr_in input source.
0000: tim_etr0: TIMx_ETR input
0001: tim_etr1
...
1111: tim_etr15
Refer to *Section 54.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

Bits 13:0 Reserved, must be kept at reset value.

54.5.22 TIMx alternate function register 2 (TIMx_AF2)(x = 2 to 5)

Address offset: 0x064

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL[2:0]		
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **OCRSEL[2:0]**: ocref_clr source selection
These bits select the ocref_clr input source.
000: tim_ocref_clr0
001: tim_ocref_clr1
...
111: tim_ocref_clr7
Refer to *Section 54.4.2: TIM2/TIM3/TIM4/TIM5 pins and internal signals* for product specific implementation.

Bits 15:0 Reserved, must be kept at reset value.

54.5.23 TIMx DMA control register (TIMx_DCR)(x = 2 to 5)

Address offset: 0x3DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **DBSS[3:0]**: DMA burst source selection

This bitfield defines the interrupt source that triggers the DMA burst transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address).

0000: Reserved
0001: Update
0010: CC1
0011: CC2
0100: CC3
0101: CC4
0110: COM
0111: Trigger
Others: reserved

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bit vector defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).

00000: 1 transfer
00001: 2 transfers
00010: 3 transfers
...
11010: 26 transfers

Example: Let us consider the following transfer: DBL = 7 bytes & DBA = TIM2_CR1.

–If DBL = 7 bytes and DBA = TIM2_CR1 represents the address of the byte to be transferred, the address of the transfer is given by the following equation:

(TIMx_CR1 address) + DBA + (DMA index), where DMA index = DBL

In this example, 7 bytes are added to (TIMx_CR1 address) + DBA, which gives us the address from/to which the data are copied. In this case, the transfer is done to 7 registers starting from the following address: (TIMx_CR1 address) + DBA

According to the configuration of the DMA Data Size, several cases may occur:

–If the DMA Data Size is configured in half-words, 16-bit data are transferred to each of the 7 registers.

–If the DMA Data Size is configured in bytes, the data are also transferred to 7 registers: the first register contains the first MSB byte, the second register, the first LSB byte and so on. So with the transfer Timer, one also has to specify the size of data transferred by DMA.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bits vector defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:
00000: TIMx_CR1,
00001: TIMx_CR2,
00010: TIMx_SMCR,
...

54.5.24 TIMx DMA address for full transfer (TIMx_DMAR)(x = 2 to 5)

Address offset: 0x3E0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMAB[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIMx_CR1 address) + (DBA + DMA index) x 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

54.5.25 TIMx register map

TIMx registers are mapped as described in the table below.

Table 519. TIM2/TIM3/TIM4/TIM5 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	TIMx_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DITHEN	UIFREMA	Res.	CKD [1:0]		ARPE	CMS [1:0]		DIR	OPM	URS	UDIS	CEN
	Reset value																				0	0		0	0	0	0	0	0	0	0	0	0
0x004	TIMx_CR2	Res.	Res.	Res.	ADSYNC	Res.	Res.	MMS[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			CCDS	Res.	Res.	Res.
	Reset value				0			0																		0	0	0	0	0			
0x008	TIMx_SMCR	Res.	Res.	Res.	Res.	Res.	Res.	SMSPS	SMSPE	Res.	Res.	TS [4:3]		Res.	Res.	Res.	SMS[3]	ETP	ECE	ETPS [1:0]		ETF[3:0]				MSM	TS[2:0]			Res.	SMS[2:0]		
	Reset value							0	0			0	0				0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
0x00C	TIMx_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRIE	IERRIE	DIRIE	IDXIE	Res.	Res.	Res.	Res.	Res.	Res.	TDE	Res.	CC4DE	CC3DE	CC2DE	CC1DE	UDE	Res.	TIE	Res.	CC4IE	CC3IE	CC2IE	CC1IE	UIE
	Reset value								0	0	0	0							0		0	0	0	0	0		0		0	0	0	0	0
0x010	TIMx_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERRF	IERRF	DIRF	IDXF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC4OF	CC3OF	CC2OF	CC1OF	Res.	Res.	TIF	Res.	CC4IF	CC3IF	CC2IF	CC1IF	UIF
	Reset value								0	0	0	0									0	0	0	0			0		0	0	0	0	0
0x014	TIMx_EGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	CC4G	CC3G	CC2G	CC1G	UG
	Reset value																										0		0	0	0	0	0
0x018	TIMx_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR1 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	TIMx_CCMR2 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC4F[3:0]				IC4 PSC [1:0]		CC4S [1:0]		IC3F[3:0]				IC3 PSC [1:0]		CC3S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR2 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC4M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC3M[3]	O24CE	OC4M [2:0]			OC4PE	OC4FE	CC4S [1:0]		OC3CE	OC3M [2:0]			OC3PE	OC3FE	CC3S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 519. TIM2/TIM3/TIM4/TIM5 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x020	TIMx_CCER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC4NP	Res.	CC4P	CC4E	CC3NP	Res.	CC3P	CC3E	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
	Reset value																	0		0	0	0		0	0	0		0	0	0		0	0
0x024	TIMx_CNT	UIFCPY_CNT[31]	CNT[30:16] (CNT[31:16] on 32-bit timers only)															CNT[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	TIMx_PSC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	TIMx_ARR (x = 2 to 5)	ARR[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x030	Reserved	Reserved																															
0x034	TIMx_CCR1	CCR1[31:20] (32-bit timers only)												CCR1[19:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	TIMx_CCR2	CCR2[31:20] (32-bit timers only)												CCR2[19:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C	TIMx_CCR3	CCR3[31:20] (32-bit timers only)												CCR3[19:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x040	TIMx_CCR4	CCR4[31:20] (32-bit timers only)												CCR4[19:0]																			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044.. 0x054	Reserved	Res.																															
0x058	TIMx_ECR	Res.	Res.	Res.	Res.	Res.	PWPRSC [2:0]			PW[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IPOS [1:0]		FIDX	IBLK [1:0]		IDIR [1:0]		IE
	Reset value						0	0	0	0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0	0
0x05C	TIMx_TISEL	Res.	Res.	Res.	Res.	TI4SEL[3:0]				Res.	Res.	Res.	Res.	TI3SEL[3:0]				Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value					0	0	0	0					0	0	0	0					0	0	0	0					0	0	0	0
0x060	TIMx_AF1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ETRSEL [3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value															0	0	0	0														
0x064	TIMx_AF2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value														0	0	0																
0x068.. 0x3D8	Reserved	Res.																															
0x3DC	TIMx_DCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]				Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value													0	0	0	0				0	0	0	0	0				0	0	0	0	0

Table 519. TIM2/TIM3/TIM4/TIM5 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x3E0	TIMx_DMAR	DMAB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

55 Basic timers (TIM6/TIM7/TIM18)

55.1 TIM6/TIM7/TIM18 introduction

The basic timers TIM6/TIM7/TIM18 consist in a 16-bit autoreload counter driven by a programmable prescaler.

They can be used as generic timers for time-base generation.

The timers are completely independent, and do not share any resources.

55.2 TIM6/TIM7/TIM18 main features

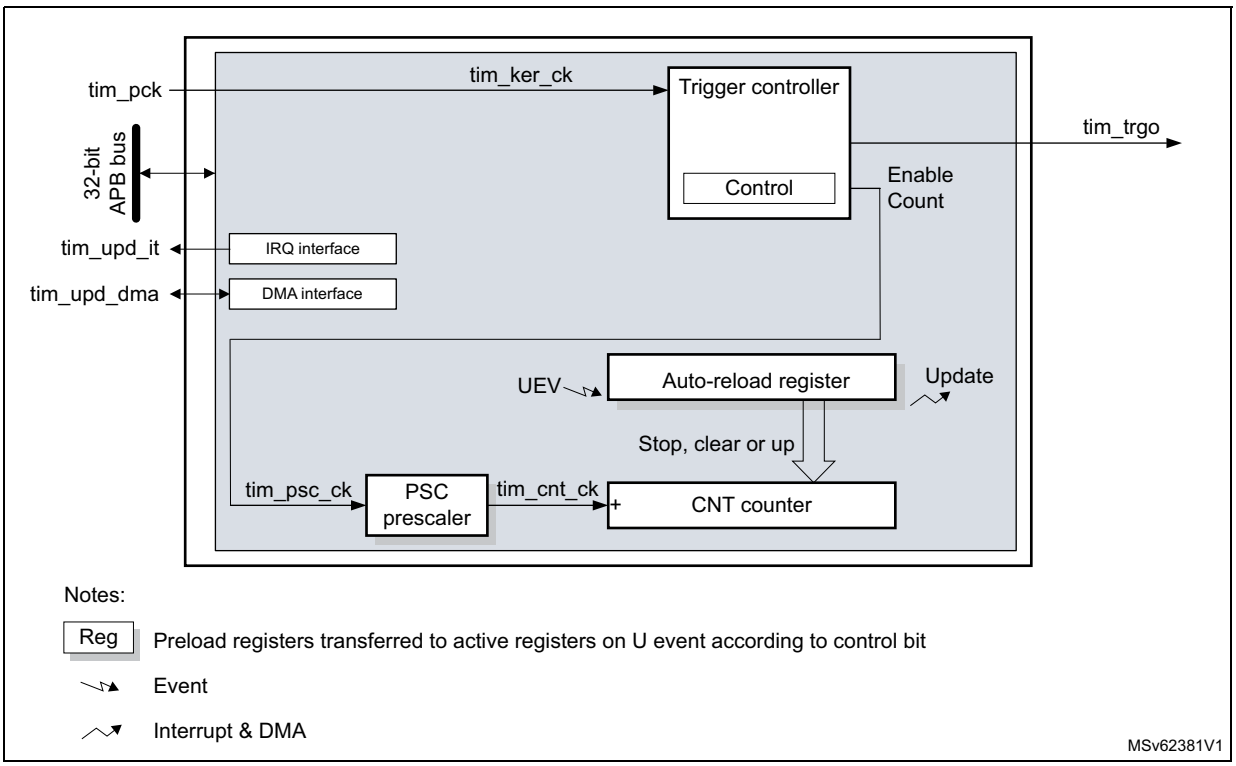
Basic timer (TIM6/TIM7/TIM18) features include:

- 16-bit autoreload upcounter.
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535.
- Interrupt/DMA generation on the update event: counter overflow.
- ADC synchronization for jitter-free sampling points.

55.3 TIM6/TIM7/TIM18 functional description

55.3.1 TIM6/TIM7/TIM18 block diagram

Figure 671. Basic timer block diagram



55.3.2 TIM6/TIM7/TIM18 internal signals

The table in this section summarizes the TIM inputs and outputs.

Table 520. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_pclk	Input	Timer APB clock
tim_ker_ck	Input	Timer kernel clock. This clock must be synchronous with tim_pclk (derived from the same source). The clock ratio tim_ker_ck/tim_pclk must be an integer: 1, 2, 3,..., 16 (maximum value)
tim_trgo	Output	Internal trigger output. This trigger can trigger other on-chip peripherals.
tim_upd_it	Output	Timer update event interrupt
tim_upd_dma	Output	Timer update dma request

55.3.3 TIM6/TIM7/TIM18 clocks

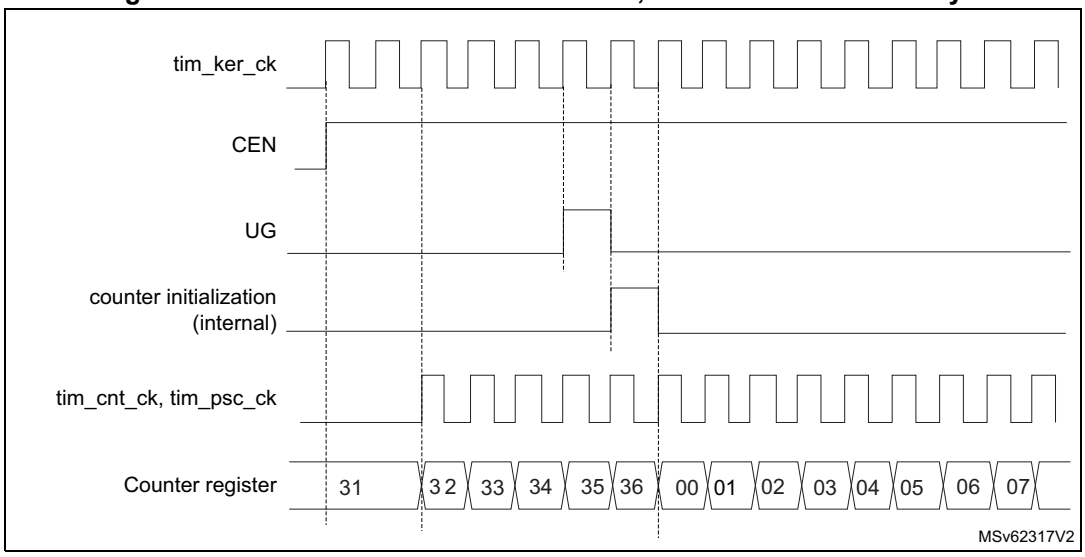
The timer bus interface is clocked by the tim_pclk APB clock.

The counter clock tim_ker_ck is connected to the tim_pclk input.

The CEN (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed only by software (except for UG that remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 672 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

Figure 672. Control circuit in normal mode, internal clock divided by 1



55.3.4 Time-base unit

The main block of the programmable timer is a 16-bit upcounter with its related autoreload register. The counter clock can be divided by a prescaler.

The counter, the autoreload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Auto-Reload register (TIMx_ARR).

The autoreload register is preloaded. The preload register is accessed each time an attempt is made to write or read the autoreload register. The contents of the preload register are transferred into the shadow register permanently or at each update event UEV, depending on the autoreload preload enable bit (ARPE) in the TIMx_CR1 register. The update event is sent when the counter reaches the overflow value and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detail for each configuration.

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in the TIMx_CR1 register is set.

Note that the actual counter enable signal tim_cnt_en is set one clock cycle after CEN bit set.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as the TIMx_PSC control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 673 and *Figure 674* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 673. Counter timing diagram with prescaler division change from 1 to 2

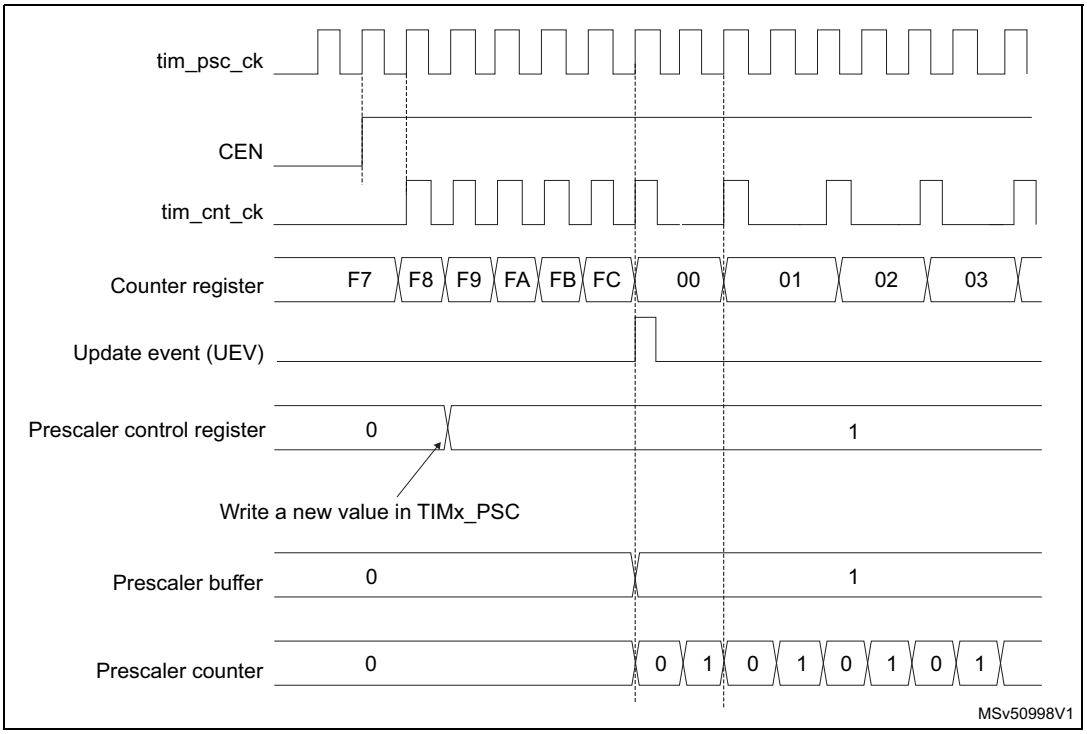
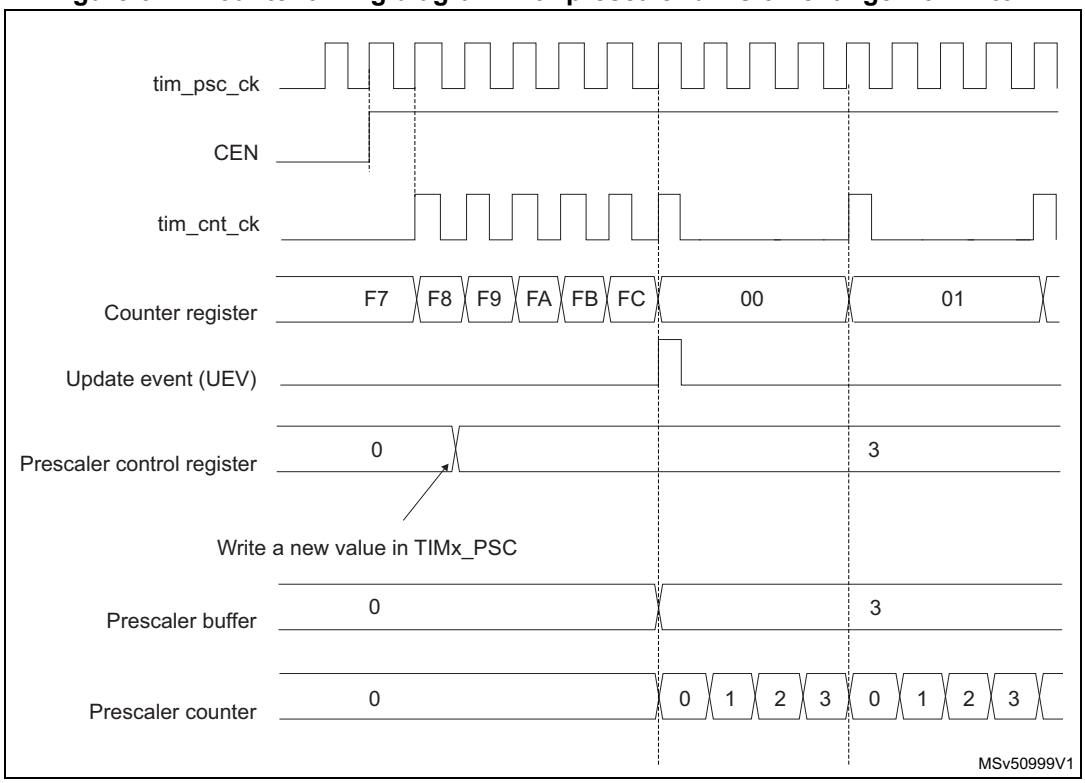


Figure 674. Counter timing diagram with prescaler division change from 1 to 4



55.3.5 Counting mode

The counter counts from 0 to the autoreload value (contents of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

An update event can be generate at each counter overflow or by setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller).

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This avoids updating the shadow registers while writing new values into the preload registers. In this way, no update event occurs until the UDIS bit has been written to 0, however, the counter and the prescaler counter both restart from 0 (but the prescale rate does not change). In addition, if the URS (update request selection) bit in the TIMx_CR1 register is set, setting the UG bit generates an update event UEV, but the UIF flag is not set (so no interrupt or DMA request is sent).

When an update event occurs, all the registers are updated and the update flag (UIF bit in the TIMx_SR register) is set (depending on the URS bit):

- The buffer of the prescaler is reloaded with the preload value (contents of the TIMx_PSC register).
- The autoreload shadow register is updated with the preload value (TIMx_ARR).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.

Figure 675. Counter timing diagram, internal clock divided by 1

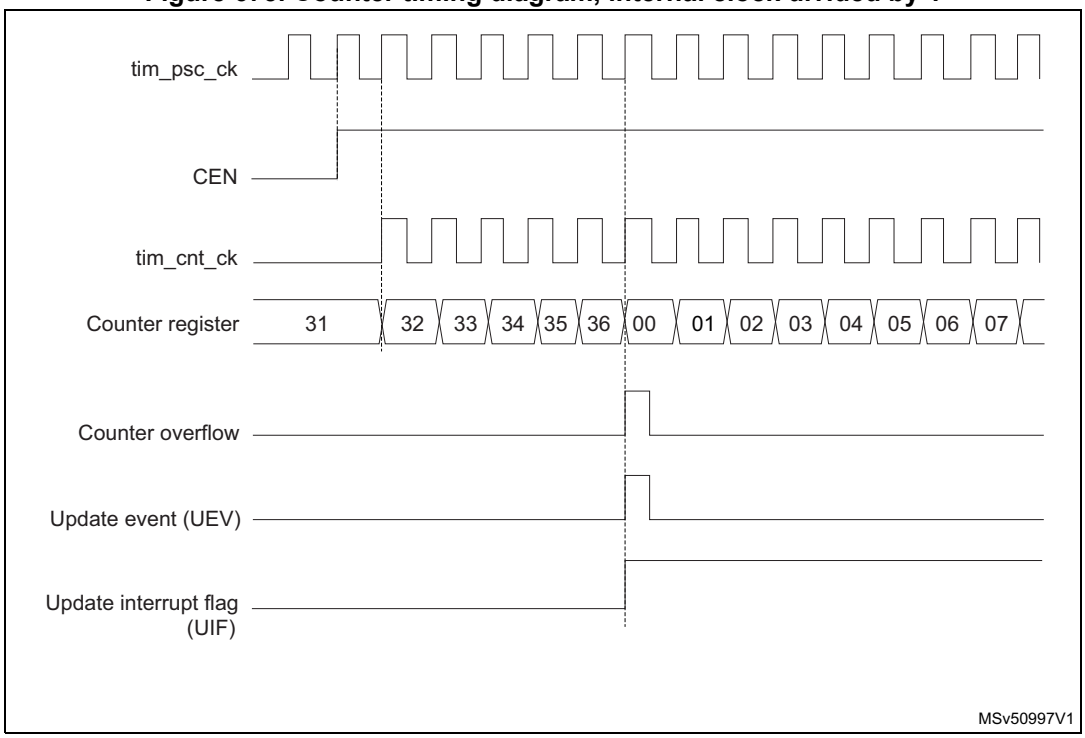


Figure 676. Counter timing diagram, internal clock divided by 2

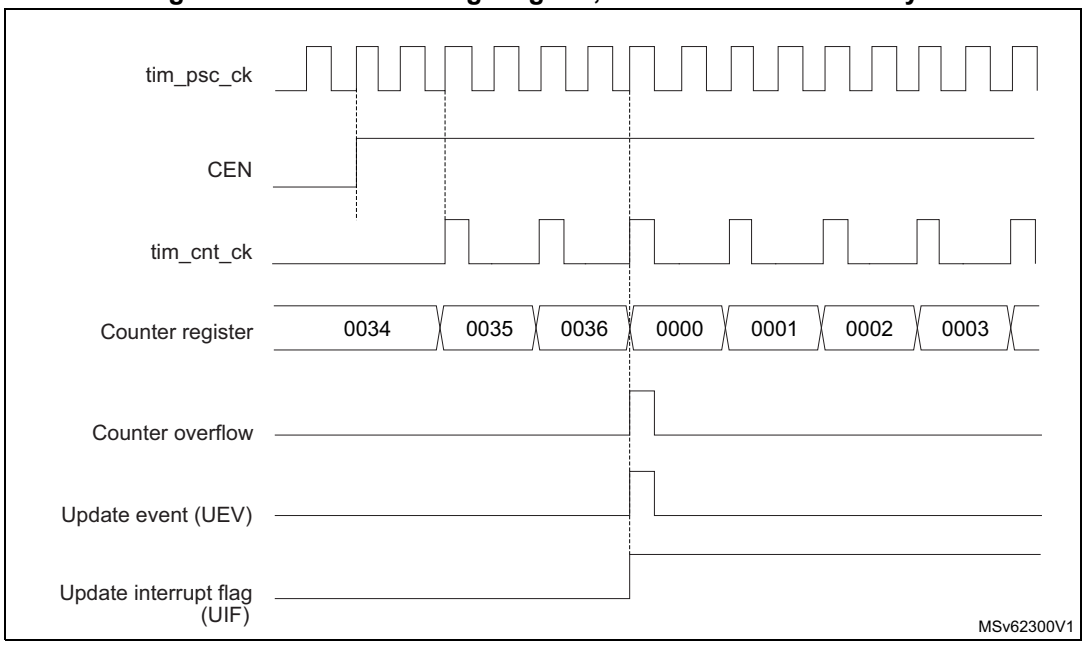


Figure 677. Counter timing diagram, internal clock divided by 4

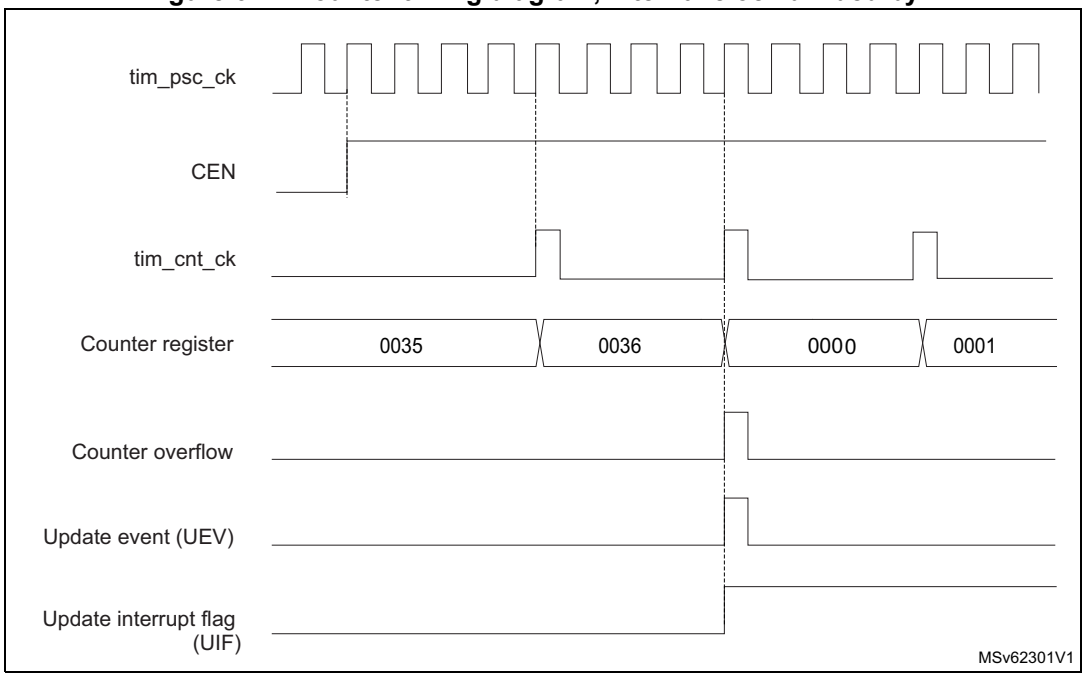


Figure 678. Counter timing diagram, internal clock divided by N

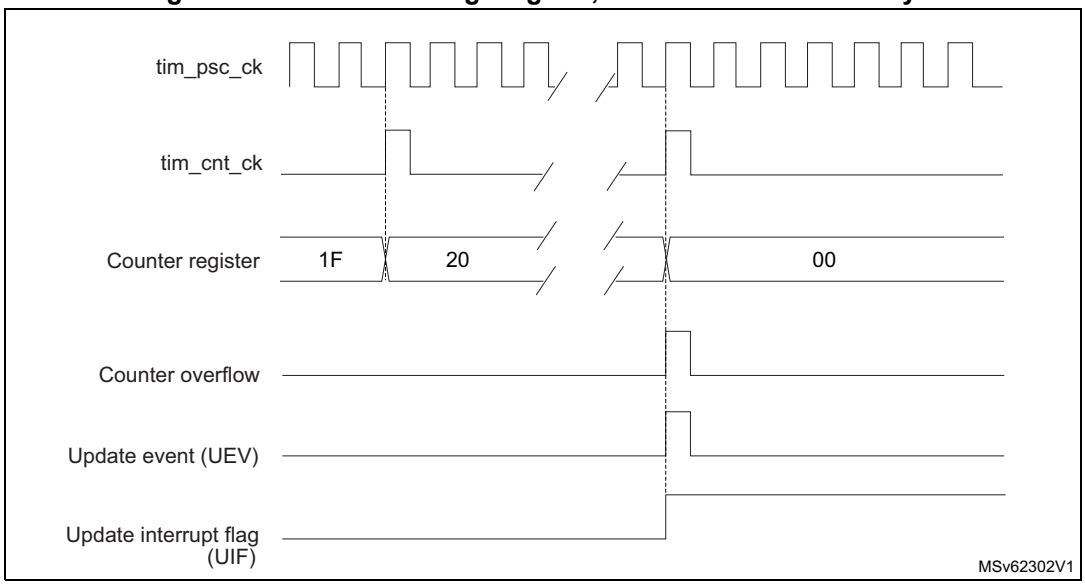


Figure 679. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded)

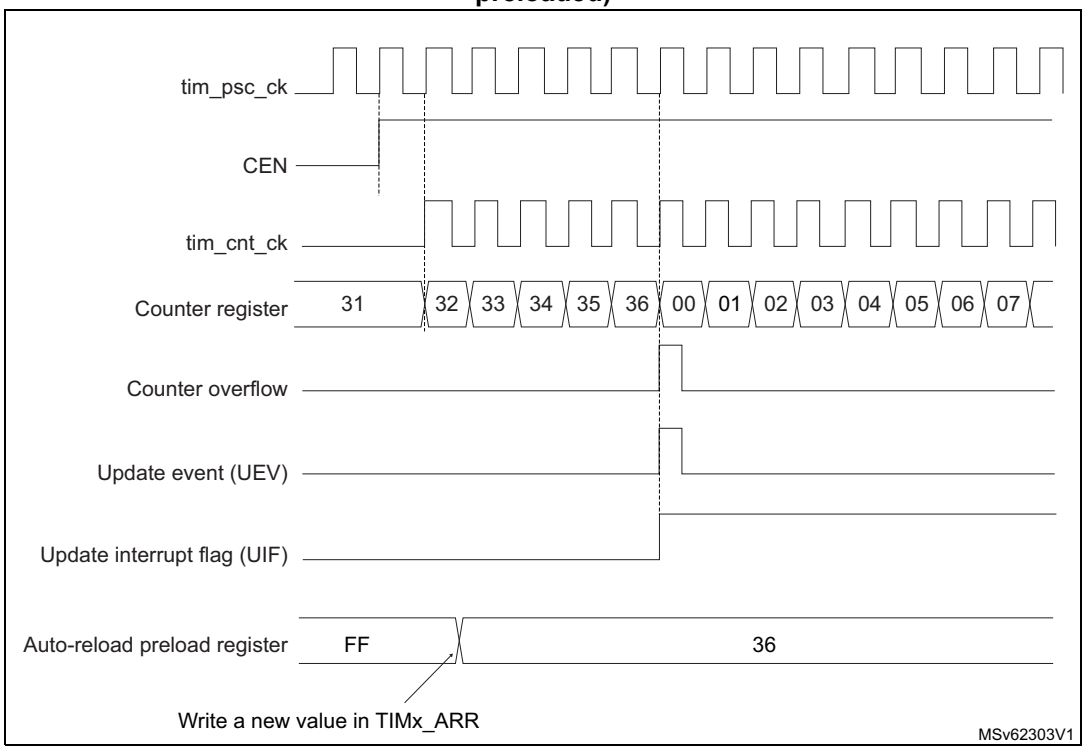
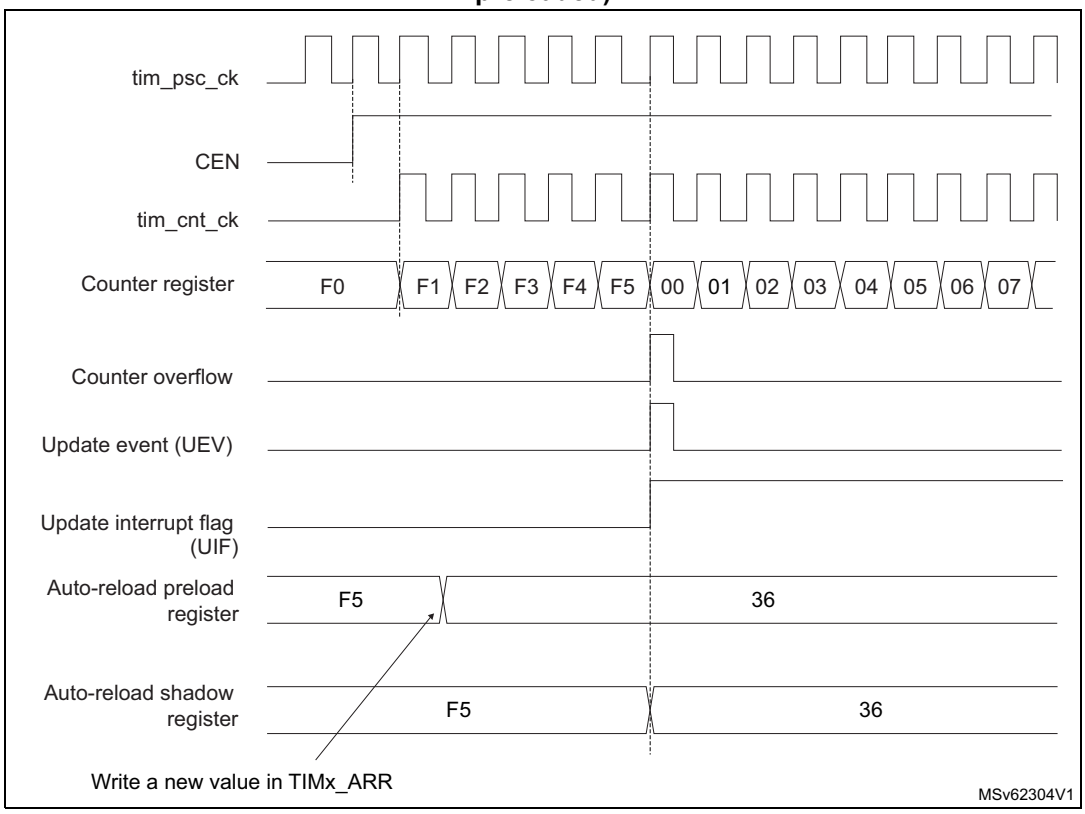


Figure 680. Counter timing diagram, update event when ARPE = 1 (TIMx_ARR preloaded)



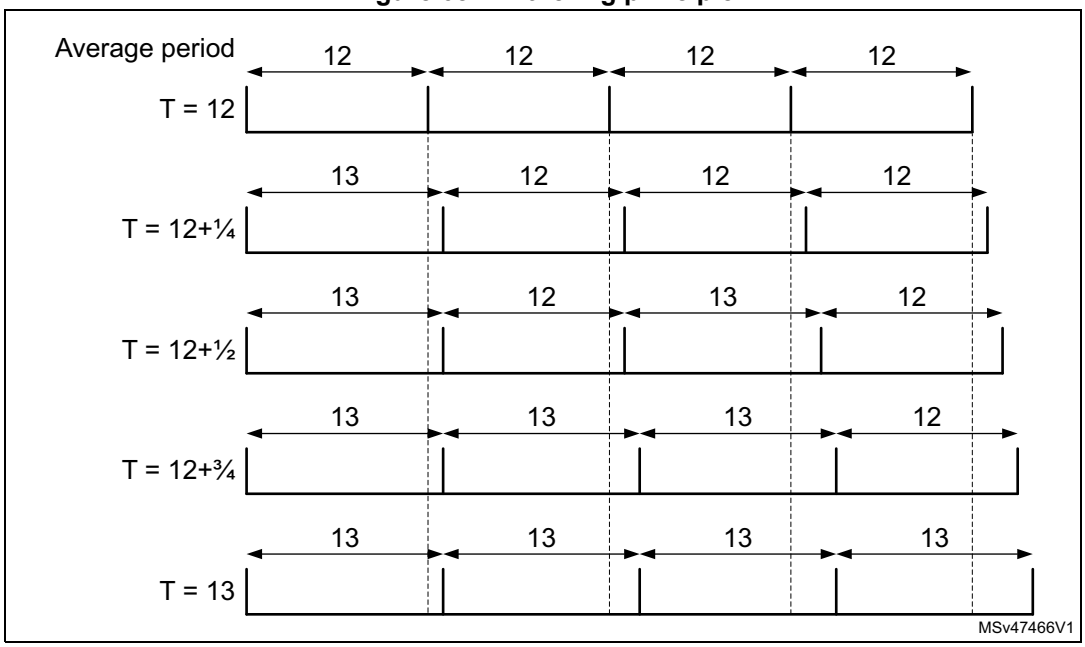
Dithering mode

The time base effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIMx_CR1 register. This affects the way the TIMx_ARR is behaving, and is useful for adjusting the average counter period when the timer is used as a trigger.

The operating principle is to have the actual ARR value slightly changed (adding or not one timer clock period) over 16 consecutive counting periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average counting period.

Figure 681 presents the dithering principle applied to four consecutive counting periods.

Figure 681. Dithering principle

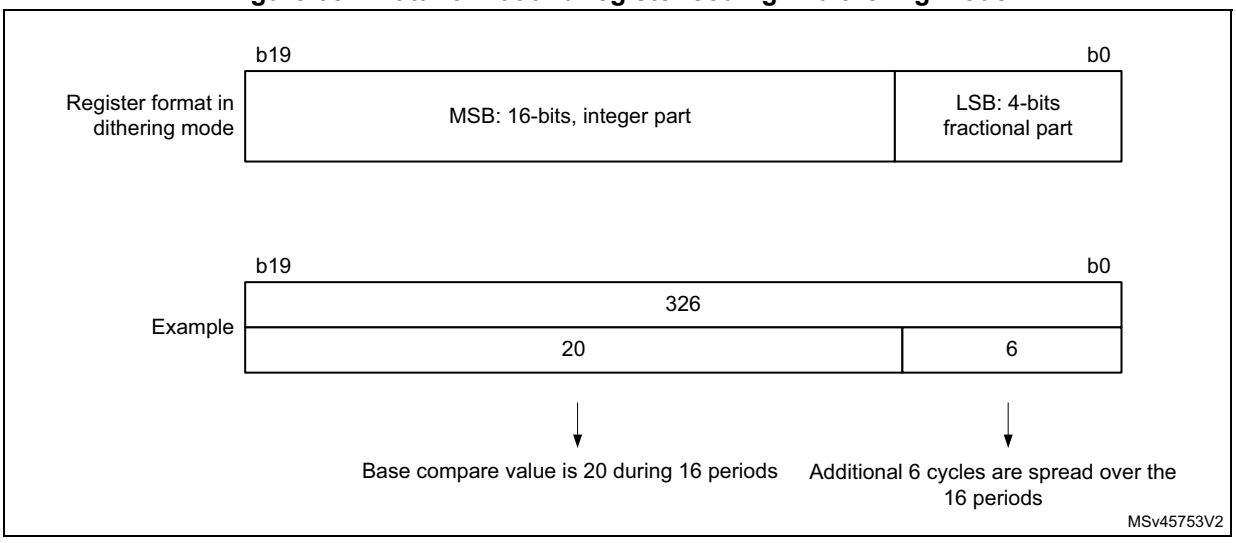


When the dithering mode is enabled, the register coding is changed as follows (see *Figure 682* for example):

- The four LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted to the bits 19:4 and are coding for the base value.

Note: The following sequence must be followed when resetting the DITHEN bit:
1. CEN and ARPE bits must be reset
2. The DITHEN bit must be reset
3. The CEN bit can be set (eventually with ARPE = 1).

Figure 682. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

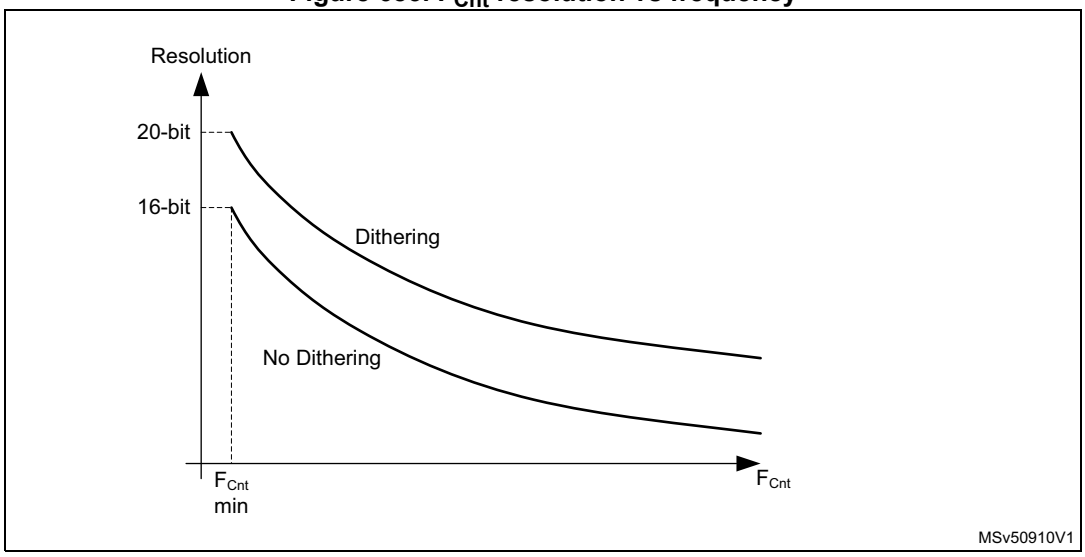
$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode enabled: } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

Note: The maximum TIMx_ARR value is limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).

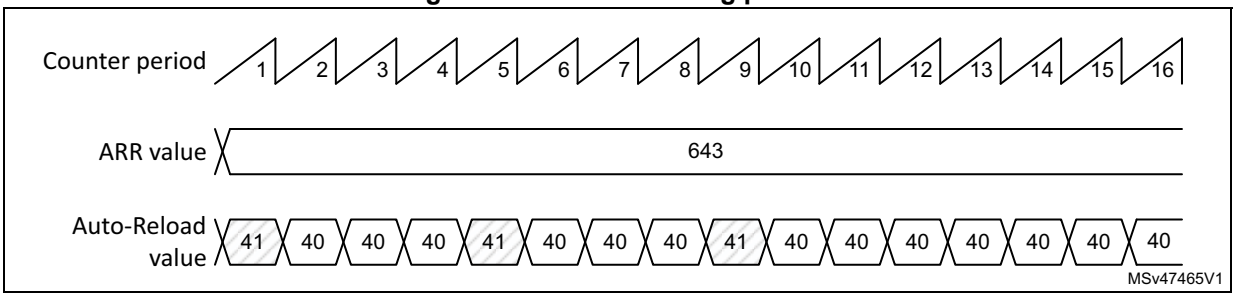
As shown on *Figure 683*, the dithering mode is used to increase the PWM resolution whatever the PWM frequency.

Figure 683. F_{Cnt} resolution vs frequency



The period changes are spread over 16 consecutive periods, as described in *Figure 684*.

Figure 684. PWM dithering pattern



The autoreload and compare values increments are spread following the specific patterns described in *Table 521*. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 521. TIMx_ARR register change dithering pattern

-	PWM period															
LSB value	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

55.3.6 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag UIF into the timer counter register's bit 31 (TIMxCNT[31]). This is used to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

There is no latency between the assertions of the UIF and UIFCPY flags.

55.3.7 ADC triggers

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events.

Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receiving events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

55.3.8 ADC synchronization

The timer operation can be synchronized to the ADC clock to trigger jitter-free ADC sampling. This function is enabled using the ADSYNC bit in the TIMx_CR2 register.

This feature is useful when the timers and the ADCs are operating with semisynchronous clocks (clocks derived from a same source with integer ratio tim_ker_ck/adc_ker_ck), for instance adc_ker_ck = 75 MHz and tim_ker_ck = 150 MHz or 300 MHz.

ADSYNC must also be set when both peripherals are operating at the same frequency from the same clock source, when jitter-free operation is needed.

ADSYNC must not be set and jitter-free operation is not supported in the following cases:

- When the clock ratio is not an integer (for example adc_ker_ck = 75 MHz and tim_ker_ck = 100 MHz): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum.
- When the ADC is operating in asynchronous mode (adc_ker_ck uncorrelated with tim_ker_ck): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum.

When ADSYNC = 1, the timer operation is slightly changed: the counter enable and counter reset events are aligned to the adc_ker_ck ADC clock, to avoid any phase shift due to clocks enable in the RCC.

Jitter-free operation is guaranteed only when one of the two requirements below is met (depending on the selected trigger source).

1. The counter period must be a multiple of the ADC clock period

$$(\text{TIMx_PSC} + 1) \times (\text{TIMx_ARR} + 1) \times T_{\text{tim_ker_ck}} = n \times T_{\text{adc_ker_ck}}$$

2. The compare value must be a multiple of the ADC clock period

$$(\text{TIMx_PSC} + 1) \times \text{TIMx_CMPy} \times T_{\text{tim_ker_ck}} = m \times T_{\text{adc_ker_ck}}$$

Note: If none of the two above requirements are met, the trigger is still generated, but the latency is not constant and varies with the timer and ADC clocks phase shift.

Programming guidelines

The ADC synchronization feature must not be modified during run-time, once the counter is enabled and once the ADC has been configured for receiving triggers from the timer.

It is mandatory to follow the procedure below to use the ADC synchronization:

1. Enable the destination ADC clock
2. Configure the timer and set the ADSYNC bit
3. Configure the ADC and enable it (using ADSTART and/or JADSTART bits)
4. Start the timer (with the CEN counter enable bit)

55.3.9 TIM6/TIM7/TIM18 DMA requests

The TIM6/TIM7/TIM18 can generate a single DMA request, as shown in *Table 522*.

Table 522. DMA request

DMA acronym	DMA request	Enable control bit
tim_upd_dma	Update	UDE

55.3.10 Debug mode

When the microcontroller enters debug mode (Cortex®-M55 core halted), the TIMx counter can either continue to work normally or be stopped.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For more details, refer to section Debug support (DBG).

55.3.11 TIM6/TIM7/TIM18 low-power modes

Table 523. Effect of low-power modes on TIM6/TIM7/TIM18

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Standby	The timer is powered-down and must be reinitialized after exiting the Standby mode.

55.3.12 TIM6/TIM7/TIM18 interrupts

The TIM6/TIM7/TIM18 can generate a single interrupt, as shown in *Table 524*.

Table 524. Interrupt request

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
TIM6 TIM7	Update	UIF	UIE	write 0 in UIF	Yes	No

55.4 TIM6/TIM7/TIM18 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

55.4.1 TIMx control register 1 (TIMx_CR1)(x = 6, 7, 18)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	Res.	Res.	ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
			rw	rw				rw				rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable
0: Dithering disabled
1: Dithering enabled
Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bits 10:8 Reserved, must be kept at reset value.

Bit 7 **ARPE**: Auto-reload preload enable
0: TIMx_ARR register is not buffered.
1: TIMx_ARR register is buffered.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the CEN bit).

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generates an update interrupt or DMA request if enabled. These events can be:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

1: Only counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

- Counter overflow/underflow
- Setting the UG bit
- Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

CEN is cleared automatically in one-pulse mode, when an update event occurs.

55.4.2 TIMx control register 2 (TIMx_CR2)(x = 6, 7, 18)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	AD SYNC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			rw												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MMS[2:0]			Res.	Res.	Res.	Res.
									rw	rw	rw				

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **ADSYNC**: ADC synchronization

0: The timer operates independently from the ADC

1: The timer operation is synchronized with the ADC clock to provide jitter-free sampling point. This mode can be enabled only with specific ADC / timer clock relationship. Refer to *Section 55.3.8* for requirements.

The ADSYNC must not modified when the counter is enabled (CEN bit is set).

Bits 27:7 Reserved, must be kept at reset value.

Bits 6:4 **MMS[2:0]**: Master mode selection

These bits are used to select the information to be sent in master mode to slave timers for synchronization (TRGO). The combination is as follows:

000:**Reset** - the UG bit from the TIMx_EGR register is used as a trigger output (tim_trgo).

001:**Enable** - the Counter enable signal, tim_cnt_en, is used as a trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enabled. The Counter Enable signal is generated when the CEN control bit is written.

010:**Update** - The update event is selected as a trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

Note: The clock of the slave timer or he peripheral receiving the tim_trgo must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

Bits 3:0 Reserved, must be kept at reset value.

55.4.3 TIMx DMA/Interrupt enable register (TIMx_DIER)(x = 6, 7, 18)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	UDE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIE
							rw								rw

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled.
1: Update DMA request enabled.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled.
1: Update interrupt enabled.

55.4.4 TIMx status register (TIMx_SR)(x = 6, 7, 18)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIF
															rc_w0

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **UIF**: Update interrupt flag
This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
–On counter overflow if UDIS = 0 in the TIMx_CR1 register.
–When CNT is reinitialized by software using the UG bit in the TIMx_EGR register, if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

55.4.5 TIMx event generation register (TIMx_EGR)(x = 6, 7, 18)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UG
															w

Bits 15:1 Reserved, must be kept at reset value.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action.
1: Re-initializes the timer counter and generates an update of the registers. Note that the prescaler counter is cleared too (but the prescaler ratio is not affected).

55.4.6 TIMx counter (TIMx_CNT)(x = 6, 7, 18)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF copy

This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in TIMx_CR1 is reset, bit 31 is reserved and read as 0.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

Non-dithering mode (DITHEN = 0)

The register holds the counter value.

Dithering mode (DITHEN = 1)

The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

55.4.7 TIMx prescaler (TIMx_PSC)(x = 6, 7, 18)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency $f_{tim_cnt_ck}$ is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).

PSC contains the value to be loaded into the active prescaler register at each update event. (including when the counter is cleared through UG bit of TIMx_EGR register.

55.4.8 TIMx autoreload register (TIMx_ARR)(x = 6, 7, 18)

Address offset: 0x2C

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Auto-reload value

ARR is the value to be loaded into the actual auto-reload register.

Refer to *Section 55.3.4: Time-base unit* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reserved.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

55.4.9 TIMx register map

TIMx registers are mapped as 16-bit addressable registers as described in the table below:

Table 525. TIMx register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TIMx_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DITHEN	UIFREMA	Res.	Res.	Res.	ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value																				0	0				0				0	0	0	0
0x04	TIMx_CR2	Res.	Res.	Res.	ADSYNC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MMS [2:0]			Res.	Res.	Res.	Res.
	Reset value				0																						0	0	0				
0x08	Reserved																																
0x0C	TIMx_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UDE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIE
	Reset value																								0								0
0x10	TIMx_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIF
	Reset value																																0
0x14	TIMx_EGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UG
	Reset value																																0
0x18-0x20	Reserved																																
0x24	TIMx_CNT	UIFCPY or Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	TIMx_PSC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	TIMx_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:0]																			
	Reset value													0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

56 General-purpose timers (TIM9/TIM10/TIM11/TIM12/TIM13/TIM14)

56.1 TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 introduction

The TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 general-purpose timers consist in a 16-bit autoreload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 timers are completely independent, and do not share any resources. They can be synchronized together as described in *Section 56.4.20: Timer synchronization (TIM9/TIM12 only)*.

56.2 TIM9/TIM12 main features

The features of the TIM9/TIM12 general-purpose timers include:

- 16-bit autoreload upcounter
- 16-bit programmable prescaler used to divide the counter clock frequency by any factor between 1 and 65536 (can be changed “on the fly”)
- Up to two independent channels for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Synchronization circuit to control the timer with external signals and to interconnect several timers together
- Interrupt generation on the following events:
  - Update: counter overflow, counter initialization (by software or internal trigger)
  - Trigger event (counter start, stop, initialization, or count by internal trigger)
  - Input capture
  - Output compare
- ADC synchronization for jitter-free sampling points

56.3 TIM10/TIM11/TIM13/TIM14 main features

The features of general-purpose timers TIM10/TIM11/TIM13/TIM14 include:

- 16-bit autoreload upcounter
- 16-bit programmable prescaler used to divide the counter clock frequency by any factor between 1 and 65536 (can be changed “on the fly”)
- Independent channel for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Interrupt generation on the following events:
  - Update: counter overflow, counter initialization (by software)
  - Input capture
  - Output compare

56.4 TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 functional description

56.4.1 Block diagram

Figure 685. General-purpose timer block diagram (TIM9/TIM12)

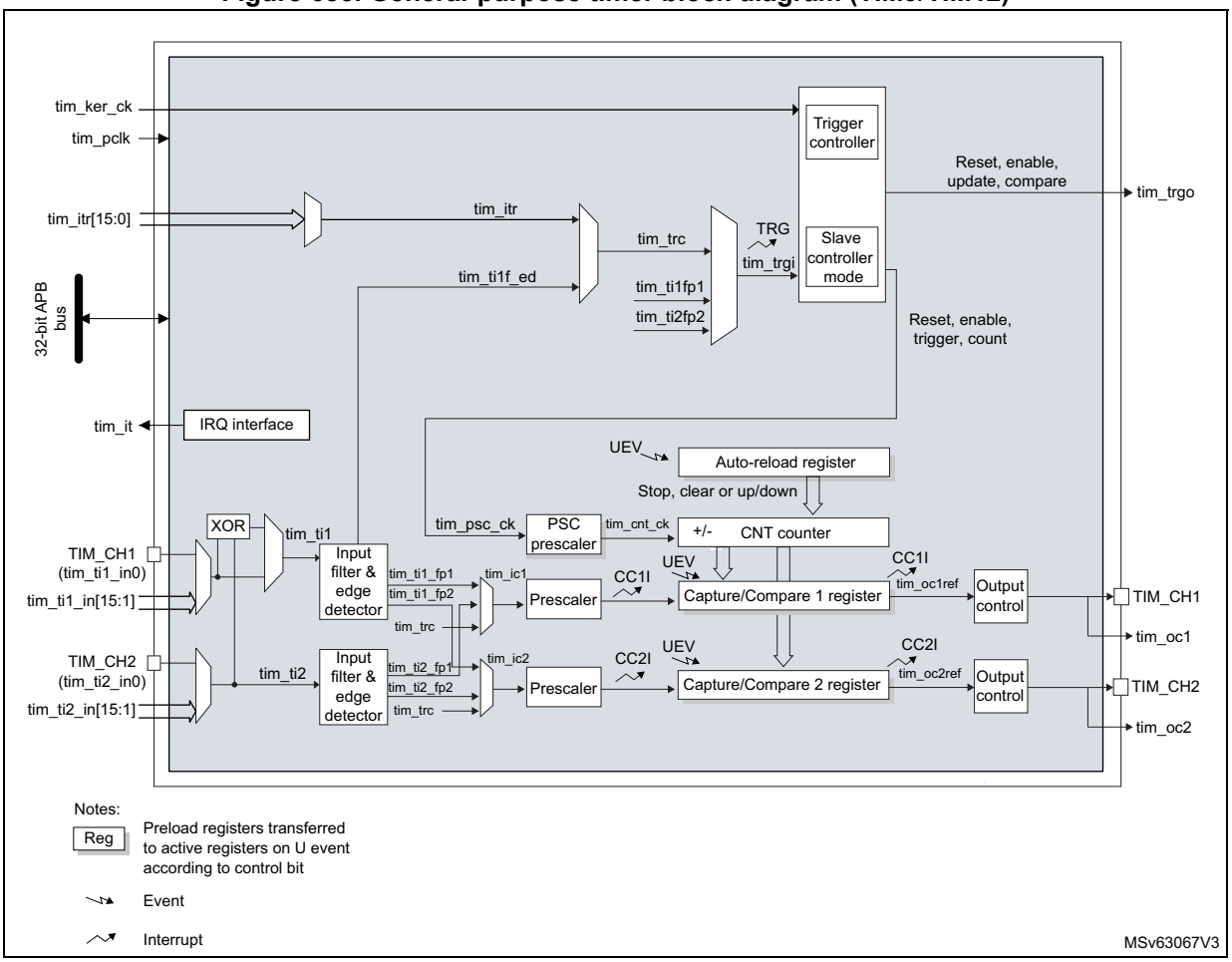
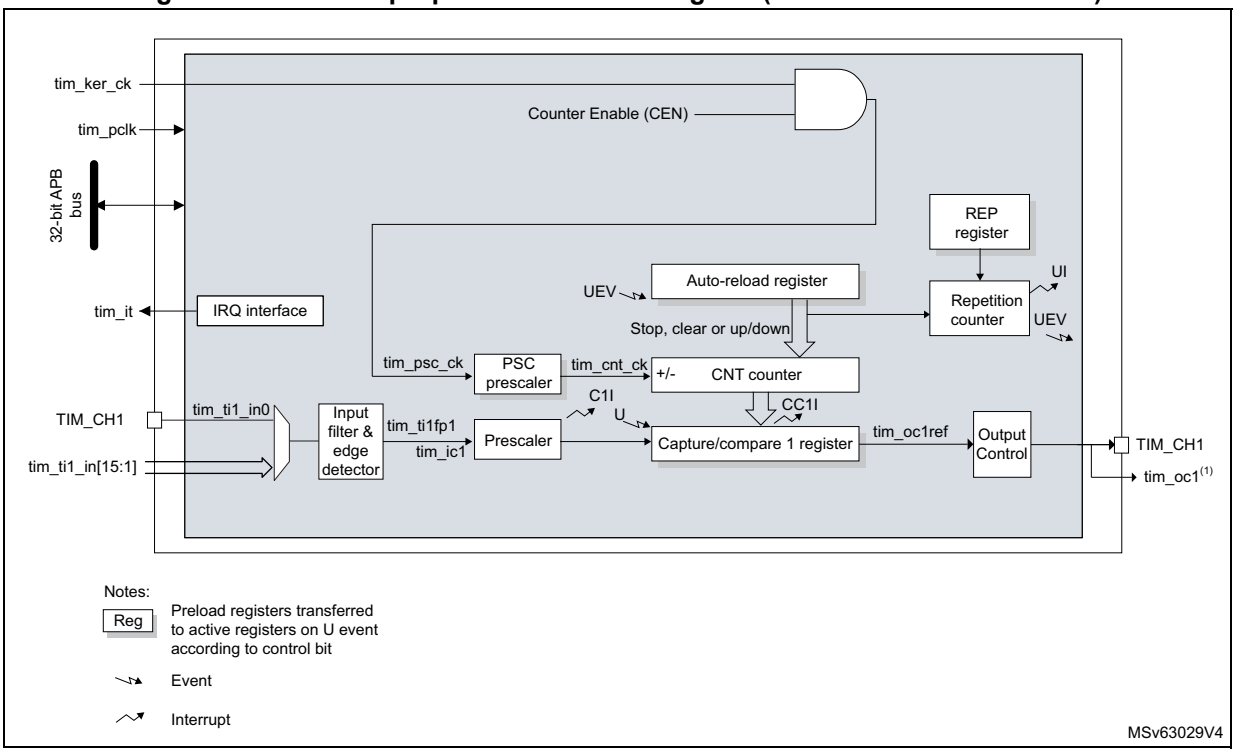


Figure 686. General-purpose timer block diagram (TIM10/TIM11/TIM13/TIM14)



1. This signal can be used as trigger for some slave timer (see internal trigger connection table in next section). See *Section 56.4.21: Using timer output as trigger for other timers (TIM10/TIM11/TIM13/TIM14 only)* for details.

56.4.2 TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 pins and internal signals

Table 526 and *Table 527* in this section summarize the TIM inputs and outputs.

Table 526. TIM input/output pins

Pin name	Signal type	Description
TIM_CH1 TIM_CH2(1)	Input/Output	Timer multi-purpose channels. Each channel be used for capture, compare, or PWM. TIM_CH1 and TIM_CH2 can also be used as external clock (below 1/4 of the tim_ker_ck clock) and external trigger inputs.

1. Available for TIM9/TIM12 only.

Table 527. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_ti1_in[15:0] tim_ti2_in[15:0](1)	Input	Internal timer inputs bus. These inputs can be used for capture or as external clock (below 1/4 of the tim_ker_ck clock).
tim_itr[15:0](1)	Input	Internal trigger input bus. These inputs can be used for the slave mode controller or as a input clock (below 1/4 of the tim_ker_ck clock).
tim_oc1 tim_oc2(1)	Output	Internal timer output. Can be used for triggering other timers or the ADC(s).

Table 527. TIM internal input/output signals (continued)

Internal signal name	Signal type	Description
tim_trgo(1)	Output	Internal trigger output. This trigger can trigger other on-chip peripherals.
tim_pclk	Input	Timer APB clock
tim_ker_ck	Input	Timer kernel clock. This clock must be synchronous with tim_pclk (derived from the same source). The clock ratio tim_ker_ck/tim_pclk must be an integer:1, 2, 3,..., 16 (maximum value)
tim_it	Output	Global Timer interrupt, gathering capture/compare, update, break trigger and commutation requests

1. Available for TIM9/TIM12 only.

Table 528 and *Table 530* list the sources connected to the tim_ti[2:1] input multiplexers.

Table 528. Interconnect to the tim_ti1 input multiplexer

tim_ti1 inputs	Sources					
	TIM9	TIM10	TIM11	TIM12	TIM13	TIM14
tim_ti1_in0	TIM9_CH1	TIM10_CH1	TIM11_CH1	TIM12_CH1	TIM13_CH1	TIM14_CH1
tim_ti1_in1	Reserved	i3c1_ibi_ack	i3c2_ibi_ack	spdifrx_ frame_sync	i3c1_ibi_ack	i3c2_ibi_ack
tim_ti1_in2		Reserved	Reserved	HSI/1024	Reserved	Reserved
tim_ti1_in3				CSI/128		
tim_ti1_in4	RCC_MCO1			RCC_MCO1		
tim_ti1_in5	RCC_MCO2			RCC_MCO2		
tim_ti1_in[15:6]	Reserved					

Table 529. Interconnect to the tim_ti2 input multiplexer

tim_ti2 inputs	Sources	
	TIM9	TIM12
tim_ti2_in0	TIM9_CH2	TIM12_CH2
tim_ti2_in[15:1]	Reserved	

Table 530 lists the internal sources connected to the tim_itr input multiplexer.

Table 530. TIMx internal trigger connection

TIMx	TIM9	TIM12
tim_itr0	tim1_trgo	tim1_trgo
tim_itr1	tim2_trgo	tim2_trgo
tim_itr2	tim3_trgo	tim3_trgo

Table 530. TIMx internal trigger connection (continued)

TIMx	TIM9	TIM12
tim_itr3	tim4_trgo	tim4_trgo
tim_itr4	tim5_trgo	tim5_trgo
tim_itr5	tim8_trgo	tim8_trgo
tim_itr6	Reserved	tim9_trgo
tim_itr7	tim12_trgo	Reserved
tim_itr8	tim13_oc1	tim13_oc1
tim_itr9	tim14_oc1	tim14_oc1
tim_itr10	tim15_trgo	tim15_trgo
tim_itr11	tim16_oc1	tim16_oc1
tim_itr12	tim17_oc1	tim17_oc1
tim_itr[15:13]	Reserved	Reserved

56.4.3 Time-base unit

The main block of the timer is a 16-bit up-counter with its related autoreload register. The counter clock can be divided by a prescaler.

The counter, the autoreload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Autoreload register (TIMx_ARR).

The autoreload register is preloaded. Writing to or reading from the auto-reload register accesses the preload register. The content of the preload register is transferred into the shadow register permanently or at each update event (UEV), depending on the autoreload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in details for each configuration.

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting one clock cycle after setting the CEN bit in the TIMx_CR1 register.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Figure 687 and *Figure 688* give some examples of the counter behavior when the prescaler ratio is changed on the fly.

Figure 687. Counter timing diagram with prescaler division change from 1 to 2

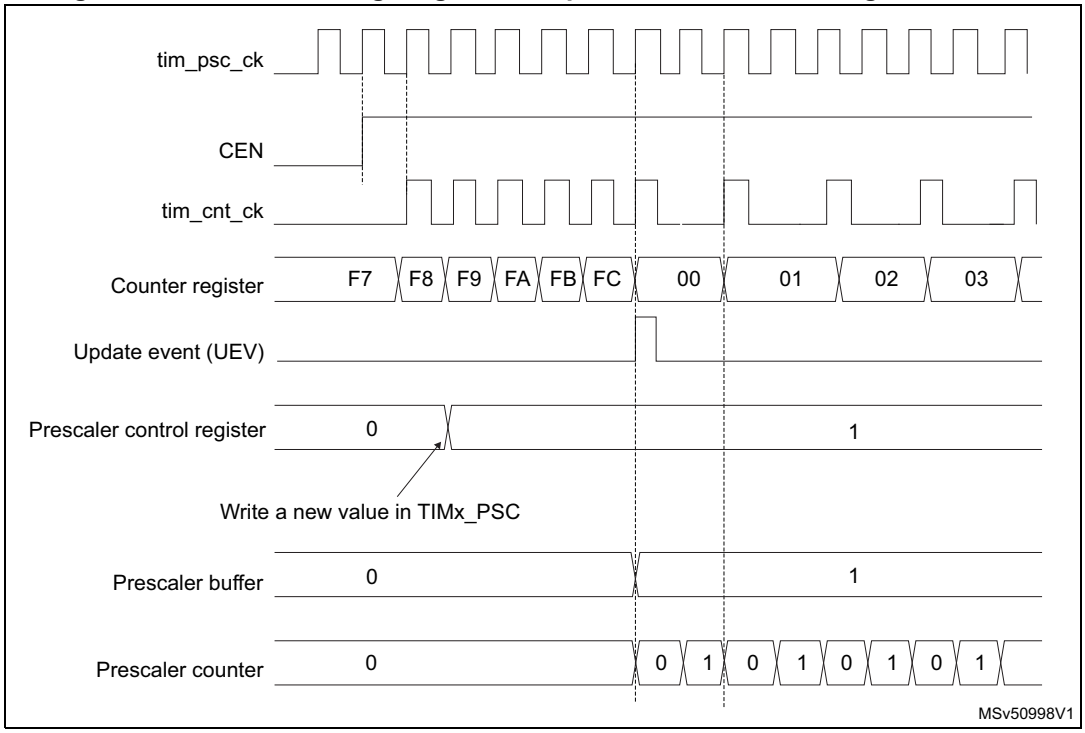
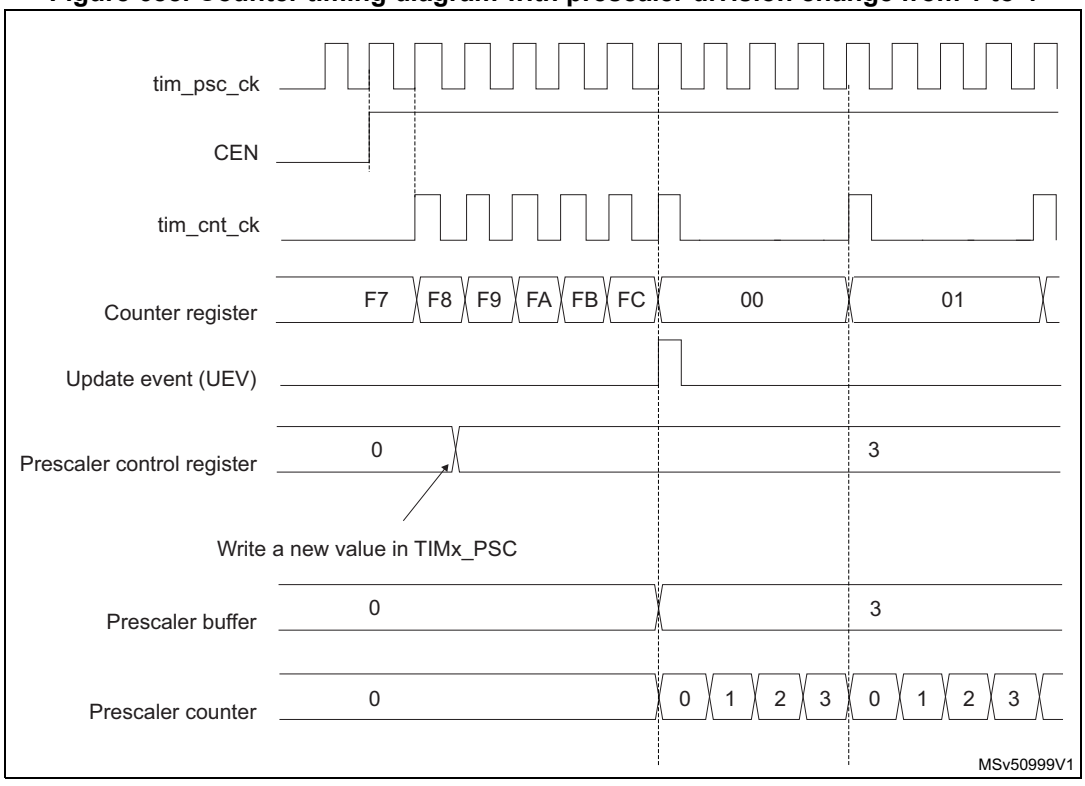


Figure 688. Counter timing diagram with prescaler division change from 1 to 4



56.4.4 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the autoreload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller on TIM9/TIM12) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The autoreload shadow register is updated with the preload value (TIMx_ARR).
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.

Figure 689. Counter timing diagram, internal clock divided by 1

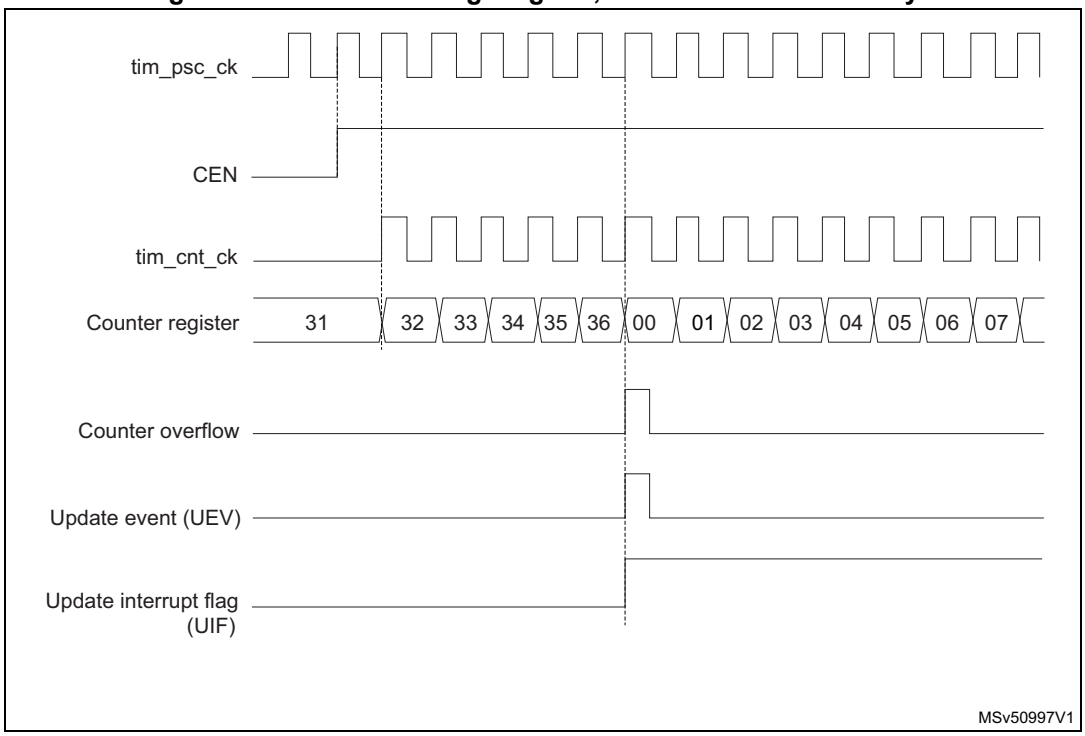


Figure 690. Counter timing diagram, internal clock divided by 2

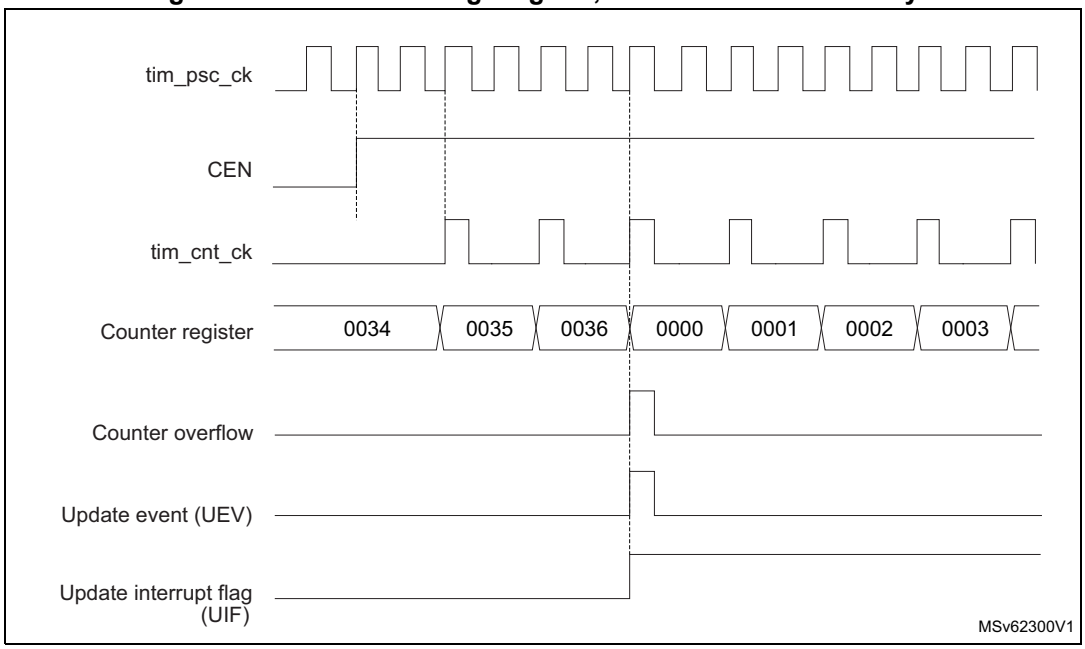


Figure 691. Counter timing diagram, internal clock divided by 4

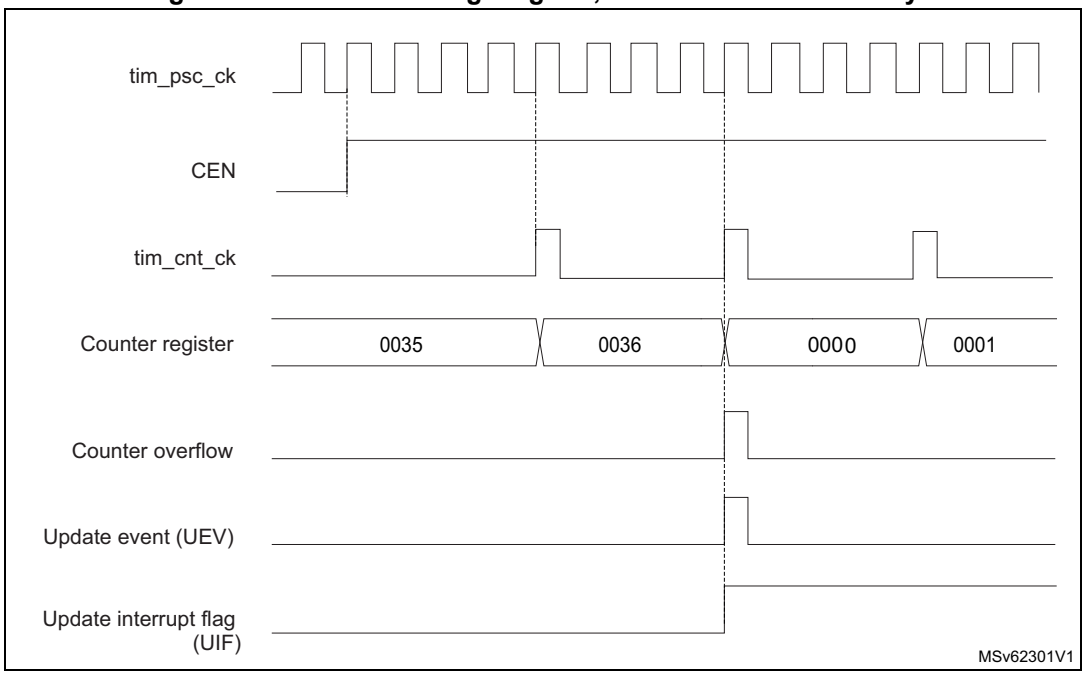


Figure 692. Counter timing diagram, internal clock divided by N

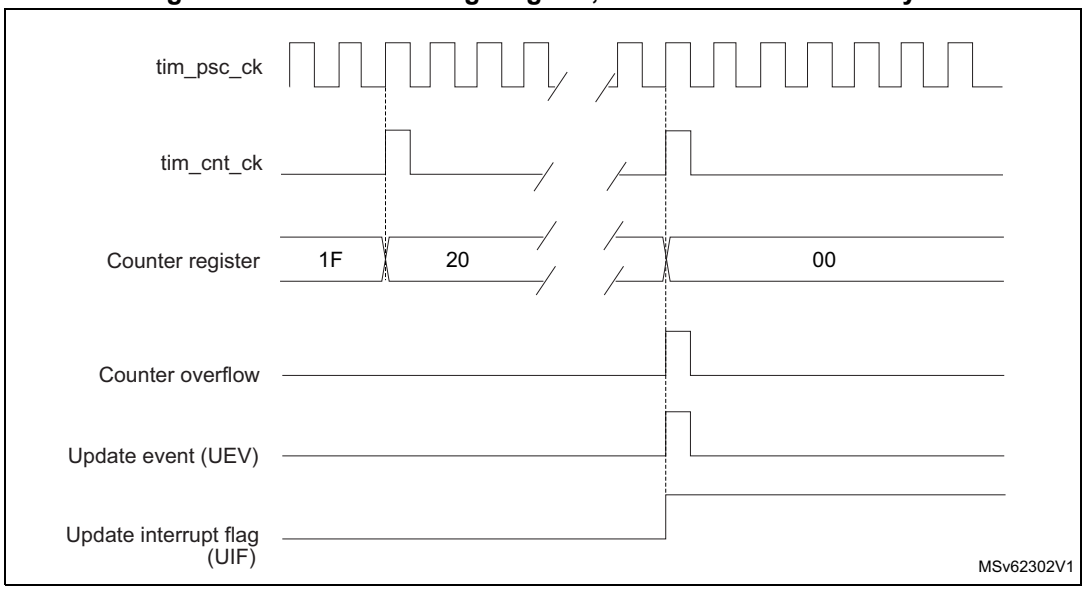


Figure 693. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded)

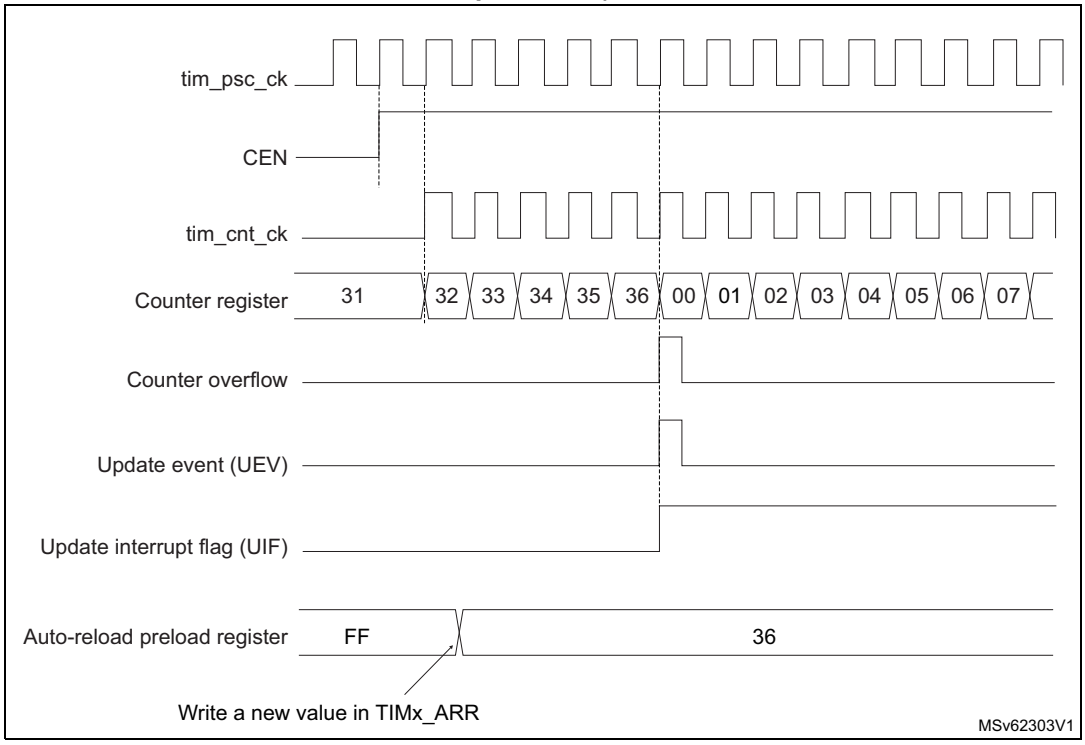
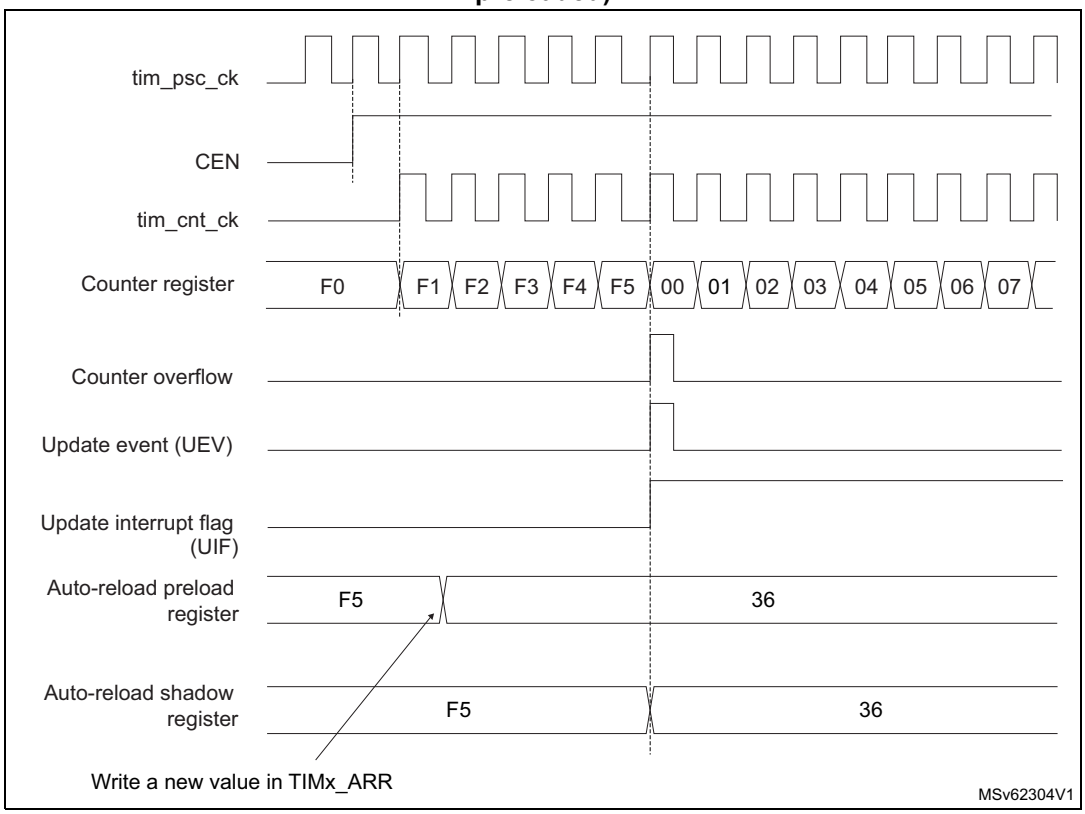


Figure 694. Counter timing diagram, update event when ARPE = 1 (TIMx_ARR preloaded)



56.4.5 Clock selection

The counter clock can be provided by the following clock sources:

- Internal clock (tim_ker_ck)
- External clock mode1 (for TIM9/TIM12): external input pin (tim_ti1 or tim_ti2, if available)
- Internal trigger inputs (tim_itrx) (for TIM9/TIM12): connecting the trigger output from another timer. For instance, another timer can be configured as a prescaler for TIM12. Refer to *Section : Using one timer as prescaler for another timer* for more details.

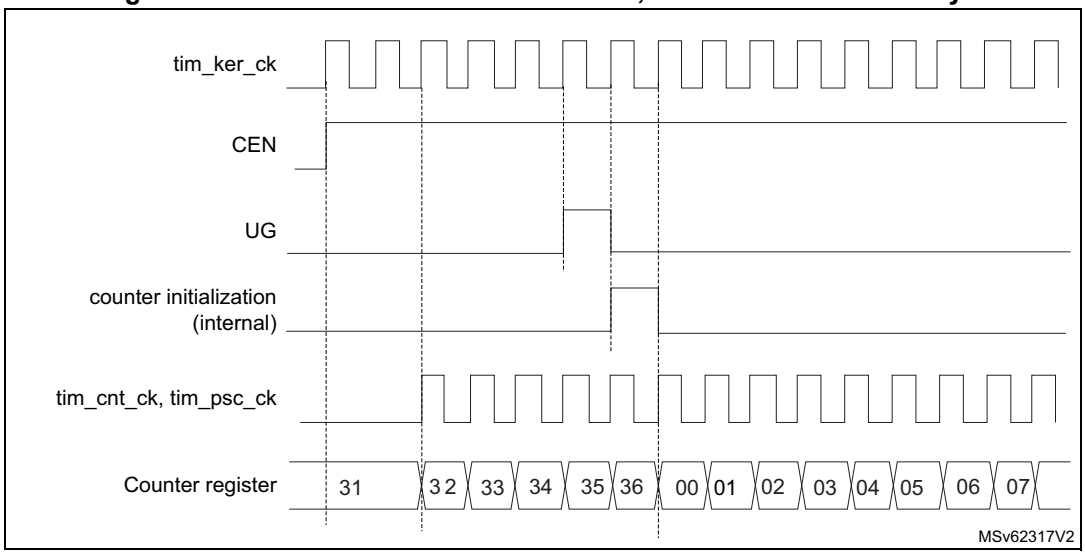
Internal clock source (tim_ker_ck)

The internal clock source is the default clock source for TIM10/TIM11/TIM13/TIM14.

For TIM9/TIM12, the internal clock source is selected when the slave mode controller is disabled (SMS = 000). The CEN bit in the TIMx_CR1 register and the UG bit in the TIMx_EGR register are then used as control bits and can be changed only by software (except for UG which remains cleared). As soon as the CEN bit is programmed to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 695 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

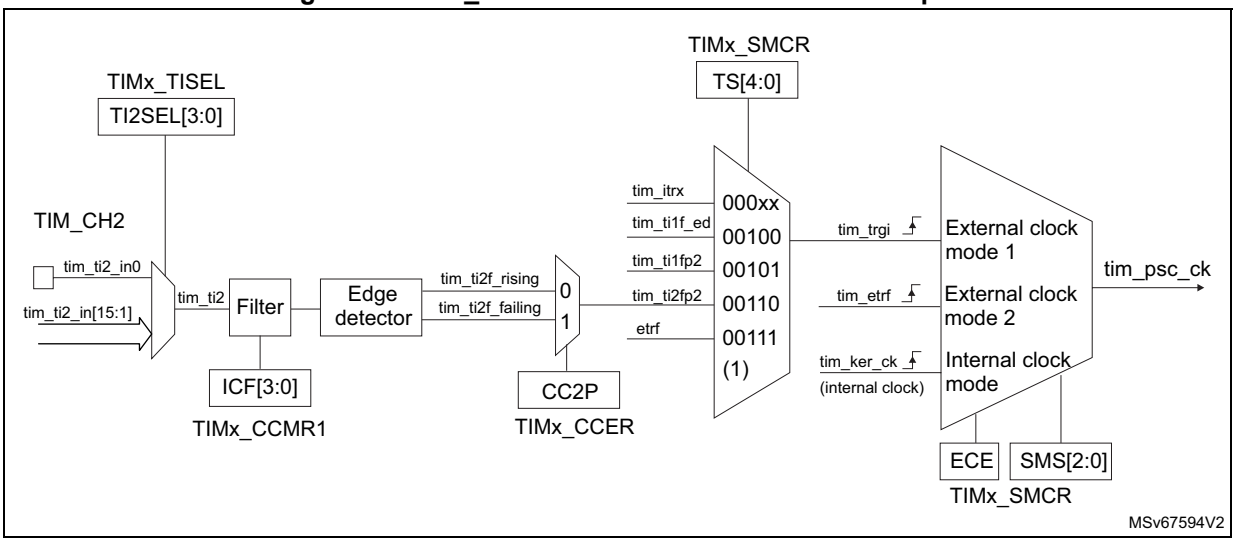
Figure 695. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1 (TIM9/TIM12 only)

This mode is selected when SMS = 111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 696. tim_ti2 external clock connection example



For example, to configure the upcounter to count in response to a rising edge on the tim_ti2 input, use the following procedure:

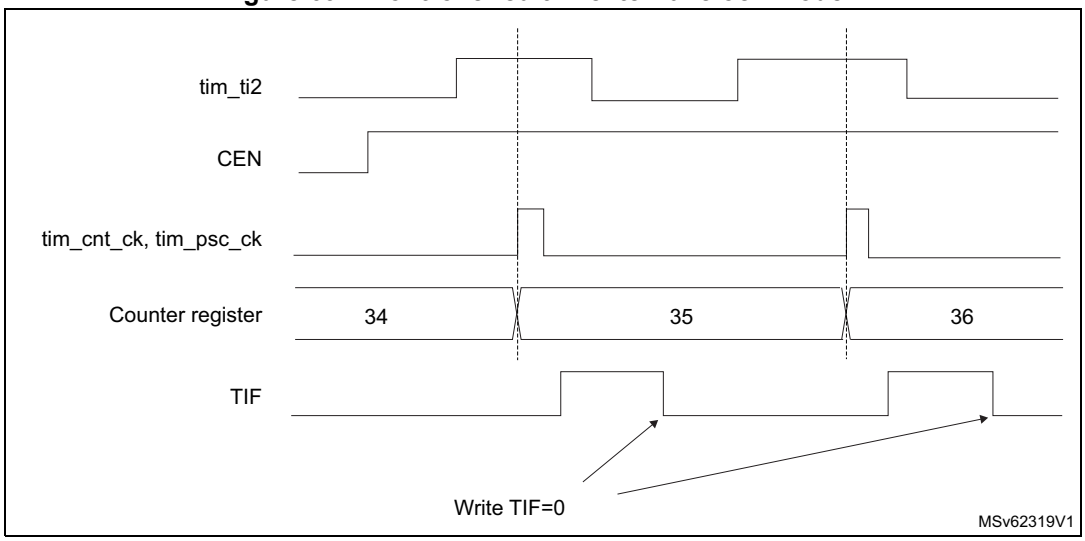
1. Select the proper tim_ti2_in[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Configure channel 2 to detect rising edges on the tim_ti2 input by writing CC2S = 01 in the TIMx_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F = 0000).
4. Select the rising edge polarity by writing CC2P = 0 and CC2NP = 0 in the TIMx_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS = 111 in the TIMx_SMCR register.
6. Select tim_ti2 as the trigger input source by writing TS = 110 in the TIMx_SMCR register.
7. Enable the counter by writing CEN = 1 in the TIMx_CR1 register.

Note: The capture prescaler is not used for triggering, it is not necessary to configure it.

When a rising edge occurs on tim_ti2, the counter counts once and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual clock of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 697. Control circuit in external clock mode 1



56.4.6 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), an input stage for capture (with digital filter, multiplexing and prescaler), and an output stage (with comparator and output control).

Figure 698, *Figure 699*, *Figure 700* and *Figure 701* give an overview of a capture/compare channel.

The input stage samples the corresponding tim_tix input to generate a filtered signal tim_tixf. Then, an edge detector with polarity selection generates a signal (tim_tixfpy) which

can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 698. Capture/compare channel 1 input stage (TIM10/TIM11/TIM13/TIM14)

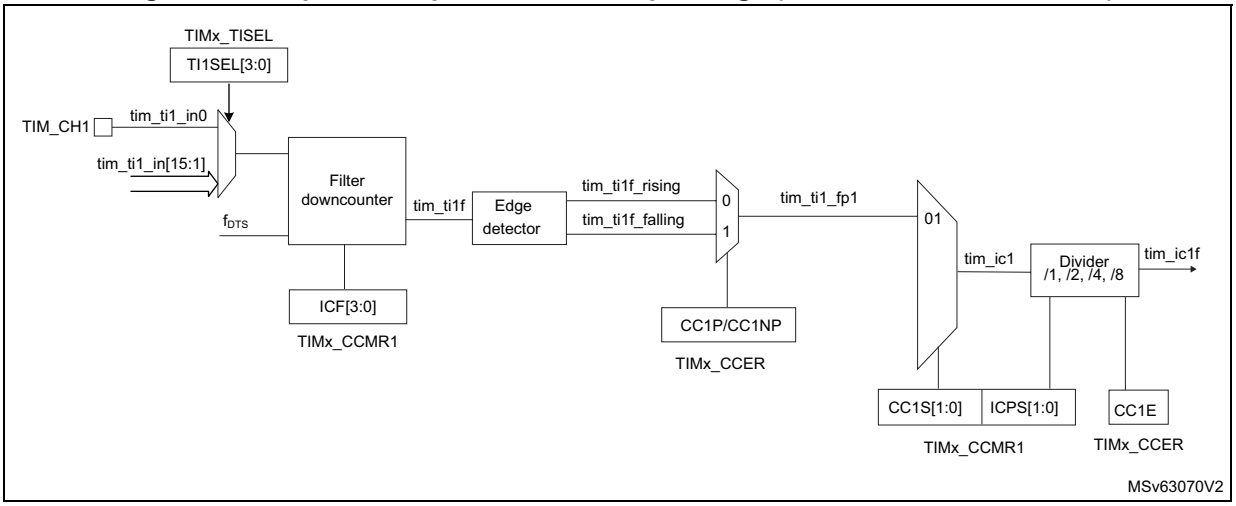
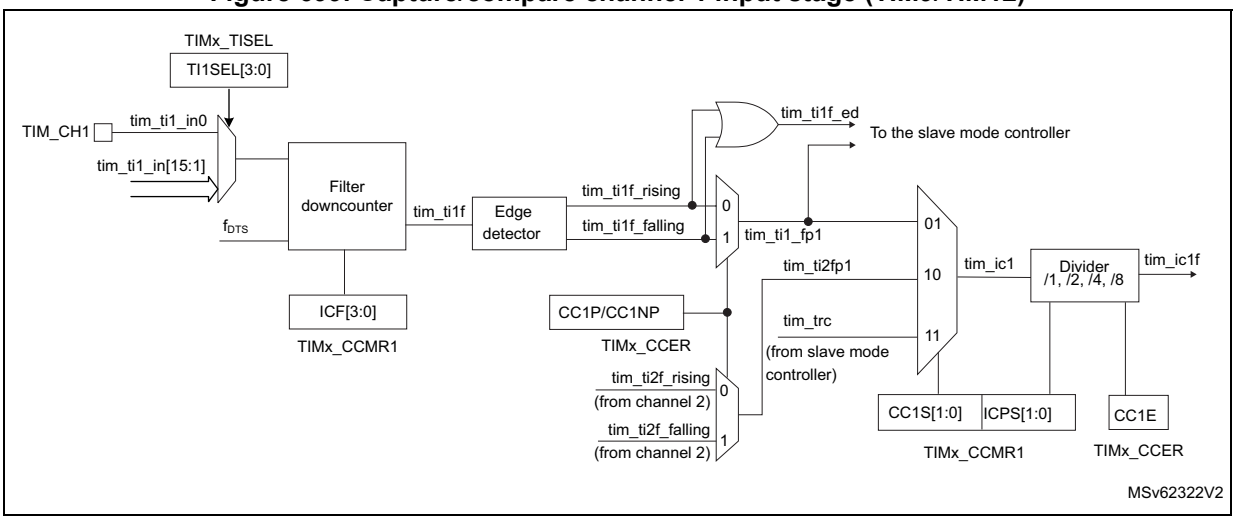


Figure 699. Capture/compare channel 1 input stage (TIM9/TIM12)



The output stage generates an intermediate waveform which is then used for reference: tim_ocxref (active high). The polarity acts at the end of the chain.

Figure 700. Capture/compare channel 1 main circuit

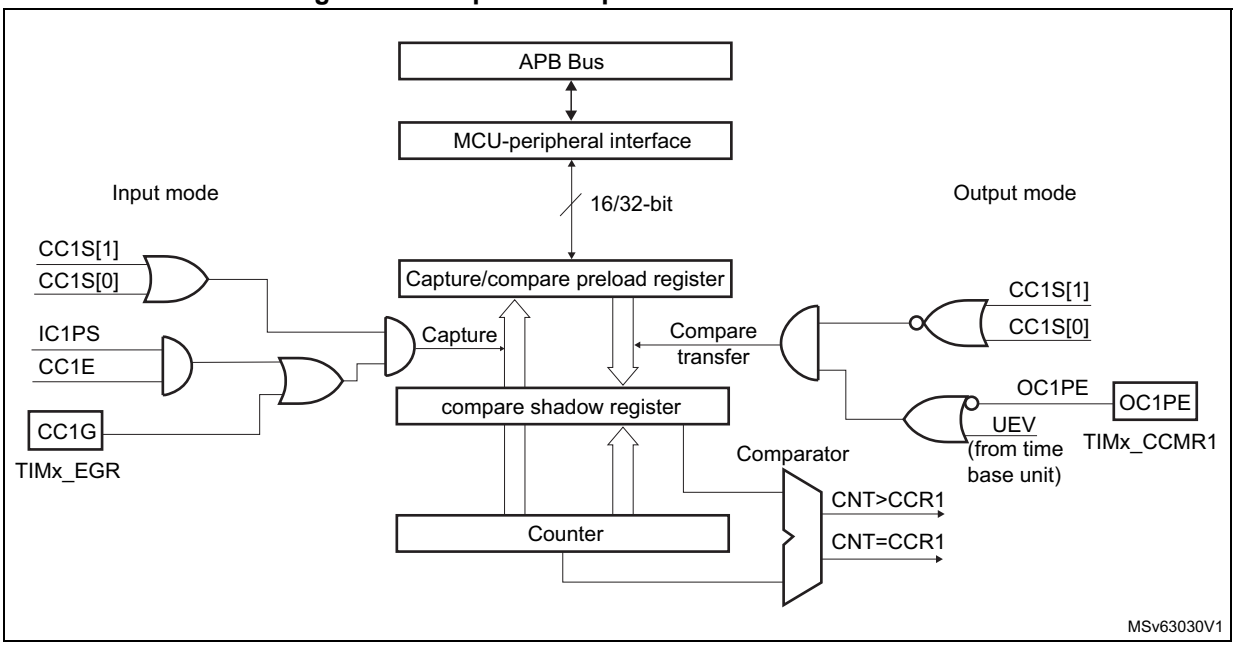
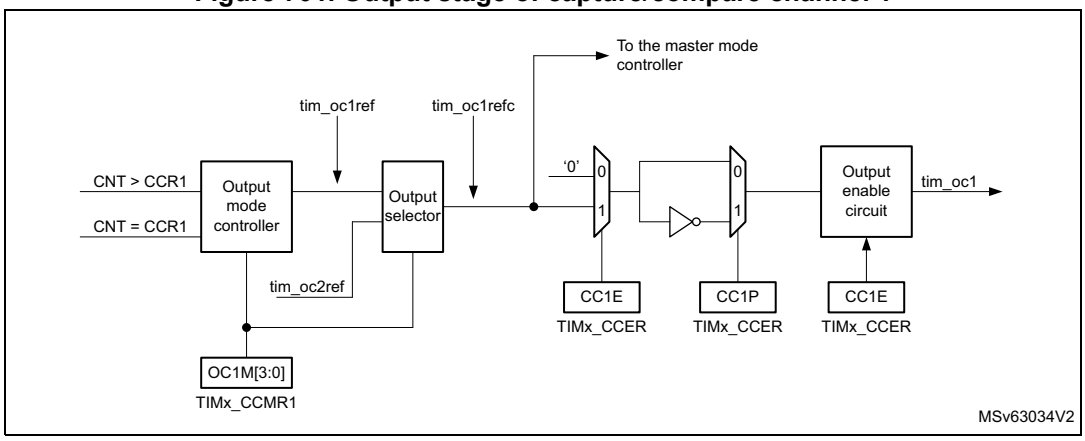


Figure 701. Output stage of capture/compare channel 1



The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

56.4.7 Input capture mode

In Input capture mode, the capture/compare registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding tim_icx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the overcapture flag CCxOF (TIMx_SR register) is set. CCxIF can be

cleared by software by writing it to 0 or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx_CCR1 when tim_ti1 input rises. To do this, use the following procedure:

1. Select the proper tim_ti1_in[15:0] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input: TIMx_CCR1 must be linked to the tim_ti1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input mode, and the TIMx_CCR1 register becomes read-only.
3. Program the appropriate input filter duration in relation with the signal connected to the timer (by programming the ICxF bits in the TIMx_CCMRx register if the input is one of the tim_tix inputs). Let's imagine that, when toggling, the input signal is not stable during at most five internal clock cycles. The user must program a filter duration longer than these five clock cycles. The user can validate a transition on tim_ti1 when eight consecutive samples with the new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.
4. Select the edge of the active transition on the tim_ti1 channel by programming CC1P and CC1NP bits to 00 in the TIMx_CCER register (rising edge in this case).
5. Program the input prescaler. In this example, the user wishes the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIMx_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which may happen after reading the flag and before reading the data.

Note: IC interrupt requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

56.4.8 PWM input mode (TIM9/TIM12 only)

This mode is used to measure both the period and the duty cycle of a PWM signal connected to single tim_tix input:

- The TIMx_CCR1 register holds the period value (interval between two consecutive rising edges).
- The TIM_CCR2 register holds the pulse width (interval between two consecutive rising and falling edges.

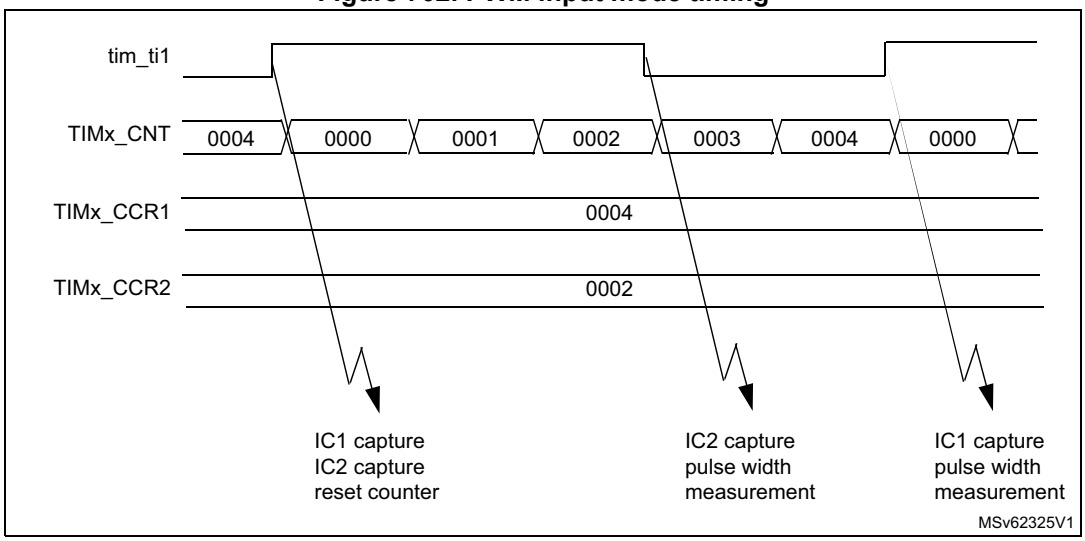
This mode is a particular case of input capture mode. The set-up procedure is similar with the following differences:

- Two tim_icx signals are mapped on the same tim_tix input.
- These two tim_icx signals are active on edges with opposite polarity.
- One of the two tim_tixfpy signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, one can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on tim_ti1 using the following procedure (depending on tim_ker_ck frequency and prescaler value):

1. Select the proper tim_ti1_in[15:0] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (tim_ti1 selected).
3. Select the active polarity for tim_ti1fp1 (used both for capture in TIMx_CCR1 and counter clear): program the CC1P and CC1NP bits to 00 (active on rising edge).
4. Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (tim_ti1 selected).
5. Select the active polarity for tim_ti1fp2 (used for capture in TIMx_CCR2): program the CC2P and CC2NP bits to 10 (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIMx_SMCR register (tim_ti1fp1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to 1 in the TIMx_CCER register.

Figure 702. PWM input mode timing



56.4.9 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (tim_ocxref and then tim_ocx) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (tim_ocxref/tim_ocx) to its active level, one just needs to write 0101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus tim_ocxref is forced high (tim_ocxref is always active high) and tim_ocx get opposite value to CCxP polarity bit.

For example: CCxP = 0 (tim_ocx active high) => tim_ocx is forced to high level.

The tim_ocxref signal can be forced low by writing the OCxM bits to 0100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt requests can be sent accordingly. This is described in the output compare mode section below.

56.4.10 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

1. Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCxM = 0000), be set active (OCxM = 0001), be set inactive (OCxM = 0010) or can toggle (OCxM = 0011) on match.
2. Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
3. Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

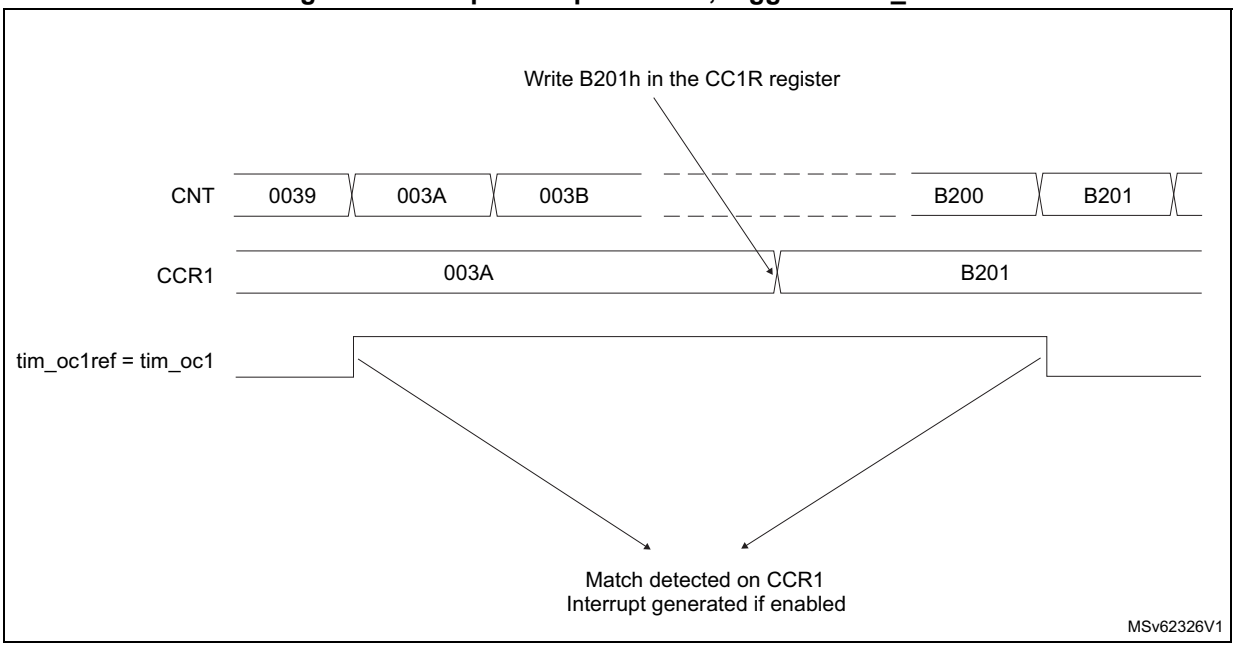
In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure:

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 0011 to toggle tim_ocx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register
   - Write CCxP = 0 to select active high polarity
   - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE = 0, else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 703*.

Figure 703. Output compare mode, toggle on tim_oc1.



56.4.11 PWM mode

Pulse width modulation mode is used to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per tim_ocx output) by writing 0110 (PWM mode 1) or 0111 (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the autoreload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

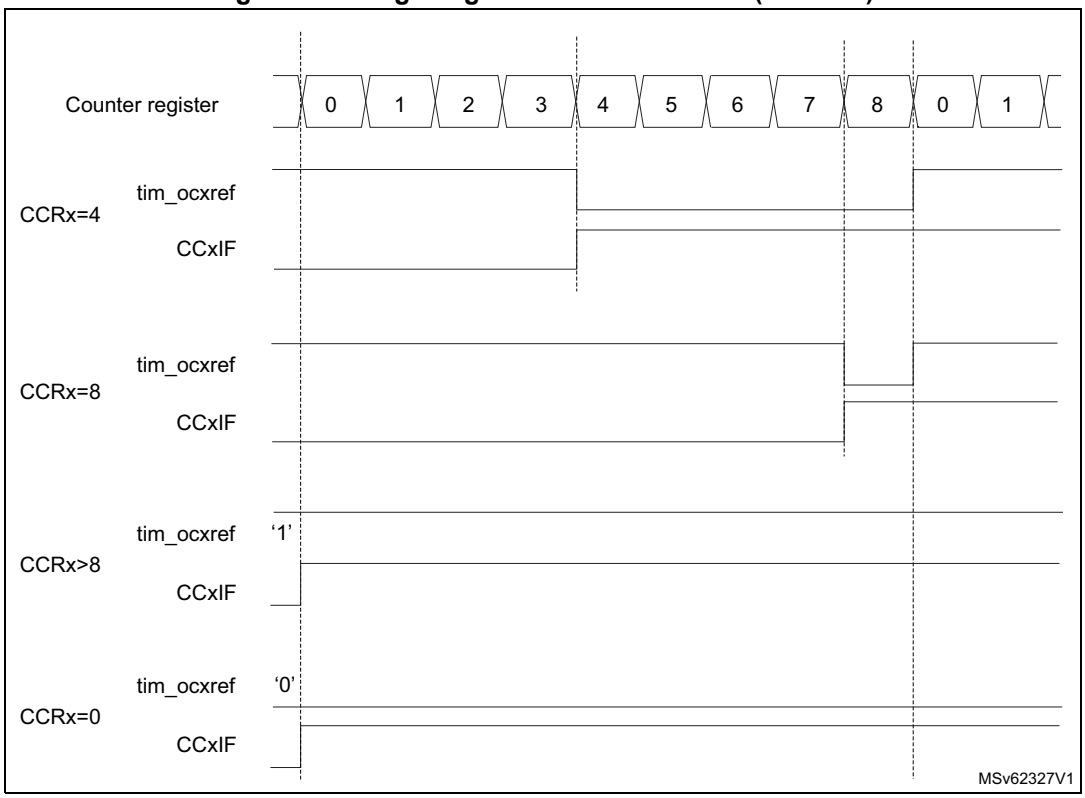
The tim_ocx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. The tim_ocx output is enabled by the CCxE bit in the TIMx_CCER register. Refer to the TIMx_CCERx register description for more details.

In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CNT ≤ TIMx_CCRx.

The timer is able to generate PWM in edge-aligned mode only since the counter is upcounting.

In the following example, the user considers PWM mode 1. The reference PWM signal tim_ocxref is high as long as TIMx_CNT < TIMx_CCRx else it becomes low. If the compare value in TIMx_CCRx is greater than the autoreload value (in TIMx_ARR) then tim_ocxref is held at 1. If the compare value is 0 then tim_ocxref is held at 0. *Figure 704* shows some edge-aligned PWM waveforms in an example where TIMx_ARR = 8.

Figure 704. Edge-aligned PWM waveforms (ARR = 8)

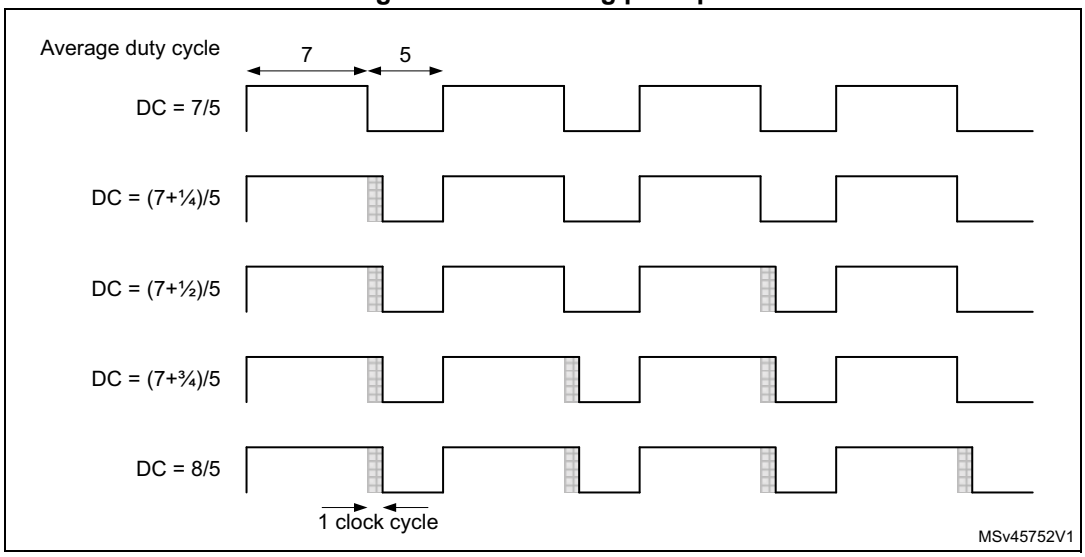


Dithering mode

The PWM mode effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIMx_CR1 register. This applies to both the CCR (for duty cycle resolution increase) and ARR (for PWM frequency resolution increase).

The operating principle is to have the actual CCR (or ARR) value slightly changed (adding or not one timer clock period) over 16 consecutive PWM periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average duty cycle or PWM period. *Figure 705* presents the dithering principle applied to four consecutive PWM cycles.

Figure 705. Dithering principle

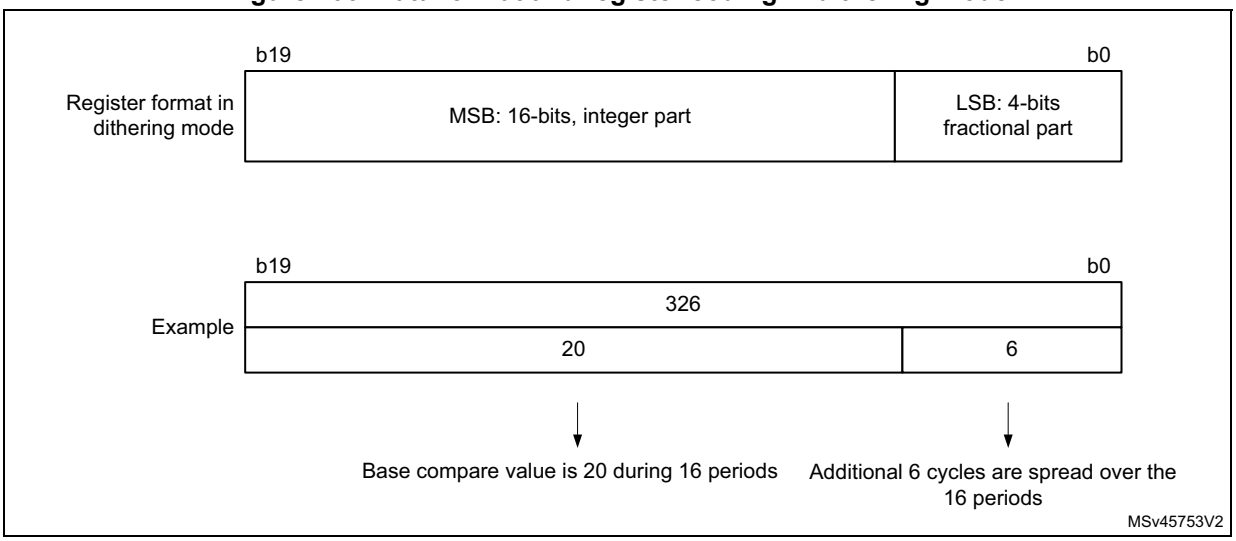


When the dithering mode is enabled, the register coding is changed as follows (see *Figure 706* for example):

- The four LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted to the bits 19:4 and are coding for the base value.

Note: The following sequence must be followed when resetting the DITHEN bit:
1. CEN and ARPE bits must be reset
2. The DITHEN bit must be reset
3. The CCIF flags must be cleared
4. The CEN bit can be set (eventually with ARPE = 1).

Figure 706. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

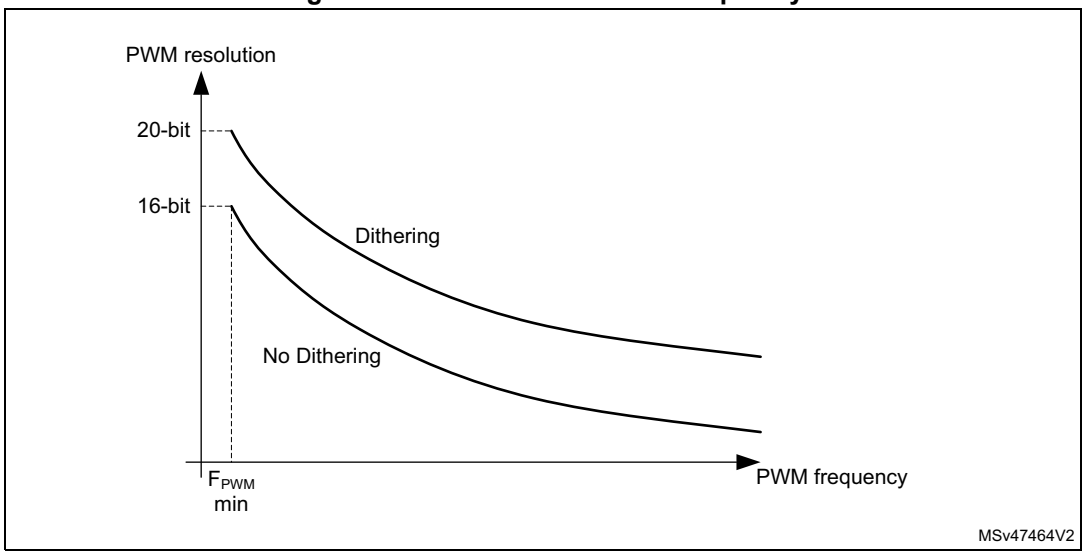
$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode enabled: } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

Note: The maximum TIMx_ARR and TIMxCCRy values are limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).

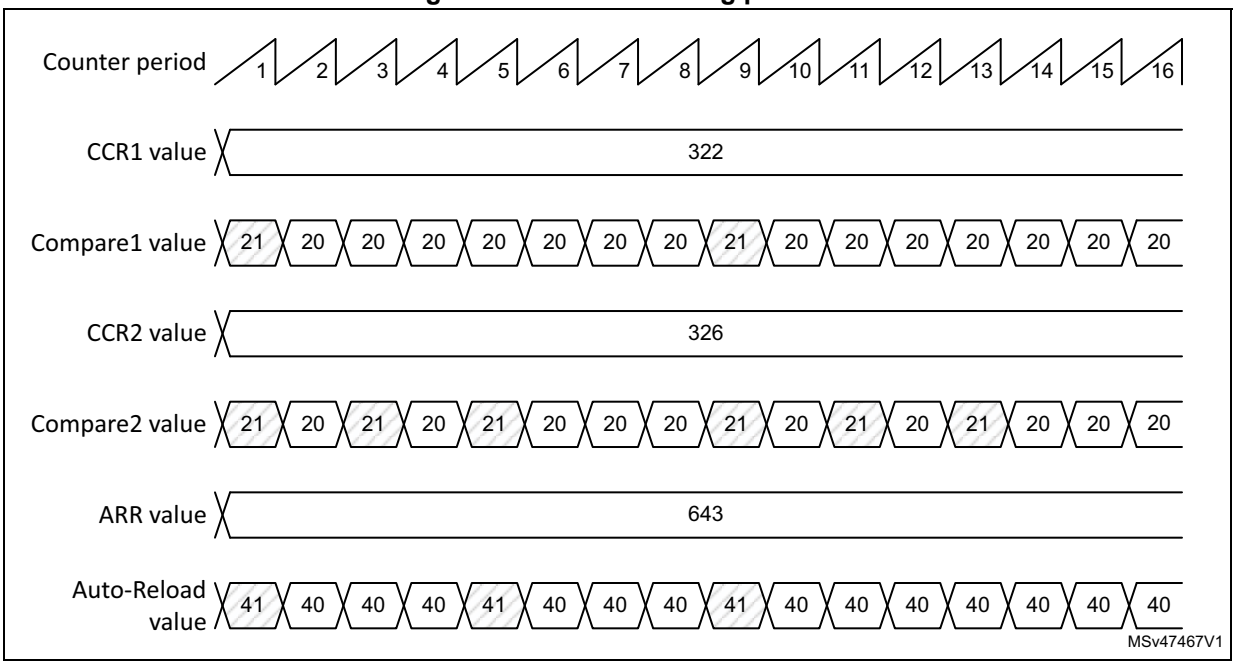
As shown on *Figure 707*, the dithering mode is used to increase the PWM resolution whatever the PWM frequency.

Figure 707. PWM resolution vs frequency



The duty cycle and/or period changes are spread over 16 consecutive periods, as described in *Figure 708*.

Figure 708. PWM dithering pattern



The autoreload and compare values increments are spread following specific patterns described in *Table 531*. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 531. CCR and ARR register change dithering pattern

-	PWM period															
LSB value	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-

Table 531. CCR and ARR register change dithering pattern (continued)

-	PWM period															
LSB value	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

56.4.12 Combined PWM mode (TIM9/TIM12 only)

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, tim_ocxrefc, are made of an OR or AND logical combination of two reference PWMs:

- tim_oc1refc (or tim_oc2refc) is controlled by the TIMx_CCR1 and TIMx_CCR2 registers.

Combined PWM mode can be selected independently on two channels (one tim_ocx output per pair of CCR registers) by writing 1100 (Combined PWM mode 1) or 1101 (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

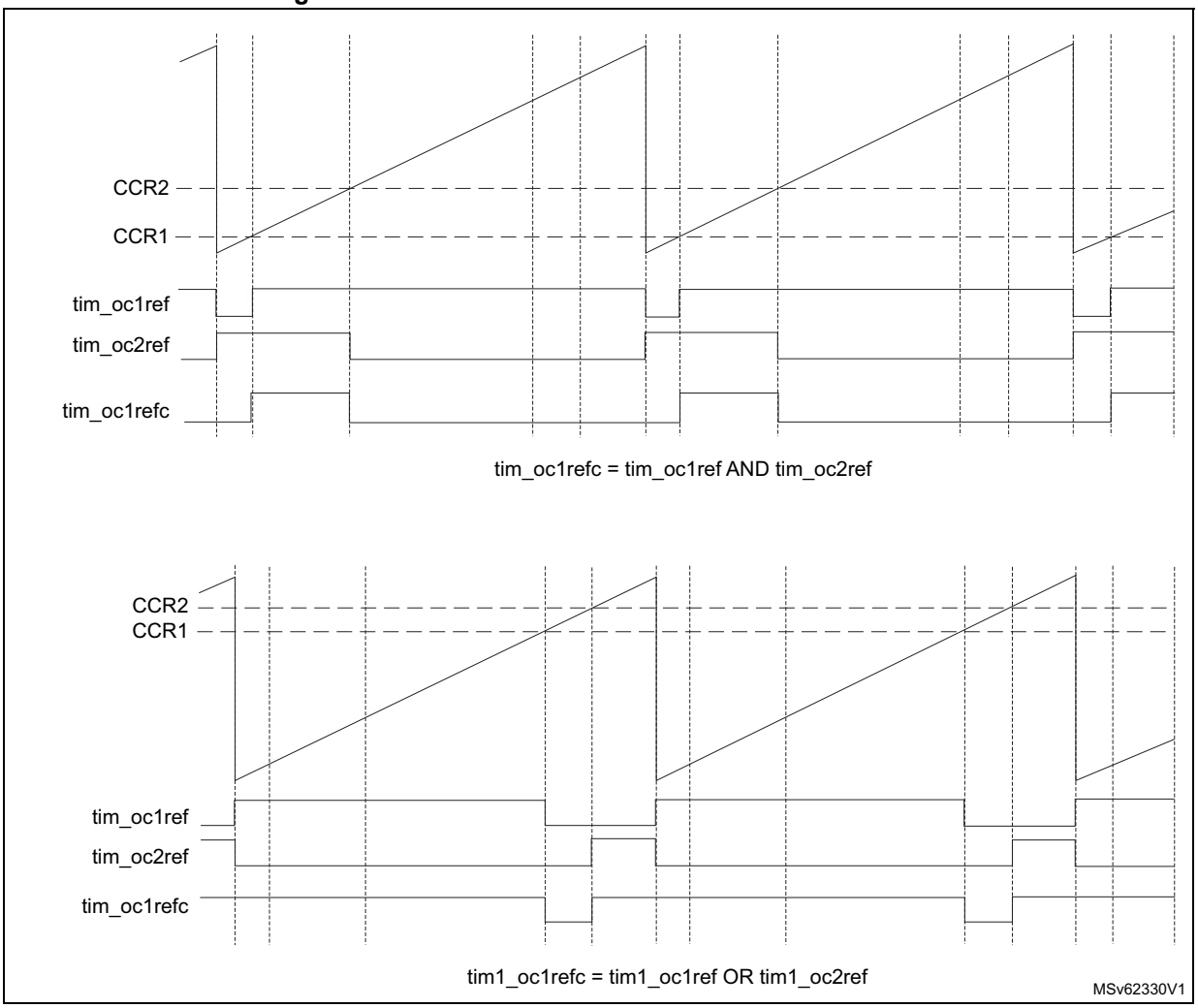
When a given channel is used as a combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in combined PWM mode 2).

Note: The OCxM[3:0] bitfield is split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

Figure 709 represents an example of signals that can be generated using combined PWM mode, obtained with the following configuration:

- Channel 1 is configured in combined PWM mode 2.
- Channel 2 is configured in PWM mode 1.

Figure 709. Combined PWM mode on channel 1 and 2



56.4.13 One-pulse mode

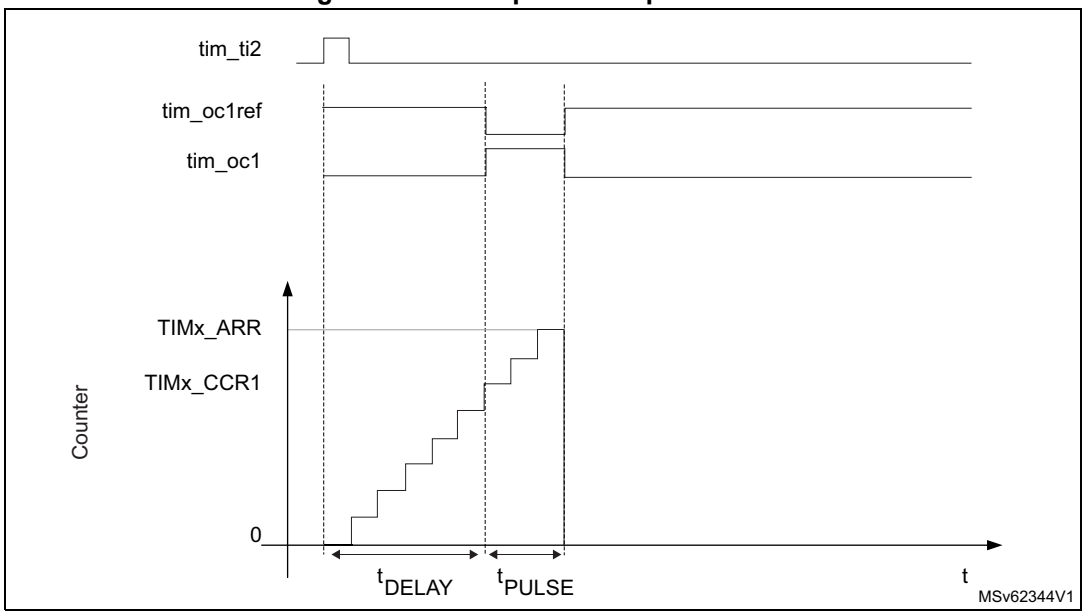
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be as follows:

CNT < CCRx ≤ ARR (in particular, 0 < CCRx)

Figure 710. Example of one pulse mode



For example one may want to generate a positive pulse on tim_oc1 with a length of tPULSE and after a delay of tDELAY as soon as a positive edge is detected on the tim_ti2 input pin.

Use tim_ti2fp2 as trigger 1:

1. Select the proper tim_ti2_in[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Map tim_ti2fp2 to tim_ti2 by writing CC2S = 01 in the TIMx_CCMR1 register.
3. tim_ti2fp2 must detect a rising edge, write CC2P = 0 and CC2NP = 0 in the TIMx_CCER register.
4. Configure tim_ti2fp2 as trigger for the slave mode controller (tim_trgi) by writing TS = 00110 in the TIMx_SMCR register.
5. tim_ti2fp2 is used to start the counter by writing SMS to 110 in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the autoreload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Assuming the user wants to build a waveform with a transition from 0 to 1 when a compare match occurs and a transition from 1 to 0 when the counter reaches the autoreload value. To do this PWM mode 2 must be enabled by writing OC1M = 0111 in the TIMx_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE = 1 in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the autoreload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on tim_ti2. CC1P is written to 0 in this example.

Since only one pulse (Single mode) is needed, a 1 must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over

from the autoreload value back to 0). When OPM bit in the TIMx_CR1 register is set to 0, so the Repetitive mode is selected.

Particular case: tim_ocx fast enable

In One-pulse mode, the edge detection on tim_tix input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min that can be obtained.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then tim_ocxref (and tim_ocx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

56.4.14 Retriggerable one pulse mode (TIM9/TIM12 only)

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with nonretriggerable one pulse mode described in *Section 56.4.13: One-pulse mode*:

- The pulse starts as soon as the trigger occurs (no programmable delay).
- The pulse is extended if a new trigger occurs before the previous one is completed.

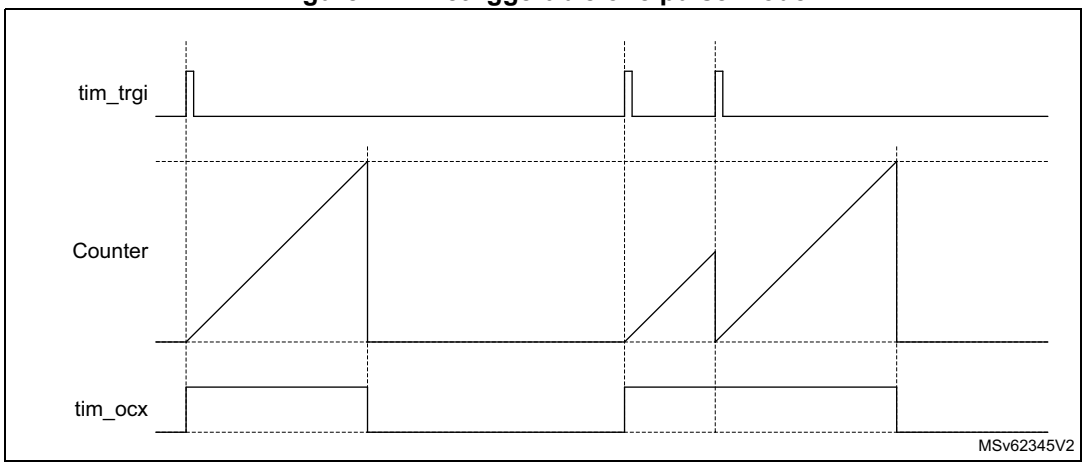
The timer must be in Slave mode, with the bits SMS[3:0] = 1000 (Combined reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to 1000 or 1001 for retriggerable OPM mode 1 or 2.

If the timer is configured in up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in down-counting mode, CCRx must be above or equal to ARR.

Note: The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the three least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.

Figure 711. Retriggerable one pulse mode



56.4.15 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag UIF into bit 31 of the timer counter register (TIMxCNT[31]). This is used to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

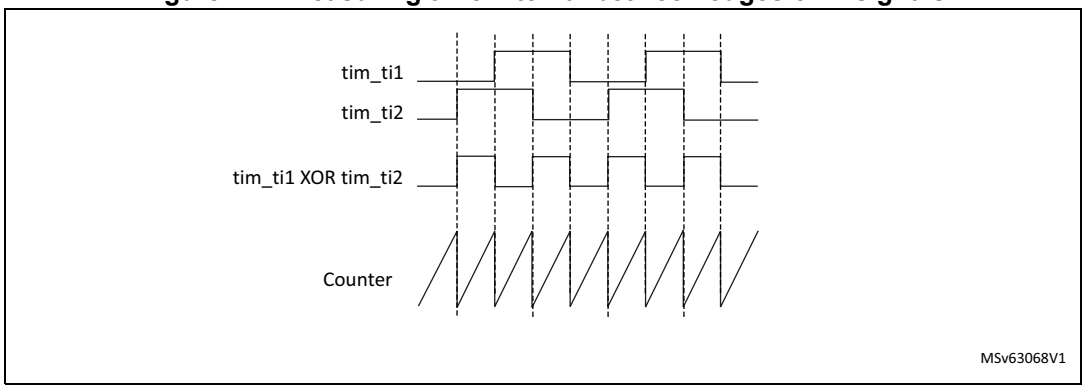
There is no latency between the assertions of the UIF and UIFCPY flags.

56.4.16 Timer input XOR function

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of an XOR gate, combining the two input pins tim_ti1 and tim_ti2.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is useful for measuring the interval between the edges on two input signals, as shown in *Figure 712*.

Figure 712. Measuring time interval between edges on 2 signals



56.4.17 TIM9/TIM12 external trigger synchronization

The TIM9/TIM12 timers can be synchronized with an external trigger in several modes: Reset mode, Gated mode, Trigger mode, Reset + trigger, and Gated + reset mode.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

In the following example, the upcounter is cleared in response to a rising edge on tim_ti1 input:

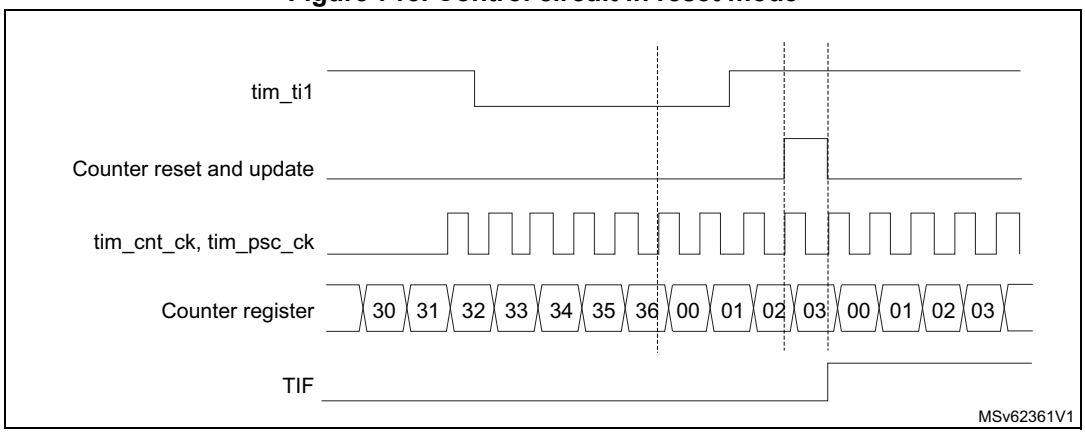
1. Configure the channel 1 to detect rising edges on tim_ti1. Configure the input filter duration (in this example, the user does not need any filter, so the user keeps IC1F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Program CC1P and CC1NP to 00 in TIMx_CCER register to validate the polarity (and detect rising edges only).

2. Configure the timer in reset mode by writing SMS = 100 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIMx_SMCR register.
3. Start the counter by writing CEN = 1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until tim_ti1 rising edge. When tim_ti1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request can be sent if enabled (depending on the TIE bit in TIMx_DIER register).

The following figure shows this behavior when the autoreload register TIMx_ARR = 0x36. The delay between the rising edge on tim_ti1 and the actual reset of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 713. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

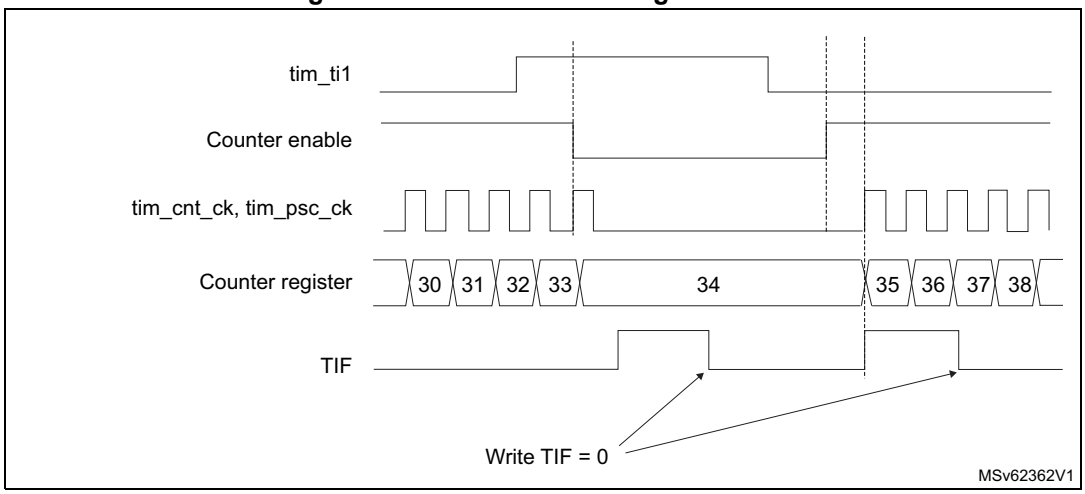
In the following example, the upcounter counts only when tim_ti1 input is low:

1. Configure the channel 1 to detect low levels on tim_ti1. Configure the input filter duration (in this example, the user does not need any filter, so the user keeps IC1F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in TIMx_CCMR1 register. Program CC1P = 1 and CC1NP = 0 in TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in gated mode by writing SMS = 101 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIMx_SMCR register.
3. Enable the counter by writing CEN = 1 in the TIMx_CR1 register (in gated mode, the counter does not start if CEN = 0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as tim_ti1 is low and stops as soon as tim_ti1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on tim_ti1 and the actual stop of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 714. Control circuit in gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

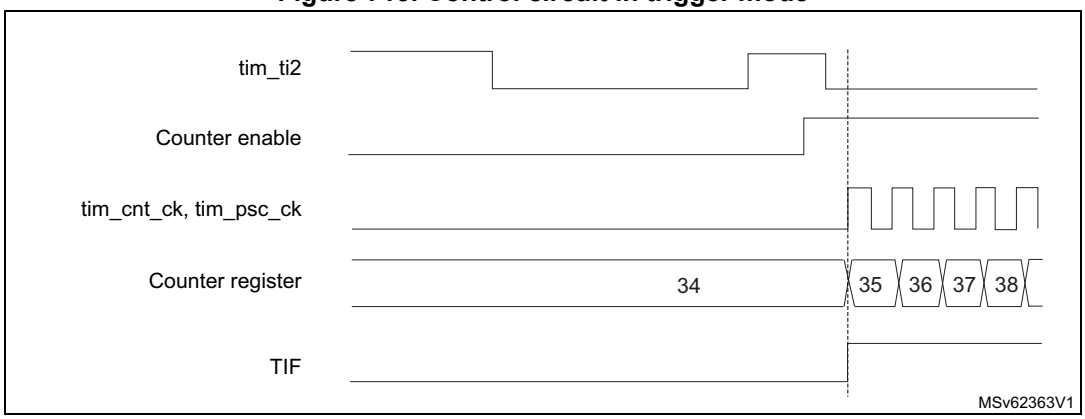
In the following example, the upcounter starts in response to a rising edge on tim_ti2 input:

1. Configure the channel 2 to detect rising edges on tim_ti2. Configure the input filter duration (in this example, the user does not need any filter, so the user keeps IC2F = 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC2S bits are configured to select the input capture source only, CC2S = 01 in TIMx_CCMR1 register. Program CC2P = 1 and CC2NP = 0 in TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in trigger mode by writing SMS = 110 in TIMx_SMCR register. Select tim_ti2 as the input source by writing TS = 00110 in TIMx_SMCR register.

When a rising edge occurs on tim_ti2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual start of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 715. Control circuit in trigger mode



56.4.18 Slave mode – combined reset + trigger mode

In this case, a rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

56.4.19 Slave mode – combined reset + gated mode

The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

56.4.20 Timer synchronization (TIM9/TIM12 only)

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to *Section 54.4.23: Timer synchronization* for details.

Note: The clock of the slave timer must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

56.4.21 Using timer output as trigger for other timers (TIM10/TIM11/TIM13/TIM14 only)

The timers with one channel only do not feature a master mode. However, the OC1 output signal can be used to trigger some other timers (including timers described in other sections of this document). Check the “TIMx internal trigger connection” table of any timer on the device to identify which timers can be targeted as slave.

The OC1 signal pulse width must be programmed to be at least two clock cycles of the destination timer, to make sure the slave timer detects the trigger.

For instance, if the destination's timer CK_INT clock is four times slower than the source timer, the OC1 pulse width must be eight clock cycles.

56.4.22 ADC triggers (TIM9/TIM12 only)

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events.

Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

56.4.23 ADC synchronization (TIM9/TIM12 only)

The timer operation can be synchronized to the ADC clock to trigger jitter-free ADC sampling. This function is enabled using the ADSYNC bit in the TIMx_CR2 register.

This feature is useful when the timers and the ADCs are operating with semisynchronous clocks (clocks derived from a same source with integer ratio tim_ker_ck/adc_ker_ck), for instance adc_ker_ck = 75 MHz and tim_ker_ck = 150 MHz or 300 MHz.

ADSYNC must also be set when both peripherals are operating at the same frequency from the same clock source, when jitter-free operation is needed.

ADSYNC must not be set and jitter-free operation is not supported in the following cases:

- When the clock ratio is not an integer (for example adc_ker_ck = 75 MHz and tim_ker_ck = 100 MHz): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum.
- When the ADC is operating in asynchronous mode (adc_ker_ck uncorrelated with tim_ker_ck): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum.

When ADSYNC = 1, the timer operation is slightly changed: the counter enable and counter reset events are aligned to the adc_ker_ck ADC clock, to avoid any phase shift due to clocks enable in the RCC.

Jitter-free operation is guaranteed only when one of the two requirements below is met (depending on the selected trigger source):

1. The counter period must be a multiple of the ADC clock period

$$(\text{TIMx_PSC} + 1) \times (\text{TIMx_ARR} + 1) \times T_{\text{tim_ker_ck}} = n \times T_{\text{adc_ker_ck}}$$

2. The compare value must be a multiple of the ADC clock period

$$(\text{TIMx_PSC} + 1) \times \text{TIMx_CMPy} \times T_{\text{tim_ker_ck}} = m \times T_{\text{adc_ker_ck}}$$

Note: If none of the two above requirements is met, the trigger is still generated, but the latency is not constant and varies with the timer and ADC clocks phase shift.

Programming guidelines

The ADC synchronization feature must not be modified during run-time, once the counter is enabled and once the ADC has been configured for receiving triggers from the timer.

It is mandatory to follow the procedure below to use the ADC synchronization:

1. Enable the destination ADC clock
2. Configure the timer and set the ADSYNC bit
3. Configure the ADC and enable it (using ADSTART and/or JADSTART bits)
4. Start the timer (with the CEN counter enable bit)

56.4.24 Debug mode

When the microcontroller enters debug mode (Cortex®-M55 core halted), the TIMx counter can either continue to work normally or stop.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For more details, refer to the Debug section.

56.5 TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 low-power modes

Table 532. Effect of low-power modes on TIM9/TIM10/TIM11/TIM12/TIM13/TIM14

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Standby	The timer is powered-down and must be reinitialized after exiting the Standby mode.

56.6 TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 interrupts

The TIM9/TIM10/TIM11/TIM12/TIM13/TIM14 can generate multiple interrupts, as shown in *Table 533*.

Table 533. Interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
TIM	Update	UIF	UIE	write 0 in UIF	Yes	No
	Capture/compare 1	CC1IF	CC1IE	write 0 in CC1IF	Yes	No
TIM	Capture/compare 2(1)	CC2IF	CC2IE	write 0 in CC2IF	Yes	No
	Trigger(1)	TIF	TIE	write 0 in TIF	Yes	No

1. Available for TIM9/TIM12 only.

56.7 TIM9/TIM12 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers have to be written by half-words (16 bits) or words (32 bits). Read accesses can be done by bytes (8 bits), half-words (16 bits), or words (32 bits).

56.7.1 TIMx control register 1 (TIMx_CR1)(x = 9, 12)

Address offset: 0x000

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw				rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable

0: Dithering disabled

1: Dithering enabled

Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping

0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.

1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division

This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and sampling clock used by the digital filters (tim_tix),

00: $t_{DTS} = t_{tim_ker_ck}$

01: $t_{DTS} = 2 \times t_{tim_ker_ck}$

10: $t_{DTS} = 4 \times t_{tim_ker_ck}$

11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable

0: TIMx_ARR register is not buffered.

1: TIMx_ARR register is buffered.

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode

0: Counter is not stopped on the update event

1: Counter stops counting on the next update event (clearing the CEN bit).

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the UEV event sources.

0: Any of the following events generates an update interrupt if enabled. These events can be:

– Counter overflow

– Setting the UG bit

– Update generation through the slave mode controller

1: Only counter overflow generates an update interrupt if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable update event (UEV) generation.

0: UEV enabled. An UEV is generated by one of the following events:

– Counter overflow

– Setting the UG bit

Buffered registers are then loaded with their preload values.

1: UEV disabled. No UEV is generated, shadow registers keep their value (ARR, PSC, CCRx). The counter and the prescaler are reinitialized if the UG bit is set.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

CEN is cleared automatically in one-pulse mode, when an update event occurs.

Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

56.7.2 TIM12 control register 2 (TIMx_CR2)(x = 9, 12)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	AD SYNC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			rw												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			Res.	Res.	Res.	Res.
								rw	rw	rw	rw				

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **ADSYNC**: ADC synchronization

0: The timer operates independently from the ADC

1: The timer operation is synchronized with the ADC clock to provide jitter-free sampling point. This mode can be enabled only with specific ADC / timer clock relationship. Refer to *Section 56.4.23* for requirements.

The ADSYNC must not modified when the counter is enabled (CEN bit is set).

Bits 27:8 Reserved, must be kept at reset value.

Bit 7 **TI1S**: tim_ti1 selection

0: The tim_ti1_in[15:0] multiplexer output is connected to tim_ti1 input

1: The tim_ti1_in[15:0] and tim_ti2_in[15:0] multiplexers output are connected to the tim_ti1 input (XOR combination)

Bits 6:4 **MMS[2:0]**: Master mode selection

These bits allow to select the information to be sent in master mode to slave timers for synchronization (tim_trgo). The combination is as follows:

000: Reset - the UG bit from the TIMx_EGR register is used as trigger output (tim_trgo). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on tim_trgo is delayed compared to the actual reset.

001: Enable - the Counter Enable signal CNT_EN is used as trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo, except if the master/slave mode is selected (see the MSM bit description in TIMx_SMCR register).

010: Update - The update event is selected as trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

011: Compare Pulse - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred (tim_trgo).

100: Compare - tim_oc1refc signal is used as trigger output (tim_trgo).

101: Compare - tim_oc2refc signal is used as trigger output (tim_trgo).

Bits 3:0 Reserved, must be kept at reset value.

56.7.3 TIMx slave mode control register (TIMx_SMCR)(x = 9, 12)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS[4:3]		Res.	Res.	Res.	SMS[3]
										rw	rw				rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSM	TS[2:0]			Res.	SMS[2:0]		
								rw	rw	rw	rw		rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 19:17 Reserved, must be kept at reset value.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **MSM:** Master/Slave mode

0: No action

1: The effect of an event on the trigger input (tim_trgi) is delayed to allow a perfect synchronization between the current timer and its slaves (through tim_trgo). It is useful in order to synchronize several timers on a single external event.

Bits 21, 20, 6, 5, 4 **TS[4:0]:** Trigger selection

This TS[4:0] bitfield selects the trigger input to be used to synchronize the counter.

00000: Internal Trigger 0 (tim_itr0)
00001: Internal Trigger 1 (tim_itr1)
00010: Internal Trigger 2 (tim_itr2)
00011: Internal Trigger 3 (tim_itr3)
00100: tim_ti1 Edge Detector (tim_ti1f_ed)
00101: Filtered Timer Input 1 (tim_ti1fp1)
00110: Filtered Timer Input 2 (tim_ti2fp2)
01000: Internal Trigger 4 (tim_itr4)
01001: Internal Trigger 5 (tim_itr5)
01010: Internal Trigger 6 (tim_itr6)
01011: Internal Trigger 7 (tim_itr7)
01100: Internal Trigger 8 (tim_itr8)
01101: Internal Trigger 9 (tim_itr9)
01110: Internal Trigger 10 (tim_itr10)
01111: Internal Trigger 10 (tim_itr11)
10000: Internal Trigger 10 (tim_itr12)
10001: Internal Trigger 10 (tim_itr13)
10010: Internal Trigger 10 (tim_itr14)
10011: Internal Trigger 10 (tim_itr15)
Others: Reserved

See for more details on the meaning of tim_itrx for each timer.

Note: These bits must be changed only when they are not used (for example when SMS = 000) to avoid wrong edge detections at the transition.

Bit 3 Reserved, must be kept at reset value.

Bits 16, 2, 1, 0 **SMS[3:0]:** Slave mode selection

When external signals are selected the active edge of the trigger signal (tim_trgi) is linked to the polarity selected on the external input (see input control register and control register description.

0000: Slave mode disabled - if CEN = 1 then the prescaler is clocked directly by the internal clock.

0001: Reserved

0010: Reserved

0011: Reserved

0100: Reset mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter and generates an update of the registers.

0101: Gated mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

0110: Trigger mode - The counter starts at a rising edge of the trigger tim_trgi (but it is not reset). Only the start of the counter is controlled.

0111: External clock mode 1 - Rising edges of the selected trigger (tim_trgi) clock the counter.

1000: Combined reset + trigger mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers and starts the counter.

1001: Combined gated + reset mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

Other codes: reserved.

Note: The gated mode (including gated + reset mode) must not be used if tim_ti1f_ed is selected as the trigger input (TS = 00100). Indeed, tim_ti1f_ed outputs 1 pulse for each transition on tim_ti1f, whereas the gated mode checks the level of the trigger signal.

The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signals must be enabled prior to receive events from the master timer, and must not be changed on-the-fly while triggers are received from the master timer.

56.7.4 TIMx interrupt enable register (TIMx_DIER)(x = 9, 12)

Address offset: 0x00C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIE	Res.	Res.	Res.	CC2IE	CC1IE	UIE
									rw				rw	rw	rw

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **TIE**: Trigger interrupt enable

0: Trigger interrupt disabled.

1: Trigger interrupt enabled.

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable
0: CC2 interrupt disabled.
1: CC2 interrupt enabled.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled.
1: CC1 interrupt enabled.

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled.
1: Update interrupt enabled.

56.7.5 TIMx status register (TIMx_SR)(x = 9, 12)

Address offset: 0x010

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CC2OF	CC1OF	Res.	Res.	TIF	Res.	Res.	Res.	CC2IF	CC1IF	UIF
					rc_w0	rc_w0			rc_w0				rc_w0	rc_w0	rc_w0

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **CC2OF**: Capture/compare 2 overcapture flag
refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.
0: No overcapture has been detected.
1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **TIF**: Trigger interrupt flag
This flag is set by hardware on the TRG trigger event (active edge detected on tim_trgi input when the slave mode controller is enabled in all modes but gated mode. It is set when the counter starts or stops when gated mode is selected. It is cleared by software.
0: No trigger event occurred.
1: Trigger interrupt pending.

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag
refer to CC1IF description

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag
This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).
0: No compare match / No input capture occurred
1: A compare match or an input capture occurred.
If channel CC1 is configured as output: this flag is set when he content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.
If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on tim_ic1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0 **UIF**: Update interrupt flag
This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
– At overflow and if UDIS = 0 in the TIMx_CR1 register.
– When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS = 0 and UDIS = 0 in the TIMx_CR1 register.
– When CNT is reinitialized by a trigger event (refer to *Section 56.7.3: TIMx slave mode control register (TIMx_SMCR)(x = 9, 12)*), if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

56.7.6 TIMx event generation register (TIMx_EGR)(x = 9, 12)

Address offset: 0x014

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	Res.	Res.	CC2G	CC1G	UG
									w				w	w	w

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **TG**: Trigger generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: The TIF flag is set in the TIMx_SR register. Related interrupt can occur if enabled

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **CC2G**: Capture/compare 2 generation
refer to CC1G description

Bit 1 **CC1G**: Capture/compare 1 generation
This bit is set by software to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
the CC1IF flag is set, the corresponding interrupt is sent if enabled.
If channel CC1 is configured as input:
The current counter value is captured in the TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Re-initializes the counter and generates an update of the registers. The prescaler counter is also cleared and the prescaler ratio is not affected. The counter is cleared.

56.7.7 TIMx capture/compare mode register 1 (TIMx_CCMR1)(x = 9, 12)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits in this register have different functions in input and output modes.

Input capture mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S[1:0]**: Capture/compare 2 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, IC2 is mapped on tim_ti2
10: CC2 channel is configured as input, IC2 is mapped on tim_ti1
11: CC2 channel is configured as input, IC2 is mapped on tim_trc. This mode works only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)
Note: The CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bitfield defines the frequency used to sample the tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on the CC1 input (tim_ic1).
The prescaler is reset as soon as CC1E = 0 (TIMx_CCER register).
00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1
10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2
11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIMx_SMCR register)

Note: The CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

56.7.8 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 9, 12)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits in this register have different functions in input and output modes.

Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M [3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	OC2M[2:0]			OC2PE	OC2FE	CC2S[1:0]		Res.	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bits 24, 14:12 **OC2M[3:0]**: Output compare 2 mode
Refer to OC1M[3:0] for bit description.

Bit 11 **OC2PE**: Output compare 2 preload enable

Bit 10 **OC2FE**: Output compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection
This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC2 channel is configured as output
01: CC2 channel is configured as input, IC2 is mapped on tim_ti2
10: CC2 channel is configured as input, IC2 is mapped on tim_ti1
11: CC2 channel is configured as input, IC2 is mapped on tim_trc. This mode works only if an internal trigger input is selected through the TS bit (TIMx_SMCR register
Note: The CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIMx_CCER).

Bit 7 Reserved, must be kept at reset value.

Bits 16, 6:4 **OC1M[3:0]**: Output compare 1 mode (refer to bit 16 for OC1M[3])

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 is derived. tim_oc1ref is active high whereas the active level of tim_oc1 depends on the CC1P.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the ouput keeps the state (active or inactive) it had before entering the frozen state.

0001: Set channel 1 to active level on match. The tim_oc1ref signal is forced high when the TIMx_CNT counter matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. The tim_oc1ref signal is forced low when the TIMx_CNT counter matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - tim_oc1ref toggles when TIMx_CNT = TIMx_CCR1

0100: Force inactive level - tim_oc1ref is forced low

0101: Force active level - tim_oc1ref is forced high

0110: PWM mode 1 - channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else it is inactive

0111: PWM mode 2 - channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else it is active

1000: Retrigerrable OPM mode 1 - The channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1001: Retrigerrable OPM mode 2 - The channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update.

1010: Reserved,

1011: Reserved,

1100: Combined PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc is the logical OR between tim_oc1ref and tim_oc2ref.

1101: Combined PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc is the logical AND between tim_oc1ref and tim_oc2ref.

1110: Reserved,

1111: Reserved

Note: In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken into account immediately

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded into the active register at each update event

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit is used to accelerate the effect of an event on the trigger in input on the CC output.

0: CC1 behaves normally depending on the counter and CCR1 values even when the trigger is ON. The minimum delay to activate the CC1 output when an edge occurs on the trigger input is 5 clock cycles

1: An active edge on the trigger input acts like a compare match on the CC1 output. Then, OC is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00:CC1 channel is configured as output

01:CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1

10:CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2

11:CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode works only if an internal trigger input is selected through the TS bit (TIMx_SMCR register)

Note: The CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

56.7.9 TIMx capture/compare enable register (TIMx_CCER)(x = 9, 12)

Address offset: 0x020

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
								rw		rw	rw	rw		rw	rw

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **CC2NP**: Capture/Compare 2 output Polarity

Refer to CC1NP description

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CC2P**: Capture/Compare 2 output Polarity

Refer to CC1P description

Bit 4 **CC2E**: Capture/Compare 2 output enable

Refer to CC1E description

Bit 3 **CC1NP**: Capture/Compare 1 complementary output Polarity

CC1 channel configured as output: CC1NP must be kept cleared

CC1 channel configured as input: CC1NP is used in conjunction with CC1P to define tim_ti1fp1/tim_ti2fp1 polarity (refer to CC1P description).

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CC1P**: Capture/Compare 1 output Polarity.

0: tim_oc1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: tim_oc1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of tim_ti1fp1 and tim_ti2fp1 for trigger or capture operations.

CC1NP = 0, CC1P = 0: non-inverted/rising edge. The circuit is sensitive to tim_tixfp1 rising edge (capture or trigger operations in reset, external clock or trigger mode), tim_tixfp1 is not inverted (trigger operation in gated mode).

CC1NP = 0, CC1P = 1: inverted/falling edge. The circuit is sensitive to tim_tixfp1 falling edge (capture or trigger operations in reset, external clock or trigger mode), tim_tixfp1 is inverted (trigger operation in gated mode).

CC1NP = 1, CC1P = 1: non-inverted/both edges/ The circuit is sensitive to both tim_tixfp1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), tim_tixfp1 is not inverted (trigger operation in gated mode).

CC1NP = 1, CC1P = 0: This configuration is reserved, it must not be used.

Bit 0 **CC1E**: Capture/Compare 1 output enable.

0: Capture mode disabled / tim_oc1 is not active

1: Capture mode enabled / tim_oc1 signal is output on the corresponding output pin

Table 534. Output control bit for standard tim_ocx channels

CCxE bit	tim_ocx output state
0	Output disabled (not driven by the timer: Hi-Z)
1	Output enabled (tim_ocx = tim_ocxref + Polarity)

Note: The states of the external I/O pins connected to the standard tim_ocx *channels depend on the state of the* tim_ocx *channel and on the GPIO registers.*

56.7.10 TIMx counter (TIMx_CNT)(x = 9, 12)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF Copy
This bit is a read-only copy of the UIF bit in the TIMx_ISR register.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value
Non-dithering mode (DITHEN = 0)
The register holds the counter value.
Dithering mode (DITHEN = 1)
The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

56.7.11 TIMx prescaler (TIMx_PSC)(x = 9, 12)

Address offset: 0x028

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value
The counter clock frequency tim_cnt_ck is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).
PSC contains the value to be loaded into the active prescaler register at each update event. (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

56.7.12 TIMx autoreload register (TIMx_ARR)(x = 9, 12)

Address offset: 0x02C

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Auto-reload value
ARR is the value to be loaded in the actual auto-reload register.
Refer to the *Section 56.4.3: Time-base unit* for more details about ARR update and behavior.
The counter is blocked while the auto-reload value is null.
Non-dithering mode (DITHEN = 0)
The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

56.7.13 TIMx capture/compare register 1 (TIMx_CCR1)(x = 9, 12)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR1[19:0]**: Capture/compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.

If channel CC1 is configured as input:

CR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIMx_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

56.7.14 TIMx capture/compare register 2 (TIMx_CCR2)(x = 9, 12)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR2[19:0]**: Capture/compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signalled on tim_oc2 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR2[19:4]. The CCR2[3:0] bitfield contains the dithered part.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 1 event (tim_ic2). The TIMx_CCR2 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR2[19:4]. The CCR2[3:0] bits are reset.

56.7.15 TIMx timer input selection register (TIMx_TISEL)(x = 9, 12)

Address offset: 0x05C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: selects tim_ti2_in[15:0] input

0000: TIMx_CH2 input (tim_ti2_in0)

0001: tim_ti2_in1

...

0100: tim_ti2_in15

Refer to for interconnects list.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects tim_ti1_in[15:0] input

0000: TIMx_CH1 input (tim_ti1_in0)

0001: tim_ti1_in1

...

1111: tim_ti1_in15

Refer to for interconnects list.

56.7.16 TIM9/TIM12 register map

TIM9/TIM12 registers are mapped as 16-bit addressable registers as described below:

Table 535. TIMx register map and reset values (x = 9, 12)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	TIMx_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DITHEN	UIFREMA	Res.	CKD [1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value																				0	0		0	0	0				0	0	0	0
0x004	TIMx_CR2	Res.	Res.	Res.	ADSYNC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1S	MMS[2:0]			Res.	Res.	Res.	Res.
	Reset value				1																					0	0	0	0				
0x008	TIMx_SMCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS [4:3]		Res.	Res.	Res.	SMS[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSM	TS[2:0]			Res.	SMS[2:0]		
	Reset value											0	0				0									0	0	0	0		0	0	0
0x00C	TIMx_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIE	Res.	Res.	Res.	CC2IE	CC1IE	UIE
	Reset value																						0	0	0		0				0	0	0
0x010	TIMx_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2OF	CC1OF	Res.	Res.	TIF	Res.	Res.	Res.	CC2IF	CC1IF	UIF
	Reset value																						0	0			0				0	0	0
0x014	TIMx_EGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TG	Res.	Res.	Res.	CC2G	CC1G	UG
	Reset value																										0				0	0	0
0x018	TIMx_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1 S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIMx_CCMR1 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	Res.	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		Res.	OC1M [2:0]			OC1PE	OC1FE	CC1 S [1:0]	
	Reset value								0								0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x01C	Reserved	Res.																															
0x020	TIMx_CCER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2NP	Res.	CC2P	CC2E	CC1NP	Res.	CC1P	CC1E
	Reset value																									0		0	0	0		0	0
0x024	TIMx_CNT	UIFCPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	TIMx_PSC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.															
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 535. TIMx register map and reset values (x = 9, 12) (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x02C	TIMx_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	Reserved	Res.																															
0x034	TIMx_CCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	TIMx_CCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C to 0x058	Reserved	Res.																															
0x05C	TIMx_TISEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value																					0	0	0	0					0	0	0	0
0x060 - 0x3E8	Reserved	Res.																															

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

56.8 TIM10/TIM11/TIM13/TIM14 registers

The peripheral registers have to be written by half-words (16 bits) or words (32 bits). Read accesses can be done by bytes (8 bits), half-words (16 bits), or words (32 bits).

56.8.1 TIMx control register 1 (TIMx_CR1)(x = 10, 11, 13, 14)

Address offset: 0x000

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw				rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable
0: Dithering disabled
1: Dithering enabled
Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division
This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and sampling clock used by the digital filters (tim_tix),
00: $t_{DTS} = t_{tim_ker_ck}$
01: $t_{DTS} = 2 \times t_{tim_ker_ck}$
10: $t_{DTS} = 4 \times t_{tim_ker_ck}$
11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable
0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped on the update event
1: Counter stops counting on the next update event (clearing the CEN bit).

Bit 2 **URS**: Update request source

This bit is set and cleared by software to select the update interrupt (UEV) sources.

0: Any of the following events generate an UEV if enabled:

– Counter overflow

– Setting the UG bit

1: Only counter overflow generates an UEV if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable update interrupt (UEV) event generation.

0: UEV enabled. An UEV is generated by one of the following events:

– Counter overflow

– Setting the UG bit.

Buffered registers are then loaded with their preload values.

1: UEV disabled. No UEV is generated, shadow registers keep their value (ARR, PSC, CCRx). The counter and the prescaler are reinitialized if the UG bit is set.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

56.8.2 TIMx interrupt enable register (TIMx_DIER)(x = 10, 11, 13, 14)

Address offset: 0x00C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1IE	UIE
														rw	rw

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable

0: CC1 interrupt disabled

1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable

0: Update interrupt disabled

1: Update interrupt enabled

56.8.3 TIMx status register (TIMx_SR)(x = 10, 11, 13, 14)

Address offset: 0x010

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC1OF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1IF	UIF
						rc_w0								rc_w0	rc_w0

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.

0: No overcapture has been detected.

1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bits 8:2 Reserved, must be kept at reset value.

Bit 1 **CC1IF**: Capture/compare 1 interrupt flag

This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).

0: No compare match / No input capture occurred

1: A compare match or an input capture occurred.

If channel CC1 is configured as output: this flag is set when he content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.

If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on tim_ic1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

- At overflow and if UDIS = 0 in the TIMx_CR1 register.
- When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

56.8.4 TIMx event generation register (TIMx_EGR)(x = 10, 11, 13, 14)

Address offset: 0x014

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1G	UG
														w	w

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **CC1G**: Capture/compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
CC1IF flag is set, Corresponding interrupt or is sent if enabled.
If channel CC1 is configured as input:
The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Re-initialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected). The counter is cleared.

56.8.5 TIMx capture/compare mode register 1 (TIMx_CCMR1)(x = 10, 11, 13, 14)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

Input capture mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bitfield defines the frequency used to sample tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on CC1 input (tim_ic1).
The prescaler is reset as soon as CC1E = 0 (TIMx_CCER register).
00: no prescaler, capture is done each time an edge is detected on the capture input
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.
00: CC1 channel is configured as output
01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1
10: Reserved
11: Reserved

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

56.8.6 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 10, 11, 13, 14)

Address offset: 0x018

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function in input and in output mode.

Output compare mode:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [3]
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
									rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bits 15:7 Reserved, must be kept at reset value.

Bits 16, 6:4 **OC1M[3:0]**: Output compare 1 mode (refer to bit 16 for OC1M[3])

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 is derived. tim_oc1ref is active high whereas tim_oc1 active level depends on CC1P bit.

0000: Frozen. The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.

0001: Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - tim_oc1ref toggles when TIMx_CNT = TIMx_CCR1.

0100: Force inactive level - tim_oc1ref is forced low.

0101: Force active level - tim_oc1ref is forced high.

0110: PWM mode 1 - Channel 1 is active as long as TIMx_CNT < TIMx_CCR1 else inactive.

0111: PWM mode 2 - Channel 1 is inactive as long as TIMx_CNT < TIMx_CCR1 else active

Others: Reserved

Note: In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.

Bit 3 **OC1PE**: Output compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

Bit 2 **OC1FE**: Output compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. OC is then set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output.

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1.

10: Reserved.

11: Reserved.

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

56.8.7 TIMx capture/compare enable register (TIMx_CCER)(x = 10, 11, 13, 14)

Address offset: 0x020

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1NP	Res.	CC1P	CC1E
												rw		rw	rw

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **CC1NP**: Capture/Compare 1 complementary output Polarity.

CC1 channel configured as output: CC1NP must be kept cleared.

CC1 channel configured as input: CC1NP bit is used in conjunction with CC1P to define tim_ti1fp1 polarity (refer to CC1P description).

Bit 2 Reserved, must be kept at reset value.

Bit 1 **CC1P**: Capture/Compare 1 output Polarity.
0: tim_oc1 active high (output mode) / Edge sensitivity selection (input mode, see below)
1: tim_oc1 active low (output mode) / Edge sensitivity selection (input mode, see below)
When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of tim_ti1fp1 for capture operations.
CC1NP = 0, CC1P = 0: non-inverted/rising edge. The circuit is sensitive to tim_ti1fp1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode).
CC1NP = 0, CC1P = 1: inverted/falling edge. The circuit is sensitive to tim_ti1fp1 falling edge (capture or trigger operations in reset, external clock or trigger mode), tim_ti1fp1 is inverted (trigger operation in gated mode).
CC1NP = 1, CC1P = 1: non-inverted/both edges/ The circuit is sensitive to both tim_ti1fp1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), tim_ti1fp1 not inverted (trigger operation in gated mode).
CC1NP = 1, CC1P = 0: This configuration is reserved, it must not be used.

Bit 0 **CC1E**: Capture/Compare 1 output enable.
0: Capture mode disabled / tim_oc1 is not active
1: Capture mode enabled / tim_oc1 signal is output on the corresponding output pin

Table 536. Output control bit for standard tim_ocx channels

CCxE bit	tim_ocx output state
0	Output Disabled (tim_ocx = 0)
1	tim_ocx = tim_ocxref + Polarity

Note: The state of the external I/O pins connected to the standard tim_ocx channels depends on the tim_ocx channel state and the GPIO registers.

56.8.8 TIMx counter (TIMx_CNT)(x = 10, 11, 13, 14)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF Copy

This bit is a read-only copy of the UIF bit in the TIMx_ISR register.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

Non-dithering mode (DITHEN = 0)

The register holds the counter value.

Dithering mode (DITHEN = 1)

The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

56.8.9 TIMx prescaler (TIMx_PSC)(x = 10, 11, 13, 14)

Address offset: 0x028

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency tim_cnt_ck is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).

PSC contains the value to be loaded in the active prescaler register at each update event.

(including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

56.8.10 TIMx autoreload register (TIMx_ARR)(x = 10, 11, 13, 14)

Address offset: 0x02C

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the *Section 56.4.3: Time-base unit* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

56.8.11 TIMx capture/compare register 1 (TIMx_CCR1)(x = 10, 11, 13, 14)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR1[19:0]**: Capture/compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.

If channel CC1 is configured as input:

CR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIMx_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

56.8.12 TIMx timer input selection register (TIMx_TISEL)(x = 10, 11, 13, 14)

Address offset: 0x05C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
												rw	rw	rw	rw

Bits 15:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects tim_ti1_in[15:0] input
0000: TIMx_CH1 input (tim_ti1_in0)
0001: tim_ti1_in1
...
1111: tim_ti1_in15
Refer to for interconnects list.

56.8.13 TIM10/TIM11/TIM13/TIM14 register map

TIMx registers are mapped as 16-bit addressable registers as described in the tables below:

Table 537. TIM10/TIM11/TIM13/TIM14 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	TIMx_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DITHEN	UIFREMA	Res.	CKD [1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value																				0	0		0	0	0				0	0	0	0
0x004 to 0x008	Reserved	Res.																															
0x00C	TIMx_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1IE	UIE
	Reset value																															0	0
0x010	TIMx_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1OF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1IF	UIF
	Reset value																							0								0	0
0x014	TIMx_EGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1G	UG
	Reset value																															0	0
0x018	TIMx_CCMR1 Output compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value																										0	0	0	0	0	0	0
	TIMx_CCMR1 Input capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																									0	0	0	0	0	0	0	0
0x01C	Reserved	Res.																															
0x020	TIMx_CCER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1NP	Res.	CC1P	CC1E
	Reset value																													0		0	0
0x024	TIMx_CNT	UIFCPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 537. TIM10/TIM11/TIM13/TIM14 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x028	TIMx_PSC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C	TIMx_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	Reserved	Res.																															
0x034	TIMx_CCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038 to 0x058	Reserved	Res.																															
0x05C	TIMx_TISEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value																													0	0	0	0
0x060 - 0x3E8	Reserved	Res.																															

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

57 General purpose timers (TIM15/TIM16/TIM17)

57.1 TIM15/TIM16/TIM17 introduction

The TIM15/TIM16/TIM17 timers consist of a 16-bit autoreload counter driven by a programmable prescaler.

They may be used for a variety of purposes, including measuring the pulse lengths of input signals (input capture) or generating output waveforms (output compare, PWM, complementary PWM with dead-time insertion).

Pulse lengths and waveform periods can be modulated from a few microseconds to several milliseconds using the timer prescaler and the RCC clock controller prescalers.

The TIM15/TIM16/TIM17 timers are completely independent, and do not share any resources. TIM15 can be synchronized as described in *Section 57.4.26: Timer synchronization (TIM15 only)*.

57.2 TIM15 main features

TIM15 includes the following features:

- 16-bit autoreload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- Up to two independent channels for:
  - Input capture
  - Output compare
  - PWM generation (edge mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time (for channel 1 only)
- Synchronization circuit to control the timer with external signals and to interconnect several timers together
- Repetition counter to update the timer registers only after a given number of cycles of the counter
- Break input to put the timer’s output signals in the reset state or a known state
- Interrupt/DMA generation on the following events:
  - Update: counter overflow counter initialization (by software or internal/external trigger)
  - Trigger event (counter start, stop, initialization, or count by internal/external trigger)
  - Input capture
  - Output compare
  - Break input (interrupt request)
- ADC synchronization for jitter-free sampling points

57.3 TIM16/TIM17 main features

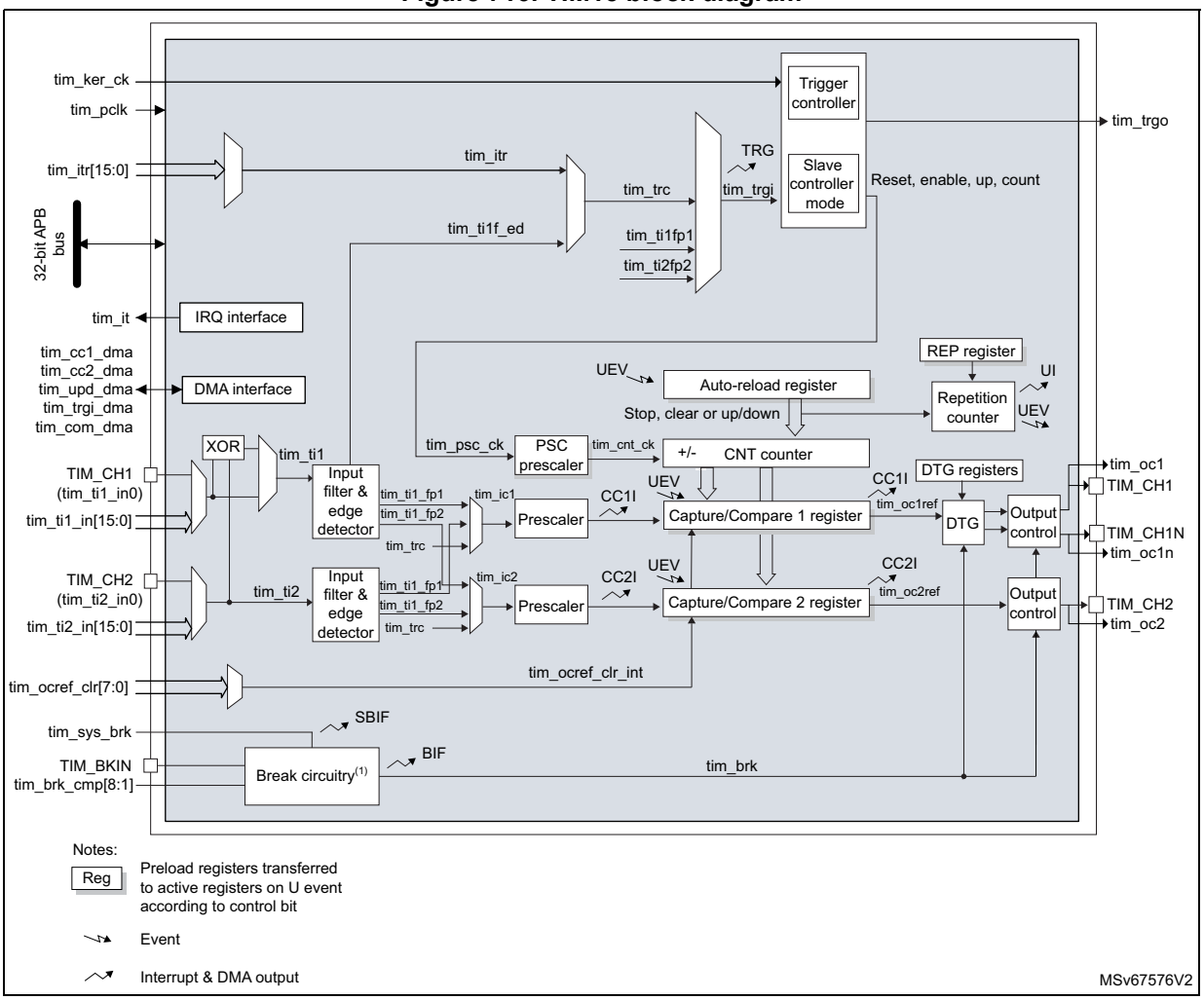
The TIM16/TIM17 timers include the following features:

- 16-bit autoreload upcounter
- 16-bit programmable prescaler used to divide (also “on the fly”) the counter clock frequency by any factor between 1 and 65535
- One channel for:
  - Input capture
  - Output compare
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Complementary outputs with programmable dead-time
- Repetition counter to update the timer registers only after a given number of cycles of the counter
- Break input to put the timer’s output signals in the reset state or a known state
- Interrupt/DMA generation on the following events:
  - Update: counter overflow
  - Input capture
  - Output compare
  - Break input

57.4 TIM15/TIM16/TIM17 functional description

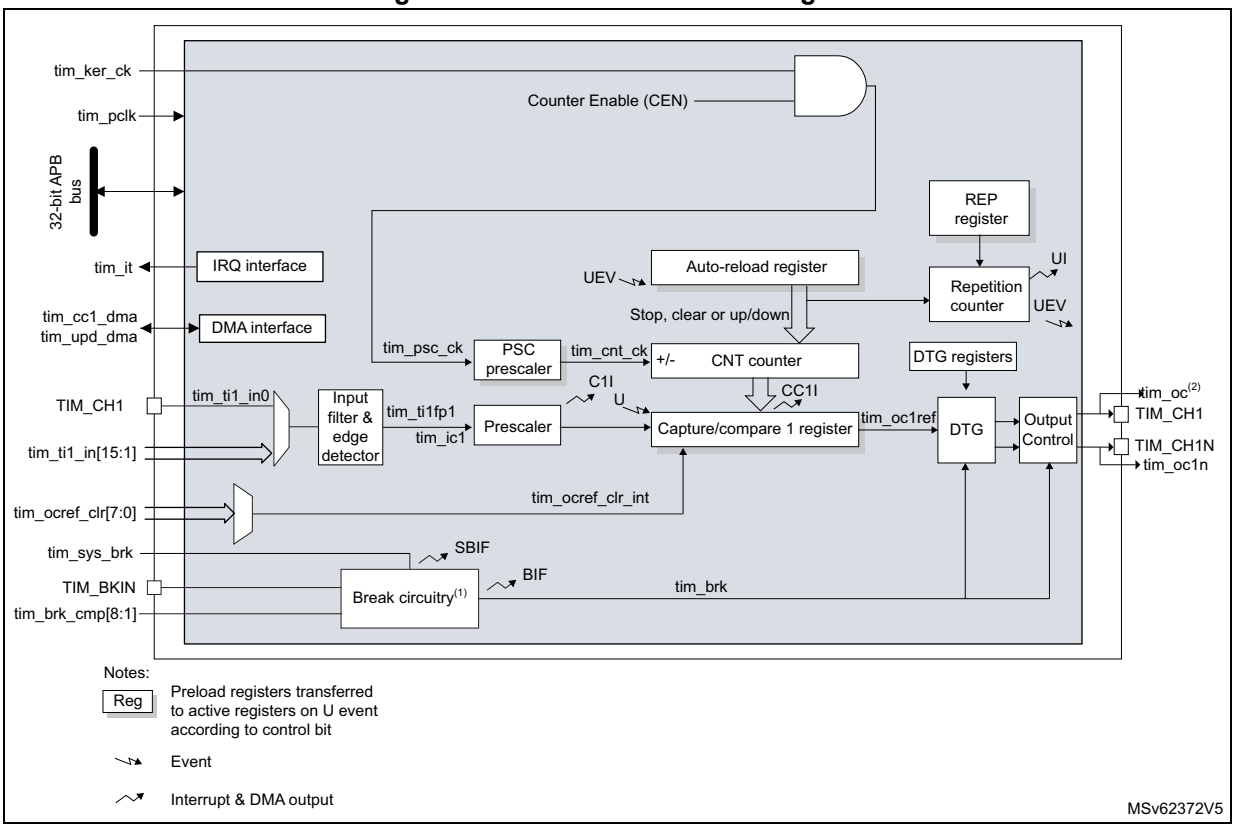
57.4.1 Block diagram

Figure 716. TIM15 block diagram



1. Refer to *Section 57.4.15: Using the break function* for details.

Figure 717. TIM16/TIM17 block diagram



1. Refer to *Section 57.4.15: Using the break function* for details.
2. This signal can be used as trigger for some slave timer (see internal trigger connection table in next section). See *Section 57.4.27: Using timer output as trigger for other timers (TIM16/TIM17 only)* for details.

57.4.2 TIM15/TIM16/TIM17 pins and internal signals

Table 538 and *Table 539* in this section summarize the TIM inputs and outputs.

Table 538. TIM input/output pins

Pin name	Signal type	Description
TIM_CH1 TIM_CH2(1)	Input/Output	Timer multi-purpose channels. Each channel be used for capture, compare, or PWM. TIM_CH1 and TIM_CH2 can also be used as external clock (below 1/4 of the tim_ker_ck clock) and external trigger inputs.
TIM_CH1N	Output	Timer complementary outputs, derived from TIM_CH1 output with the possibility to have deadtime insertion.
TIM_BKIN	Input / Output	Break input. This input can also be configured in bidirectional mode.

1. Available for TIM15 only.

Table 539. TIM internal input/output signals

Internal signal name	Signal type	Description
tim_ti1_in[15:0] tim_ti2_in[15:0](1)	Input	Internal timer inputs bus. These inputs can be used for capture or as external clock (below 1/4 of the tim_ker_ck clock).
tim_itr[15:0](1)	Input	Internal trigger input bus. These inputs can be used for the slave mode controller or as a input clock (below 1/4 of the tim_ker_ck clock).
tim_trgo(1)	Output	Internal trigger output. This trigger can trigger other on-chip peripherals.
tim_ocref_clr[7:0]	Input	Timer tim_ocref_clr input bus. These inputs can be used to clear the tim_ocxref signals, typically for hardware cycle-by-cycle pulsewidth control.
tim_brk_cmp[8:1]	Input	Break input for internal signals
tim_sys_brk[n:0]	Input	System break input. This input gathers the MCU's system level errors.
tim_pclk	Input	Timer APB clock
tim_ker_ck	Input	Timer kernel clock. This clock must be synchronous with tim_pclk (derived from the same source). The clock ratio tim_ker_ck/tim_pclk must be an integer:1, 2, 3,..., 16 (maximum value)
tim_it	Output	Global Timer interrupt, gathering capture/compare, update, break trigger and commutation requests
tim_cc1_dma	Output	Timer capture / compare 1 dma request
tim_cc2_dma	Output	Timer capture / compare 2 dma request
tim_upd_dma	Output	Timer update dma request
tim_trgi_dma	Output	Timer trigger dma request
tim_com_dma	Output	Timer commutation dma request

1. Available for TIM15 only.

Tables below list the sources connected to the tim_ti[2:1] input multiplexers.

Table 540. Interconnect to the tim_ti1 input multiplexer

tim_ti1 inputs	Sources		
	TIM15	TIM16	TIM17
tim_ti1_in0	TIM15_CH1	TIM16_CH1	TIM17_CH1
tim_ti1_in1	TIM2_CH1	Reserved	spdifrx_frame_sync
tim_ti1_in2	TIM3_CH1		Reserved
tim_ti1_in3	TIM4_CH1	RTC Wake-up	
tim_ti1_in4	MCO1	Reserved	
tim_ti1_in5	MCO2		
tim_ti1_in[15:6]	Reserved		

Table 541. Interconnect to the tim_ti2 input multiplexer

tim_ti2 inputs	TIM 15 sources
tim_ti2_in0	TIM15_CH2
tim_ti2_in1	TIM2_CH2
tim_ti2_in2	TIM3_CH2
tim_ti2_in3	TIM4_CH2
tim_ti2_in[15:4]	Reserved

The table below lists the internal sources connected to the tim_itr input multiplexer.

Table 542. TIMx internal trigger connection

tim_itrx inputs	TIM15
tim_itr0	tim1_trgo
tim_itr1	tim2_trgo
tim_itr2	tim3_trgo
tim_itr3	tim4_trgo
tim_itr4	tim5_trgo
tim_itr5	tim8_trgo
tim_itr6	tim9_trgo
tim_itr7	tim12_trgo
tim_itr8	tim13_oc1
tim_itr9	tim14_oc1
tim_itr10	Reserved
tim_itr11	tim16_oc1

Table 542. TIMx internal trigger connection (continued)

tim_itrx inputs	TIM15
tim_itr12	tim17_oc1
tim_itr[15:13]	Reserved

Tables below list the sources connected to the tim_brk input.

Table 543. Timer break interconnect

tim_brk inputs	TIM15	TIM16	TIM17
TIM_BKIN	TIM15_BKIN pin	TIM16_BKIN pin	TIM17_BKIN pin
tim_brk_cmp[8:1]	Reserved		

Table 544. System break interconnect

tim_sys_brk inputs	TIM15/16/17	Enable bit in SYSCFG_BKLOCKR register
tim_sys_brk0	Programmable voltage detector (PVD)	PVD_BL
tim_sys_brk1	Cortex-M55 LOCKUP	CM55LU_BL
tim_sys_brk2	Backup RAM double ECC error	BKPRAM_BL
tim_sys_brk3	Cortex®-M55 TCM double ECC error	CM55TCM_BL
tim_sys_brk4	Cortex®-M55 cache double ECC error	CM55CACHE_BL
CSS	Clock security system	None (always enabled)

57.4.3 Time-base unit

The main block of the programmable advanced-control timer is a 16-bit upcounter with its related autoreload register. The counter clock can be divided by a prescaler.

The counter, the autoreload register and the prescaler register can be written or read by software. This is true even when the counter is running.

The time-base unit includes:

- Counter register (TIMx_CNT)
- Prescaler register (TIMx_PSC)
- Autoreload register (TIMx_ARR)
- Repetition counter register (TIMx_RCR)

The autoreload register is preloaded. Writing to or reading from the autoreload register accesses the preload register. The content of the preload register is transferred into the shadow register permanently or at each update event (UEV), depending on the autoreload preload enable bit (ARPE) in TIMx_CR1 register. The update event is sent when the counter reaches the overflow and if the UDIS bit equals 0 in the TIMx_CR1 register. It can also be generated by software. The generation of the update event is described in detailed for each configuration.

The counter is clocked by the prescaler output tim_cnt_ck, which is enabled only when the counter enable bit (CEN) in TIMx_CR1 register is set (refer also to the slave mode controller description to get more details on counter enabling).

Note that the counter starts counting one clock cycle after setting the CEN bit in the TIMx_CR1 register.

Prescaler description

The prescaler can divide the counter clock frequency by any factor between 1 and 65536. It is based on a 16-bit counter controlled through a 16-bit register (in the TIMx_PSC register). It can be changed on the fly as this control register is buffered. The new prescaler ratio is taken into account at the next update event.

Table 728 and *Table 719* give some examples of the counter behavior when the prescaler ratio is changed on the fly:

Figure 718. Counter timing diagram with prescaler division change from 1 to 2

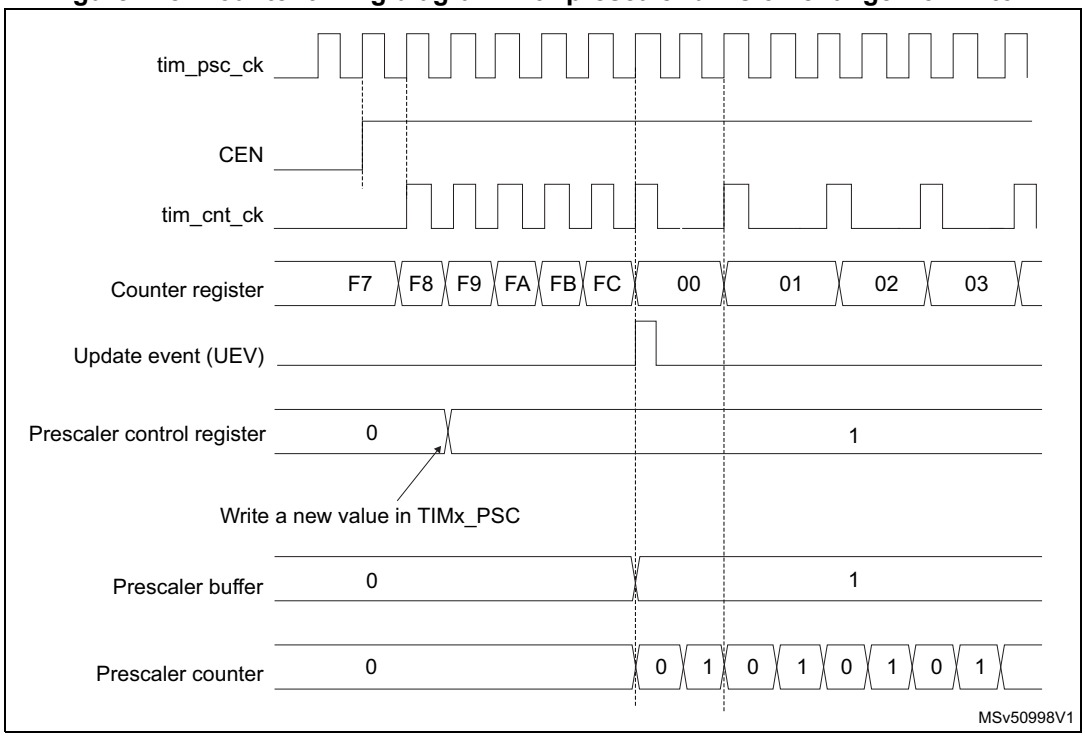
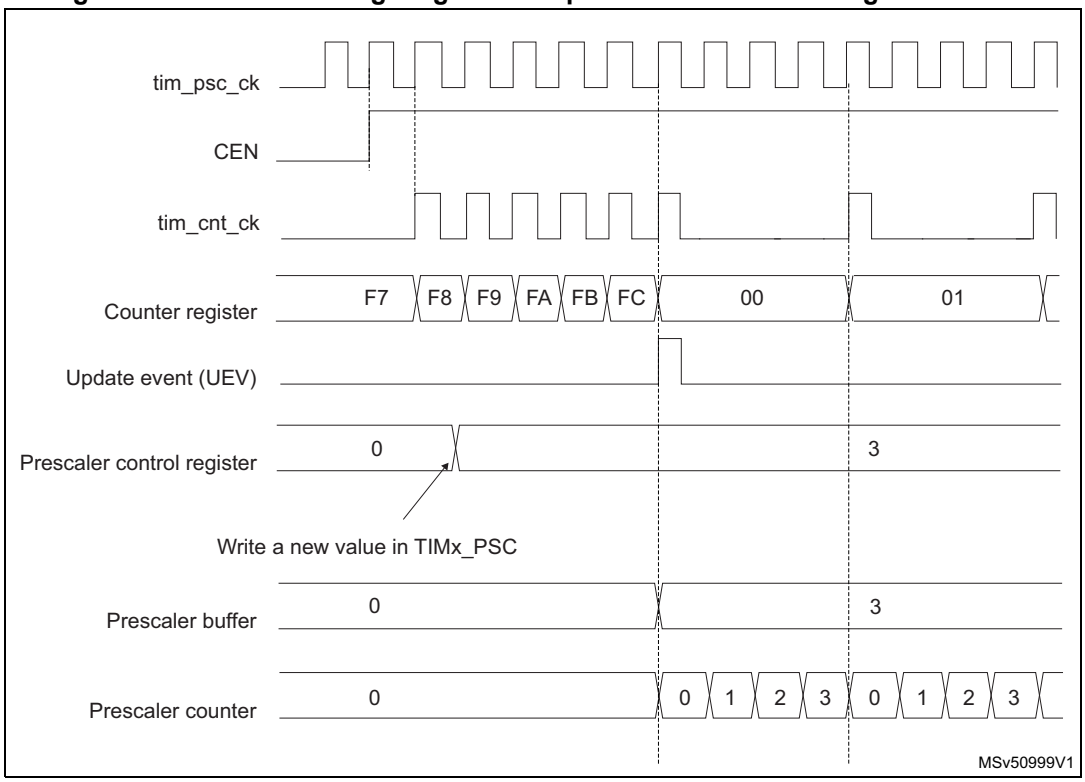


Figure 719. Counter timing diagram with prescaler division change from 1 to 4



57.4.4 Counter modes

Upcounting mode

In upcounting mode, the counter counts from 0 to the autoreload value (content of the TIMx_ARR register), then restarts from 0 and generates a counter overflow event.

If the repetition counter is used, the update event (UEV) is generated after upcounting is repeated for the number of times programmed in the repetition counter register (TIMx_RCR). Else the update event is generated at each counter overflow.

Setting the UG bit in the TIMx_EGR register (by software or by using the slave mode controller) also generates an update event.

The UEV event can be disabled by software by setting the UDIS bit in the TIMx_CR1 register. This is to avoid updating the shadow registers while writing new values in the preload registers. Then no update event occurs until the UDIS bit has been written to 0. However, the counter restarts from 0, as well as the counter of the prescaler (but the prescale rate does not change). In addition, if the URS bit (update request selection) in TIMx_CR1 register is set, setting the UG bit generates an update event UEV but without setting the UIF flag (thus no interrupt or DMA request is sent). This is to avoid generating both update and capture interrupts when clearing the counter on the capture event.

When an update event occurs, all the registers are updated and the update flag (UIF bit in TIMx_SR register) is set (depending on the URS bit):

- The repetition counter is reloaded with the content of TIMx_RCR register,
- The autoreload shadow register is updated with the preload value (TIMx_ARR),
- The buffer of the prescaler is reloaded with the preload value (content of the TIMx_PSC register).

The following figures show some examples of the counter behavior for different clock frequencies when TIMx_ARR = 0x36.

Figure 720. Counter timing diagram, internal clock divided by 1

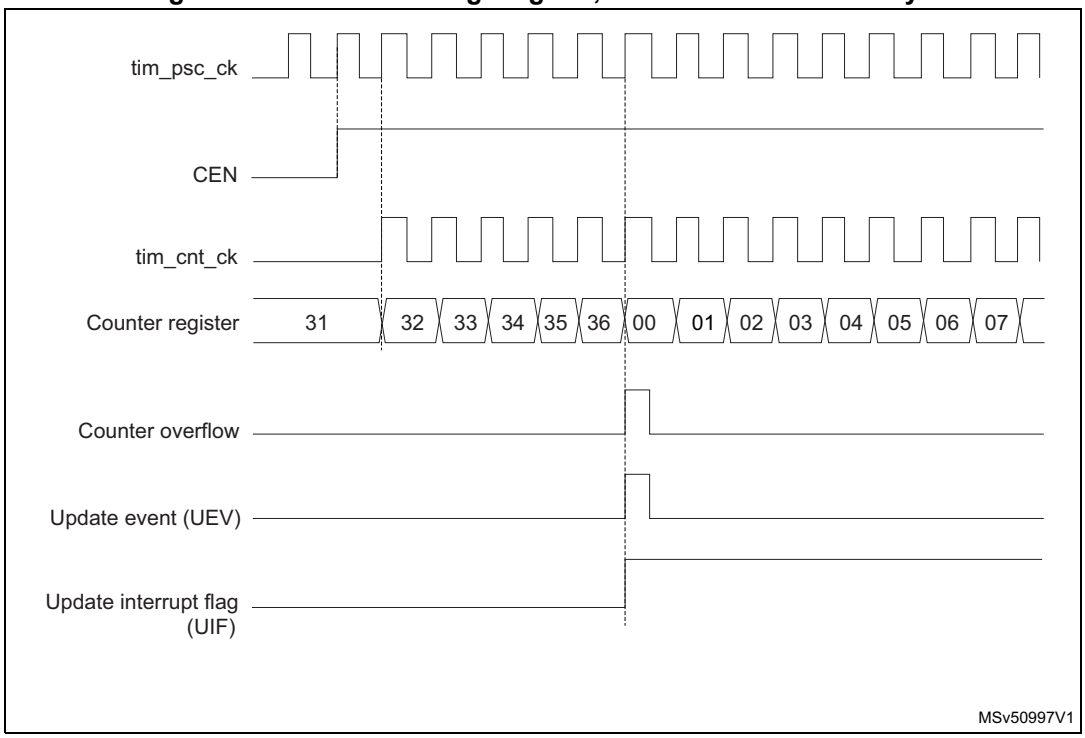


Figure 721. Counter timing diagram, internal clock divided by 2

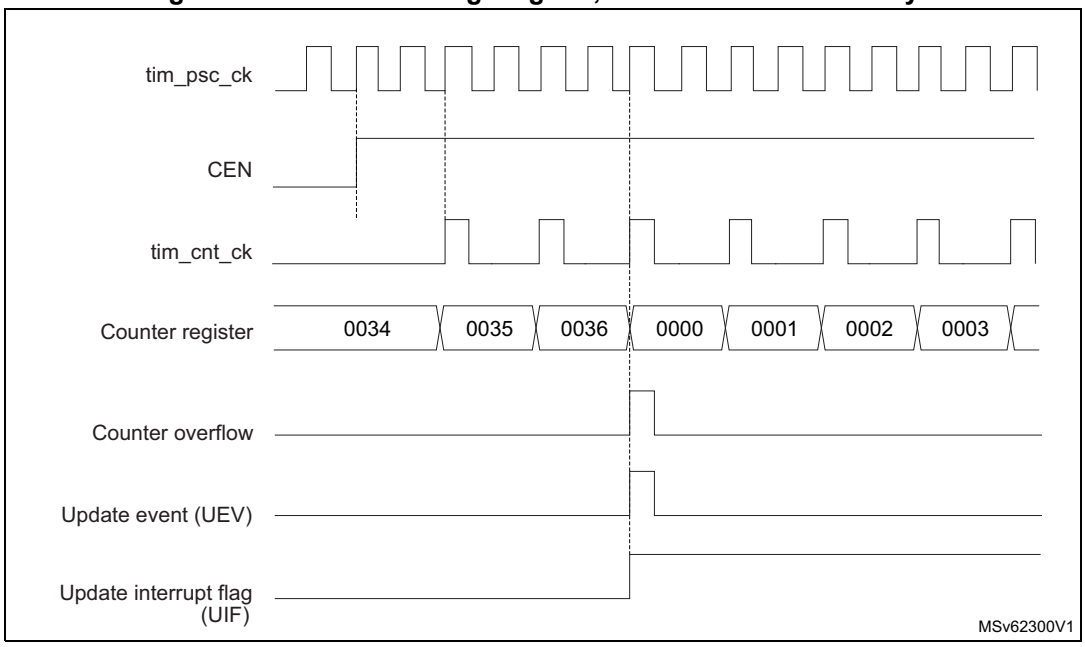


Figure 722. Counter timing diagram, internal clock divided by 4

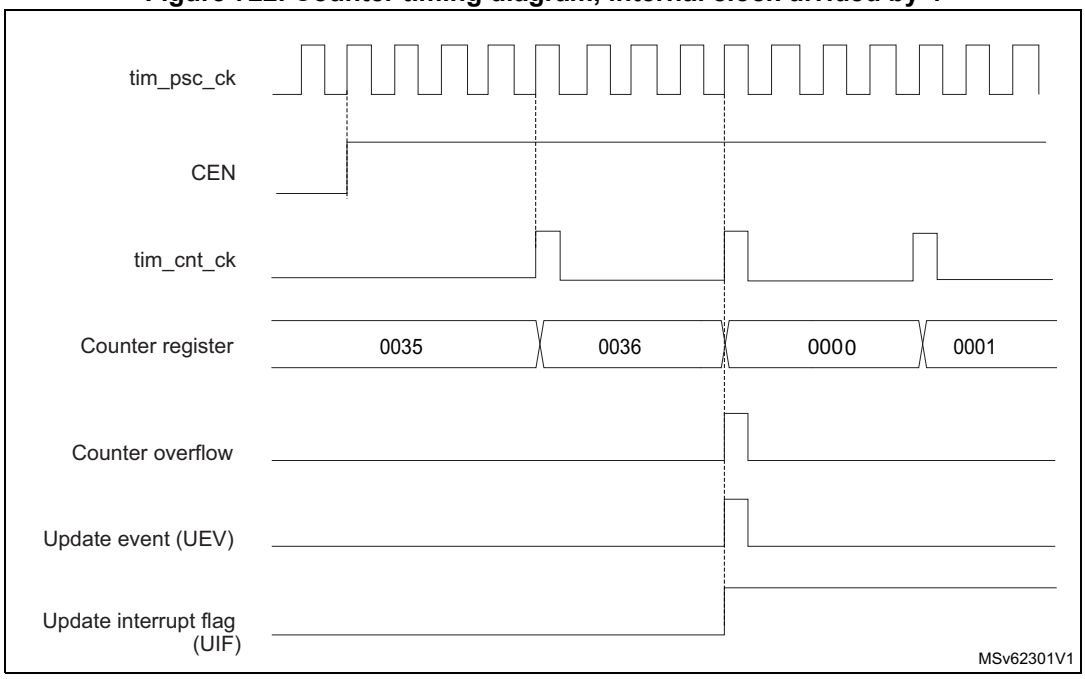


Figure 723. Counter timing diagram, internal clock divided by N

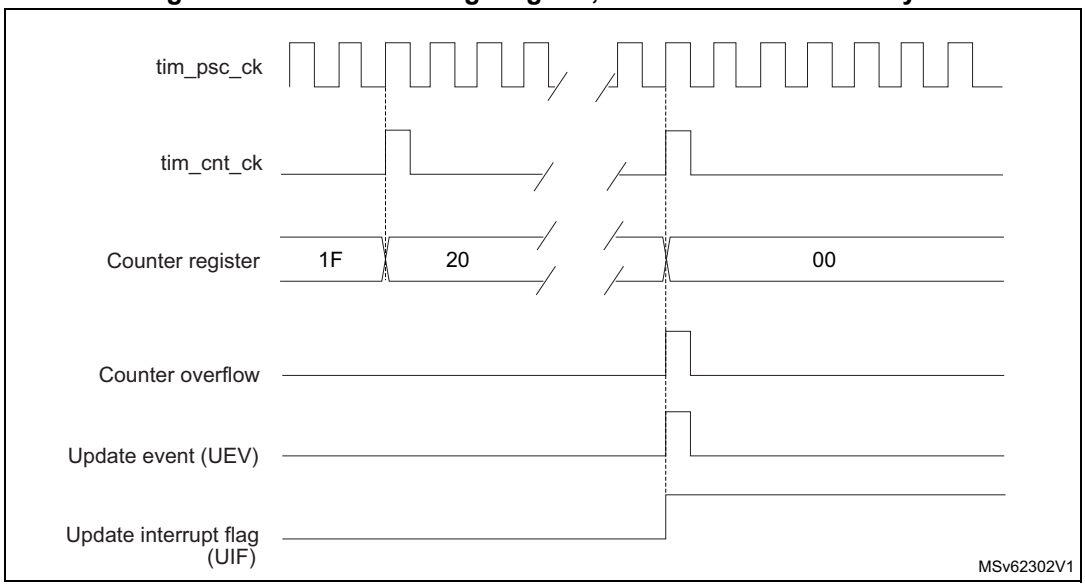


Figure 724. Counter timing diagram, update event when ARPE = 0 (TIMx_ARR not preloaded)

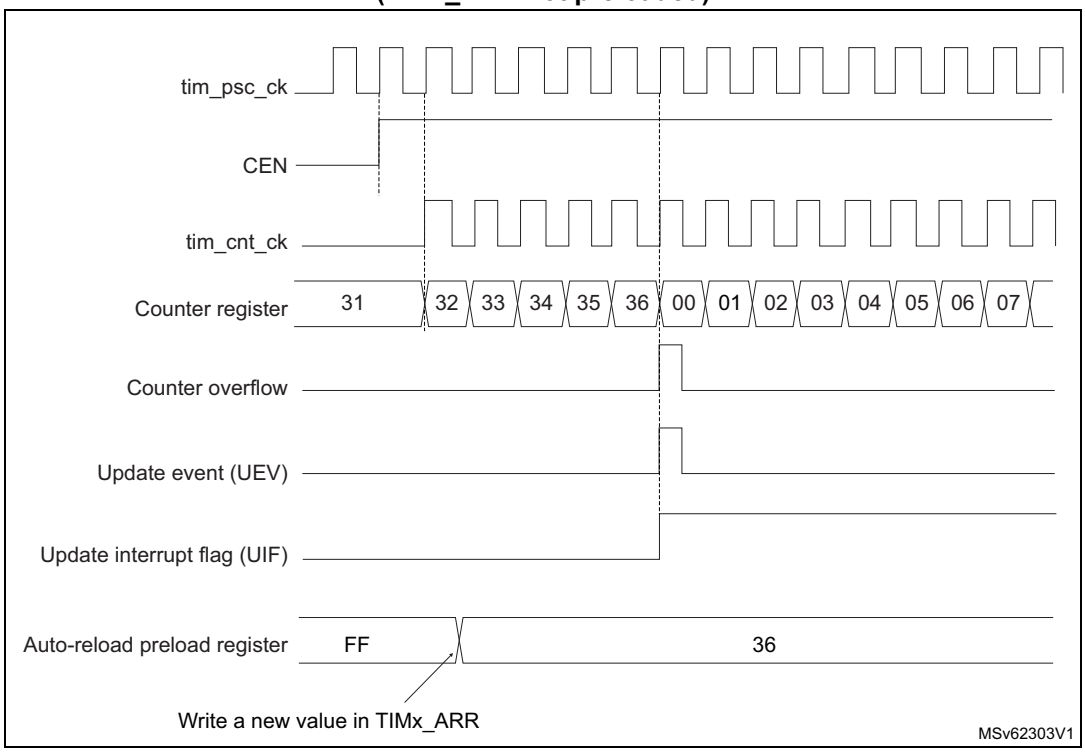
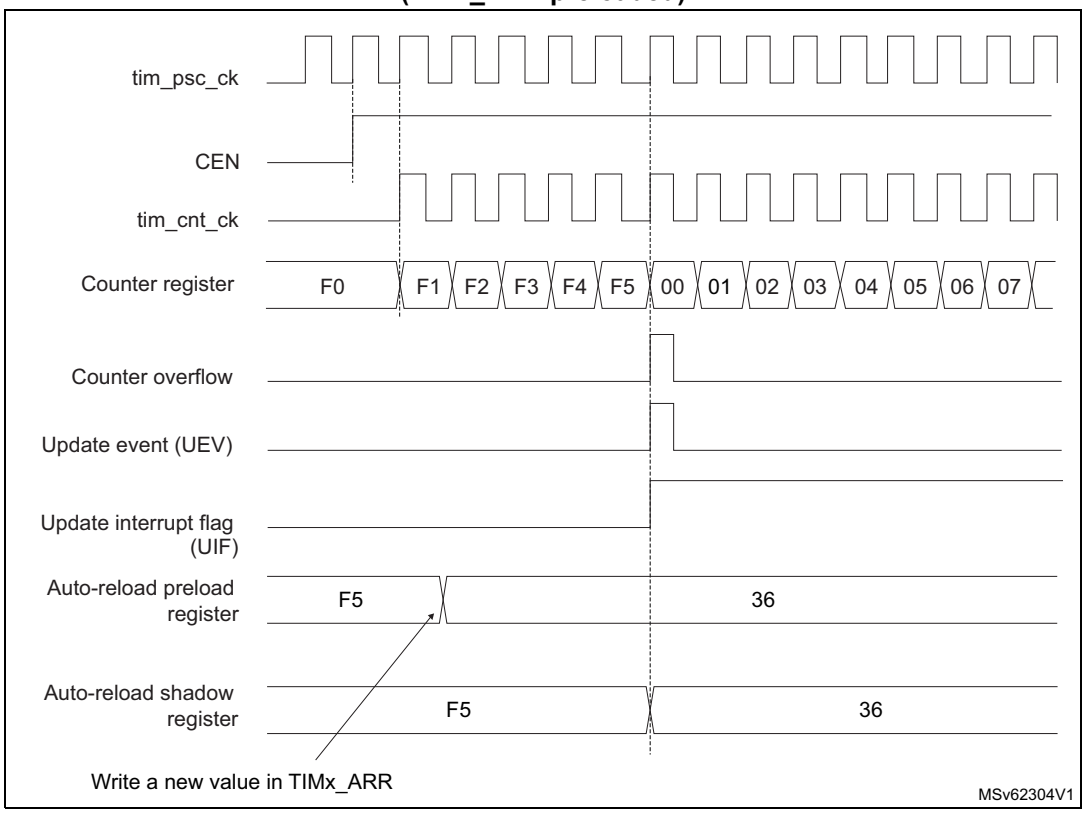


Figure 725. Counter timing diagram, update event when ARPE = 1 (TIMx_ARR preloaded)



57.4.5 Repetition counter

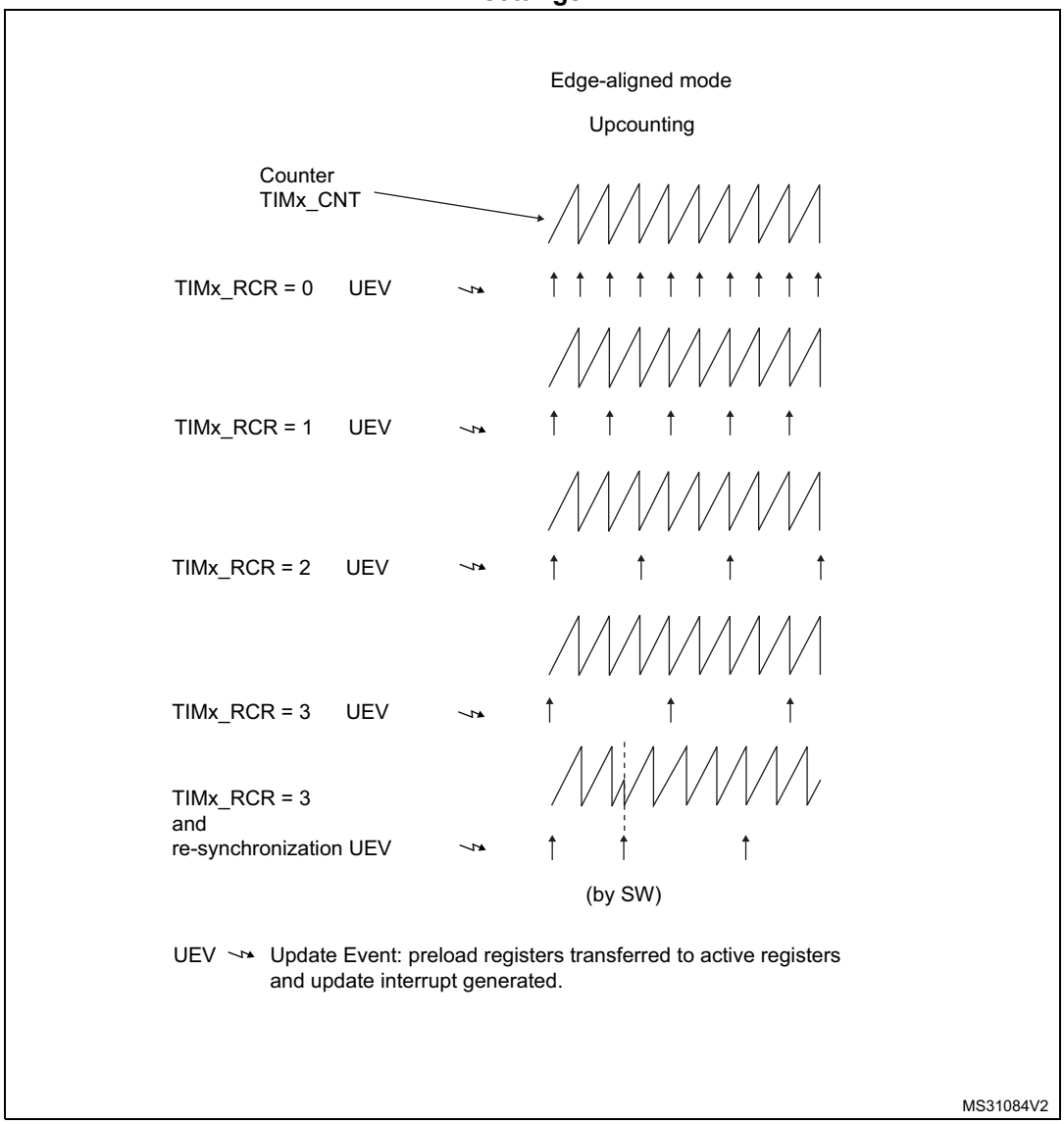
Section 57.4.3: Time-base unit describes how the update event (UEV) is generated with respect to the counter overflows. It is actually generated only when the repetition counter has reached zero. This can be useful when generating PWM signals.

This means that data are transferred from the preload registers to the shadow registers (TIMx_ARR autoreload register, TIMx_PSC prescaler register, but also TIMx_CCRx capture/compare registers in compare mode) every N counter overflows, where N is the value in the TIMx_RCR repetition counter register.

The repetition counter is decremented at each counter overflow.

The repetition counter is an autoreload type; the repetition rate is maintained as defined by the TIMx_RCR register value (refer to *Figure 726*). When the update event is generated by software (by setting the UG bit in TIMx_EGR register) or by hardware through the slave mode controller, it occurs immediately whatever the value of the repetition counter is and the repetition counter is reloaded with the content of the TIMx_RCR register.

Figure 726. Update rate examples depending on mode and TIMx_RCR register settings



57.4.6 Clock selection

The counter clock can be provided by the following clock sources:

- Internal clock (tim_ker_ck)
- External clock mode1: external input pin (tim_ti1 or tim_ti2, if available)
- Internal trigger inputs (tim_itrx) (only for TIM15): using one timer as the prescaler for another timer, for example, TIM1 can be configured to act as a prescaler for TIM15. Refer to *Using one timer as prescaler for another timer* for more details.

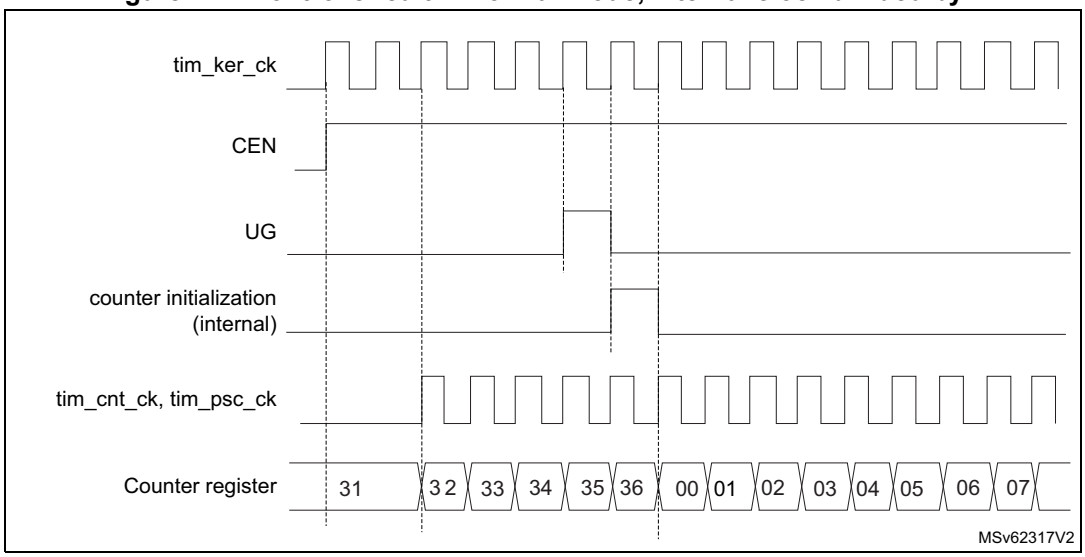
Internal clock source (tim_ker_ck)

If the slave mode controller is disabled (SMS = 000), then the CEN (in the TIMx_CR1 register) and UG bits (in the TIMx_EGR register) are actual control bits and can be changed

only by software (except UG which remains cleared automatically). As soon as the CEN bit is written to 1, the prescaler is clocked by the internal clock tim_ker_ck.

Figure 727 shows the behavior of the control circuit and the upcounter in normal mode, without prescaler.

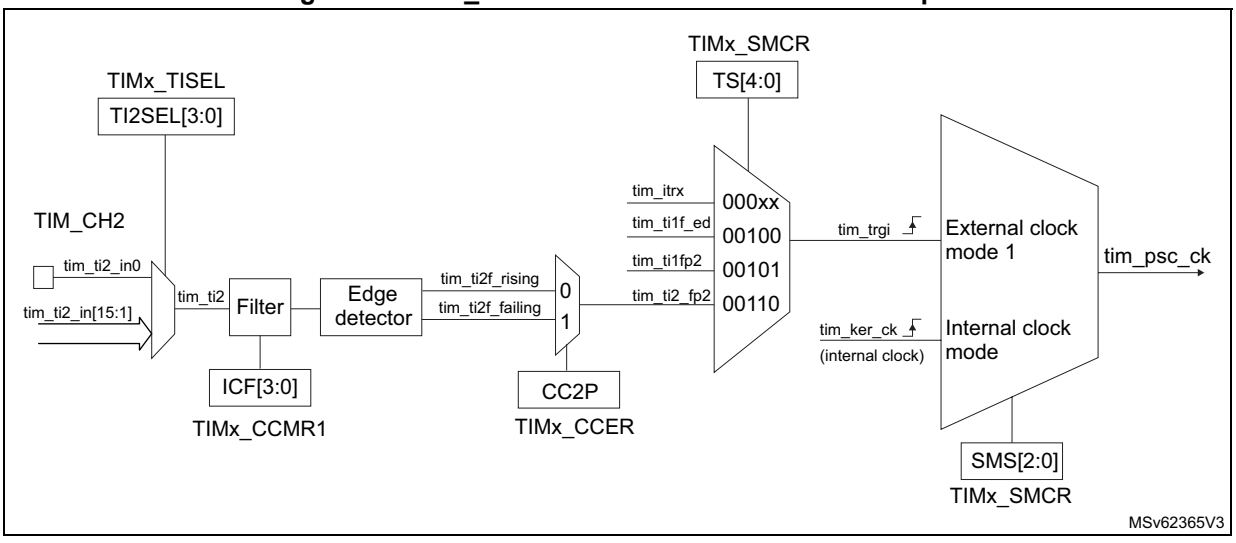
Figure 727. Control circuit in normal mode, internal clock divided by 1



External clock source mode 1

This mode is selected when SMS = 111 in the TIMx_SMCR register. The counter can count at each rising or falling edge on a selected input.

Figure 728. tim_ti2 external clock connection example



For example, to configure the upcounter to count in response to a rising edge on the tim_ti2 input, use the following procedure:

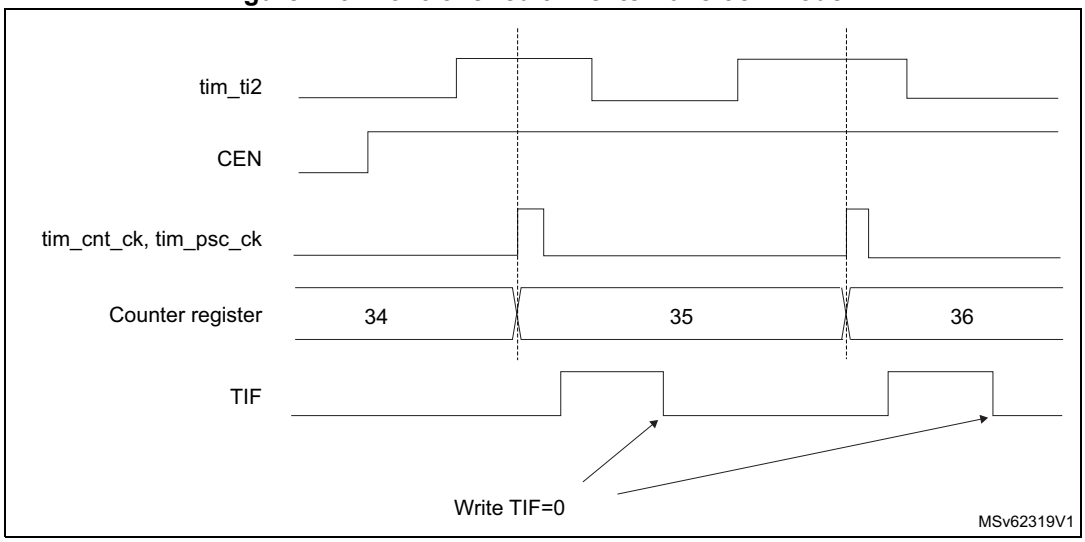
1. Select the proper tim_ti2_in[15:0] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Configure channel 2 to detect rising edges on the tim_ti2 input by writing CC2S = 01 in the TIMx_CCMR1 register.
3. Configure the input filter duration by writing the IC2F[3:0] bits in the TIMx_CCMR1 register (if no filter is needed, keep IC2F = 0000).
4. Select rising edge polarity by writing CC2P = 0 in the TIMx_CCER register.
5. Configure the timer in external clock mode 1 by writing SMS = 111 in the TIMx_SMCR register.
6. Select tim_ti2 as the trigger input source by writing TS = 00110 in the TIMx_SMCR register.
7. Enable the counter by writing CEN = 1 in the TIMx_CR1 register.

Note: The capture prescaler is not used for triggering, it is not necessary to configure it.

When a rising edge occurs on tim_ti2, the counter counts once and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual clock of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 729. Control circuit in external clock mode 1



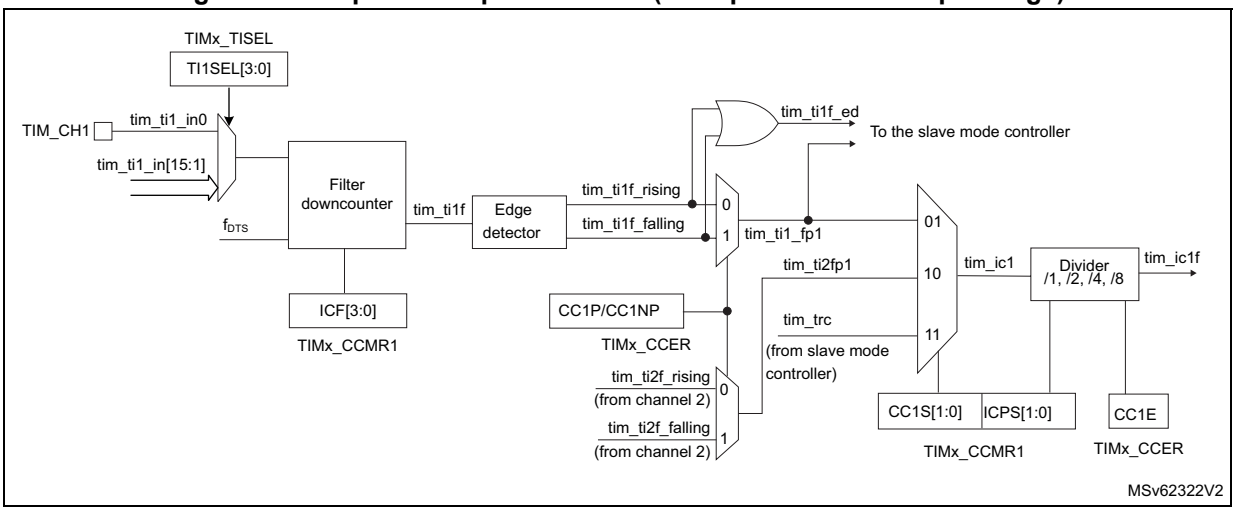
57.4.7 Capture/compare channels

Each Capture/Compare channel is built around a capture/compare register (including a shadow register), a input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control).

Figure 730 to *Figure 733* give an overview of one Capture/Compare channel.

The input stage samples the corresponding tim_tix input to generate a filtered signal tim_tixf. Then, an edge detector with polarity selection generates a signal (tim_tixfpy) which can be used as trigger input by the slave mode controller or as the capture command. It is prescaled before the capture register (ICxPS).

Figure 730. Capture/compare channel (example: channel 1 input stage)



The output stage generates an intermediate waveform which is then used for reference: tim_ocxref (active high). The polarity acts at the end of the chain.

Figure 731. Capture/compare channel 1 main circuit

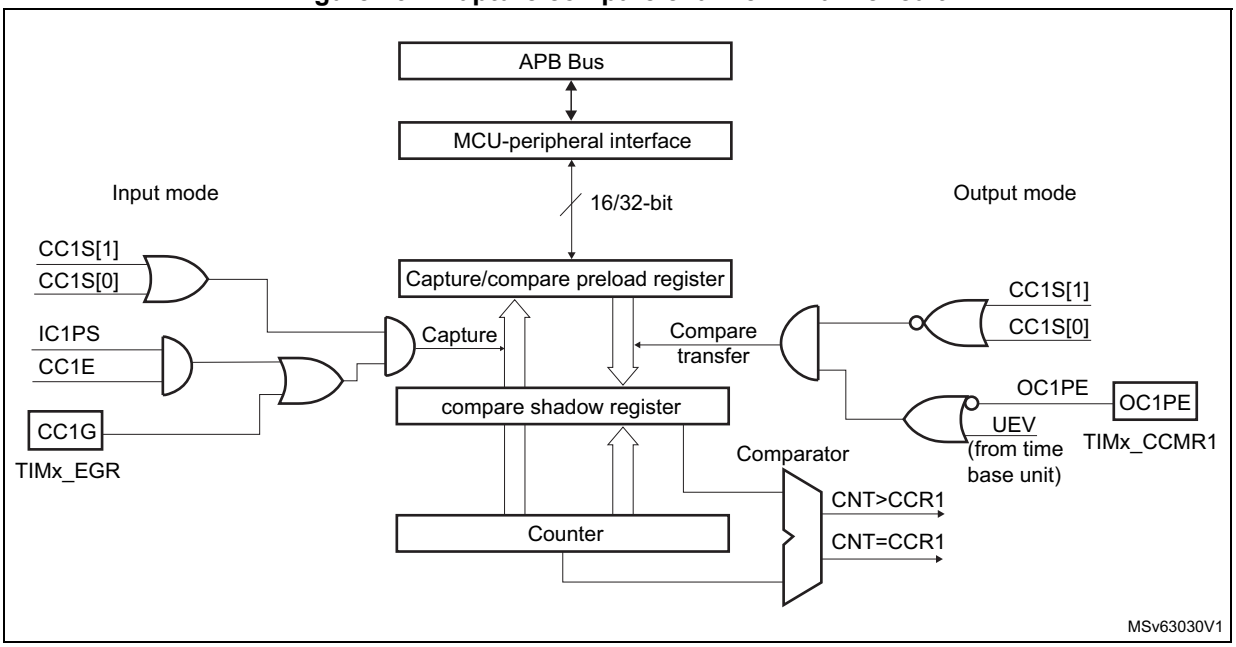


Figure 732. Output stage of capture/compare channel (channel 1)

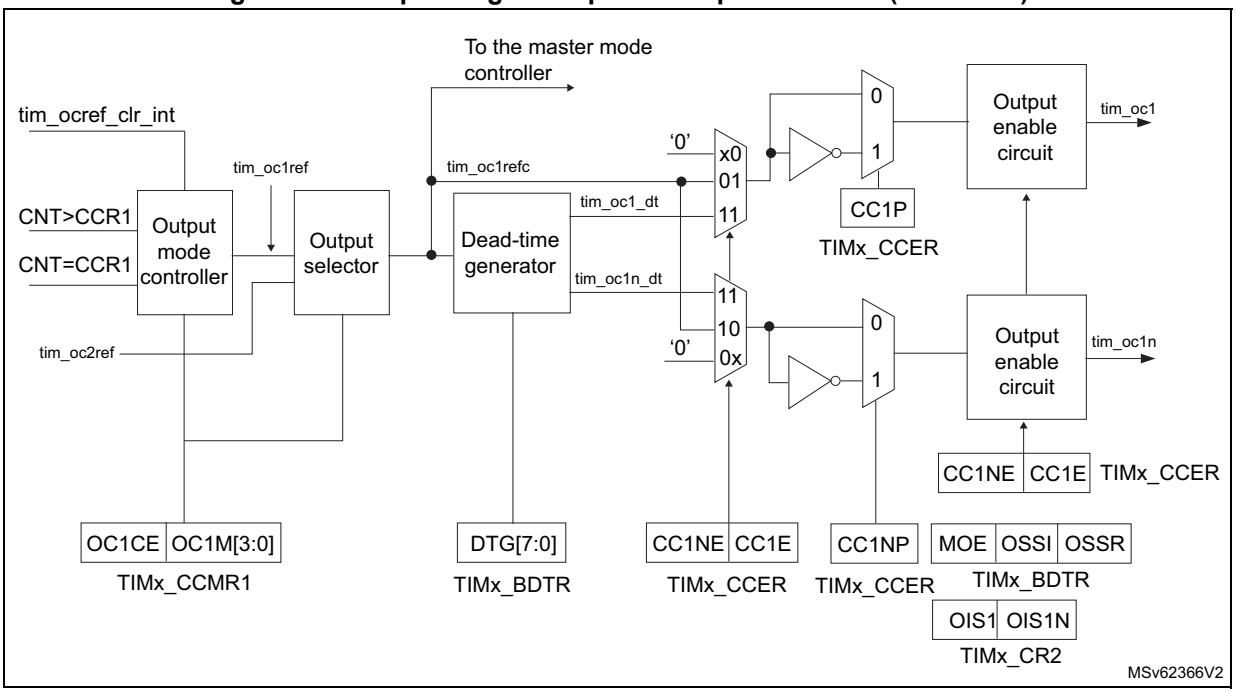
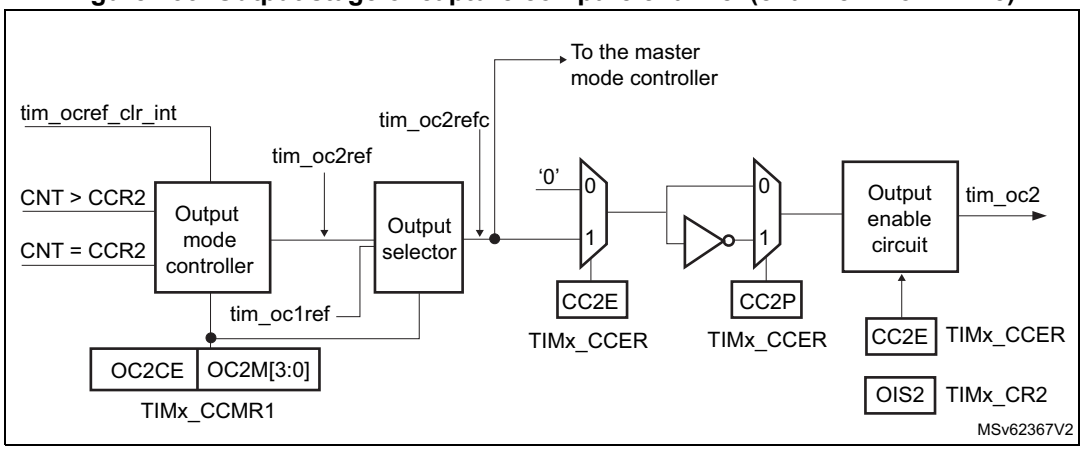


Figure 733. Output stage of capture/compare channel (channel 2 for TIM15)



The capture/compare block is made of one preload register and one shadow register. Write and read always access the preload register.

In capture mode, captures are actually done in the shadow register, which is copied into the preload register.

In compare mode, the content of the preload register is copied into the shadow register which is compared to the counter.

57.4.8 Input capture mode

In Input capture mode, the capture/compare registers (TIMx_CCRx) are used to latch the value of the counter after a transition detected by the corresponding tim_icx signal. When a capture occurs, the corresponding CCXIF flag (TIMx_SR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was

already high, then the overcapture flag CCxOF (TIMx_SR register) is set. CCxIF can be cleared by software by writing it to 0 or by reading the captured data stored in the TIMx_CCRx register. CCxOF is cleared when it is written with 0.

The following example shows how to capture the counter value in TIMx_CCR1 when tim_ti1 input rises. To do this, use the following procedure:

1. Select the proper tim_ti1_in[15:1] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input: TIMx_CCR1 must be linked to the tim_ti1 input, so write the CC1S bits to 01 in the TIMx_CCMR1 register. As soon as CC1S becomes different from 00, the channel is configured in input, and the TIMx_CCR1 register becomes read-only.
3. Program the appropriate input filter duration in relation with the signal connected to the timer (when the input is one of the tim_tix (ICxF bits in the TIMx_CCMRx register). Let's imagine that, when toggling, the input signal is not stable during at least 5 internal clock cycles. The user must program a filter duration longer than these five clock cycles. The user can validate a transition on tim_ti1 when eight consecutive samples with the new level have been detected (sampled at f_{DTS} frequency). Then write IC1F bits to 0011 in the TIMx_CCMR1 register.
4. Select the edge of the active transition on the tim_ti1 channel by writing CC1P bit to 0 in the TIMx_CCER register (rising edge in this case).
5. Program the input prescaler. In this example, the user wants the capture to be performed at each valid transition, so the prescaler is disabled (write IC1PS bits to 00 in the TIMx_CCMR1 register).
6. Enable capture from the counter into the capture register by setting the CC1E bit in the TIMx_CCER register.
7. If needed, enable the related interrupt request by setting the CC1IE bit in the TIMx_DIER register, and/or the DMA request by setting the CC1DE bit in the TIMx_DIER register.

When an input capture occurs:

- The TIMx_CCR1 register gets the value of the counter on the active transition.
- CC1IF flag is set (interrupt flag). CC1OF is also set if at least two consecutive captures occurred whereas the flag was not cleared.
- An interrupt is generated depending on the CC1IE bit.
- A DMA request is generated depending on the CC1DE bit.

In order to handle the overcapture, it is recommended to read the data before the overcapture flag. This is to avoid missing an overcapture which may happen after reading the flag and before reading the data.

Note: IC interrupt and/or DMA requests can be generated by software by setting the corresponding CCxG bit in the TIMx_EGR register.

57.4.9 PWM input mode (only for TIM15)

This mode is used to measure both the period and the duty cycle of a PWM signal connected to single tim_tix input:

- The TIMx_CCR1 register holds the period value (interval between two consecutive rising edges)
- The TIMx_CCR2 register holds the pulse width (interval between two consecutive rising and falling edges)

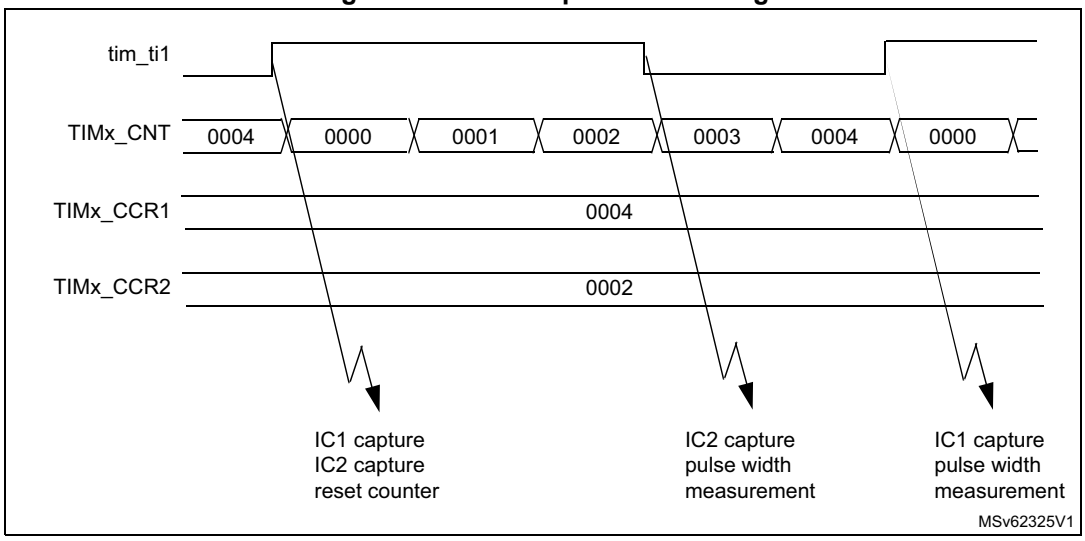
This mode is a particular case of input capture mode. The set-up procedure is similar with the following differences:

- Two tim_icx signals are mapped on the same tim_tix input.
- These two tim_icx signals are active on edges with opposite polarity.
- One of the two tim_tixfpy signals is selected as trigger input and the slave mode controller is configured in reset mode.

For example, one can measure the period (in TIMx_CCR1 register) and the duty cycle (in TIMx_CCR2 register) of the PWM applied on tim_ti1 using the following procedure (depending on tim_ker_ck frequency and prescaler value):

1. Select the proper tim_ti1_in[15:0] source (internal or external) with the TI1SEL[3:0] bits in the TIMx_TISEL register.
2. Select the active input for TIMx_CCR1: write the CC1S bits to 01 in the TIMx_CCMR1 register (tim_ti1 selected).
3. Select the active polarity for tim_ti1fp1 (used both for capture in TIMx_CCR1 and counter clear): write the CC1P and CC1NP bits to 0 (active on rising edge).
4. Select the active input for TIMx_CCR2: write the CC2S bits to 10 in the TIMx_CCMR1 register (tim_ti1 selected).
5. Select the active polarity for tim_ti1fp2 (used for capture in TIMx_CCR2): write the CC2P and CC2NP bits to 10 (active on falling edge).
6. Select the valid trigger input: write the TS bits to 00101 in the TIMx_SMCR register (tim_ti1fp1 selected).
7. Configure the slave mode controller in reset mode: write the SMS bits to 100 in the TIMx_SMCR register.
8. Enable the captures: write the CC1E and CC2E bits to 1 in the TIMx_CCER register.

Figure 734. PWM input mode timing



1. The PWM input mode can be used only with the TIMx_CH1/TIMx_CH2 signals due to the fact that only tim_ti1fp1 and tim_ti2fp2 are connected to the slave mode controller.

57.4.10 Forced output mode

In output mode (CCxS bits = 00 in the TIMx_CCMRx register), each output compare signal (tim_ocxref and then tim_ocx/tim_ocxn) can be forced to active or inactive level directly by software, independently of any comparison between the output compare register and the counter.

To force an output compare signal (tim_ocxref/tim_ocx) to its active level, one just needs to write 101 in the OCxM bits in the corresponding TIMx_CCMRx register. Thus tim_ocxref is forced high (tim_ocxref is always active high) and tim_ocx get opposite value to CCxP polarity bit.

For example: CCxP = 0 (tim_ocx active high) → tim_ocx is forced to high level.

The tim_ocxref signal can be forced low by writing the OCxM bits to 100 in the TIMx_CCMRx register.

Anyway, the comparison between the TIMx_CCRx shadow register and the counter is still performed and allows the flag to be set. Interrupt and DMA requests can be sent accordingly. This is described in the output compare mode section below.

57.4.11 Output compare mode

This function is used to control an output waveform or indicating when a period of time has elapsed.

When a match is found between the capture/compare register and the counter, the output compare function:

- Assigns the corresponding output pin to a programmable value defined by the output compare mode (OCxM bits in the TIMx_CCMRx register) and the output polarity (CCxP bit in the TIMx_CCER register). The output pin can keep its level (OCXM = 000), be set

active (OCxM = 001), be set inactive (OCxM = 010) or can toggle (OCxM = 011) on match.

- Sets a flag in the interrupt status register (CCxIF bit in the TIMx_SR register).
- Generates an interrupt if the corresponding interrupt mask is set (CCXIE bit in the TIMx_DIER register).
- Sends a DMA request if the corresponding enable bit is set (CCxDE bit in the TIMx_DIER register, CCDS bit in the TIMx_CR2 register for the DMA request selection).

The TIMx_CCRx registers can be programmed with or without preload registers using the OCxPE bit in the TIMx_CCMRx register.

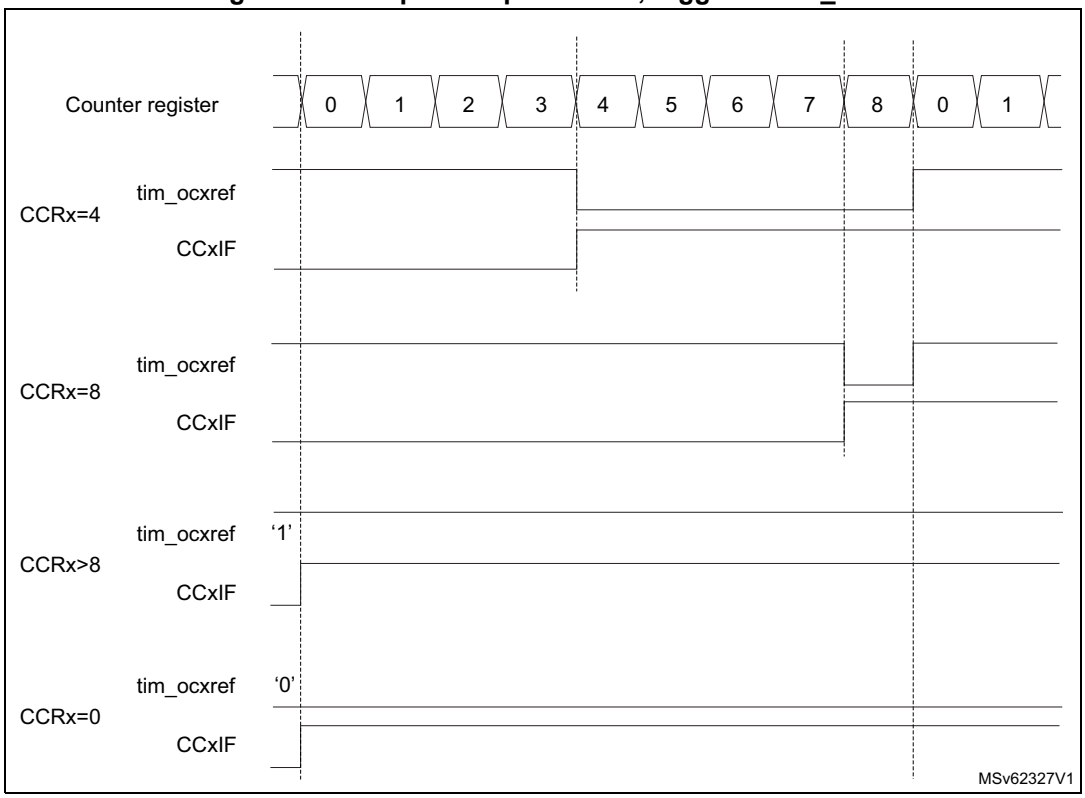
In output compare mode, the update event UEV has no effect on tim_ocxref and tim_ocx output. The timing resolution is one count of the counter. Output compare mode can also be used to output a single pulse (in One-pulse mode).

Procedure

1. Select the counter clock (internal, external, prescaler).
2. Write the desired data in the TIMx_ARR and TIMx_CCRx registers.
3. Set the CCxIE bit if an interrupt request is to be generated.
4. Select the output mode. For example:
   - Write OCxM = 011 to toggle tim_ocx output pin when CNT matches CCRx
   - Write OCxPE = 0 to disable preload register
   - Write CCxP = 0 to select active high polarity
   - Write CCxE = 1 to enable the output
5. Enable the counter by setting the CEN bit in the TIMx_CR1 register.

The TIMx_CCRx register can be updated at any time by software to control the output waveform, provided that the preload register is not enabled (OCxPE = 0, else TIMx_CCRx shadow register is updated only at the next update event UEV). An example is given in *Figure 735*.

Figure 735. Output compare mode, toggle on tim_oc1



57.4.12 PWM mode

Pulse width modulation mode is used to generate a signal with a frequency determined by the value of the TIMx_ARR register and a duty cycle determined by the value of the TIMx_CCRx register.

The PWM mode can be selected independently on each channel (one PWM per tim_ocx output) by writing 110 (PWM mode 1) or 111 (PWM mode 2) in the OCxM bits in the TIMx_CCMRx register. The corresponding preload register must be enabled by setting the OCxPE bit in the TIMx_CCMRx register, and eventually the autoreload preload register (in upcounting or center-aligned modes) by setting the ARPE bit in the TIMx_CR1 register.

As the preload registers are transferred to the shadow registers only when an update event occurs, before starting the counter, all registers must be initialized by setting the UG bit in the TIMx_EGR register.

tim_ocx polarity is software programmable using the CCxP bit in the TIMx_CCER register. It can be programmed as active high or active low. tim_ocx output is enabled by a combination of the CCxE, CCxNE, MOE, OSSI, and OSSR bits (TIMx_CCER and TIMx_BDTR registers). Refer to the TIMx_CCER register description for more details.

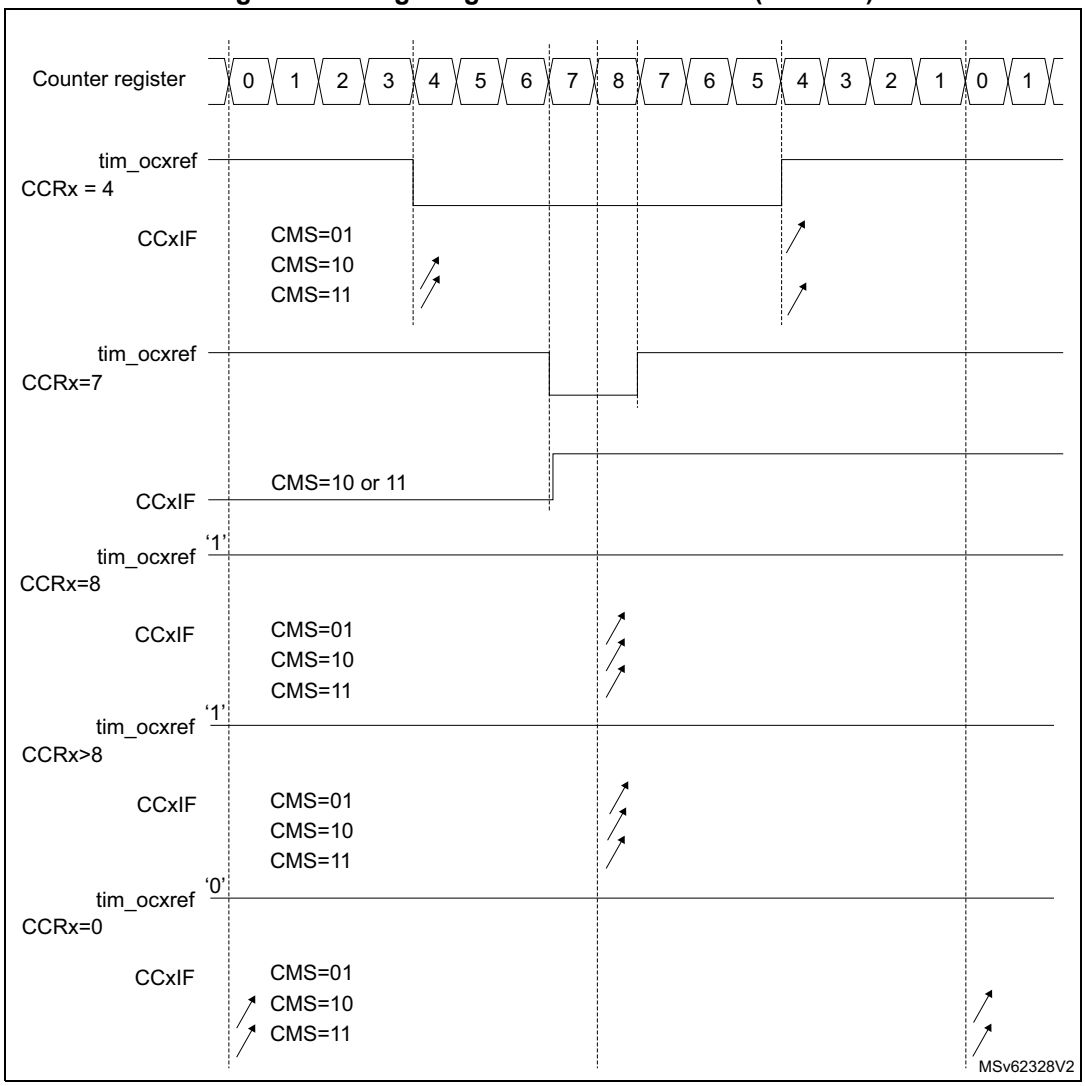
In PWM mode (1 or 2), TIMx_CNT and TIMx_CCRx are always compared to determine whether TIMx_CCRx ≤ TIMx_CNT or TIMx_CNT ≤ TIMx_CCRx (depending on the direction of the counter).

The TIM15/TIM16/TIM17 are capable of upcounting only. Refer to *Upcounting mode*.

In the following example applies to PWM mode 1. The reference PWM signal tim_ocxref is high as long as TIMx_CNT < TIMx_CCRx else it becomes low. If the compare value in

TIMx_CCRx is greater than the autoreload value (in TIMx_ARR) then tim_ocxref is held at 1. If the compare value is 0 then tim_ocxref is held at 0. *Figure 736* shows some edge-aligned PWM waveforms in an example where TIMx_ARR = 8.

Figure 736. Edge-aligned PWM waveforms (ARR = 8)

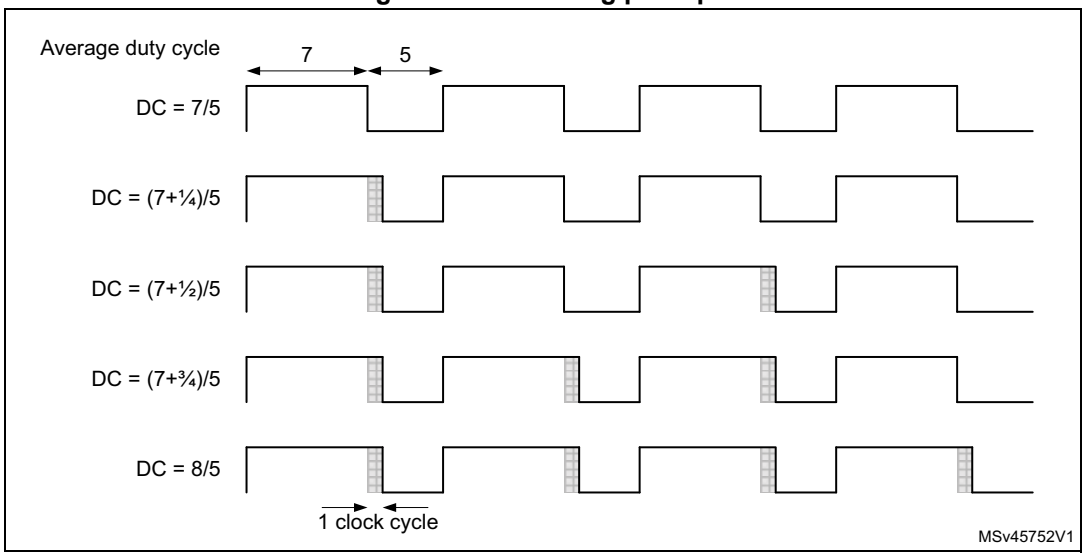


Dithering mode

The PWM mode effective resolution can be increased by enabling the dithering mode, using the DITHEN bit in the TIMx_CR1 register. This applies to both the CCR (for duty cycle resolution increase) and ARR (for PWM frequency resolution increase).

The operating principle is to have the actual CCR (or ARR) value slightly changed (adding or not one timer clock period) over 16 consecutive PWM periods, with predefined patterns. This allows a 16-fold resolution increase, considering the average duty cycle or PWM period. The *Figure 737* below presents the dithering principle applied to four consecutive PWM cycles.

Figure 737. Dithering principle

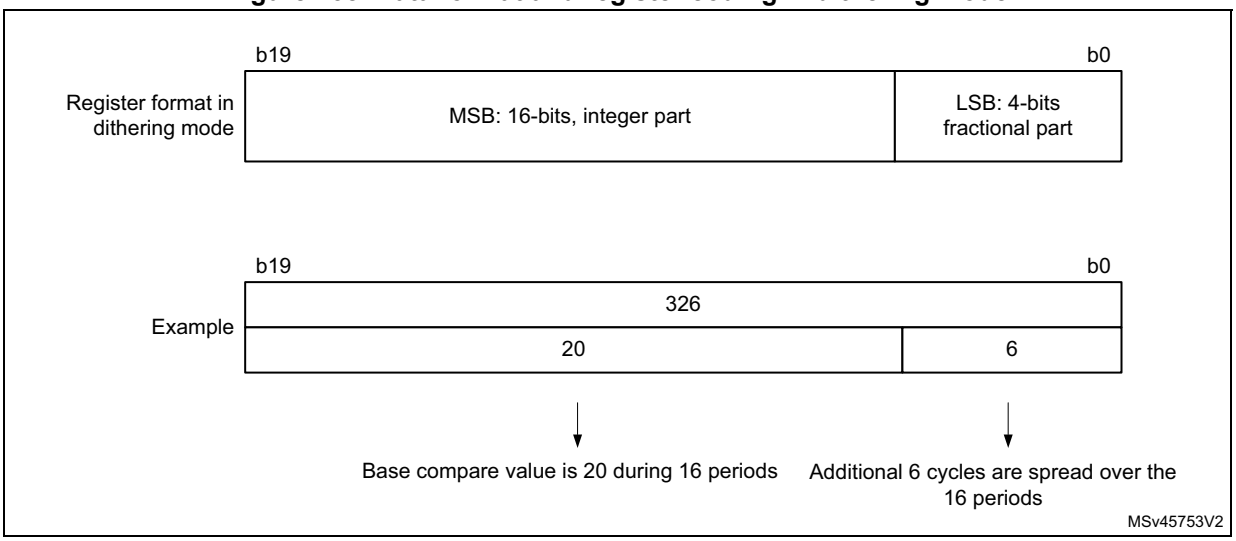


When the dithering mode is enabled, the register coding is changed as follows (see *Figure 738* for example):

- The four LSBs are coding for the enhanced resolution part (fractional part).
- The MSBs are left-shifted to the bits 19:4 and are coding for the base value.

Note: The following sequence must be followed when resetting the DITHEN bit:
1. CEN and ARPE bits must be reset
2. The ARR[3:0] bits must be reset
3. The CCIF flags must be cleared
4. The CEN bit can be set (eventually with ARPE = 1).

Figure 738. Data format and register coding in dithering mode



The minimum frequency is given by the following formula:

$$\text{Resolution} = \frac{F_{Tim}}{F_{pwm}} \Rightarrow F_{pwmMin} = \frac{F_{Tim}}{Max_{Resolution}}$$

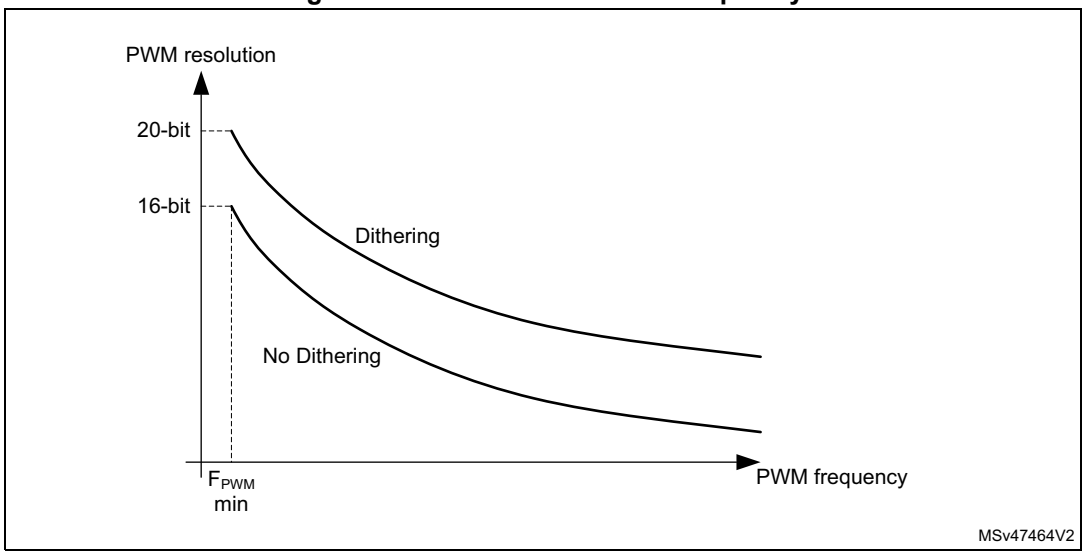
$$\text{Dithering mode disabled: } F_{pwmMin} = \frac{F_{Tim}}{65536}$$

$$\text{Dithering mode enabled: } F_{pwmMin} = \frac{F_{Tim}}{65535 + \frac{15}{16}}$$

Note: The maximum TIMx_ARR and TIMx_CCRy values are limited to 0xFFFEF in dithering mode (corresponds to 65534 for the integer part and 15 for the dithered part).

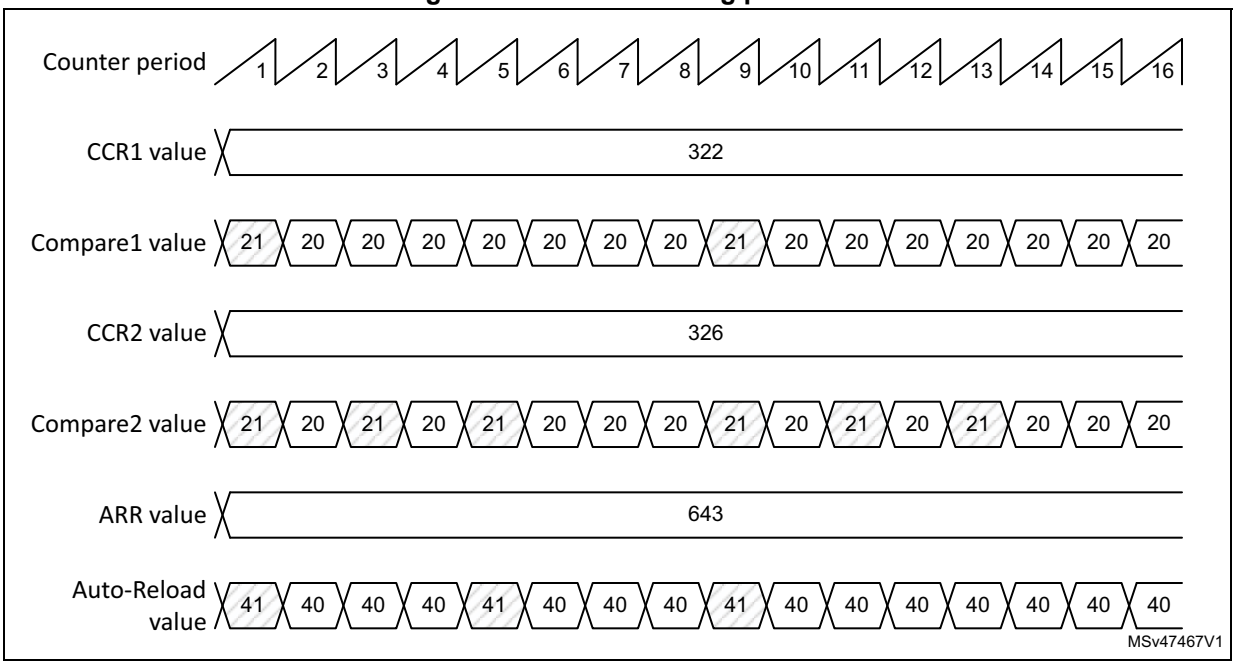
As shown on the *Figure 739* below, the dithering mode is used to increase the PWM resolution whatever the PWM frequency.

Figure 739. PWM resolution vs frequency



The duty cycle and/or period changes are spread over 16 consecutive periods, as described in the *Figure 740* below.

Figure 740. PWM dithering pattern



The autoreload and compare values increments are spread following specific patterns described in the *Table 545* below. The dithering sequence is done to have increments distributed as evenly as possible and minimize the overall ripple.

Table 545. CCR and ARR register change dithering pattern

-	PWM period															
LSB value	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
0000	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0001	+1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
0010	+1	-	-	-	-	-	-	-	+1	-	-	-	-	-	-	-
0011	+1	-	-	-	+1	-	-	-	+1	-	-	-	-	-	-	-
0100	+1	-	-	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0101	+1	-	+1	-	+1	-	-	-	+1	-	-	-	+1	-	-	-
0110	+1	-	+1	-	+1	-	-	-	+1	-	+1	-	+1	-	-	-
0111	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	-	-
1000	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1001	+1	+1	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-	+1	-
1010	+1	+1	+1	-	+1	-	+1	-	+1	+1	+1	-	+1	-	+1	-
1011	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	-	+1	-
1100	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-
1101	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	-	+1	+1	+1	-

Table 545. CCR and ARR register change dithering pattern (continued)

-	PWM period															
LSB value	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
1110	+1	+1	+1	+1	+1	+1	+1	-	+1	+1	+1	+1	+1	+1	+1	-
1111	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	+1	-

57.4.13 Combined PWM mode (TIM15 only)

Combined PWM mode allows two edge or center-aligned PWM signals to be generated with programmable delay and phase shift between respective pulses. While the frequency is determined by the value of the TIMx_ARR register, the duty cycle and delay are determined by the two TIMx_CCRx registers. The resulting signals, tim_ocxrefc, are made of an OR or AND logical combination of two reference PWMs:

- tim_oc1refc (or tim_oc2refc) is controlled by the TIMx_CCR1 and TIMx_CCR2 registers.

Combined PWM mode can be selected independently on two channels (one tim_ocx output per pair of CCR registers) by writing 1100 (Combined PWM mode 1) or 1101 (Combined PWM mode 2) in the OCxM bits in the TIMx_CCMRx register.

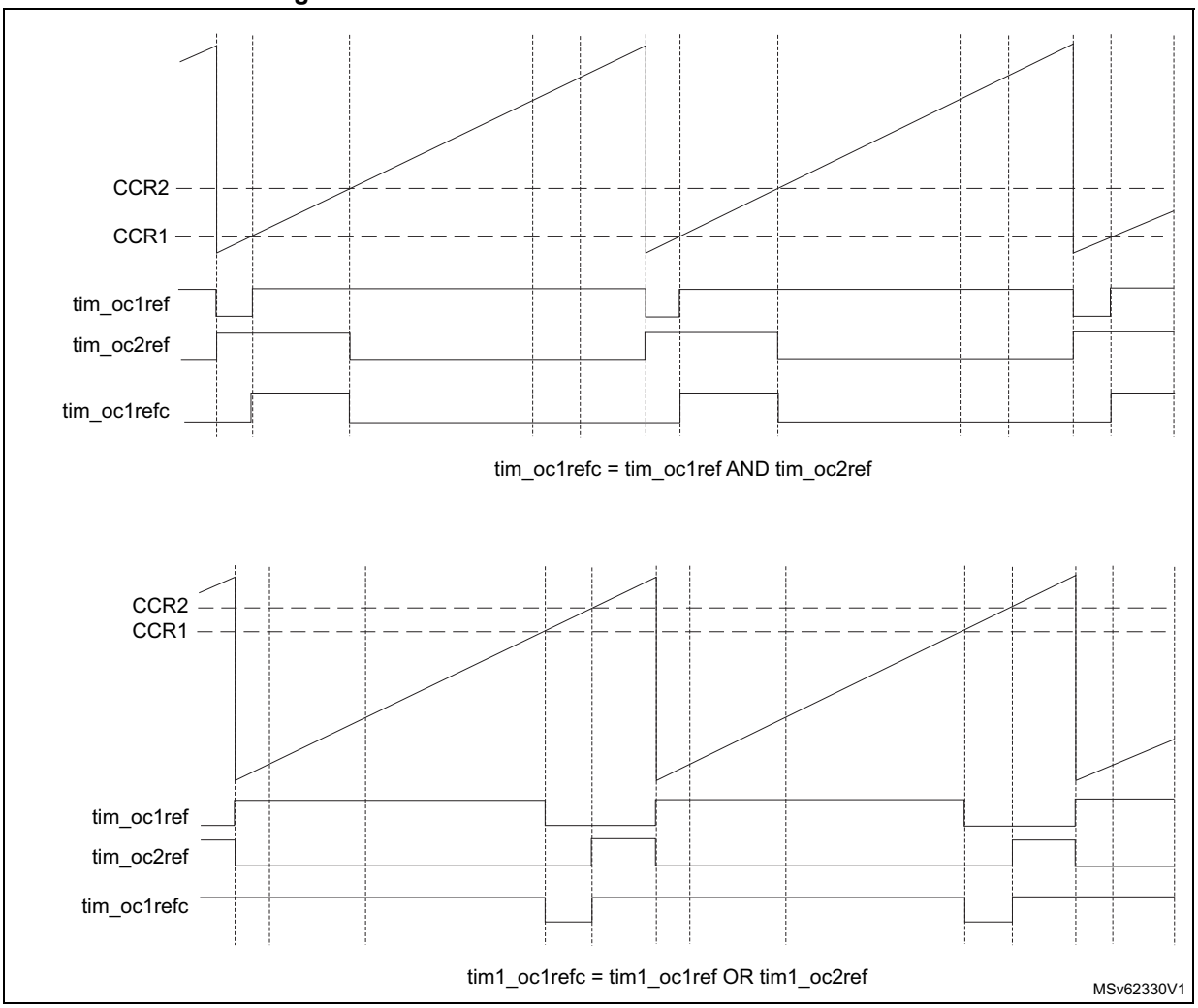
When a given channel is used as a combined PWM channel, its complementary channel must be configured in the opposite PWM mode (for instance, one in Combined PWM mode 1 and the other in Combined PWM mode 2).

Note: The OCxM[3:0] bitfield is split into two parts for compatibility reasons, the most significant bit is not contiguous with the three least significant ones.

Table 741 represents an example of signals that can be generated using Combined PWM mode, obtained with the following configuration:

- Channel 1 is configured in Combined PWM mode 2.
- Channel 2 is configured in PWM mode 1.

Figure 741. Combined PWM mode on channel 1 and 2



57.4.14 Complementary outputs and dead-time insertion

The TIM15/TIM16/TIM17 general-purpose timers can output one complementary signal and manage the switching-off and switching-on of the outputs.

This time is generally known as dead-time and it has to be adjusted depending on the devices that are connected to the outputs and their characteristics (such as intrinsic delays of level-shifters and delays due to power switches)

The polarity of the outputs (main output tim_ocx or complementary tim_ocxn) can be selected independently for each output. This is done by writing to the CCxP and CCxNP bits in the TIMx_CCER register.

The complementary signals tim_ocx and tim_ocxn are activated by a combination of several control bits: the CCxE and CCxNE bits in the TIMx_CCER register and the MOE, OISx, OISxN, OSSI and OSSR bits in the TIMx_BDTR and TIMx_CR2 registers. Refer to *Table 552: Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17)* for more details. In particular, the dead-time is activated when switching to the idle state (MOE falling down to 0).

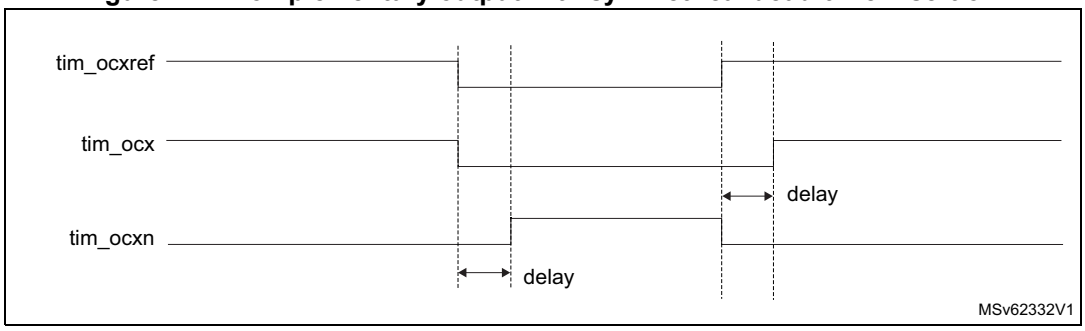
Dead-time insertion is enabled by setting both CCxE and CCxNE bits, and the MOE bit if the break circuit is present. There is one 10-bit dead-time generator for each channel. From a reference waveform tim_ocxref, it generates two outputs tim_ocx and tim_ocxn. If tim_ocx and tim_ocxn are active high:

- The tim_ocx output signal is the same as the reference signal except for the rising edge, which is delayed relative to the reference rising edge.
- The tim_ocxn output signal is the opposite of the reference signal except for the rising edge, which is delayed relative to the reference falling edge.

If the delay is greater than the width of the active output (tim_ocx or tim_ocxn) then the corresponding pulse is not generated.

The following figures show the relationships between the output signals of the dead-time generator and the reference signal tim_ocxref. (in these examples CCxP = 0, CCxNP = 0, MOE = 1, CCxE = 1 and CCxNE = 1)

Figure 742. Complementary output with symmetrical dead-time insertion.



The DTAE bit in the TIMx_DTR2 is used to differentiate the deadtime values for rising and falling edges of the reference signal, as shown on *Figure 743*.

In asymmetrical mode (DTAE = 1), the rising edge-referred deadtime is defined by the DTG[7:0] bitfield in the TIMx_BDTR register, while the falling edge-referred is defined by the DTGF[7:0] bitfield in the TIMx_DTR2 register. The DTAE bit must be written before enabling the counter and must not be modified while CEN = 1.

It is possible to have the deadtime value updated on-the-fly during pwm operation, using a preload mechanism. The deadtime bitfield DTG[7:0] and DTGF[7:0] are preloaded when the DTPE bit is set in the TIMX_DTR2 register. The preload value is loaded in the active register on the next update event.

Note: If the DTPE bit is enabled while the counter is enabled, any new value written since last update is discarded and previous value is used.

Figure 743. Asymmetrical deadtime

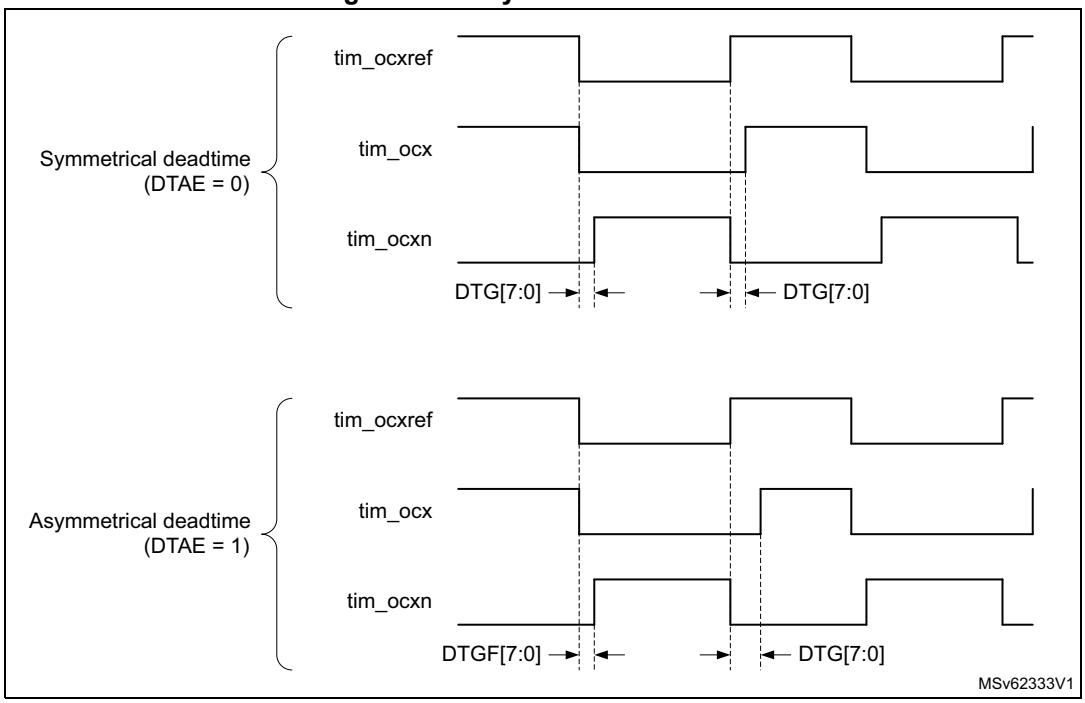


Figure 744. Dead-time waveforms with delay greater than the negative pulse.

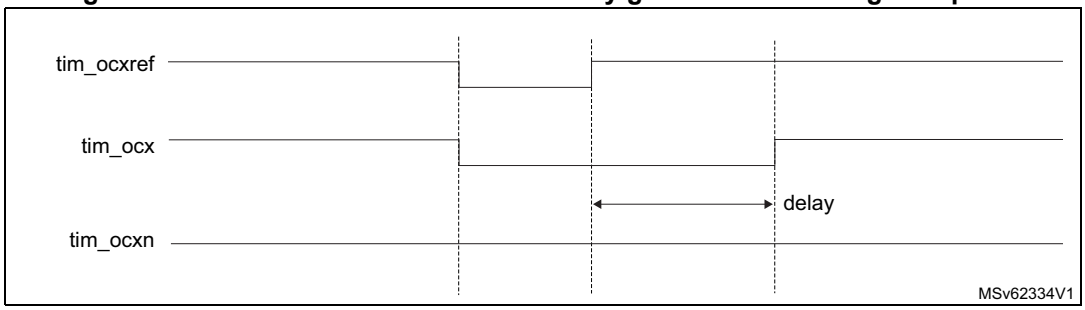
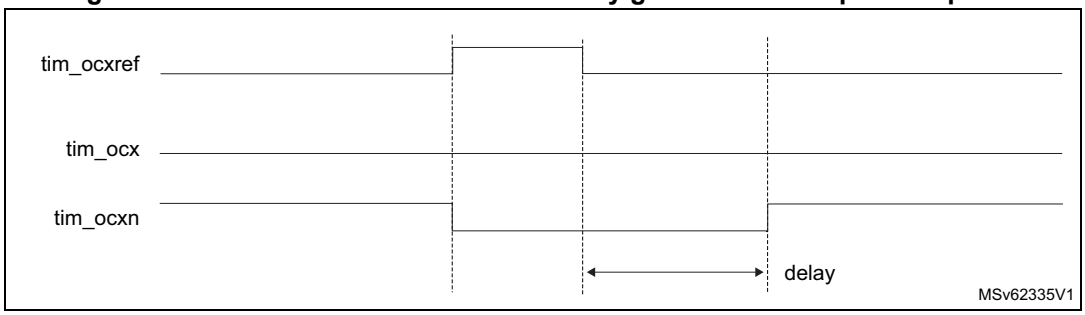


Figure 745. Dead-time waveforms with delay greater than the positive pulse.



The dead-time delay is the same for each of the channels and is programmable with the DTG bits in the TIMx_BDTR register. Refer to *Section 57.8.14: TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17)* for delay calculation.

Redirecting tim_ocxref to tim_ocx or tim_ocxn

In output mode (forced, output compare or PWM), tim_ocxref can be redirected to the tim_ocx output or to tim_ocxn output by configuring the CCxE and CCxNE bits in the TIMx_CCER register.

This is used to send a specific waveform (such as PWM or static active level) on one output while the complementary remains at its inactive level. Other alternative possibilities are to have both outputs at inactive level or both outputs active and complementary with dead-time.

Note: When only tim_ocxn is enabled (CCxE = 0, CCxNE = 1), it is not complemented and becomes active as soon as tim_ocxref is high. For example, if CCxNP = 0 then tim_ocxn = tim_ocxref. On the other hand, when both tim_ocx and tim_ocxn are enabled (CCxE = CCxNE = 1) tim_ocx becomes active when tim_ocxref is high whereas tim_ocxn is complemented and becomes active when tim_ocxref is low.

57.4.15 Using the break function

The purpose of the break function is to protect power switches driven by PWM signals generated with the timers. The break input is usually connected to fault outputs of power stages and 3-phase inverters. When activated, the break circuitry shuts down the PWM outputs and forces them to a predefined safe state.

The break channel gathers both system-level fault (such as clock failure, ECC/parity, and errors) and application fault (from input pins and built-in comparator), and can force the outputs to a predefined level (either active or inactive) after a deadtime duration.

The output enable signal and output levels during break are depending on several control bits:

- The MOE bit in TIMx_BDTR register is used to enable /disable the outputs by software and is reset in case of break or break2 event.
- The OSSI bit in the TIMx_BDTR register defines whether the timer controls the output in inactive state or releases the control to the GPIO controller (typically to have it in Hi-Z mode).
- The OISx and OISxN bits in the TIMx_CR2 register which are setting the output shut-down level, either active or inactive. The tim_ocx and tim_ocxn outputs cannot be set both to active level at a given time, whatever the OISx and OISxN values. Refer to *Table 552: Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17)* for more details.

When exiting from reset, the break circuit is disabled and the MOE bit is low. The break function is enabled by setting the BKE bit in the TIMx_BDTR register. The break input polarity can be selected by configuring the BKP bit in the same register. BKE and BKP can be modified at the same time. When the BKE and BKP bits are written, a delay of one APB clock cycle is applied before the writing is effective. Consequently, it is necessary to wait one APB clock period to correctly read back the bit after the write operation.

Because MOE falling edge can be asynchronous, a resynchronization circuit has been inserted between the actual signal (acting on the outputs) and the synchronous control bit (accessed in the TIMx_BDTR register). It results in some delays between the asynchronous and the synchronous signals. In particular, if MOE is set to 1 whereas it was low, a delay must be inserted (dummy instruction) before reading it correctly. This is because the write acts on the asynchronous signal whereas the read reflects the synchronous signal.

The break is generated by the tim_brk inputs which have:

- Programmable polarity (BKP bit in the TIMx_BDTR register).
- Programmable enable bit (BKE bit in the TIMx_BDTR register).
- Programmable filter (BKF[3:0] bits in the TIMx_BDTR register) to avoid spurious events.

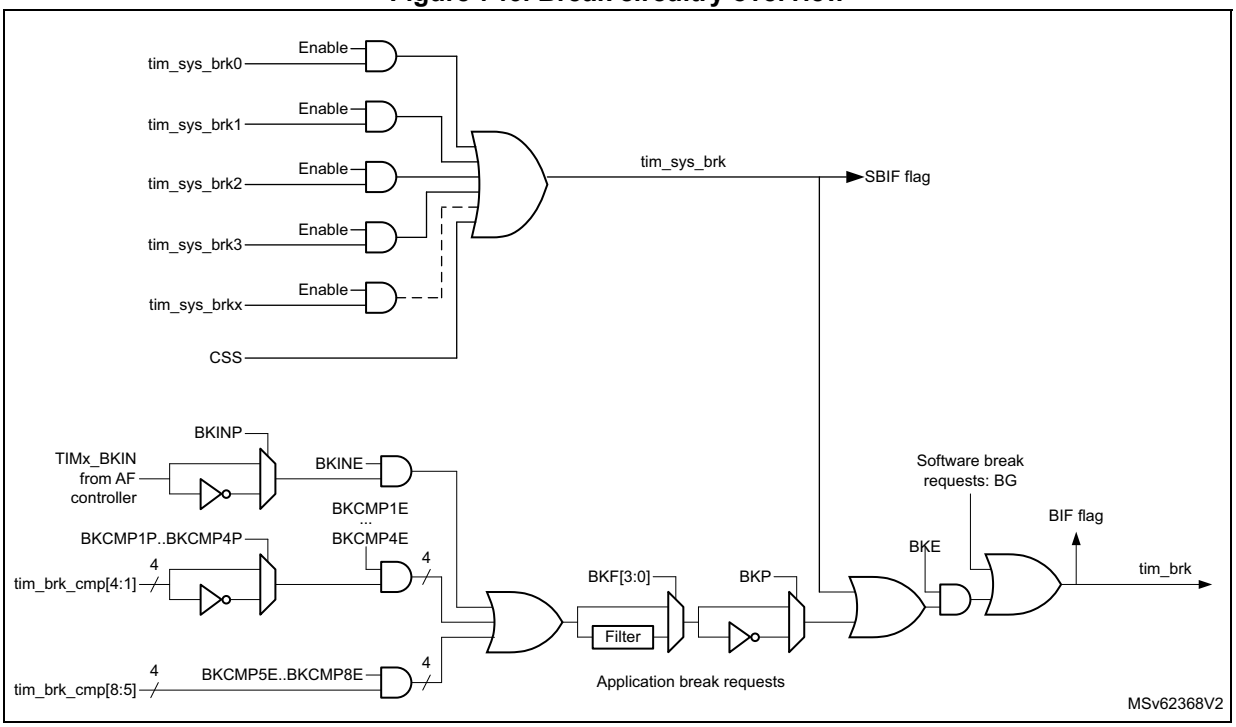
The break can be generated from multiple sources which can be individually enabled and with programmable edge sensitivity, using the TIMx_AF1 register.

The sources for break (tim_brk) channel are:

- External sources connected to one of the TIM_BKIN pins (as per selection done in the GPIO alternate function selection registers), with polarity selection and optional digital filtering.
- Internal sources:
  - Coming from a tim_brk_cmpx input (refer to *Section 57.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation).
  - Coming from a system break request on the tim_sys_brk inputs (refer to *Section 57.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation).

Break events can also be generated by software using BG bit in the TIMx_EGR register. All sources are ORed before entering the timer tim_brk inputs, as per *Figure 746* below.

Figure 746. Break circuitry overview



Caution: An asynchronous (clockless) operation is only guaranteed when the programmable filter is disabled. If it is enabled, a fail-safe clock mode (for example, using the internal PLL and/or the CSS) must be used to guarantee that break events are handled.

When a break occurs (selected level on the break input):

- The MOE bit is cleared asynchronously, putting the outputs in inactive state, idle state, or even releasing the control to the GPIO (selected by the OSSI bit). This feature functions even if the MCU oscillator is off.
- Each output channel is driven with the level programmed in the OISx bit in the TIMx_CR2 register as soon as MOE = 0. If OSSI = 0, the timer releases the output control (taken over by the GPIO) else the enable output remains high.
- When complementary outputs are used:
  - The outputs are first put in reset state inactive state (depending on the polarity). This is done asynchronously so that it works even if no clock is provided to the timer.
  - If the timer clock is still present, then the dead-time generator is reactivated in order to drive the outputs with the level programmed in the OISx and OISxN bits after a dead-time. Even in this case, tim_ocx and tim_ocxn cannot be driven to their active level together. Note that because of the resynchronization on MOE, the dead-time duration is a bit longer than usual (around 2 tim_ker_ck clock cycles).
  - If OSSI = 0 then the timer releases the enable outputs (taken over by the GPIO which forces a Hi-Z state) else the enable outputs remain or become high as soon as one of the CCxE or CCxNE bits is high.
- The break status flag (BIF bit in the TIMx_SR register) is set. An interrupt can be generated if the BIE bit in the TIMx_DIER register is set. A DMA request can be sent if the BDE bit in the TIMx_DIER register is set.
- If the AOE bit in the TIMx_BDTR register is set, the MOE bit is automatically set again at the next update event UEV. This can be used to perform a regulation, for instance. Else, MOE remains low until it is written with 1 again. In this case, it can be used for security and the break input can be connected to an alarm from power drivers, thermal sensors or any security components.

Note: If the MOE is reset by the CPU while the AOE bit is set, the outputs are in idle state and forced to inactive level or Hi-Z depending on OSSI value. If both the MOE and AOE bits are reset by the CPU, the outputs are in disabled state and driven with the level programmed in the OISx bit in the TIMx_CR2 register.

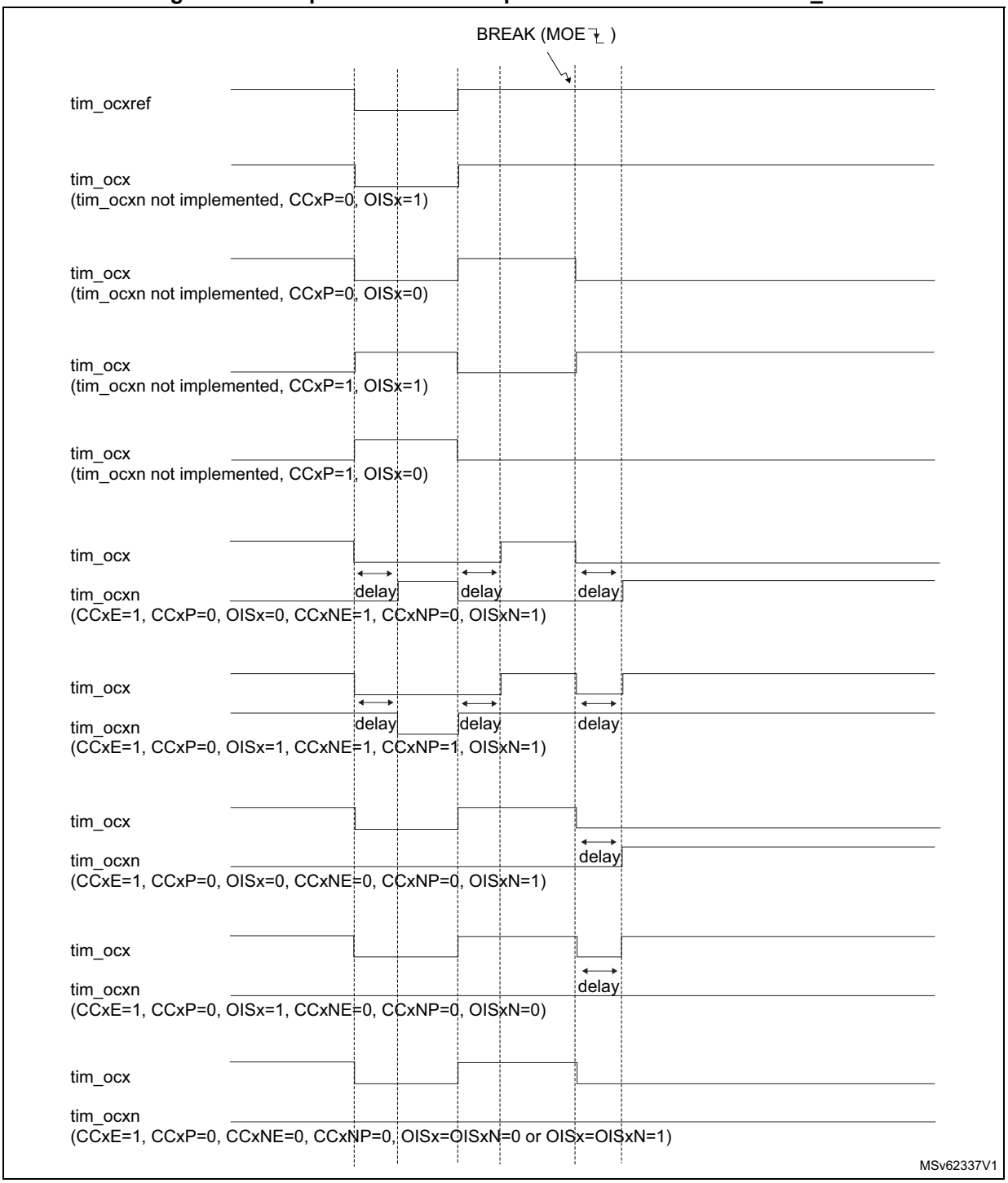
The break inputs are acting on level. Thus, the MOE cannot be set while the break input is active (neither automatically nor by software). In the meantime, the status flag BIF cannot be cleared.

The break can be generated by the tim_brk input which has a programmable polarity and an enable bit BKE in the TIMx_BDTR register.

In addition to the break input and the output management, a write protection has been implemented inside the break circuit to safeguard the application. It is used to freeze the configuration of several parameters (dead-time duration, tim_ocx/tim_ocxn polarities and state when disabled, OCxM configurations, break enable, and polarity). The protection can be selected among 3 levels with the LOCK bits in the TIMx_BDTR register. Refer to *Section 57.8.14: TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17)*. The LOCK bits can be written only once after an MCU reset.

The *Figure 747* shows an example of behavior for the outputs in response to a break.

Figure 747. Output behavior in response to a break event on tim_brk



57.4.16 Bidirectional break input

The TIM15/TIM16/TIM17 are featuring bidirectional break I/Os, as represented on *Figure 748*.

They are used to have:

- A board-level global break signal available for signaling faults to external MCUs or gate drivers, with a unique pin being both an input and an output status pin.
- Internal break sources and multiple external open drain sources ORed together to trigger a unique break event, when multiple internal and external break sources must be merged.

The tim_brk input is configured in bidirectional mode using the BKBID bit in the TIMx_BDTR register. The BKBID programming bit can be locked in read-only mode using the LOCK bits in the TIMx_BDTR register (in LOCK level 1 or above).

The bidirectional mode requires the I/O to be configured in open-drain mode with active low polarity (using BKINP and BKP bits). Any break request coming either from system (for example CSS), from on-chip peripherals, or from break inputs forces a low level on the break input to signal the fault event. The bidirectional mode is inhibited if the polarity bits are not correctly set (active high polarity), for safety purposes.

The break software event (triggered by setting the BG bit) also causes the break I/O to be forced to '0' to indicate to the external components that the timer has entered in break state. However, this is valid only if the break is enabled (BKE = 1). When a software break event is generated with BKE = 0, the outputs are put in safe state and the break flag is set, but there is no effect on the TIM_BKIN I/O.

A safe disarming mechanism prevents the system to be definitively locked-up (a low level on the break input triggers a break which enforces a low level on the same input).

When the BKDSRM bit is set to 1, this releases the break output to clear a fault signal and to give the possibility to re-arm the system.

At no point the break protection circuitry can be disabled:

- The break input path is always active: a break event is active even if the BKDSRM bit is set and the open drain control is released. This prevents the PWM output to be restarted as long as the break condition is present.
- The BKDSRM bit cannot disarm the break protection as long as the outputs are enabled (MOE bit is set) (see *Table 546*).

Table 546. Break protection disarming conditions

MOE	BKBID	BKDSRM	Break protection state
0	0	X	Armed
0	1	0	Armed
0	1	1	Disarmed
1	X	X	Armed

Arming and rearming break circuitry

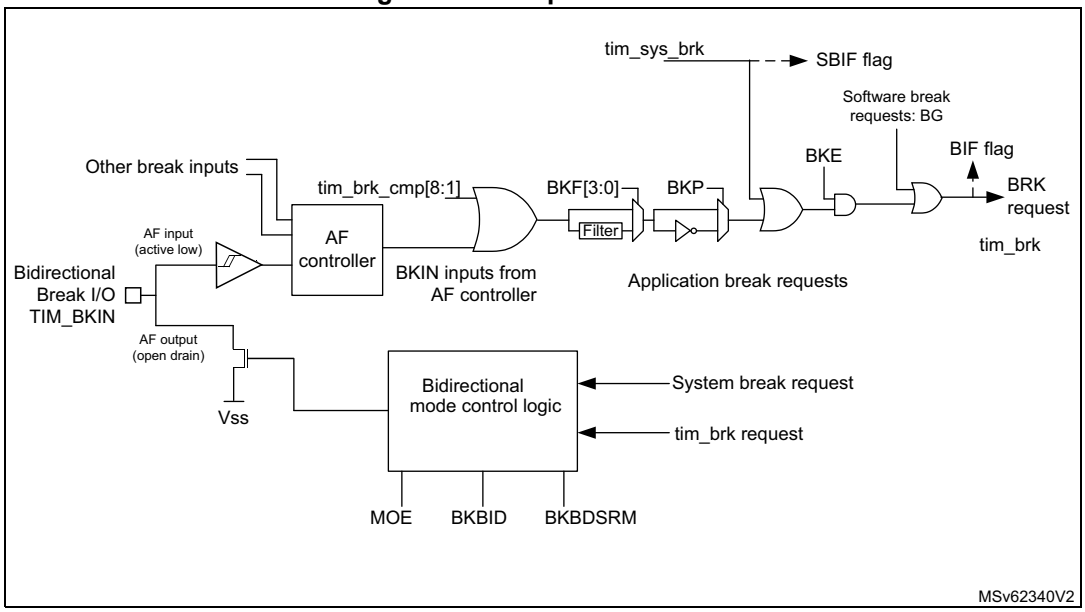
The break circuitry (in input or bidirectional mode) is armed by default (peripheral reset configuration).

The following procedure must be followed to re-arm the protection after a break event:

- The BKDSRM bit must be set to release the output control.
- The software must wait until the system break condition disappears (if any) and clear the SBIF status flag (or clear it systematically before rearming).
- The software must poll the BKDSRM bit until it is cleared by hardware (when the application break condition disappears).

From this point, the break circuitry is armed and active, and the MOE bit can be set to re-enable the PWM outputs.

Figure 748. Output redirection

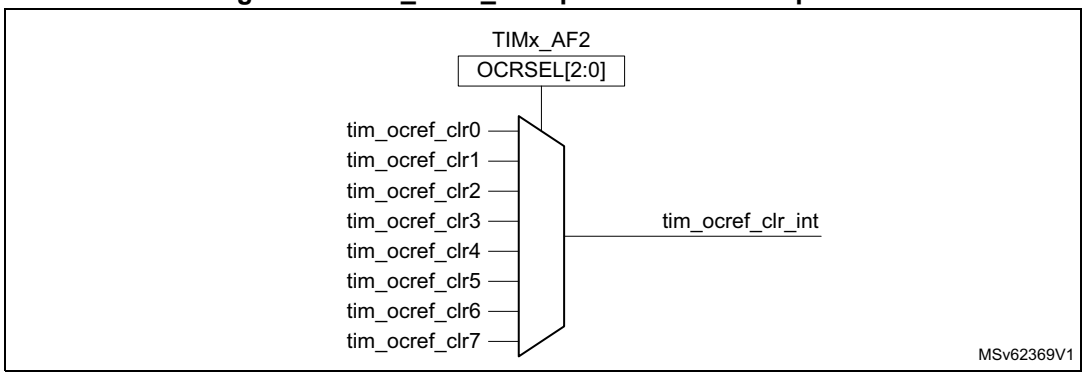


57.4.17 Clearing the tim_ocxref signal on an external event

The tim_ocxref signal of a given channel can be cleared when a high level is applied on the tim_ocref_clr_int input (OCxCE enable bit in the corresponding TIMx_CCMRx register set to 1). tim_ocxref remains low until the next transition to the active state, on the following PWM cycle. This function can only be used in Output compare and PWM modes. It does not work in Forced mode.

The tim_ocref_clr_int input can be selected among several inputs, as shown on *Figure 509* below.

Figure 749. tim_ocref_clr input selection multiplexer



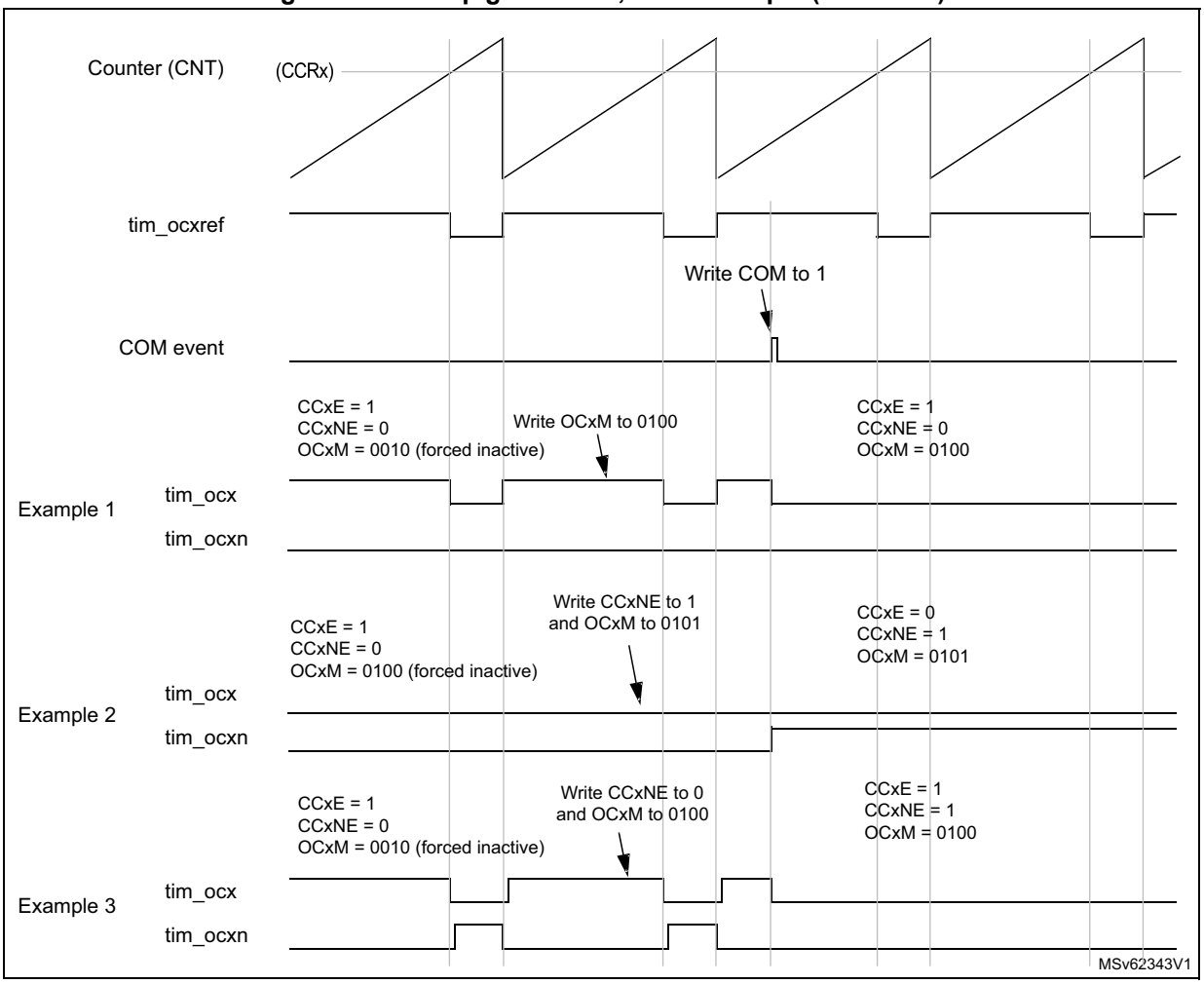
57.4.18 6-step PWM generation

When complementary outputs are used on a channel, preload bits are available on the OCxM, CCxE, and CCxNE bits. The preload bits are transferred to the shadow bits at the COM commutation event. Thus one can program in advance the configuration for the next step and change the configuration of all the channels at the same time. COM can be generated by software by setting the COM bit in the TIMx_EGR register or by hardware (on tim_trgi rising edge).

A flag is set when the COM event occurs (COMIF bit in the TIMx_SR register), which can generate an interrupt (if the COMIE bit is set in the TIMx_DIER register) or a DMA request (if the COMDE bit is set in the TIMx_DIER register).

The *Figure 750* describes the behavior of the tim_ocx and tim_ocxn outputs when a COM event occurs, in 3 different examples of programmed configurations.

Figure 750. 6-step generation, COM example (OSSR = 1)



57.4.19 One-pulse mode

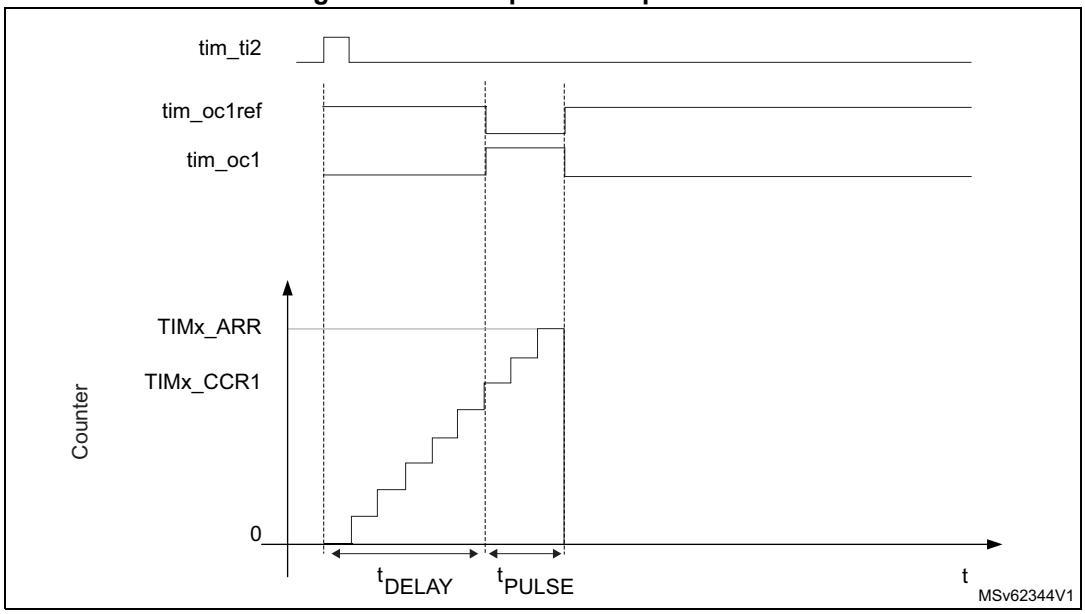
One-pulse mode (OPM) is a particular case of the previous modes. It allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length after a programmable delay.

Starting the counter can be controlled through the slave mode controller. Generating the waveform can be done in output compare mode or PWM mode. One-pulse mode is selected by setting the OPM bit in the TIMx_CR1 register. This makes the counter stop automatically at the next update event UEV.

A pulse can be correctly generated only if the compare value is different from the counter initial value. Before starting (when the timer is waiting for the trigger), the configuration must be:

- CNT < CCRx ≤ ARR (in particular, 0 < CCRx).

Figure 751. Example of one pulse mode.



For example one may want to generate a positive pulse on tim_oc1 with a length of t_{PULSE} and after a delay of t_{DELAY} as soon as a positive edge is detected on the tim_ti2 input pin.

Let's use tim_ti2fp2 as trigger 1:

1. Select the proper tim_ti2_in[15:1] source (internal or external) with the TI2SEL[3:0] bits in the TIMx_TISEL register.
2. Map tim_ti2fp2 to tim_ti2 by writing CC2S = 01 in the TIMx_CCMR1 register.
3. tim_ti2fp2 must detect a rising edge, write CC2P = 0 and CC2NP = 0 in the TIMx_CCER register.
4. Configure tim_ti2fp2 as trigger for the slave mode controller (tim_trgi) by writing TS = 00110 in the TIMx_SMCR register.
5. tim_ti2fp2 is used to start the counter by writing SMS to 110 in the TIMx_SMCR register (trigger mode).

The OPM waveform is defined by writing the compare registers (taking into account the clock frequency and the counter prescaler).

- The t_{DELAY} is defined by the value written in the TIMx_CCR1 register.
- The t_{PULSE} is defined by the difference between the autoreload value and the compare value (TIMx_ARR - TIMx_CCR1).
- Let's say one want to build a waveform with a transition from 0 to 1 when a compare match occurs and a transition from 1 to 0 when the counter reaches the autoreload value. To do this PWM mode 2 must be enabled by writing OC1M = 111 in the TIMx_CCMR1 register. Optionally the preload registers can be enabled by writing OC1PE = 1 in the TIMx_CCMR1 register and ARPE in the TIMx_CR1 register. In this case one has to write the compare value in the TIMx_CCR1 register, the autoreload value in the TIMx_ARR register, generate an update by setting the UG bit and wait for external trigger event on tim_ti2. CC1P is written to 0 in this example.

Since only one pulse is needed, a 1 must be written in the OPM bit in the TIMx_CR1 register to stop the counter at the next update event (when the counter rolls over from the autoreload value back to 0).

Particular case: tim_ocx fast enable

In One-pulse mode, the edge detection on tim_tix input set the CEN bit which enables the counter. Then the comparison between the counter and the compare value makes the output toggle. But several clock cycles are needed for these operations and it limits the minimum delay t_{DELAY} min that can be obtained.

If one wants to output a waveform with the minimum delay, the OCxFE bit can be set in the TIMx_CCMRx register. Then tim_ocxref (and tim_ocx) are forced in response to the stimulus, without taking in account the comparison. Its new level is the same as if a compare match had occurred. OCxFE acts only if the channel is configured in PWM1 or PWM2 mode.

57.4.20 Retriggerable one pulse mode (TIM15 only)

This mode allows the counter to be started in response to a stimulus and to generate a pulse with a programmable length, but with the following differences with non-retriggerable one pulse mode described in *Section 57.4.19*:

- The pulse starts as soon as the trigger occurs (no programmable delay)
- The pulse is extended if a new trigger occurs before the previous one is completed

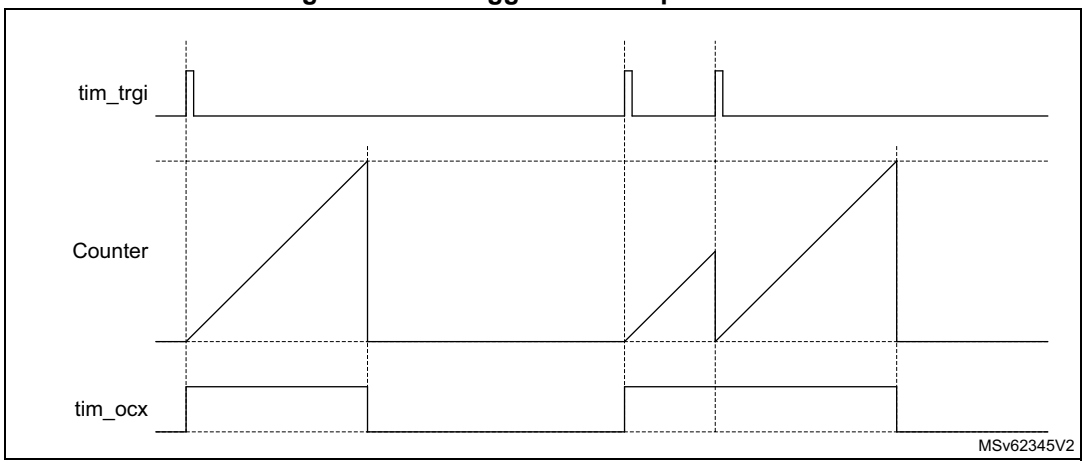
The timer must be in Slave mode, with the bits SMS[3:0] = 1000 (Combined reset + trigger mode) in the TIMx_SMCR register, and the OCxM[3:0] bits set to 1000 or 1001 for Retriggerable OPM mode 1 or 2.

If the timer is configured in Up-counting mode, the corresponding CCRx must be set to 0 (the ARR register sets the pulse length). If the timer is configured in Down-counting mode, CCRx must be above or equal to ARR.

Note: The OCxM[3:0] and SMS[3:0] bit fields are split into two parts for compatibility reasons, the most significant bit are not contiguous with the three least significant ones.

This mode must not be used with center-aligned PWM modes. It is mandatory to have CMS[1:0] = 00 in TIMx_CR1.

Figure 752. Retriggerable one pulse mode



57.4.21 UIF bit remapping

The IUFREMAP bit in the TIMx_CR1 register forces a continuous copy of the update interrupt flag UIF into bit 31 of the timer counter register (TIMx_CNT[31]). This is used to atomically read both the counter value and a potential roll-over condition signaled by the UIFCPY flag. In particular cases, it can ease the calculations by avoiding race conditions caused for instance by a processing shared between a background task (counter reading) and an interrupt (update interrupt).

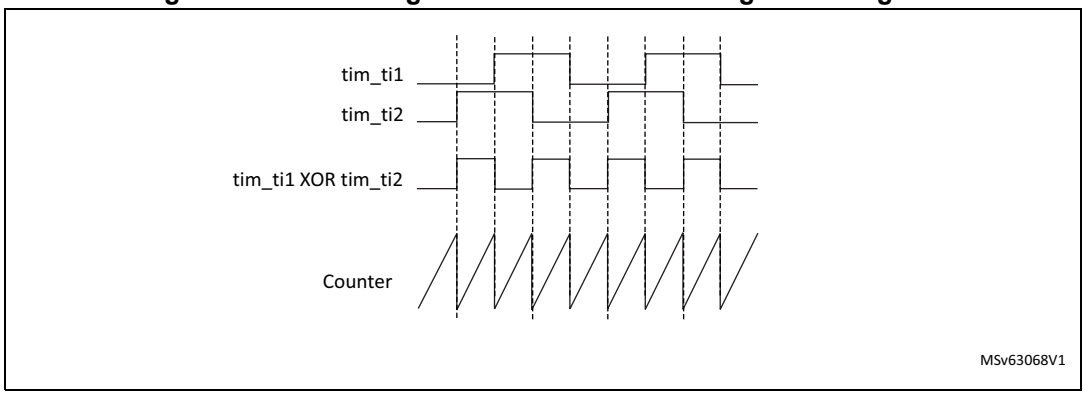
There is no latency between the assertions of the UIF and UIFCPY flags.

57.4.22 Timer input XOR function (TIM15 only)

The TI1S bit in the TIMx_CR2 register, allows the input filter of channel 1 to be connected to the output of a XOR gate, combining the two input pins tim_ti1 and tim_ti2.

The XOR output can be used with all the timer input functions such as trigger or input capture. It is useful for measuring the interval between the edges on two input signals, as shown in *Figure 753*.

Figure 753. Measuring time interval between edges on 2 signals



57.4.23 External trigger synchronization (TIM15 only)

The TIM timers are linked together internally for timer synchronization or chaining.

The TIM15 timer can be synchronized with an external trigger in several modes: Reset mode, Gated mode, Trigger mode, Reset + trigger, and gated + reset modes.

Slave mode: Reset mode

The counter and its prescaler can be reinitialized in response to an event on a trigger input. Moreover, if the URS bit from the TIMx_CR1 register is low, an update event UEV is generated. Then all the preloaded registers (TIMx_ARR, TIMx_CCRx) are updated.

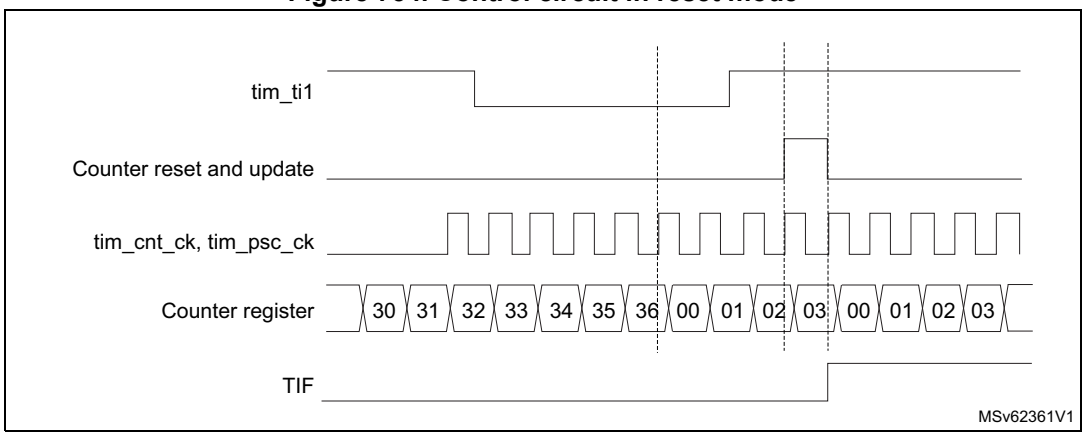
In the following example, the upcounter is cleared in response to a rising edge on tim_ti1 input:

1. Configure the channel 1 to detect rising edges on tim_ti1. Configure the input filter duration (in this example, no need for any filter, so IC1F is kept at 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in the TIMx_CCMR1 register. Write CC1P = 0 and CC1NP = 0 in the TIMx_CCER register to validate the polarity (and detect rising edges only).
2. Configure the timer in reset mode by writing SMS = 100 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIMx_SMCR register.
3. Start the counter by writing CEN = 1 in the TIMx_CR1 register.

The counter starts counting on the internal clock, then behaves normally until tim_ti1 rising edge. When tim_ti1 rises, the counter is cleared and restarts from 0. In the meantime, the trigger flag is set (TIF bit in the TIMx_SR register) and an interrupt request, or a DMA request can be sent if enabled (depending on the TIE and TDE bits in TIMx_DIER register).

The following figure shows this behavior when the autoreload register TIMx_ARR = 0x36. The delay between the rising edge on tim_ti1 and the actual reset of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 754. Control circuit in reset mode



Slave mode: Gated mode

The counter can be enabled depending on the level of a selected input.

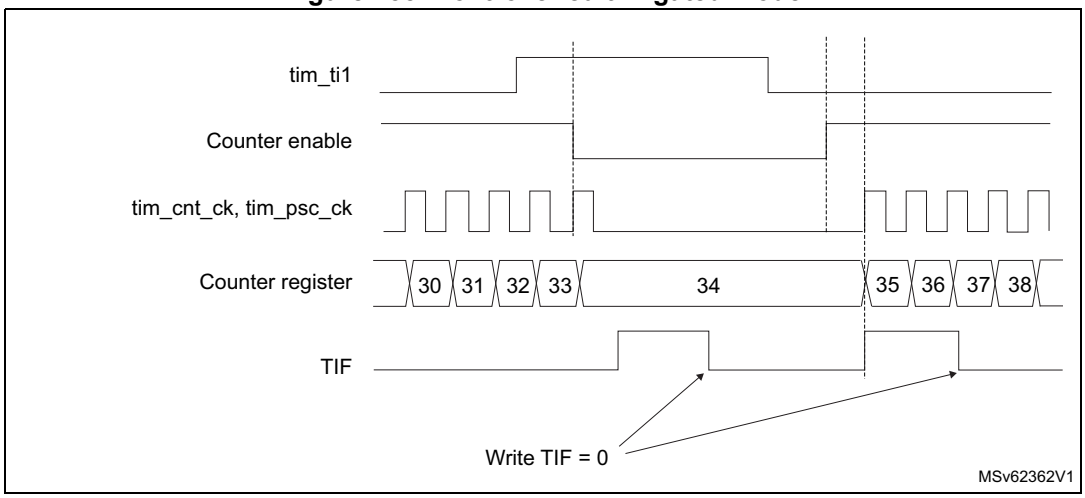
In the following example, the upcounter counts only when tim_ti1 input is low:

1. Configure the channel 1 to detect low levels on tim_ti1. Configure the input filter duration (in this example, no need for any filter, so IC1F is kept at 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC1S bits select the input capture source only, CC1S = 01 in TIMx_CCMR1 register. Write CC1P = 1 and CC1NP = 0 in the TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in gated mode by writing SMS = 101 in TIMx_SMCR register. Select tim_ti1 as the input source by writing TS = 00101 in TIMx_SMCR register.
3. Enable the counter by writing CEN = 1 in the TIMx_CR1 register (in gated mode, the counter does not start if CEN = 0, whatever is the trigger input level).

The counter starts counting on the internal clock as long as tim_ti1 is low and stops as soon as tim_ti1 becomes high. The TIF flag in the TIMx_SR register is set both when the counter starts or stops.

The delay between the rising edge on tim_ti1 and the actual stop of the counter is due to the resynchronization circuit on tim_ti1 input.

Figure 755. Control circuit in gated mode



Slave mode: Trigger mode

The counter can start in response to an event on a selected input.

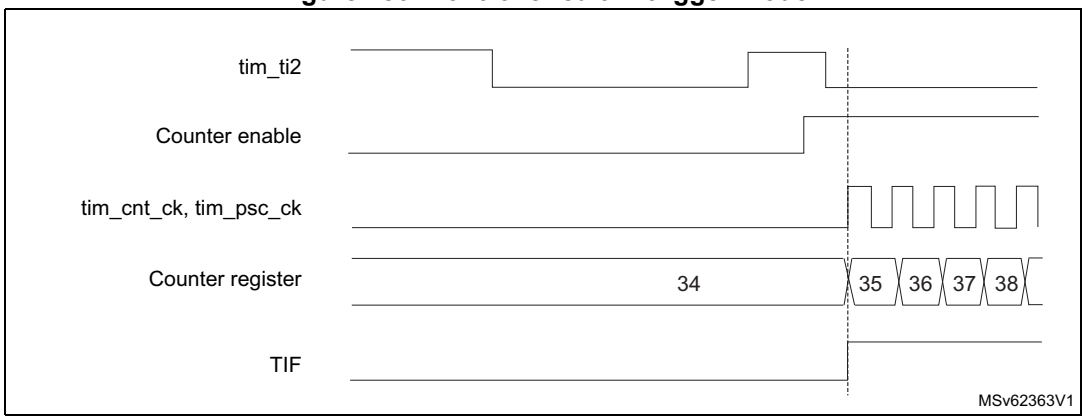
In the following example, the upcounter starts in response to a rising edge on tim_ti2 input:

1. Configure the channel 2 to detect rising edges on tim_ti2. Configure the input filter duration (in this example, no need for any filter, so IC2F is kept at 0000). The capture prescaler is not used for triggering, so it does not need to be configured. The CC2S bits are configured to select the input capture source only, CC2S = 01 in TIMx_CCMR1 register. Write CC2P = 1 and CC2NP = 0 in the TIMx_CCER register to validate the polarity (and detect low level only).
2. Configure the timer in trigger mode by writing SMS = 110 in the TIMx_SMCR register. Select tim_ti2 as the input source by writing TS = 00110 in the TIMx_SMCR register.

When a rising edge occurs on tim_ti2, the counter starts counting on the internal clock and the TIF flag is set.

The delay between the rising edge on tim_ti2 and the actual start of the counter is due to the resynchronization circuit on tim_ti2 input.

Figure 756. Control circuit in trigger mode



Slave mode selection preload for run-time update

The SMS[3:0] bit can be preloaded. This is enabled by setting the SMSPE enable bit in the TIMx_SMCR register. The trigger for the transfer from SMS[3:0] preload to active value is the update event (UEV) occurring when the counter overflows.

57.4.24 Slave mode – combined reset + trigger mode (TIM15 only)

In this case, a rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers, and starts the counter.

This mode is used for one-pulse mode.

57.4.25 Slave mode – combined reset + gated mode (TIM15 only)

The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

This mode is used to detect out-of-range PWM signal (duty cycle exceeding a maximum expected value).

57.4.26 Timer synchronization (TIM15 only)

The TIMx timers are linked together internally for timer synchronization or chaining. Refer to *Section 54.4.23: Timer synchronization* for details.

Note: The clock of the slave peripherals (such as timer and ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

57.4.27 Using timer output as trigger for other timers (TIM16/TIM17 only)

The timers with one channel only do not feature a master mode. However, the OC1 output signal can be used to trigger some other timers (including timers described in other sections of this document). Check the “TIMx internal trigger connection” table of any timer on the device to identify which timers can be targeted as slave.

The OC1 signal pulse width must be programmed to be at least two clock cycles of the destination timer, to make sure the slave timer detects the trigger.

For instance, if the destination's timer CK_INT clock is four times slower than the source timer, the OC1 pulse width must be eight clock cycles.

57.4.28 ADC triggers (TIM15 only)

The timer can generate an ADC triggering event with various internal signals, such as reset, enable or compare events.

Note: The clock of the slave peripherals (such as timer, ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

57.4.29 ADC synchronization (TIM15 only)

The timer operation can be synchronized to the ADC clock to trigger jitter-free ADC sampling. This function is enabled using the ADSYNC bit in the TIMx_CR2 register.

This feature is useful when the timers and the ADCs are operating with semi-synchronous clocks (clocks derived from a same source with integer ratio tim_ker_ck/adc_ker_ck), for instance adc_ker_ck = 75 MHz and tim_ker_ck = 150 MHz or 300 MHz.

ADSYNC must also be set when both peripherals are operating at the same frequency from the same clock source, when jitter-free operation is needed.

ADSYNC must not be set and jitter-free operation is not supported in the following cases:

- when the clock ratio is not an integer (for example adc_ker_ck = 75 MHz and tim_ker_ck = 100 MHz): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum
- when the ADC is operating in asynchronous mode (adc_ker_ck uncorrelated with tim_ker_ck): in this case, the sampling point jitter due to the timer to ADC signal resynchronization is 1 adc_ker_ck period maximum.

When ADSYNC = 1, the timer operation is slightly changed: the counter enable and counter reset events are aligned to the adc_ker_ck ADC clock, to avoid any phase shift due to clocks enable in the RCC.

Jitter-free operation is guaranteed only when one of the 2 requirements below is met (depending on the selected trigger source):

1. the counter period must be a multiple of the ADC clock period

$$(\text{TIMx_PSC} + 1) \times (\text{TIMx_ARR} + 1) \times T_{\text{tim_ker_ck}} = n \times T_{\text{adc_ker_ck}}$$

2. the compare value must be a multiple of the ADC clock period

$$(\text{TIMx_PSC} + 1) \times \text{TIMx_CMPy} \times T_{\text{tim_ker_ck}} = m \times T_{\text{adc_ker_ck}}$$

If none of the 2 above requirement are met, the trigger is still generated, but the latency is not constant and varies with the timer and ADC clocks phase shift.

Programming guidelines

The ADC synchronization feature must not be modified during run-time, once the counter is enabled and once the ADC has been configured for receiving triggers from the timer.

It is mandatory to follow the procedure below to use the ADC synchronization:

1. Enable the destination ADC clock
2. Configure the timer and set the ADSYNC bit
3. Configure the ADC and enable it (using ADSTART and/or JADSTART bits)
4. Start the timer (with the CEN counter enable bit)

57.4.30 DMA burst mode

The TIMx timers have the capability to generate multiple DMA requests on a single event. The main purpose is to be able to reprogram several timer registers multiple times without software overhead, but it can also be used to read several registers in a row, at regular intervals.

The DMA controller destination is unique and must point to the virtual register TIMx_DMAR. On a given timer event, the timer launches a sequence of DMA requests (burst). Each write into the TIMx_DMAR register is actually redirected to one of the timer registers.

The DBL[4:0] bits in the TIMx_DCR register set the DMA burst length. The timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers (either in half-words or in bytes).

The DBA[4:0] bits in the TIMx_DCR registers define the DMA base address for DMA transfers (when read/write accesses are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:

00000: TIMx_CR1

00001: TIMx_CR2

00010: TIMx_SMCR

The DBSS[3:0] bits in the TIMx_DCR register defines the interrupt source that triggers the DMA burst transfers (see *Section 57.8.19: TIMx DMA control register (TIMx_DCR)(x = 16 to 17)* for details).

For example, the timer DMA burst feature can be used to update the contents of the CCRx registers (x = 2, 3, 4) on an update event, with the DMA transferring half words into the CCRx registers.

This is done in the following steps:

1. Configure the corresponding DMA channel as follows:
   - DMA channel peripheral address is the DMAR register address
   - DMA channel memory address is the address of the buffer in the RAM containing the data to be transferred by DMA into the CCRx registers.
   - Number of data to transfer = 3 (See note below).
   - Circular mode disabled.
2. Configure the DCR register by configuring the DBA and DBL bit fields as follows: DBL = 3 transfers, DBA = 0xE and DBSS = 1.
3. Enable the TIMx update DMA request (set the UDE bit in the DIER register).
4. Enable TIMx
5. Enable the DMA channel

This example is for the case where every CCRx register is to be updated once. If every CCRx register is to be updated twice for example, the number of data to transfer must be 6. Let's take the example of a buffer in the RAM containing data1, data2, data3, data4, data5, and data6. The data is transferred to the CCRx registers as follows: on the first update DMA request, data1 is transferred to CCR2, data2 is transferred to CCR3, data3 is transferred to CCR4 and on the second update DMA request, data4 is transferred to CCR2, data5 is transferred to CCR3, and data6 is transferred to CCR4.

Note: A null value can be written to the reserved registers.

57.4.31 TIM15/TIM16/TIM17 DMA requests

The TIM15/TIM16/TIM17 can generate a DMA request, as shown in *Table 547*.

Table 547. DMA request

DMA request signal	DMA request	Enable control bit
tim_upd_dma	Update	UDE
tim_cc1_dma	Capture/compare 1	CC1DE
tim_cc2_dma	Capture/compare 2	CC2DE
tim_com_dma(1)	Commutation (COM)	COMDE
tim_trgi_dma(1)	Trigger	TDE

1. Available for TIM15 only.

57.4.32 Debug mode

When the microcontroller enters debug mode (Cortex®-M55 core halted), the TIMx counter can either continue to work normally or stop.

The behavior in debug mode can be programmed with a dedicated configuration bit per timer in the Debug support (DBG) module.

For safety purposes, when the counter is stopped, the outputs are disabled (as if the MOE bit was reset). The outputs can either be forced to an inactive state (OSSI bit = 1), or have their control taken over by the GPIO controller (OSSI bit = 0) to force them to Hi-Z.

For more details, refer to the Debug section.

57.5 TIM15/TIM16/TIM17 low-power modes

Table 548. Effect of low-power modes on TIM15/TIM16/TIM17

Mode	Description
Sleep	No effect, peripheral is active. The interrupts can cause the device to exit from Sleep mode.
Stop	The timer operation is stopped and the register content is kept. No interrupt can be generated.
Standby	The timer is powered-down and must be reinitialized after exiting the Standby mode.

57.6 TIM15/TIM16/TIM17 interrupts

The TIM15/TIM16/TIM17 can generate multiple interrupts, as shown in *Table 549*.

Table 549. Interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop and Standby mode
TIM	Update	UIF	UIE	write 0 in UIF	Yes	No
	Capture/compare 1	CC1IF	CC1IE	write 0 in CC1IF	Yes	No
	Capture/compare 2(1)	CC2IF	CC2IE	write 0 in CC2IF	Yes	No
	Commutation (COM)	COMIF	COMIE	write 0 in COMIF	Yes	No
	Trigger(1)	TIF	TIE	write 0 in TIF	Yes	No
	Break	BIF	BIE	write 0 in BIF	Yes	No

1. Available for TIM15 only.

57.7 TIM15 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

57.7.1 TIM15 control register 1 (TIM15_CR1)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw				rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 DITHEN: Dithering enable
0: Dithering disabled
1: Dithering enabled
Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIM15_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIM15_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division
This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and the dead-time and sampling clock (t_{DTS}) used by the dead-time generators and the digital filters (tim_tix)
00: $t_{DTS} = t_{tim_ker_ck}$
01: $t_{DTS} = 2*t_{tim_ker_ck}$
10: $t_{DTS} = 4*t_{tim_ker_ck}$
11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable
0: TIM15_ARR register is not buffered
1: TIM15_ARR register is buffered

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One-pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source
This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generate an update interrupt if enabled. These events can be:
– Counter overflow/underflow
– Setting the UG bit
– Update generation through the slave mode controller
1: Only counter overflow/underflow generates an update interrupt if enabled

Bit 1 **UDIS**: Update disable
This bit is set and cleared by software to enable/disable UEV event generation.
0: UEV enabled. The Update (UEV) event is generated by one of the following events:
– Counter overflow/underflow
– Setting the UG bit
– Update generation through the slave mode controller
Buffered registers are then loaded with their preload values.
1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable
0: Counter disabled
1: Counter enabled
Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

57.7.2 TIM15 control register 2 (TIM15_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	AD SYNC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			rw												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
					rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **ADSYNC**: ADC synchronization

0: The timer operates independently from the ADC

1: The timer operation is synchronized with the ADC clock to provide jitter-free sampling point. This mode can be enabled only with specific ADC / timer clock relationship. Refer to *Section 57.4.29* for requirements.

The ADSYNC must not modified when the counter is enabled (CEN bit is set).

Bits 27:11 Reserved, must be kept at reset value.

Bit 10 **OIS2:** Output idle state 2 (tim_oc2 output)

0: tim_oc2 = 0 when MOE = 0

1: tim_oc2 = 1 when MOE = 0

Note: This bit cannot be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in the TIM15_BKR register).

Bit 9 **OIS1N**: Output Idle state 1 (tim_oc1n output)

0: tim_oc1n = 0 after a dead-time when MOE = 0

1: tim_oc1n = 1 after a dead-time when MOE = 0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BKR register).

Bit 8 **OIS1**: Output Idle state 1 (tim_oc1 output)

0: tim_oc1 = 0 after a dead-time when MOE = 0

1: tim_oc1 = 1 after a dead-time when MOE = 0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BKR register).

Bit 7 **TI1S**: tim_ti1 selection

0: The tim_ti1_in[15:0] multiplexer output is connected to tim_ti1 input

1: The tim_ti1_in[15:0] and tim_ti2_in[15:0] multiplexers output are connected to the tim_ti1 input (XOR combination)

Bits 6:4 **MMS[2:0]**: Master mode selection

These bits are used to select the information to be sent in master mode to slave timers for synchronization (tim_trgo). The combination is as follows:

000: **Reset** - the UG bit from the TIM15_EGR register is used as trigger output (tim_trgo). If the reset is generated by the trigger input (slave mode controller configured in reset mode) then the signal on tim_trgo is delayed compared to the actual reset.

001: **Enable** - the Counter Enable signal CNT_EN is used as trigger output (tim_trgo). It is useful to start several timers at the same time or to control a window in which a slave timer is enable. The Counter Enable signal is generated by a logic AND between CEN control bit and the trigger input when configured in gated mode. When the Counter Enable signal is controlled by the trigger input, there is a delay on tim_trgo, except if the master/slave mode is selected (see the MSM bit description in TIM15_SMCR register).

010: **Update** - The update event is selected as trigger output (tim_trgo). For instance a master timer can then be used as a prescaler for a slave timer.

011: **Compare Pulse** - The trigger output send a positive pulse when the CC1IF flag is to be set (even if it was already high), as soon as a capture or a compare match occurred (tim_trgo).

100: **Compare** - tim_oc1refc signal is used as trigger output (tim_trgo).

101: **Compare** - tim_oc2refc signal is used as trigger output (tim_trgo).

Bit 3 **CCDS**: Capture/compare DMA selection
0: CCx DMA request sent when CCx event occurs
1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection
0: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit only.
1: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit or when an rising edge occurs on tim_trgi.
Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control
0: CCxE, CCxNE and OCxM bits are not preloaded
1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when a commutation event (COM) occurs (COMG bit set or rising edge detected on tim_trgi, depending on the CCUS bit).
Note: This bit acts only on channels that have a complementary output.

57.7.3 TIM15 slave mode control register (TIM15_SMCR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMSPE	Res.	Res.	TS[4:3]		Res.	Res.	Res.	SMS[3]
							rw			rw	rw				rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSM	TS[2:0]			Res.	SMS[2:0]		
								rw	rw	rw	rw		rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **SMSPE:** SMS preload enable
This bit selects whether the SMS[3:0] bitfield is preloaded.
0: SMS[3:0] bitfield is not preloaded
1: SMS[3:0] preload is enabled

Bits 23:22 Reserved, must be kept at reset value.

Bits 19:17 Reserved, must be kept at reset value.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **MSM:** Master/slave mode
0: No action
1: The effect of an event on the trigger input (tim_trgi) is delayed to allow a perfect synchronization between the current timer and its slaves (through tim_trgo). It is useful if the user wants to synchronize several timers on a single external event.

Bits 21, 20, 6, 5, 4 **TS[4:0]:** Trigger selection

This bitfield selects the trigger input to be used to synchronize the counter.

00000: Internal Trigger 0 (tim_itr0)
00001: Internal Trigger 1 (tim_itr1)
00010: Internal Trigger 2 (tim_itr2)
00011: Internal Trigger 3 (tim_itr3)
00100: tim_ti1 Edge Detector (tim_ti1f_ed)
00101: Filtered Timer Input 1 (tim_ti1fp1)
00110: Filtered Timer Input 2 (tim_ti2fp2)
00111: Reserved
01000: Internal Trigger 4 (tim_itr4)
01001: Internal Trigger 5 (tim_itr5)
01010: Internal Trigger 6 (tim_itr6)
01011: Internal Trigger 7 (tim_itr7)
01100: Internal Trigger 8 (tim_itr8)
01101: Internal Trigger 9 (tim_itr9)
01110: Internal Trigger 10 (tim_itr10)
10000: Internal trigger 12 (tim_itr12)
10001: Internal trigger 13 (tim_itr13)
10010: Internal trigger 14 (tim_itr14)
10011: Internal trigger 15 (tim_itr15)
Others: Reserved

See *Section 57.4.2: TIM15/TIM16/TIM17 pins and internal signals* for more details on tim_itrx meaning for each timer.

Note: These bits must be changed only when they are not used (for example when SMS = 000) to avoid wrong edge detections at the transition.

Bit 3 Reserved, must be kept at reset value.

Bits 16, 2, 1, 0 **SMS[3:0]:** Slave mode selection

When external signals are selected the active edge of the trigger signal (tim_trgi) is linked to the polarity selected on the external input (refer to ETP bit in TIMx_SMCR for tim_etr_in and CCxP/CCxNP bits in TIMx_CCER register for tim_ti1fp1 and tim_ti2fp2).

0000: Slave mode disabled - if CEN = 1 then the prescaler is clocked directly by the internal clock.

0001: Reserved

0010: Reserved

0011: Reserved

0100: Reset mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter and generates an update of the registers.

0101: Gated mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops (but is not reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

0110: Trigger mode - The counter starts at a rising edge of the trigger tim_trgi (but it is not reset). Only the start of the counter is controlled.

0111: External Clock mode 1 - Rising edges of the selected trigger (tim_trgi) clock the counter.

1000: Combined reset + trigger mode - Rising edge of the selected trigger input (tim_trgi) reinitializes the counter, generates an update of the registers and starts the counter.

1001: Combined gated + reset mode - The counter clock is enabled when the trigger input (tim_trgi) is high. The counter stops and is reset) as soon as the trigger becomes low. Both start and stop of the counter are controlled.

Others: Reserved.

Note: The gated mode must not be used if tim_ti1f_ed is selected as the trigger input (TS = 00100). Indeed, tim_ti1f_ed outputs 1 pulse for each transition on tim_ti1f, whereas the gated mode checks the level of the trigger signal.

The clock of the slave peripherals (such as timer and ADC) receiving the tim_trgo signal must be enabled prior to receive events from the master timer, and the clock frequency (prescaler) must not be changed on-the-fly while triggers are received from the master timer.

57.7.4 TIM15 DMA/interrupt enable register (TIM15_DIER)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDE	COMDE	Res.	Res.	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	Res.	Res.	CC2IE	CC1IE	UIE
	rw	rw			rw	rw	rw	rw	rw	rw			rw	rw	rw

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TDE**: Trigger DMA request enable

0: Trigger DMA request disabled

1: Trigger DMA request enabled

Bit 13 **COMDE**: COM DMA request enable

0: COM DMA request disabled

1: COM DMA request enabled

Bits 12:11 Reserved, must be kept at reset value.

Bit 10 **CC2DE**: Capture/Compare 2 DMA request enable
0: CC2 DMA request disabled
1: CC2 DMA request enabled

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable
0: Break interrupt disabled
1: Break interrupt enabled

Bit 6 **TIE**: Trigger interrupt enable
0: Trigger interrupt disabled
1: Trigger interrupt enabled

Bit 5 **COMIE:** COM interrupt enable
0: COM interrupt disabled
1: COM interrupt enabled

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **CC2IE**: Capture/Compare 2 interrupt enable
0: CC2 interrupt disabled
1: CC2 interrupt enabled

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

57.7.5 TIM15 status register (TIM15_SR)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CC2OF	CC1OF	Res.	BIF	TIF	COMIF	Res.	Res.	CC2IF	CC1IF	UIF
					rc_w0	rc_w0		rc_w0	rc_w0	rc_w0			rc_w0	rc_w0	rc_w0

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **CC2OF**: Capture/Compare 2 overcapture flag
Refer to CC1OF description

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag
This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.
0: No overcapture has been detected
1: The counter value has been captured in TIM15_CCR1 register while CC1IF flag was already set

Bit 8 Reserved, must be kept at reset value.

Bit 7 **BIF**: Break interrupt flag
This flag is set by hardware as soon as the break input goes active. It can be cleared by software if the break input is not active.
0: No break event occurred
1: An active level has been detected on the break input

Bit 6 **TIF**: Trigger interrupt flag
This flag is set by hardware on the TRG trigger event (active edge detected on tim_trgi input when the slave mode controller is enabled in all modes but gated mode, both edges in case gated mode is selected). It is set when the counter starts or stops when gated mode is selected. It is cleared by software.
0: No trigger event occurred
1: Trigger interrupt pending

Bit 5 **COMIF:** COM interrupt flag
This flag is set by hardware on a COM event (once the capture/compare control bits –CCxE, CCxNE, OCxM– have been updated). It is cleared by software.
0: No COM event occurred
1: COM interrupt pending

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **CC2IF**: Capture/Compare 2 interrupt flag
refer to CC1IF description

Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag
This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).
0: No compare match / No input capture occurred
1: A compare match or an input capture occurred
If channel CC1 is configured as output: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in downcounting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.
If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0 **UIF**: Update interrupt flag
This bit is set by hardware on an update event. It is cleared by software.
0: No update occurred.
1: Update interrupt pending. This bit is set by hardware when the registers are updated:
– At overflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS = 0 in the TIM15_CR1 register.
– When CNT is reinitialized by software using the UG bit in TIM15_EGR register, if URS = 0 and UDIS = 0 in the TIM15_CR1 register.
– When CNT is reinitialized by a trigger event (refer to *Section 57.7.3: TIM15 slave mode control register (TIM15_SMCR)*), if URS = 0 and UDIS = 0 in the TIM15_CR1 register.

57.7.6 TIM15 event generation register (TIM15_EGR)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	TG	COMG	Res.	Res.	CC2G	CC1G	UG
								w	w	rw			w	w	w

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **BG**: Break generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 **TG**: Trigger generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1: The TIF flag is set in TIM15_SR register. Related interrupt or DMA transfer can occur if enabled

Bit 5 **COMG:** Capture/Compare control update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: When the CCPC bit is set, it is possible to update the CCxE, CCxNE and OCxM bits
Note: This bit acts only on channels that have a complementary output.

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **CC2G**: Capture/Compare 2 generation
Refer to CC1G description

Bit 1 **CC1G**: Capture/Compare 1 generation
This bit is set by software in order to generate an event, it is automatically cleared by hardware.
0: No action
1 A capture/compare event is generated on channel 1:
If channel CC1 is configured as output:
CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.
If channel CC1 is configured as input:
The current value of the counter is captured in TIM15_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation
This bit can be set by software, it is automatically cleared by hardware.
0: No action
1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected).

57.7.7 TIM15 capture/compare mode register 1 (TIM15_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IC2F[3:0]				IC2PSC[1:0]		CC2S[1:0]		IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:12 **IC2F[3:0]**: Input capture 2 filter

Bits 11:10 **IC2PSC[1:0]**: Input capture 2 prescaler

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output

01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2

10: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti1

11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM15_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIM15_CCER).

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bitfield defines the frequency used to sample tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}

0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2

0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4

0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8

0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6

0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8

0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6

0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8

1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6

1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8

1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5

1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6

1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8

1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5

1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6

1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on CC1 input (tim_ic1). The prescaler is reset as soon as CC1E = 0 (TIM15_CCER register).

00: no prescaler, capture is done each time an edge is detected on the capture input

01: capture is done once every 2 events

10: capture is done once every 4 events

11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1

10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2

11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM15_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIM15_CCER).

57.7.8 TIM15 capture/compare mode register 1 [alternate] (TIM15_CCMR1)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M [3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M [3]
							rw								rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OC2 CE	OC2M[2:0]			OC2 PE	OC2 FE	CC2S[1:0]		OC1 CE	OC1M[2:0]			OC1 PE	OC1 FE	CC1S[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode:

Bits 31:25 Reserved, must be kept at reset value.

Bits 23:17 Reserved, must be kept at reset value.

Bit 15 **OC2CE**: Output compare 2 clear enable

Bits 24, 14:12 **OC2M[3:0]**: Output compare 2 mode

Bit 11 **OC2PE**: Output compare 2 preload enable

Bit 10 **OC2FE**: Output compare 2 fast enable

Bits 9:8 **CC2S[1:0]**: Capture/Compare 2 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC2 channel is configured as output.

01: CC2 channel is configured as input, tim_ic2 is mapped on tim_ti2.

10: C2 channel is configured as input, tim_ic2 is mapped on tim_ti1.

11: CC2 channel is configured as input, tim_ic2 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through the TS bit (TIM15_SMCR register)

Note: CC2S bits are writable only when the channel is OFF (CC2E = 0 in TIM15_CCER).

Bit 7 **OC1CE**: Output compare 1 clear enable

0: tim_oc1ref is not affected by the tim_ocref_clr_int input.

1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr_int input.

Bits 16, 6:4 **OC1M[3:0]**: Output compare 1 mode

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 and tim_oc1n are derived. tim_oc1ref is active high whereas tim_oc1 and tim_oc1n active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIM15_CCR1 and the counter TIM15_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.

0001: Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIM15_CNT matches the capture/compare register 1 (TIM15_CCR1).

0010: Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIM15_CNT matches the capture/compare register 1 (TIM15_CCR1).

0011: Toggle - tim_oc1ref toggles when TIM15_CNT = TIM15_CCR1.

0100: Force inactive level - tim_oc1ref is forced low.

0101: Force active level - tim_oc1ref is forced high.

0110: PWM mode 1 - Channel 1 is active as long as TIM15_CNT<TIM15_CCR1 else inactive.

0111: PWM mode 2 - Channel 1 is inactive as long as TIM15_CNT<TIM15_CCR1 else active.

1000: Retrigerrable OPM mode 1 - In up-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update. In down-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes inactive again at the next update.

1001: Retrigerrable OPM mode 2 - In up-counting mode, the channel is inactive until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 2 and the channels becomes inactive again at the next update. In down-counting mode, the channel is active until a trigger event is detected (on tim_trgi signal). Then, a comparison is performed as in PWM mode 1 and the channels becomes active again at the next update.

1010: Reserved

1011: Reserved

1100: Combined PWM mode 1 - tim_oc1ref has the same behavior as in PWM mode 1. tim_oc1refc is the logical OR between tim_oc1ref and tim_oc2ref.

1101: Combined PWM mode 2 - tim_oc1ref has the same behavior as in PWM mode 2. tim_oc1refc is the logical AND between tim_oc1ref and tim_oc2ref.

1110: Reserved,

1111: Reserved,

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM15_BDTR register) and CC1S = 00 (the channel is configured in output).

In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.

On channels that have a complementary output, this bitfield is preloaded. If the CCPC bit is set in the TIM15_CR2 register then the OC1M active bits take the new value from the preloaded bits only when a COM event is generated.

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIM15_CCR1 disabled. TIM15_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIM15_CCR1 enabled. Read/Write operations access the preload register. TIM15_CCR1 preload value is loaded in the active register at each update event.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIM15_BDTR register) and CC1S = 00 (the channel is configured in output).

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, tim_ocx is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OCFE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output.

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1.

10: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti2.

11: CC1 channel is configured as input, tim_ic1 is mapped on tim_trc. This mode is working only if an internal trigger input is selected through TS bit (TIM15_SMCR register)

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIM15_CCER).

57.7.9 TIM15 capture/compare enable register (TIM15_CCER)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2NP	Res.	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
								rw		rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **CC2NP**: Capture/Compare 2 complementary output polarity
Refer to CC1NP description

Bit 6 Reserved, must be kept at reset value.

Bit 5 **CC2P**: Capture/Compare 2 output polarity
Refer to CC1P description

Bit 4 **CC2E**: Capture/Compare 2 output enable
Refer to CC1E description

Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity

CC1 channel configured as output:

0: tim_oc1n active high

1: tim_oc1n active low

CC1 channel configured as input:

This bit is used in conjunction with CC1P to define the polarity of tim_ti1fp1 and tim_ti2fp1. Refer to CC1P description.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register) and CC1S = 00 (the channel is configured in output).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIM15_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 2 **CC1NE**: Capture/Compare 1 complementary output enable

0: Off - tim_oc1n is not active. tim_oc1n level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

1: On - tim_oc1n signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Bit 1 **CC1P**: Capture/Compare 1 output polarity

CC1 channel configured as output:

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP = 0, CC1P = 0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode).

CC1NP = 0, CC1P = 1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode).

CC1NP = 1, CC1P = 1: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode).

CC1NP = 1, CC1P = 0: this configuration is reserved, it must not be used.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIM15_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0 **CC1E**: Capture/Compare 1 output enable

0: Capture mode disabled / OC1 is not active (see below)

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

When CC1 channel is configured as output, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to *Table 550* for details.

Table 550. Output control bits for complementary tim_ocx and tim_ocxn channels with break feature (TIM15)

Control bits					Output states(1)	
MOE bit	OSSI bit	OSSR bit	CCxE bit	CCxNE bit	tim_ocx output state	tim_ocxn output state
1	X	X	0	0	Output Disabled (not driven by the timer: Hi-Z) tim_ocx = 0 tim_ocxn = 0	
		0	0	1	Output Disabled (not driven by the timer: Hi-Z) tim_ocx = 0	tim_ocxref + Polarity tim_ocxn = tim_ocxref XOR CCxNP
		0	1	0	tim_ocxref + Polarity tim_ocx = tim_ocxref XOR CCxP	Output Disabled (not driven by the timer: Hi-Z) tim_ocxn = 0
		X	1	1	tim_ocxref + Polarity + dead-time	Complementary to tim_ocxref (not OCREF) + Polarity + dead-time
		1	0	1	Off-State (output enabled with inactive state) tim_ocx = CCxP	tim_ocxref + Polarity tim_ocxn = tim_ocxref XOR CCxNP
		1	1	0	tim_ocxref + Polarity tim_ocx = tim_ocxref xor CCxP	Off-State (output enabled with inactive state) tim_ocxn = CCxNP
0	0	X	X	X	Output disabled (not driven by the timer: Hi-Z)	
	1		0	0		
			0	1	Off-State (output enabled with inactive state) Asynchronously: tim_ocx = CCxP, tim_ocxn = CCxNP Then if the clock is present: tim_ocx = OISx and tim_ocxn = OISxN after a dead-time, assuming that OISx and OISxN do not correspond to tim_ocx and tim_ocxn both in active state	
			1	0		
			1	1		

1. When both outputs of a channel are not used (control taken over by GPIO controller), the OISx, OISxN, CCxP and CCxNP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary tim_ocx and tim_ocxn channels depends on the tim_ocx and tim_ocxn channel state and GPIO control and alternate function selection registers.

57.7.10 TIM15 counter (TIM15_CNT)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF Copy
This bit is a read-only copy of the UIF bit in the TIM15_ISR register.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value
Non-dithering mode (DITHEN = 0)
The register holds the counter value.
Dithering mode (DITHEN = 1)
The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

57.7.11 TIM15 prescaler (TIM15_PSC)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value
The counter clock frequency ($f_{tim_cnt_ck}$) is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).
PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIM15_EGR register or through trigger controller when configured in “reset mode”).

57.7.12 TIM15 autoreload register (TIM15_ARR)

Address offset: 0x2C

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the *Section 57.4.3: Time-base unit* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

57.7.13 TIM15 repetition counter register (TIM15_RCR)

Address offset: 0x30

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **REP[7:0]**: Repetition counter reload value

This bitfield defines the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable. It also defines the update interrupt generation rate, if this interrupt is enable.

When the repetition down-counter reaches zero, an update event is generated and it restarts counting from REP value. As the reptition counter is reloaded with REP value only at the repetition update event UEV, any write to the TIM15_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to the number of PWM periods in edge-aligned mode:

– The number of PWM periods in edge-aligned mode.
– The number of half PWM period in center-aligned mode.

57.7.14 TIM15 capture/compare register 1 (TIM15_CCR1)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR1[19:0]**: Capture/compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIM15_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIM15_CNT and signaled on tim_oc1 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.

If channel CC1 is configured as input:

CR1 is the counter value transferred by the last input capture 1 event (tim_ic1). The TIMx_CCR1 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

57.7.15 TIM15 capture/compare register 2 (TIM15_CCR2)

Address offset: 0x38

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR2[19:0]**: Capture/compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the actual capture/compare 2 register (preload value).

It is loaded permanently if the preload feature is not selected in the TIM15_CCMR2 register (bit OC2PE). Else the preload value is copied in the active capture/compare 2 register when an update event occurs.

The active capture/compare register contains the value to be compared to the counter TIM15_CNT and signalled on tim_oc2 output.

Non-dithering mode (DITHEN = 0)

The register holds the compare value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in CCR2[19:4]. The CCR2[3:0] bitfield contains the dithered part.

If channel CC2 is configured as input:

CCR2 is the counter value transferred by the last input capture 1 event (tim_ic2). The TIMx_CCR2 register is read-only and cannot be programmed.

Non-dithering mode (DITHEN = 0)

The register holds the capture value in CCR2[15:0]. The CCR2[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the capture in CCR2[19:4]. The CCR2[3:0] bits are reset.

57.7.16 TIM15 break and dead-time register (TIM15_BDTR)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	BKBID	Res.	BK DSRM	Res.	Res.	Res.	Res.	Res.	Res.	BKF[3:0]			
			rw		rw							rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the BKBID, BKDSRM, BKF[3:0], AOE, BKP, BKE, OSSI, OSSR, and DTG[7:0] bits may be write-locked depending on the LOCK configuration, it may be necessary to configure all of them during the first write access to the TIM15_BDTR register.

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **BKBID**: Break bidirectional

0: Break input tim_brk in input mode
1: Break input tim_brk in bidirectional mode
In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **BKDSRM**: Break disarm

0: Break input tim_brk is armed
1: Break input tim_brk is disarmed
This bit is cleared by hardware when no break source is active.
The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled until it is reset by hardware, indicating that the fault condition has disappeared.

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bits 25:20 Reserved, must be kept at reset value.

Bits 19:16 **BKF[3:0]**: Break filter

This bitfield defines the frequency used to sample the tim_brk input signal and the length of the digital filter applied to tim_brk. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:
0000: No filter, tim_brk acts asynchronously
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as the tim_brk input is active. It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: tim_ocx and tim_ocxn outputs are disabled or forced to idle state depending on the OSSI bit.

1: tim_ocx and tim_ocxn outputs are enabled if their respective enable bits are set (CCxE, CCxNE in TIM15_CCER register)

See tim_ocx/tim_ocxn enable description for more details (*Section 57.7.9: TIM15 capture/compare enable register (TIM15_CCER) on page 2885*).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if the break input is not be active)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input tim_brk is active low

1: Break input tim_brk is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable

0: Break inputs (tim_brk and tim_sys_brk clock failure event) disabled

1; Break inputs (tim_brk and tim_sys_brk clock failure event) enabled

This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE = 1 on channels that have a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See tim_ocx/tim_ocxn enable description for more details (*Section 57.7.9: TIM15 capture/compare enable register (TIM15_CCER) on page 2885*).

0: When inactive, tim_ocx/tim_ocxn outputs are disabled (the timer releases the output control which is taken over by the GPIO, which forces a Hi-Z state)

1: When inactive, tim_ocx/tim_ocxn outputs are enabled with their inactive level as soon as CCxE = 1 or CCxNE = 1 (the output is still controlled by the timer).

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 10 **OSSI**: Off-state selection for Idle mode

This bit is used when MOE = 0 on channels configured as outputs.

See tim_ocx/tim_ocxn enable description for more details (*Section 57.7.9: TIM15 capture/compare enable register (TIM15_CCER) on page 2885*).

0: When inactive, tim_ocx/tim_ocxn outputs are disabled (tim_ocx/tim_ocxn enable output signal = 0)

1: When inactive, tim_ocx/tim_ocxn outputs are forced first with their idle level as soon as CCxE = 1 or CCxNE = 1. tim_ocx/tim_ocxn enable output signal = 1)

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIM15_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected

01: LOCK Level 1 = DTG bits in TIM15_BDTR register, OISx and OISxN bits in TIM15_CR2 register and BKBID/BKE/BKP/AOE bits in TIM15_BDTR register can no longer be written

10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIM15_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.

11: OCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIM15_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

Note: The LOCK bits can be written only once after the reset. Once the TIM15_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bitfield defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.

DTG[7:5] = 0xx → DT = DTG[7:0]x t_{dtg} with $t_{dtg} = t_{DTS}$

DTG[7:5] = 10x → DT = (64+DTG[5:0])xt_{dtg} with $T_{dtg} = 2xt_{DTS}$

DTG[7:5] = 110 → DT = (32+DTG[4:0])xt_{dtg} with $T_{dtg} = 8xt_{DTS}$

DTG[7:5] = 111 → DT = (32+DTG[4:0])xt_{dtg} with $T_{dtg} = 16xt_{DTS}$

Example if T_{DTS} = 125 ns (8MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register).

57.7.17 TIM15 timer deadtime register 2 (TIM15_DTR2)

Address offset: 0x054

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPE	DTAE
														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTGF[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **DTPE**: Deadtime preload enable

0: Deadtime value is not preloaded

1: Deadtime value preload is enabled

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 16 **DTAE**: Deadtime asymmetric enable

0: Deadtime on rising and falling edges are identical, and defined with DTG[7:0] register

1: Deadtime on rising edge is defined with DTG[7:0] register and deadtime on falling edge is defined with DTGF[7:0] bits.

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **DTGF[7:0]**: Dead-time falling edge generator setup

This bitfield defines the duration of the dead-time inserted between the complementary outputs, on the falling edge.

DTGF[7:5] = 0xx → DTF = DTGF[7:0]x t_{dtg} with $t_{dtg} = t_{DTS}$.

DTGF[7:5] = 10x → DTF = (64+DTGF[5:0])xt_{dtg} with $T_{dtg} = 2xt_{DTS}$.

DTGF[7:5] = 110 → DTF = (32+DTGF[4:0])xt_{dtg} with $T_{dtg} = 8xt_{DTS}$.

DTGF[7:5] = 111 → DTF = (32+DTGF[4:0])xt_{dtg} with $T_{dtg} = 16xt_{DTS}$.

Example if T_{DTS} = 125 ns (8 MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIM15_BDTR register).

57.7.18 TIM15 input selection register (TIM15_TISEL)

Address offset: 0x5C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **TI2SEL[3:0]**: selects tim_ti2_in[15:0] input

0000: TIM15_CH2 input (tim_ti2_in0)

0001: tim_ti2_in1

...

1111: tim_ti2_in15

Refer to *Section 57.4.2: TIM15/TIM16/TIM17 pins and internal signals* for interconnects list.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects tim_ti1_in[15:0] input

0000: TIM15_CH1 input (tim_ti1_in0)

0001: tim_ti1_in1

...

1111: tim_ti1_in15

Refer to *Section 57.4.2: TIM15/TIM16/TIM17 pins and internal signals* for interconnects list.

57.7.19 TIM15 alternate function register 1 (TIM15_AF1)

Address offset: 0x060

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	BK CMP4P	BK CMP3P	BK CMP2P	BK CMP1P	BKINP	BK CMP8E	BK CMP7E	BK CMP6E	BK CMP5E	BK CMP4E	BK CMP3E	BK CMP2E	BK CMP1E	BKINE
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Refer to *Section 57.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation.

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **BKCMP4P**: tim_brk_cmp4 input polarity

This bit selects the tim_brk_cmp4 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp4 input is active high

1: tim_brk_cmp4 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 12 **BKCMP3P**: tim_brk_cmp3 input polarity

This bit selects the tim_brk_cmp3 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp3 input is active high

1: tim_brk_cmp3 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 11 **BKCMP2P**: tim_brk_cmp2 input polarity

This bit selects the tim_brk_cmp2 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp2 input is active high

1: tim_brk_cmp2 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 10 **BKCMP1P**: tim_brk_cmp1 input polarity

This bit selects the tim_brk_cmp1 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp1 input is active high

1: tim_brk_cmp1 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 9 **BKINP**: TIMx_BKIN input polarity

This bit selects the TIMx_BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: TIMx_BKIN input is active high

1: TIMx_BKIN input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 8 **BKCMP8E**: tim_brk_cmp8 enable

This bit enables the tim_brk_cmp8 for the timer's tim_brk input. tim_brk_cmp8 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp8 input disabled

1: tim_brk_cmp8 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 7 **BKCMP7E**: tim_brk_cmp7 enable

This bit enables the tim_brk_cmp7 for the timer's tim_brk input. tim_brk_cmp7 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp7 input disabled

1: tim_brk_cmp7 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 6 **BKCMP6E**: tim_brk_cmp6 enable

This bit enables the tim_brk_cmp6 for the timer's tim_brk input. tim_brk_cmp6 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp6 input disabled

1: tim_brk_cmp6 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 5 **BKCMP5E**: tim_brk_cmp5 enable

This bit enables the tim_brk_cmp5 for the timer's tim_brk input. tim_brk_cmp5 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp5 input disabled

1: tim_brk_cmp5 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 4 **BKCMP4E**: tim_brk_cmp4 enable

This bit enables the tim_brk_cmp4 for the timer's tim_brk input. tim_brk_cmp4 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp4 input disabled

1: tim_brk_cmp4 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 3 **BKCMP3E**: tim_brk_cmp3 enable

This bit enables the tim_brk_cmp3 for the timer's tim_brk input. tim_brk_cmp3 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp3 input disabled

1: tim_brk_cmp3 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 2 **BKCMP2E**: tim_brk_cmp2 enable

This bit enables the tim_brk_cmp2 for the timer's tim_brk input. tim_brk_cmp2 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp2 input disabled

1: tim_brk_cmp2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 1 **BKCMP1E**: tim_brk_cmp1 enable

This bit enables the tim_brk_cmp1 for the timer's tim_brk input. tim_brk_cmp1 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp1 input disabled

1: tim_brk_cmp1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bit 0 **BKINE**: TIMx_BKIN input enable

This bit enables the TIMx_BKIN alternate function input for the timer's tim_brk input. TIMx_BKIN input is 'ORed' with the other tim_brk sources.

0: TIMx_BKIN input disabled

1: TIMx_BKIN input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

57.7.20 TIM15 alternate function register 2 (TIM15_AF2)

Address offset: 0x064

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL[2:0]		
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **OCRSEL[2:0]**: ocref_clr source selection

These bits select the ocref_clr input source.

000: tim_ocref_clr0

001: tim_ocref_clr1

010: tim_ocref_clr2

011: tim_ocref_clr3

100: tim_ocref_clr4

101: tim_ocref_clr5

110: tim_ocref_clr6

111: tim_ocref_clr7

Refer to *Section 57.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation.

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIM15_BDTR register).

Bits 15:0 Reserved, must be kept at reset value.

57.7.21 TIM15 DMA control register (TIM15_DCR)

Address offset: 0x3DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **DBSS[3:0]**: DMA burst source selection

This bitfield defines the interrupt source that triggers the DMA burst transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address).

0000: Reserved

0001: Update

0010: CC1

0011: CC2

0110: COM

0111: Trigger

Other: reserved

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bitfield defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIM15_DMAR address).

00000: 1 transfer,
00001: 2 transfers,
00010: 3 transfers,
...
10001: 18 transfers.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bitfield defines the base-address for DMA transfers (when read/write access are done through the TIM15_DMAR address). DBA is defined as an offset starting from the address of the TIM15_CR1 register.

Example:

00000: TIM15_CR1,
00001: TIM15_CR2,
00010: TIM15_SMCR,
...

57.7.22 TIM15 DMA address for full transfer (TIM15_DMAR)

Address offset: 0x3E0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMAB[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIM15_CR1 address) + (DBA + DMA index) x 4

where TIM15_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIM15_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIM15_DCR).

57.7.23 TIM15 register map

TIM15 registers are mapped as 16-bit addressable registers as described in the table below:

Table 551. TIM15 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TIM15_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMA	Res.	CKD [1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value																					0		0	0	0				0	0	0	0
0x04	TIM15_CR2	Res.	Res.	Res.	ADSYNC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIS2	OIS1N	OIS1	TI1S	MMS[2:0]			CCDS	CCUS	Res.	CCPC
	Reset value				0																		0	0	0	0	0	0	0	0	0		0
0x08	TIM15_SMCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SMSPE	Res.	Res.	TS [4:3]		Res.	Res.	Res.	SMS[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSM	TS[2:0]			Res.	SMS[2:0]		
	Reset value								0			0	0				0								0	0	0	0			0	0	0
0x0C	TIM15_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDE	COMDE	Res.	Res.	CC2DE	CC1DE	UDE	BIE	TIE	COMIE	Res.	Res.	CC2IE	CC1IE	UIE
	Reset value																		0	0			0	0	0	0	0	0			0	0	0
0x10	TIM15_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2OF	CC1OF	Res.	BIF	TIF	COMIF	Res.	Res.	CC2IF	CC1IF	UIF
	Reset value																						0	0		0	0	0			0	0	0

Table 551. TIM15 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x14	TIM15_EGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	TG	COMG	Res.	Res.	CC2G	CC1G	UG
	Reset value																									0	0	0			0	0	0
0x18	TIM15_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2F[3:0]				IC2 PSC [1:0]		CC2S [1:0]		IC1F[3:0]				IC1 PSC [1:0]		CC1S [1:0]	
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	TIM15_CCMR1 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC2M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	OC2CE	OC2M [2:0]			OC2PE	OC2FE	CC2S [1:0]		OC1CE	OC1M [2:0]			OC1PE	OC1FE	CC1S [1:0]	
	Reset value								0								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20	TIM15_CCER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2NP	Res.	CC2P	CC2E	CC1NP	CC1NE	CC1P	CC1E
	Reset value																									0		0	0	0	0	0	0
0x24	TIM15_CNT	UIFCPY or Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	TIM15_PSC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x2C	TIM15_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:0]																			
	Reset value													0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30	TIM15_RCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x34	TIM15_CCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38	TIM15_CCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38 - 0x40	Reserved	Res.																															
0x44	TIM15_BDTR	Res.	Res.	Res.	BKBID	Res.	BKDSRM	Res.	Res.	Res.	Res.	Res.	Res.	BKF[3:0]				MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]							
	Reset value				0		0							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48 - 0x50	Reserved	Res.																															

Table 551. TIM15 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x54	TIM15_DTR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPE	DTAE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTGF[7:0]							
	Reset value															0	0									0	0	0	0	0	0	0	0
0x58	Reserved	Res.																															
0x5C	TIM15_TISEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI2SEL[3:0]				Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value																					0	0	0	0					0	0	0	0
0x60	TIM15_AF1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKCMP4P	BKCMP3P	BKCMP2P	BKCMP1P	BKINP	BKCMP8E	BKCMP7E	BKCMP6E	BKCMP5E	BKCMP4E	BKCMP3E	BKCMP2E	BKCMP1E	BKINE
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x64	TIM15_AF2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCR SEL[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value														0	0	0																
0x68 - 0x3D8	Reserved	Res.																															
0x3DC	TIM15_DCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]				Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value													0	0	0	0				0	0	0	0	0				0	0	0	0	0
0x3E0	TIM15_DMAR	DMAB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

57.8 TIM16/TIM17 registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

57.8.1 TIMx control register 1 (TIMx_CR1)(x = 16 to 17)

Address offset: 0x00

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DITHEN	UIFREMAP	Res.	CKD[1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
			rw	rw		rw	rw	rw				rw	rw	rw	rw

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DITHEN**: Dithering enable
0: Dithering disabled
1: Dithering enabled
Note: The DITHEN bit can only be modified when CEN bit is reset.

Bit 11 **UIFREMAP**: UIF status bit remapping
0: No remapping. UIF status bit is not copied to TIMx_CNT register bit 31.
1: Remapping enabled. UIF status bit is copied to TIMx_CNT register bit 31.

Bit 10 Reserved, must be kept at reset value.

Bits 9:8 **CKD[1:0]**: Clock division
This bitfield indicates the division ratio between the timer clock (tim_ker_ck) frequency and the dead-time and sampling clock (t_{DTS})used by the dead-time generators and the digital filters (tim_tix),
00: $t_{DTS} = t_{tim_ker_ck}$
01: $t_{DTS} = 2*t_{tim_ker_ck}$
10: $t_{DTS} = 4*t_{tim_ker_ck}$
11: Reserved

Bit 7 **ARPE**: Auto-reload preload enable
0: TIMx_ARR register is not buffered
1: TIMx_ARR register is buffered

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **OPM**: One pulse mode
0: Counter is not stopped at update event
1: Counter stops counting at the next update event (clearing the bit CEN)

Bit 2 **URS**: Update request source
This bit is set and cleared by software to select the UEV event sources.
0: Any of the following events generate an update interrupt or DMA request if enabled. These events can be:
– Counter overflow/underflow
– Setting the UG bit
– Update generation through the slave mode controller
1: nly counter overflow/underflow generates an update interrupt or DMA request if enabled.

Bit 1 **UDIS**: Update disable

This bit is set and cleared by software to enable/disable UEV event generation.

0: UEV enabled. The Update (UEV) event is generated by one of the following events:

– Counter overflow/underflow

– Setting the UG bit

– Update generation through the slave mode controller

Buffered registers are then loaded with their preload values.

1: UEV disabled. The Update event is not generated, shadow registers keep their value (ARR, PSC, CCRx). However the counter and the prescaler are reinitialized if the UG bit is set or if a hardware reset is received from the slave mode controller.

Bit 0 **CEN**: Counter enable

0: Counter disabled

1: Counter enabled

Note: External clock and gated mode can work only if the CEN bit has been previously set by software. However trigger mode can set the CEN bit automatically by hardware.

57.8.2 TIMx control register 2 (TIMx_CR2)(x = 16 to 17)

Address offset: 0x04

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	OIS1N	OIS1	Res.	Res.	Res.	Res.	CCDS	CCUS	Res.	CCPC
						rw	rw					rw	rw		rw

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **OIS1N**: Output Idle state 1 (tim_oc1n output)

0: tim_oc1n = 0 after a dead-time when MOE = 0

1: tim_oc1n = 1 after a dead-time when MOE = 0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BKR register).

Bit 8 **OIS1**: Output Idle state 1 (tim_oc1 output)

0: tim_oc1 = 0 after a dead-time when MOE = 0

1: tim_oc1 = 1 after a dead-time when MOE = 0

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BKR register).

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **CCDS**: Capture/compare DMA selection

0: CCx DMA request sent when CCx event occurs

1: CCx DMA requests sent when update event occurs

Bit 2 **CCUS**: Capture/compare control update selection

0: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit only.

1: When capture/compare control bits are preloaded (CCPC = 1), they are updated by setting the COMG bit or when a rising edge occurs on tim_trgi (if available).

Note: This bit acts only on channels that have a complementary output.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **CCPC**: Capture/compare preloaded control
0: CCxE, CCxNE and OCxM bits are not preloaded
1: CCxE, CCxNE and OCxM bits are preloaded, after having been written, they are updated only when COM bit is set.
Note: This bit acts only on channels that have a complementary output.

57.8.3 TIMx DMA/interrupt enable register (TIMx_DIER)(x = 16 to 17)

Address offset: 0x0C

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC1DE	UDE	BIE	Res.	COMIE	Res.	Res.	Res.	CC1IE	UIE
						rw	rw	rw		rw				rw	rw

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **CC1DE**: Capture/Compare 1 DMA request enable
0: CC1 DMA request disabled
1: CC1 DMA request enabled

Bit 8 **UDE**: Update DMA request enable
0: Update DMA request disabled
1: Update DMA request enabled

Bit 7 **BIE**: Break interrupt enable
0: Break interrupt disabled
1: Break interrupt enabled

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMIE:** COM interrupt enable
0: COM interrupt disabled
1: COM interrupt enabled

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1IE**: Capture/Compare 1 interrupt enable
0: CC1 interrupt disabled
1: CC1 interrupt enabled

Bit 0 **UIE**: Update interrupt enable
0: Update interrupt disabled
1: Update interrupt enabled

57.8.4 TIMx status register (TIMx_SR)(x = 16 to 17)

Address offset: 0x10

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC1OF	Res.	BIF	Res.	COMIF	Res.	Res.	Res.	CC1IF	UIF
						rc_w0		rc_w0		rc_w0				rc_w0	rc_w0

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **CC1OF**: Capture/Compare 1 overcapture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing it to 0.

0: No overcapture has been detected

1: The counter value has been captured in TIMx_CCR1 register while CC1IF flag was already set

Bit 8 Reserved, must be kept at reset value.

Bit 7 **BIF**: Break interrupt flag

This flag is set by hardware as soon as the tim_brk input goes active. It can be cleared by software if the break input is not active.

0: No break event occurred

1: An active level has been detected on the break input

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMIF:** COM interrupt flag

This flag is set by hardware on a COM event (once the capture/compare control bits –CCxE, CCxNE, OCxM– have been updated). It is cleared by software.

0: No COM event occurred

1: COM interrupt pending

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1IF**: Capture/Compare 1 interrupt flag

This flag is set by hardware. It is cleared by software (input capture or output compare mode) or by reading the TIMx_CCR1 register (input capture mode only).

0: No compare match / No input capture occurred

1: A compare match or an input capture occurred

If channel CC1 is configured as output: this flag is set when the content of the counter TIMx_CNT matches the content of the TIMx_CCR1 register. When the content of TIMx_CCR1 is greater than the content of TIMx_ARR, the CC1IF bit goes high on the counter overflow (in up-counting and up/down-counting modes) or underflow (in down-counting mode). There are 3 possible options for flag setting in center-aligned mode, refer to the CMS bits in the TIMx_CR1 register for the full description.

If channel CC1 is configured as input: this bit is set when counter value has been captured in TIMx_CCR1 register (an edge has been detected on IC1, as per the edge sensitivity defined with the CC1P and CC1NP bits setting, in TIMx_CCER).

Bit 0 **UIF**: Update interrupt flag

This bit is set by hardware on an update event. It is cleared by software.

0: No update occurred.

1: Update interrupt pending. This bit is set by hardware when the registers are updated:

– At overflow regarding the repetition counter value (update if repetition counter = 0) and if the UDIS = 0 in the TIMx_CR1 register.

– When CNT is reinitialized by software using the UG bit in TIMx_EGR register, if URS = 0 and UDIS = 0 in the TIMx_CR1 register.

57.8.5 TIMx event generation register (TIMx_EGR)(x = 16 to 17)

Address offset: 0x14

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	Res.	COMG	Res.	Res.	Res.	CC1G	UG
								w		w				w	w

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **BG**: Break generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action.

1: A break event is generated. MOE bit is cleared and BIF flag is set. Related interrupt or DMA transfer can occur if enabled.

Bit 6 Reserved, must be kept at reset value.

Bit 5 **COMG:** Capture/Compare control update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action

1: When the CCPC bit is set, it is possible to update the CCxE, CCxNE and OCxM bits

Note: This bit acts only on channels that have a complementary output.

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **CC1G**: Capture/Compare 1 generation

This bit is set by software in order to generate an event, it is automatically cleared by hardware.

0: No action.

1: A capture/compare event is generated on channel 1:

If channel CC1 is configured as output:

CC1IF flag is set, Corresponding interrupt or DMA request is sent if enabled.

If channel CC1 is configured as input:

The current value of the counter is captured in TIMx_CCR1 register. The CC1IF flag is set, the corresponding interrupt or DMA request is sent if enabled. The CC1OF flag is set if the CC1IF flag was already high.

Bit 0 **UG**: Update generation

This bit can be set by software, it is automatically cleared by hardware.

0: No action.

1: Reinitialize the counter and generates an update of the registers. Note that the prescaler counter is cleared too (anyway the prescaler ratio is not affected).

57.8.6 TIMx capture/compare mode register 1 (TIMx_CCMR1) (x = 16 to 17)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for input capture mode (this section) or for output compare mode (next section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC1F[3:0]				IC1PSC[1:0]		CC1S[1:0]	
								rw	rw	rw	rw	rw	rw	rw	rw

Input capture mode

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **IC1F[3:0]**: Input capture 1 filter

This bitfield defines the frequency used to sample tim_ti1 input and the length of the digital filter applied to tim_ti1. The digital filter is made of an event counter in which N consecutive events are needed to validate a transition on the output:

0000: No filter, sampling is done at f_{DTS}
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N =
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

Bits 3:2 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on CC1 input (tim_ic1).
The prescaler is reset as soon as CC1E = 0 (TIMx_CCER register).
00: no prescaler, capture is done each time an edge is detected on the capture input.
01: capture is done once every 2 events
10: capture is done once every 4 events
11: capture is done once every 8 events

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1

Others: Reserved

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

57.8.7 TIMx capture/compare mode register 1 [alternate] (TIMx_CCMR1)(x = 16 to 17)

Address offset: 0x18

Reset value: 0x0000 0000

The same register can be used for output compare mode (this section) or for input capture mode (previous section). The direction of a channel is defined by configuring the corresponding CCxS bits. All the other bits of this register have a different function for input capture and for output compare modes. It is possible to combine both modes independently (for example channel 1 in input capture mode and channel 2 in output compare mode).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1CE	OC1M[2:0]			OC1PE	OC1FE	CC1S[1:0]	
								rw	rw	rw	rw	rw	rw	rw	rw

Output compare mode:

Bits 31:17 Reserved, must be kept at reset value.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **OC1CE**: Output Compare 1 clear enable

0: tim_oc1ref is not affected by the tim_ocref_clr input.

1: tim_oc1ref is cleared as soon as a High level is detected on tim_ocref_clr input.

Bits 16, 6:4 **OC1M[3:0]**: Output Compare 1 mode

These bits define the behavior of the output reference signal tim_oc1ref from which tim_oc1 and tim_oc1n are derived. tim_oc1ref is active high whereas tim_oc1 and tim_oc1n active level depends on CC1P and CC1NP bits.

0000: Frozen - The comparison between the output compare register TIMx_CCR1 and the counter TIMx_CNT has no effect on the outputs. This mode can be used when the timer serves as a software timebase. When the frozen mode is enabled during timer operation, the output keeps the state (active or inactive) it had before entering the frozen state.

0001: Set channel 1 to active level on match. tim_oc1ref signal is forced high when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0010: Set channel 1 to inactive level on match. tim_oc1ref signal is forced low when the counter TIMx_CNT matches the capture/compare register 1 (TIMx_CCR1).

0011: Toggle - tim_oc1ref toggles when TIMx_CNT = TIMx_CCR1.

0100: Force inactive level - tim_oc1ref is forced low.

0101: Force active level - tim_oc1ref is forced high.

0110: PWM mode 1 - Channel 1 is active as long as TIMx_CNT<TIMx_CCR1 else inactive.

0111: PWM mode 2 - Channel 1 is inactive as long as TIMx_CNT<TIMx_CCR1 else active.

Others: Reserved

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output).

In PWM mode, the OCREF level changes when the result of the comparison changes, when the output compare mode switches from “frozen” mode to “PWM” mode and when the output compare mode switches from “force active/inactive” mode to “PWM” mode.

Bit 3 **OC1PE**: Output Compare 1 preload enable

0: Preload register on TIMx_CCR1 disabled. TIMx_CCR1 can be written at anytime, the new value is taken in account immediately.

1: Preload register on TIMx_CCR1 enabled. Read/Write operations access the preload register. TIMx_CCR1 preload value is loaded in the active register at each update event.

Note: These bits can not be modified as long as LOCK level 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output).

Bit 2 **OC1FE**: Output Compare 1 fast enable

This bit decreases the latency between a trigger event and a transition on the timer output. It must be used in one-pulse mode (OPM bit set in TIMx_CR1 register), to have the output pulse starting as soon as possible after the starting trigger.

0: CC1 behaves normally depending on counter and CCR1 values even when the trigger is ON. The minimum delay to activate CC1 output when an edge occurs on the trigger input is 5 clock cycles.

1: An active edge on the trigger input acts like a compare match on CC1 output. Then, tim_ocx is set to the compare level independently of the result of the comparison. Delay to sample the trigger input and to activate CC1 output is reduced to 3 clock cycles. OC1FE acts only if the channel is configured in PWM1 or PWM2 mode.

Bits 1:0 **CC1S[1:0]**: Capture/Compare 1 selection

This bitfield defines the direction of the channel (input/output) as well as the used input.

00: CC1 channel is configured as output

01: CC1 channel is configured as input, tim_ic1 is mapped on tim_ti1

Others: Reserved

Note: CC1S bits are writable only when the channel is OFF (CC1E = 0 in TIMx_CCER).

57.8.8 TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17)

Address offset: 0x20

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1NP	CC1NE	CC1P	CC1E
												rw	rw	rw	rw

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **CC1NP**: Capture/Compare 1 complementary output polarity

CC1 channel configured as output:

0: tim_oc1n active high

1: tim_oc1n active low

CC1 channel configured as input:

This bit is used in conjunction with CC1P to define the polarity of tim_ti1fp1. Refer to the description of CC1P.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register) and CC1S = 00 (the channel is configured in output).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1NP active bit takes the new value from the preloaded bit only when a commutation event is generated.

Bit 2 **CC1NE**: Capture/Compare 1 complementary output enable

0: Off - tim_oc1n is not active. tim_oc1n level is then function of MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

1: On - tim_oc1n signal is output on the corresponding output pin depending on MOE, OSSI, OSSR, OIS1, OIS1N and CC1E bits.

Bit 1 **CC1P**: Capture/Compare 1 output polarity

0: OC1 active high (output mode) / Edge sensitivity selection (input mode, see below)

1: OC1 active low (output mode) / Edge sensitivity selection (input mode, see below)

When CC1 channel is configured as input, both CC1NP/CC1P bits select the active polarity of TI1FP1 and TI2FP1 for trigger or capture operations.

CC1NP = 0, CC1P = 0: non-inverted/rising edge. The circuit is sensitive to TIxFP1 rising edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is not inverted (trigger operation in gated mode).

CC1NP = 0, CC1P = 1: inverted/falling edge. The circuit is sensitive to TIxFP1 falling edge (capture or trigger operations in reset, external clock or trigger mode), TIxFP1 is inverted (trigger operation in gated mode).

CC1NP = 1, CC1P = 1: non-inverted/both edges/ The circuit is sensitive to both TIxFP1 rising and falling edges (capture or trigger operations in reset, external clock or trigger mode), TIxFP1is not inverted (trigger operation in gated mode).

CC1NP = 1, CC1P = 0: this configuration is reserved, it must not be used.

Note: This bit is not writable as soon as LOCK level 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

On channels that have a complementary output, this bit is preloaded. If the CCPC bit is set in the TIMx_CR2 register then the CC1P active bit takes the new value from the preloaded bit only when a Commutation event is generated.

Bit 0 **CC1E**: Capture/Compare 1 output enable

0: Capture mode disabled / OC1 is not active (see below)

1: Capture mode enabled / OC1 signal is output on the corresponding output pin

When CC1 channel is configured as output, the OC1 level depends on MOE, OSSI, OSSR, OIS1, OIS1N and CC1NE bits, regardless of the CC1E bits state. Refer to *Table 552* for details.

Table 552. Output control bits for complementary tim_oc1 and tim_oc1n channels with break feature (TIM16/TIM17)

<table>
<tr><th colspan="5">Control bits</th><th colspan="2">Output states(1)</th></tr>
<tr><th>MOE bit</th><th>OSSI bit</th><th>OSSR bit</th><th>CC1E bit</th><th>CC1NE bit</th><th>tim_oc1 output state</th><th>tim_oc1n output state</th></tr>
<tr><td rowspan="7">1</td><td rowspan="7">X</td><td>X</td><td>0</td><td>0</td><td colspan="2">Output Disabled (not driven by the timer: Hi-Z)
tim_oc1 = 0
tim_oc1n = 0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Output Disabled (not driven by the timer: Hi-Z)
tim_oc1 = 0</td><td>tim_oc1ref + Polarity
tim_oc1n = tim_oc1ref XOR CC1NP</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>tim_oc1ref + Polarity
tim_oc1 = tim_oc1ref XOR CC1P</td><td>Output Disabled (not driven by the timer: Hi-Z)
tim_oc1n = 0</td></tr>
<tr><td>X</td><td>1</td><td>1</td><td>tim_oc1ref + Polarity + dead-time</td><td>Complementary to tim_oc1ref (not tim_oc1ref) + Polarity + dead-time</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Off-State (output enabled with inactive state)
tim_oc1 = CC1P</td><td>tim_oc1ref + Polarity
tim_oc1n = tim_oc1ref XOR CC1NP</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>tim_oc1ref + Polarity
tim_oc1 = tim_oc1ref XOR CC1P</td><td>Off-State (output enabled with inactive state)
tim_oc1n = CC1NP</td></tr>
<tr><td rowspan="5">0</td><td>0</td><td rowspan="5">X</td><td>X</td><td>X</td><td colspan="2" rowspan="2">Output disabled (not driven by the timer: Hi-Z)</td></tr>
<tr><td rowspan="4">1</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td colspan="2" rowspan="3">Off-State (output enabled with inactive state)
Asynchronously: tim_oc1 = CC1P, tim_oc1n = CC1NP
Then if the clock is present: tim_oc1 = OIS1 and tim_oc1n = OIS1N after a dead-time, assuming that OIS1 and OIS1N do not correspond to tim_oc1 and tim_oc1n both in active state</td></tr>
<tr><td>1</td><td>0</td></tr>
<tr><td>1</td><td>1</td></tr>
</table>

1. When both outputs of a channel are not used (control taken over by the GPIO controller), the OIS1, OIS1N, CC1P and CC1NP bits must be kept cleared.

Note: The state of the external I/O pins connected to the complementary tim_oc1 and tim_oc1n channels depends on the tim_oc1 and tim_oc1n channel state and GPIO control and alternate function selection registers.

57.8.9 TIMx counter (TIMx_CNT)(x = 16 to 17)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UIF CPY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UIFCPY**: UIF Copy

This bit is a read-only copy of the UIF bit of the TIMx_ISR register. If the UIFREMAP bit in TIMx_CR1 is reset, bit 31 is reserved.

Bits 30:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]**: Counter value

Non-dithering mode (DITHEN = 0)

The register holds the counter value.

Dithering mode (DITHEN = 1)

The register only holds the non-dithered part in CNT[15:0]. The fractional part is not available.

57.8.10 TIMx prescaler (TIMx_PSC)(x = 16 to 17)

Address offset: 0x28

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:0 **PSC[15:0]**: Prescaler value

The counter clock frequency (tim_cnt_ck) is equal to $f_{tim_psc_ck}$ / (PSC[15:0] + 1).

PSC contains the value to be loaded in the active prescaler register at each update event (including when the counter is cleared through UG bit of TIMx_EGR register or through trigger controller when configured in “reset mode”).

57.8.11 TIMx auto-reautoreload register (TIMx_ARR)(x = 16 to 17)

Address offset: 0x2C

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **ARR[19:0]**: Auto-reload value

ARR is the value to be loaded in the actual auto-reload register.

Refer to the *Section 57.4.3: Time-base unit on page 2829* for more details about ARR update and behavior.

The counter is blocked while the auto-reload value is null.

Non-dithering mode (DITHEN = 0)

The register holds the auto-reload value in ARR[15:0]. The ARR[19:16] bits are reset.

Dithering mode (DITHEN = 1)

The register holds the integer part in ARR[19:4]. The ARR[3:0] bitfield contains the dithered part.

57.8.12 TIMx repetition counter register (TIMx_RCR)(x = 16 to 17)

Address offset: 0x30

Reset value: 0x0000

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **REP[7:0]**: Repetition counter reload value

This bitfield defines the update rate of the compare registers (i.e. periodic transfers from preload to active registers) when preload registers are enable. It also defines the update interrupt generation rate, if this interrupt is enable.

When the repetition down-counter reaches zero, an update event is generated and it restarts counting from REP value. As the repetition counter is reloaded with REP value only at the repetition update event UEV, any write to the TIMx_RCR register is not taken in account until the next repetition update event.

It means in PWM mode (REP+1) corresponds to the number of PWM periods in edge-aligned mode:

– The number of PWM periods in edge-aligned mode.
– The number of half PWM period in center-aligned mode.

57.8.13 TIMx capture/compare register 1 (TIMx_CCR1)(x = 16 to 17)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **CCR1[19:0]**: Capture/Compare 1 value

If channel CC1 is configured as output:
CCR1 is the value to be loaded in the actual capture/compare 1 register (preload value).
It is loaded permanently if the preload feature is not selected in the TIMx_CCMR1 register (bit OC1PE). Else the preload value is copied in the active capture/compare 1 register when an update event occurs.
The active capture/compare register contains the value to be compared to the counter TIMx_CNT and signaled on tim_oc1 output.
Non-dithering mode (DITHEN = 0)
The register holds the compare value in CCR1[15:0]. The CCR1[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the integer part in CCR1[19:4]. The CCR1[3:0] bitfield contains the dithered part.
If channel CC1 is configured as input:
CCR1 is the counter value transferred by the last input capture 1 event (tim_ic1).
Non-dithering mode (DITHEN = 0)
The register holds the capture value in CCR1[15:0]. The CCR1[19:16] bits are reset.
Dithering mode (DITHEN = 1)
The register holds the capture in CCR1[19:4]. The CCR1[3:0] bits are reset.

57.8.14 TIMx break and dead-time register (TIMx_BDTR)(x = 16 to 17)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	BKBID	Res.	BK DSRM	Res.	Res.	Res.	Res.	Res.	Res.	BKF[3:0]			
			rw		rw							rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK[1:0]		DTG[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Note: As the BKBID, BKDSRM, BKF[3:0], AOE, BKP, BKE, OSSI, OSSR, and DTG[7:0] bits may be write-locked depending on the LOCK configuration, it may be necessary to configure all of them during the first write access to the TIMx_BDTR register.

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **BKBID**: Break Bidirectional

0: Break input tim_brk in input mode
1: Break input tim_brk in bidirectional mode
In the bidirectional mode (BKBID bit set to 1), the break input is configured both in input mode and in open drain output mode. Any active break event asserts a low logic level on the Break input to indicate an internal break event to external devices.

Note: This bit cannot be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **BKDSRM**: Break Disarm

0: Break input tim_brk is armed
1: Break input tim_brk is disarmed
This bit is cleared by hardware when no break source is active.
The BKDSRM bit must be set by software to release the bidirectional output control (open-drain output in Hi-Z state) and then be polled it until it is reset by hardware, indicating that the fault condition has disappeared.

Note: Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bits 25:20 Reserved, must be kept at reset value.

Bits 19:16 **BKF[3:0]**: Break filter

This bitfield defines the frequency used to sample tim_brk input and the length of the digital filter applied to tim_brk. The digital filter is made of an event counter in which N events are needed to validate a transition on the output:

0000: No filter, tim_brk acts asynchronously
0001: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 2
0010: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 4
0011: $f_{SAMPLING} = f_{tim_ker_ck}$, N = 8
0100: $f_{SAMPLING} = f_{DTS}/2$, N = 6
0101: $f_{SAMPLING} = f_{DTS}/2$, N = 8
0110: $f_{SAMPLING} = f_{DTS}/4$, N = 6
0111: $f_{SAMPLING} = f_{DTS}/4$, N = 8
1000: $f_{SAMPLING} = f_{DTS}/8$, N = 6
1001: $f_{SAMPLING} = f_{DTS}/8$, N = 8
1010: $f_{SAMPLING} = f_{DTS}/16$, N = 5
1011: $f_{SAMPLING} = f_{DTS}/16$, N = 6
1100: $f_{SAMPLING} = f_{DTS}/16$, N = 8
1101: $f_{SAMPLING} = f_{DTS}/32$, N = 5
1110: $f_{SAMPLING} = f_{DTS}/32$, N = 6
1111: $f_{SAMPLING} = f_{DTS}/32$, N = 8

This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 15 **MOE**: Main output enable

This bit is cleared asynchronously by hardware as soon as the tim_brk input is active. It is set by software or automatically depending on the AOE bit. It is acting only on the channels which are configured in output.

0: tim_oc1 and tim_oc1n outputs are disabled or forced to idle state depending on the OSSI bit.

1: tim_oc1 and tim_oc1n outputs are enabled if their respective enable bits are set (CC1E, CC1NE in TIMx_CCER register)

See tim_oc1/tim_oc1n enable description for more details (*Section 57.8.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17)*).

Bit 14 **AOE**: Automatic output enable

0: MOE can be set only by software

1: MOE can be set by software or automatically at the next update event (if the tim_brk input is not active)

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 13 **BKP**: Break polarity

0: Break input tim_brk is active low
1: Break input tim_brk is active high

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 12 **BKE**: Break enable

0: Break inputs (tim_brk and tim_sys_brk event) disabled
1; Break inputs (tim_brk and tim_sys_brk event) enabled

Note: This bit cannot be modified when LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Any write operation to this bit takes a delay of 1 APB clock cycle to become effective.

Bit 11 **OSSR**: Off-state selection for Run mode

This bit is used when MOE = 1 on channels that have a complementary output which are configured as outputs. OSSR is not implemented if no complementary output is implemented in the timer.

See tim_oc1/tim_oc1n enable description for more details (*Section 57.8.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17)*).

0: When inactive, tim_oc1/tim_oc1n outputs are disabled (the timer releases the output control which is taken over by the GPIO, which forces a Hi-Z state)

1: When inactive, tim_oc1/tim_oc1n outputs are enabled with their inactive level as soon as CC1E = 1 or CC1NE = 1 (the output is still controlled by the timer).

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **OSSI**: Off-state selection for Idle mode

This bit is used when MOE = 0 on channels configured as outputs.

See tim_oc1/tim_oc1n enable description for more details (*Section 57.8.8: TIMx capture/compare enable register (TIMx_CCER)(x = 16 to 17)*).

0: When inactive, tim_oc1/tim_oc1n outputs are disabled (tim_oc1/tim_oc1n enable output signal = 0)

1: When inactive, tim_oc1/tim_oc1n outputs are forced first with their idle level as soon as CC1E = 1 or CC1NE = 1. tim_oc1/tim_oc1n enable output signal = 1)

Note: This bit can not be modified as soon as the LOCK level 2 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 9:8 **LOCK[1:0]**: Lock configuration

These bits offer a write protection against software errors.

00: LOCK OFF - No bit is write protected

01: LOCK Level 1 = DTG bits in TIMx_BDTR register, OISx and OISxN bits in TIMx_CR2 register and BKBID/BKE/BKP/AOE bits in TIMx_BDTR register can no longer be written.

10: LOCK Level 2 = LOCK Level 1 + CC Polarity bits (CCxP/CCxNP bits in TIMx_CCER register, as long as the related channel is configured in output through the CCxS bits) as well as OSSR and OSSI bits can no longer be written.

11: LOCK Level 3 = LOCK Level 2 + CC Control bits (OCxM and OCxPE bits in TIMx_CCMRx registers, as long as the related channel is configured in output through the CCxS bits) can no longer be written.

Note: The LOCK bits can be written only once after the reset. Once the TIMx_BDTR register has been written, their content is frozen until the next reset.

Bits 7:0 **DTG[7:0]**: Dead-time generator setup

This bitfield defines the duration of the dead-time inserted between the complementary outputs. DT correspond to this duration.

DTG[7:5] = 0xx → DT = DTG[7:0]x t_{dtg} with $t_{dtg} = t_{DTS}$

DTG[7:5] = 10x → DT = (64+DTG[5:0])xt_{dtg} with $T_{dtg} = 2xt_{DTS}$

DTG[7:5] = 110 → DT = (32+DTG[4:0])xt_{dtg} with $T_{dtg} = 8xt_{DTS}$

DTG[7:5] = 111 → DT = (32+DTG[4:0])xt_{dtg} with $T_{dtg} = 16xt_{DTS}$

Example if T_{DTS}=125 ns (8 MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

57.8.15 TIMx timer deadtime register 2 (TIMx_DTR2)(x = 16 to 17)

Address offset: 0x054

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPE	DTAE
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTGF[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **DTPE**: Deadtime preload enable

0: Deadtime value is not preloaded

1: Deadtime value preload is enabled

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 16 **DTAE**: Deadtime asymmetric enable

0: Deadtime on rising and falling edges are identical, and defined with DTG[7:0] register

1: Deadtime on rising edge is defined with DTG[7:0] register and deadtime on falling edge is defined with DTGF[7:0] bits.

Note: This bit can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **DTGF[7:0]**: Dead-time falling edge generator setup

This bitfield defines the duration of the dead-time inserted between the complementary outputs, on the falling edge.

DTGF[7:5] = 0xx → DTF = DTGF[7:0]x t_{dtg} with $t_{dtg} = t_{DTS}$.

DTGF[7:5] = 10x → DTF = (64+DTGF[5:0])xt_{dtg} with $T_{dtg} = 2xt_{DTS}$.

DTGF[7:5] = 110 → DTF = (32+DTGF[4:0])xt_{dtg} with $T_{dtg} = 8xt_{DTS}$.

DTGF[7:5] = 111 → DTF = (32+DTGF[4:0])xt_{dtg} with $T_{dtg} = 16xt_{DTS}$.

Example if T_{DTS} = 125 ns (8 MHz), dead-time possible values are:

0 to 15875 ns by 125 ns steps,

16 µs to 31750 ns by 250 ns steps,

32 µs to 63 µs by 1 µs steps,

64 µs to 126 µs by 2 µs steps

Note: This bitfield can not be modified as long as LOCK level 1, 2 or 3 has been programmed (LOCK bits in TIMx_BDTR register).

57.8.16 TIMx input selection register (TIMx_TISEL)(x = 16 to 17)

Address offset: 0x5C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **TI1SEL[3:0]**: selects tim_ti1_in[15:0] input
0000: TIMx_CH1 input (tim_ti1_in0)
0001: tim_ti1_in1
...
1111: tim_ti1_in15
Refer to *Section 57.4.2: TIM15/TIM16/TIM17 pins and internal signals* for interconnects list.

57.8.17 TIMx alternate function register 1 (TIMx_AF1)(x = 16 to 17)

Address offset: 0x060

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	BK CMP4P	BK CMP3P	BK CMP2P	BK CMP1P	BKINP	BK CMP8E	BK CMP7E	BK CMP6E	BK CMP5E	BK CMP4E	BK CMP3E	BK CMP2E	BK CMP1E	BKINE
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Refer to *Section 57.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation.

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **BKCMP4P**: tim_brk_cmp4 input polarity
This bit selects the tim_brk_cmp4 input sensitivity. It must be programmed together with the BKP polarity bit.
0: tim_brk_cmp4 input is active high
1: tim_brk_cmp4 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 12 **BKCMP3P**: tim_brk_cmp3 input polarity

This bit selects the tim_brk_cmp3 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp3 input is active high

1: tim_brk_cmp3 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 11 **BKCMP2P**: tim_brk_cmp2 input polarity

This bit selects the tim_brk_cmp2 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp2 input is active high

1: tim_brk_cmp2 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 10 **BKCMP1P**: tim_brk_cmp1 input polarity

This bit selects the tim_brk_cmp1 input sensitivity. It must be programmed together with the BKP polarity bit.

0: tim_brk_cmp1 input is active high

1: tim_brk_cmp1 input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 9 **BKINP**: TIMx_BKIN input polarity

This bit selects the TIMx_BKIN alternate function input sensitivity. It must be programmed together with the BKP polarity bit.

0: TIMx_BKIN input is active high

1: TIMx_BKIN input is active low

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 8 **BKCMP8E**: tim_brk_cmp8 enable

This bit enables the tim_brk_cmp8 for the timer's tim_brk input. tim_brk_cmp8 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp8 input disabled

1: tim_brk_cmp8 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 7 **BKCMP7E**: tim_brk_cmp7 enable

This bit enables the tim_brk_cmp7 for the timer's tim_brk input. tim_brk_cmp7 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp7 input disabled

1: tim_brk_cmp7 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 6 **BKCMP6E**: tim_brk_cmp6 enable

This bit enables the tim_brk_cmp6 for the timer's tim_brk input. tim_brk_cmp6 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp6 input disabled

1: tim_brk_cmp6 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 5 **BKCMP5E**: tim_brk_cmp5 enable

This bit enables the tim_brk_cmp5 for the timer's tim_brk input. tim_brk_cmp5 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp5 input disabled

1: tim_brk_cmp5 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 4 **BKCMP4E**: tim_brk_cmp4 enable

This bit enables the tim_brk_cmp4 for the timer's tim_brk input. tim_brk_cmp4 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp4 input disabled

1: tim_brk_cmp4 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 3 **BKCMP3E**: tim_brk_cmp3 enable

This bit enables the tim_brk_cmp3 for the timer's tim_brk input. tim_brk_cmp3 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp3 input disabled

1: tim_brk_cmp3 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 2 **BKCMP2E**: tim_brk_cmp2 enable

This bit enables the tim_brk_cmp2 for the timer's tim_brk input. tim_brk_cmp2 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp2 input disabled

1: tim_brk_cmp2 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 1 **BKCMP1E**: tim_brk_cmp1 enable

This bit enables the tim_brk_cmp1 for the timer's tim_brk input. tim_brk_cmp1 output is 'ORed' with the other tim_brk sources.

0: tim_brk_cmp1 input disabled

1: tim_brk_cmp1 input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bit 0 **BKINE**: TIMx_BKIN input enable

This bit enables the TIMx_BKIN alternate function input for the timer's tim_brk input. TIMx_BKIN input is 'ORed' with the other tim_brk sources.

0: TIMx_BKIN input disabled

1: TIMx_BKIN input enabled

Note: This bit can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

57.8.18 TIMx alternate function register 2 (TIMx_AF2)(x = 16 to 17)

Address offset: 0x064

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCRSEL[2:0]		
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **OCRSEL[2:0]**: tim_ocref_clr source selection

These bits select the tim_ocref_clr input source.

000: tim_ocref_clr0

001: tim_ocref_clr1

010: tim_ocref_clr2

011: tim_ocref_clr3

100: tim_ocref_clr4

101: tim_ocref_clr5

110: tim_ocref_clr6

111: tim_ocref_clr7

Refer to *Section 57.4.2: TIM15/TIM16/TIM17 pins and internal signals* for product specific implementation.

Note: These bits can not be modified as long as LOCK level 1 has been programmed (LOCK bits in TIMx_BDTR register).

Bits 15:0 Reserved, must be kept at reset value.

57.8.19 TIMx DMA control register (TIMx_DCR)(x = 16 to 17)

Address offset: 0x3DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
			rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **DBSS[3:0]**: DMA burst source selection

This bitfield defines the interrupt source that triggers the DMA burst transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address).

0000: Reserved

0001: Update

0010: CC1

Other: reserved

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DBL[4:0]**: DMA burst length

This 5-bitfield defines the length of DMA transfers (the timer recognizes a burst transfer when a read or a write access is done to the TIMx_DMAR address), i.e. the number of transfers. Transfers can be in half-words or in bytes (see example below).

00000: 1 transfer,
00001: 2 transfers,
00010: 3 transfers,
...
10001: 18 transfers.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **DBA[4:0]**: DMA base address

This 5-bitfield defines the base-address for DMA transfers (when read/write access are done through the TIMx_DMAR address). DBA is defined as an offset starting from the address of the TIMx_CR1 register.

Example:
00000: TIMx_CR1,
00001: TIMx_CR2,
00010: TIMx_SMCR,
...

Example: Let us consider the following transfer: DBL = 7 transfers and DBA = TIMx_CR1. In this case the transfer is done to/from 7 registers starting from the TIMx_CR1 address.

57.8.20 TIM16/TIM17 DMA address for full transfer (TIMx_DMAR)(x = 16 to 17)

Address offset: 0x3E0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMAB[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAB[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DMAB[31:0]**: DMA register for burst accesses

A read or write operation to the DMAR register accesses the register located at the address

(TIMx_CR1 address) + (DBA + DMA index) x 4

where TIMx_CR1 address is the address of the control register 1, DBA is the DMA base address configured in TIMx_DCR register, DMA index is automatically controlled by the DMA transfer, and ranges from 0 to DBL (DBL configured in TIMx_DCR).

57.8.21 TIM16/TIM17 register map

TIM16/TIM17 registers are mapped as 16-bit addressable registers as described in the table below:

Table 553. TIM16/TIM17 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TIMx_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UIFREMA	Res.	CKD [1:0]		ARPE	Res.	Res.	Res.	OPM	URS	UDIS	CEN
	Reset value																					0		0	0	0				0	0	0	0
0x04	TIMx_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OIS1N	OIS1	Res.	Res.	Res.	Res.	CCDS	CCUS	Res.	CCPC
	Reset value																							0	0					0	0		0
0x08	Reserved	Res.																															
0x0C	TIMx_DIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1DE	UDE	BIE		COMIE	Res.	Res.	Res.	CC1IE	UIE
	Reset value																							0	0	0		0				0	0
0x10	TIMx_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1OF	Res.	BIF	Res.	COMIF	Res.	Res.	Res.	CC1IF	UIF
	Reset value																							0		0		0				0	0
0x14	TIMx_EGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BG	Res.	COMG	Res.	Res.	Res.	CC1G	UG
	Reset value																									0		0				0	0
0x18	TIMx_CCMR1 Input Capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC1F[3:0]				IC1 PSC [1:0]		CC1 S [1:0]	
	Reset value																									0	0	0	0	0	0	0	0
	TIMx_CCMR1 Output Compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1M[3]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OC1CE	OC1M [2:0]				OC1PE	OC1FE	CC1 S [1:0]	
	Reset value																0									0	0	0	0		0	0	0	0
0x1C	Reserved	Res.																															
0x20	TIMx_CCER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC1NP	CC1NE	CC1P	CC1E
	Reset value																													0	0	0	0
0x24	TIMx_CNT	UIFCPY or Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value	0																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x28	TIMx_PSC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 553. TIM16/TIM17 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2C	TIMx_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[19:0]																			
	Reset value													0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30	TIMx_RCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x34	TIMx_CCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x38 - 0x40	Reserved	Res.																															
0x44	TIMx_BDTR	Res.	Res.	Res.	BKBID	Res.	BKDSRM	Res.	Res.	Res.	Res.	Res.	Res.	BKF[3:0]				MOE	AOE	BKP	BKE	OSSR	OSSI	LOCK [1:0]		DT[7:0]							
	Reset value				0		0							0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48 - 0x50	Reserved	Res.																															
0x54	TIMx_DTR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTPE	DTAE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTGF[7:0]							
	Reset value															0	0									0	0	0	0	0	0	0	0
0x58	Reserved	Res.																															
0x5C	TIMx_TISEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TI1SEL[3:0]			
	Reset value																													0	0	0	0
0x60	TIMx_AF1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKCMP4P	BKCMP3P	BKCMP2P	BKCMP1P	BKINP	BKCMP8E	BKCMP7E	BKCMP6E	BKCMP5E	BKCMP4E	BKCMP3E	BKCMP2E	BKCMP1E	BKINE	
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	1	
0x64	TIMx_AF2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OCR SEL[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value														0	0	0																
0x6 - 0x3D8	Reserved	Res.																															
0x3DC	TIMx_DCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBSS[3:0]				Res.	Res.	Res.	DBL[4:0]					Res.	Res.	Res.	DBA[4:0]				
	Reset value													0	0	0	0				0	0	0	0	0				0	0	0	0	0
0x3E0	TIMx_DMAR	DMAB[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

58 Low-power timer (LPTIM)

58.1 LPTIM introduction

The LPTIM is a 16-bit timer that benefits from the ultimate developments in power consumption reduction. Thanks to its diversity of clock sources, the LPTIM is able to keep running in all power modes except for Standby mode. Given its capability to run even with no internal clock source, the LPTIM can be used as a “Pulse Counter” which can be useful in some applications. The LPTIM capability to wake up the system from low-power modes, makes it suitable to realize “Timeout functions” with extremely low power consumption.

The LPTIM introduces a flexible clock scheme that provides the needed functionalities and performance, while minimizing the power consumption.

58.2 LPTIM main features

- 16-bit upcounter
- 3-bit prescaler with 8 possible dividing factors (1,2,4,8,16,32,64,128)
- Selectable clock
  - Internal clock sources: configurable internal clock source (see RCC section)
  - External clock source over LPTIM input (working with no LP oscillator running, used by pulse counter application)
- 16-bit ARR autoreload register
- 16-bit capture/compare register
- Continuous/One-shot mode
- Selectable software/hardware input trigger
- Programmable digital glitch filter
- Configurable output: Pulse, PWM
- Configurable I/O polarity
- Encoder mode
- Repetition counter
- Up to 2 independent channels for:
  - Input capture
  - PWM generation (edge-aligned mode)
  - One-pulse mode output
- Interrupt generation on 10 events
- DMA request generation on the following events:
  - Update event
  - Input capture

58.3 LPTIM implementation

The table below describes LPTIM implementation on STM32N6x5/x7xx devices. The full set of features is implemented in LPTIM1, LPTIM2 and LPTIM3. LPTIM4 and LPTIM5 support a smaller set of features.

Table 554. STM32N6x5/x7xx LPTIM features

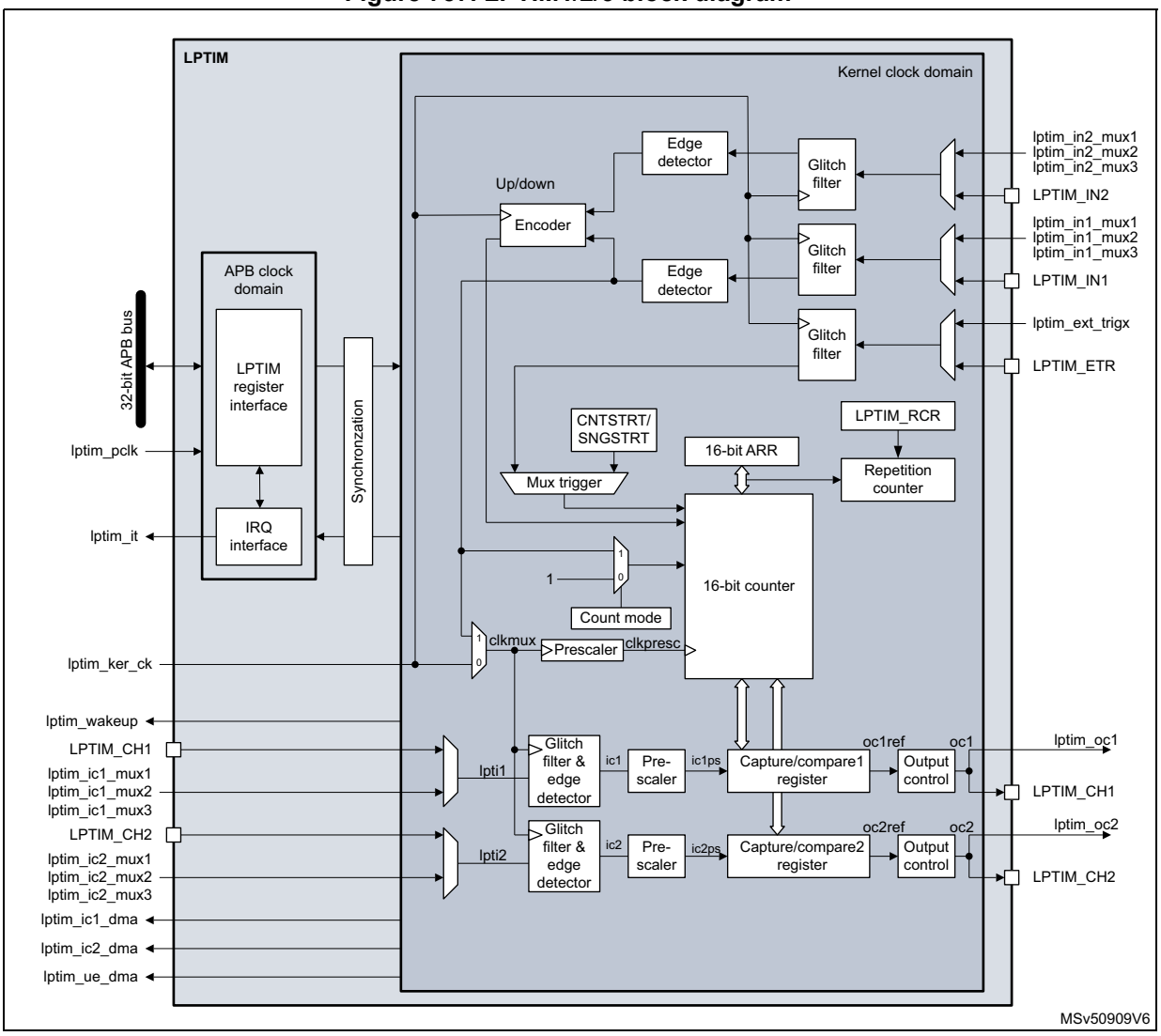
LPTIM modes/features[1]	LPTIM1	LPTIM2	LPTIM3	LPTIM4	LPTIM5
Encoder mode	X	X	X	-	-
PWM mode	X	X	X	X	X
Input Capture	X	X	X	-	-
Number of channels	2	2	2	0	0
Number of DMA requests	3	3	3	0	0
Wake-up in Stop mode	X	X	X	X	X

1. X = supported.

58.4 LPTIM functional description

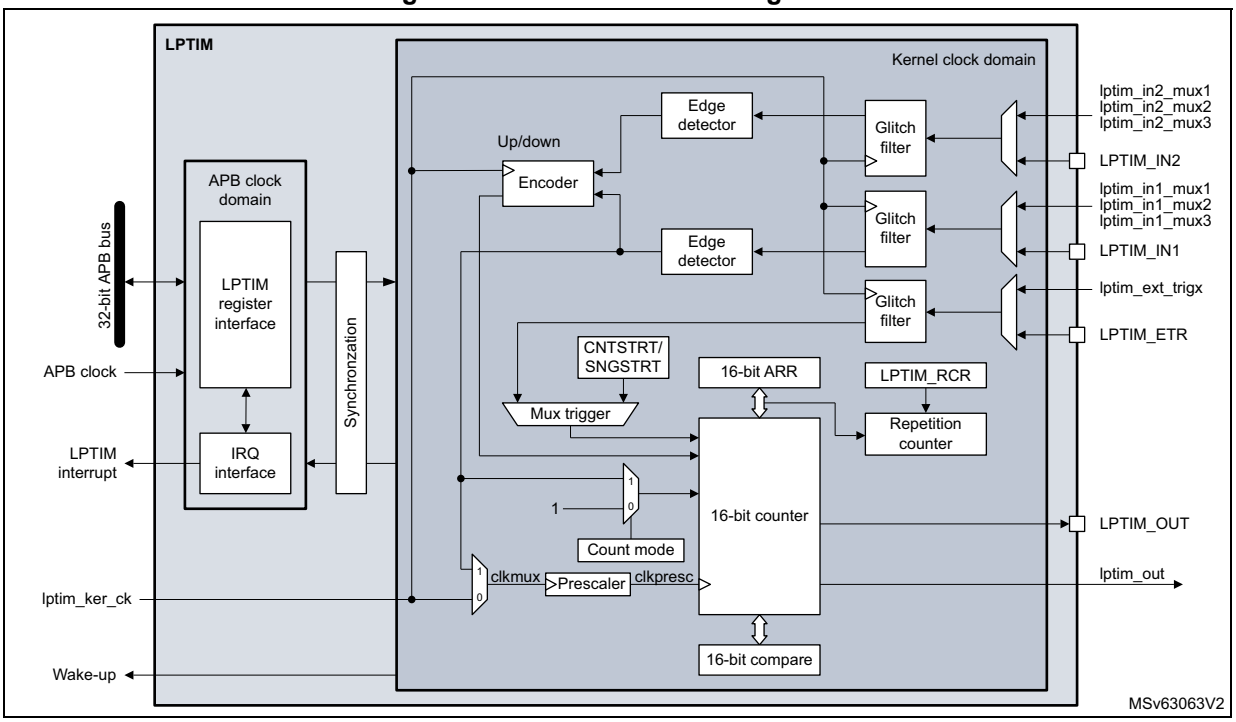
58.4.1 LPTIM block diagram

Figure 757. LPTIM1/2/3 block diagram(1)



1. Some I/Os may not be available, refer to *Section 58.4.2: LPTIM pins and internal signals*.

Figure 758. LPTIM4/5 block diagram(1)



1. Some I/Os may not be available, refer to *Section 58.4.2: LPTIM pins and internal signals*.

58.4.2 LPTIM pins and internal signals

The following tables provide the list of LPTIM pins and internal signals, respectively.

Table 555. LPTIM1/2/3 input/output pins

Pin name	Pin type	Description
LPTIM_IN1	Digital input	LPTIM Input 1 from GPIO pin on mux input 0
LPTIM_IN2	Digital input	LPTIM Input 2 from GPIO pin on mux input 0
LPTIM_ETR	Digital input	LPTIM external trigger GPIO pin
LPTIM_CH1	Digital input/output	LPTIM channel 1 input/output GPIO pin
LPTIM_CH2	Digital input/output	LPTIM channel 2 input/output GPIO pin

Table 556. LPTIM4/5 input/output pins

Pin name	Pin type	Description
LPTIM_IN1	Digital input	LPTIM Input 1 from GPIO pin on mux input 0
LPTIM_ETR	Digital input	LPTIM external trigger GPIO pin
LPTIM_OUT	Digital output	LPTIM output GPIO pin

Table 557. LPTIM1/2/3 internal signals

Signal name	Signal type	Description
lptim_pclk	Digital input	LPTIM APB clock domain
lptim_ker_ck	Digital input	LPTIM kernel clock
lptim_in1_mux1	Digital input	Internal LPTIM input 1 connected to mux input 1
lptim_in1_mux2	Digital input	Internal LPTIM input 1 connected to mux input 2
lptim_in1_mux3	Digital input	Internal LPTIM input 1 connected to mux input 3
lptim_in2_mux1	Digital input	Internal LPTIM input 2 connected to mux input 1
lptim_in2_mux2	Digital input	Internal LPTIM input 2 connected to mux input 2
lptim_in2_mux3	Digital input	Internal LPTIM input 2 connected to mux input 3
lptim_ic1_mux1	Digital input	Internal LPTIM input capture 1 connected to mux input 1
lptim_ic1_mux2	Digital input	Internal LPTIM input capture 1 connected to mux input 2
lptim_ic1_mux3	Digital input	Internal LPTIM input capture 1 connected to mux input 3
lptim_ic2_mux1	Digital input	Internal LPTIM input capture 2 connected to mux input 1
lptim_ic2_mux2	Digital input	Internal LPTIM input capture 2 connected to mux input 2
lptim_ic2_mux3	Digital input	Internal LPTIM input capture 2 connected to mux input 3
lptim_ext_trigx	Digital input	LPTIM external trigger input x
lptim_it	Digital output	LPTIM global interrupt
lptim_wakeup	Digital output	LPTIM wake-up event
lptim_ic1_dma	Digital output	LPTIM input capture 1 DMA request
lptim_ic2_dma	Digital output	LPTIM input capture 2 DMA request
lptim_ue_dma	Digital output	LPTIM update event DMA request

Table 558. LPTIM4/5 internal signals

Signal name	Signal type	Description
lptim_pclk	Digital input	LPTIM APB clock domain
lptim_ker_ck	Digital input	LPTIM kernel clock
lptim_in1_mux1	Digital input	Internal LPTIM input 1 connected to mux input 1
lptim_in1_mux2	Digital input	Internal LPTIM input 1 connected to mux input 2
lptim_in1_mux3	Digital input	Internal LPTIM input 1 connected to mux input 3
lptim_ext_trigx	Digital input	LPTIM external trigger input x
lptim_out	Digital output	LPTIM counter output
lptim_it	Digital output	LPTIM global interrupt
lptim_wakeup	Digital output	LPTIM wake-up event

58.4.3 LPTIM input and trigger mapping

The LPTIM external trigger and input connections are detailed hereafter.

Table 559. LPTIM1/2/3/4/5 external trigger connections

TRIGSEL	External trigger				
	LPTIM1	LPTIM2	LPTIM3	LPTIM4	LPTIM5
lptim_ext_trig0	GPIO				
lptim_ext_trig1	rtc_alra_trg		lptim2_ch1	lptim2_ch1	lptim2_ch1
lptim_ext_trig2	rtc_alrb_trg		Reserved	lptim3_ch1	lptim3_ch1
lptim_ext_trig3	tamp_trg1		lptim4_out	Reserved	lptim4_out
lptim_ext_trig4	tamp_trg2		lptim5_out	lptim5_out	Reserved
lptim_ext_trig5	tamp_trg3		sai1_fs_a_in	sai2_fs_a_in	Reserved
lptim_ext_trig6	Reserved		sai1_fs_b_in	sai2_fs_b_in	Reserved
lptim_ext_trig7	Reserved	Reserved	Reserved	Reserved	Reserved

Table 560. LPTIM1/2/3 input 1 connections

lptim_in1_mux	LPTIM1/2/3 input 1 connected to
lptim_in1_mux0	GPIO
lptim_in1_mux1	Reserved
lptim_in1_mux2	Reserved
lptim_in1_mux3	Reserved

Table 561. LPTIM1/2/3 input 2 connections

lptim_in2_mux	LPTIM1/2 input 2 connected to
lptim_in2_mux0	GPIO
lptim_in2_mux1	Reserved
lptim_in2_mux2	Reserved
lptim_in2_mux3	Reserved

Table 562. LPTIM1/2/3 input capture 1 connections

lptim_ic1_mux	LPTIM1 input capture 1 connected to	LPTIM2 input capture 1 connected to	LPTIM3 input capture 1 connected to
lptim_ic1_mux0	GPIO	GPIO	GPIO
lptim_ic1_mux1	Reserved	I3C1 IBI request (_IBIAck)	I3C2 IBI request (_IBIAck)
lptim_ic1_mux2	Reserved	Reserved	Reserved
lptim_ic1_mux3	Reserved	Reserved	Reserved

Table 563. LPTIM1 input capture 2 connections

lptim_ic2_mux	LPTIM1 input capture 2 connected to
lptim_ic2_mux0	I/O
lptim_ic2_mux1	LSI
lptim_ic2_mux2	LSE
lptim_ic2_mux3	Reserved

Table 564. LPTIM2 input capture 2 connections

lptim_ic2_mux	LPTIM2 input capture 2 connected to
lptim_ic2_mux0	I/O
lptim_ic2_mux1	HSI/1024
lptim_ic2_mux2	CSI/128
lptim_ic2_mux3	I3C2 IBI request (_IBIAck)

Table 565. LPTIM3 input capture 2 connections

lptim_ic2_mux	LPTIM3 input capture 2 connected to
lptim_ic2_mux0	I/O
lptim_ic2_mux1	Reserved
lptim_ic2_mux2	Reserved
lptim_ic2_mux3	I3C1 IBI request (_IBIAck)

58.4.4 LPTIM reset and clocks

The LPTIM can be clocked using several clock sources. It can be clocked using an internal clock signal which can be any configurable internal clock source selectable through the RCC (see RCC section for more details). Also, the LPTIM can be clocked using an external clock signal injected on its external Input1. When clocked with an external clock source, the LPTIM can run in one of the following configurations:

- The first configuration is when the LPTIM is clocked by an external signal but in the same time an internal clock signal is provided to the LPTIM from configurable internal clock source (see RCC section).
- The second configuration is when the LPTIM is solely clocked by an external clock source through its external Input1. This configuration is the one used to realize Timeout function or pulse counter function when all the embedded oscillators are turned off after entering a low-power mode.

Programming the CKSEL and COUNTMODE bits allows controlling whether the LPTIM uses an external clock source or an internal one.

When configured to use an external clock source, the CKPOL bits are used to select the external clock signal active edge. If both edges are configured to be active ones, an internal clock signal must also be provided (first configuration). In this case, the internal clock signal frequency must be at least four times higher than the external clock signal frequency.

58.4.5 Glitch filter

The LPTIM inputs, either external (mapped to GPIOs) or internal (mapped on the chip-level to other embedded peripherals), are protected with digital filters that prevent any glitches and noise perturbations to propagate inside the LPTIM. This is in order to prevent spurious counts or triggers.

Before activating the digital filters, an internal clock source must first be provided to the LPTIM. This is necessary to guarantee the proper operation of the filters.

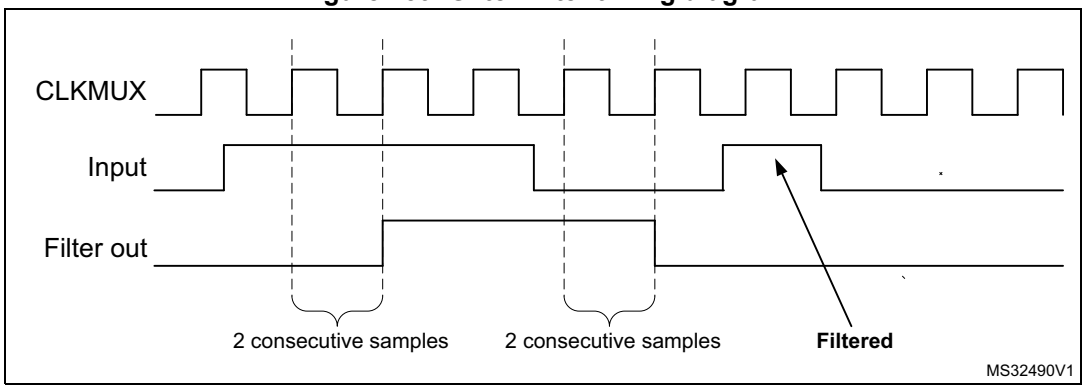
The digital filters are divided into three groups:

- The first group of digital filters protects the LPTIM internal or external inputs. The digital filters sensitivity is controlled by the CKFLT bits
- The second group of digital filters protects the LPTIM internal or external trigger inputs. The digital filters sensitivity is controlled by the TRGFLT bits.
- The third group of digital filters protects the LPTIM internal or external input captures. The digital filters sensitivity is controlled by the ICxF bits.

Note: The digital filters sensitivity is controlled by groups. It is not possible to configure each digital filter sensitivity separately inside the same group.

The filter sensitivity acts on the number of consecutive equal samples that is detected on one of the LPTIM inputs to consider a signal level change as a valid transition. *Figure 759* shows an example of glitch filter behavior in case of a two consecutive samples programmed.

Figure 759. Glitch filter timing diagram



Note: In case no internal clock signal is provided, the digital filter must be deactivated by setting the CKFLT, ICxF and TRGFLT bits to 0. In this case, an external analog filter can be used to protect the LPTIM external inputs against glitches.

58.4.6 Prescaler

The LPTIM 16-bit counter is preceded by a configurable power-of-2 prescaler. The prescaler division ratio is controlled by the PRESC[2:0] field. The table below lists all the possible division ratios:

Table 566. Prescaler division ratios

Programming	Dividing factor
000	/1
001	/2
010	/4
011	/8
100	/16
101	/32
110	/64
111	/128

58.4.7 Trigger multiplexer

The LPTIM counter can be started either by software or after the detection of an active edge on one of the eight trigger inputs.

TRIGEN[1:0] is used to determine the LPTIM trigger source:

- When TRIGEN[1:0] equals 00, the LPTIM counter is started as soon as one of the CNTSTRT or the SNGSTRT bits is set by software. The three remaining possible values for the TRIGEN[1:0] are used to configure the active edge used by the trigger inputs. The LPTIM counter starts as soon as an active edge is detected.
- When TRIGEN[1:0] is different than 00, TRIGSEL[2:0] is used to select which of the eight trigger inputs is used to start the counter.

The external triggers are considered asynchronous signals for the LPTIM. After a trigger detection, a two-counter-clock period latency is needed before the timer starts running due to the synchronization.

If a new trigger event occurs when the timer is already started it is ignored (unless timeout function is enabled).

Note: The timer must be enabled before setting the SNGSTRT/CNTSTRT bits. Any write on these bits when the timer is disabled is discarded by hardware.

Note: When starting the counter by software (TRIGEN[1:0] = 00), there is a delay of 3 kernel clock cycles between the LPTIM_CR register update (set one of SNGSTRT or CNTSTRT bits) and the effective start of the counter.

58.4.8 Operating mode

The LPTIM features two operating modes:

- Continuous mode: the timer is free running, the timer is started from a trigger event and never stops until the timer is disabled
- One-shot mode: the timer is started from a trigger event and stops when an LPTIM update event is generated.

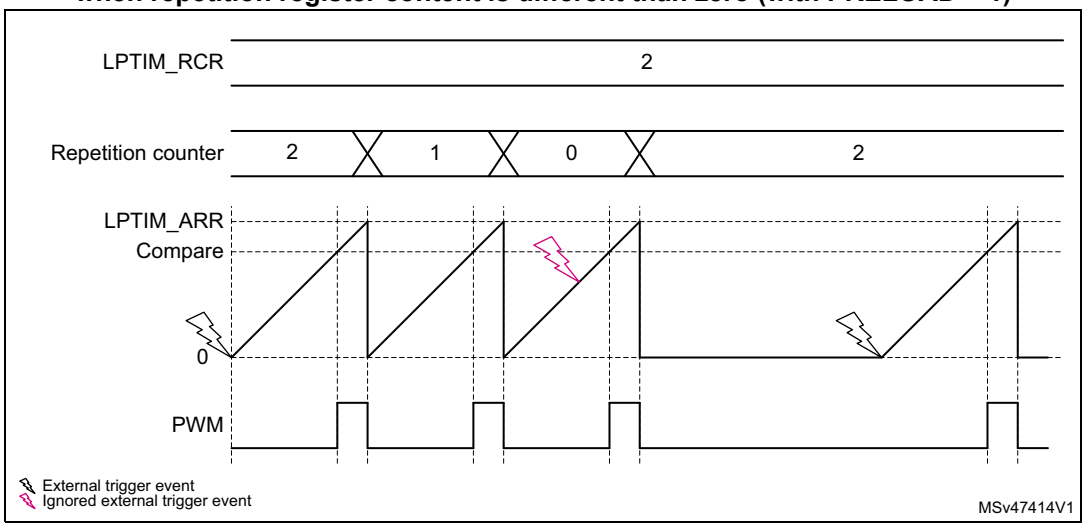
One-shot mode

To enable the one-shot counting, the SNGSTRT bit must be set.

A new trigger event re-starts the timer. Any trigger event occurring after the counter starts and before the next LPTIM update event, is discarded.

In case an external trigger is selected, each external trigger event arriving after the SNGSTRT bit is set, and after the repetition counter has stopped (after the update event), and if the repetition register content is different from zero, the repetition counter gets reloaded with the value already contained by the repetition register and a new one-shot counting cycle is started as shown in *Figure 760*.

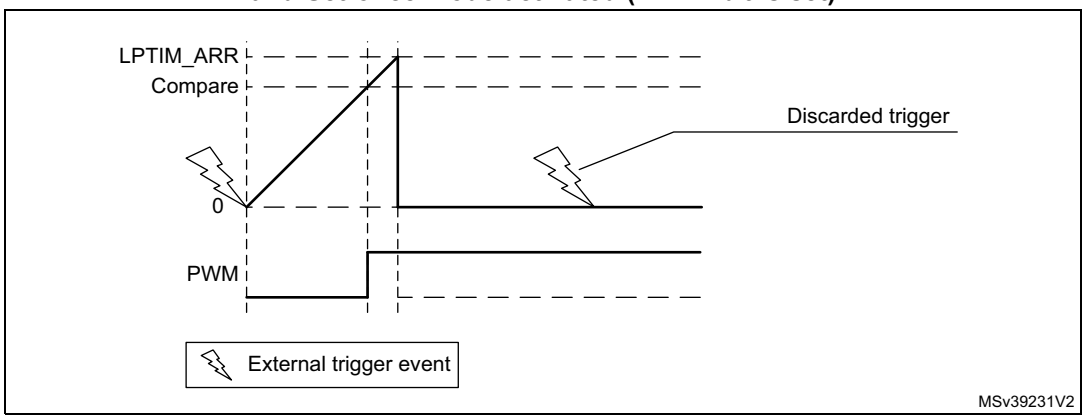
Figure 760. LPTIM output waveform, single-counting mode configuration when repetition register content is different than zero (with PRELOAD = 1)



- Set-once mode activated:

Note that when the WAVE bitfield in the LPTIM_CFGR register is set, the Set-once mode is activated. In this case, the counter is only started once following the first trigger, and any subsequent trigger event is discarded as shown in *Figure 761*.

Figure 761. LPTIM output waveform, single-counting mode configuration and Set-once mode activated (WAVE bit is set)



In case of software start (TRIGEN[1:0] = 00), the SNGSTRT setting starts the counter for one-shot counting.

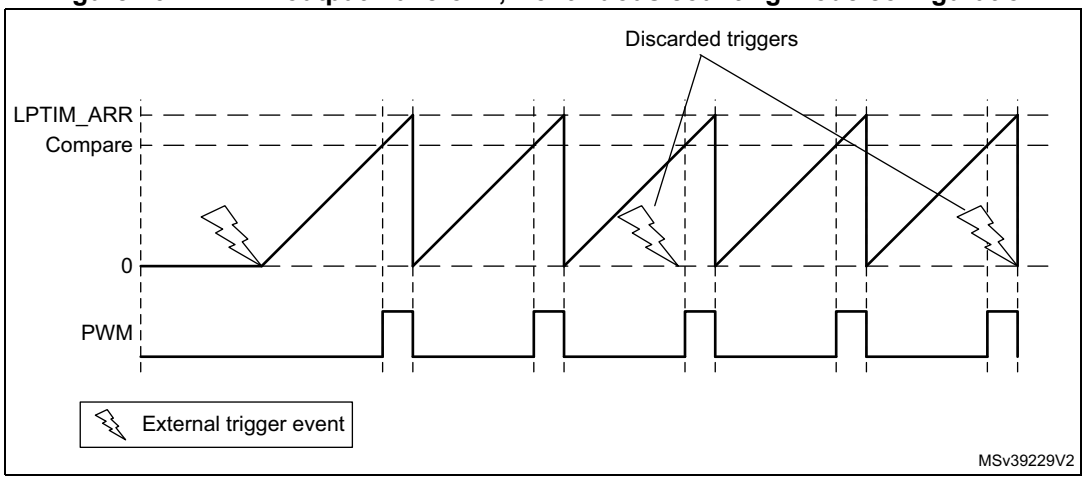
Continuous mode

To enable the continuous counting, the CNTSTRT bit must be set.

In case an external trigger is selected, an external trigger event arriving after CNTSTRT is set, starts the counter for continuous counting. Any subsequent external trigger event is discarded as shown in *Figure 762*.

In case of software start (TRIGEN[1:0] = 00), setting CNTSTRT starts the counter for continuous counting.

Figure 762. LPTIM output waveform, Continuous counting mode configuration



SNGSTRT and CNTSTRT bits can only be set when the timer is enabled (ENABLE bit set to 1). It is possible to change “on the fly” from One-shot mode to Continuous mode.

If the Continuous mode was previously selected, setting SNGSTRT switches the LPTIM to the One-shot mode. The counter (if active) stops as soon as an LPTIM update event is generated.

If the One-shot mode was previously selected, setting CNTSTRT switches the LPTIM to the Continuous mode. The counter (if active) restarts as soon as it reaches ARR.

58.4.9 Timeout function

The detection of an active edge on one selected trigger input can be used to reset the LPTIM counter. This feature is controlled through the TIMOUT bit.

The first trigger event starts the timer, any successive trigger event resets the LPTIM counter and the repetition counter and the timer restarts.

A low-power timeout function can be realized. The timeout value corresponds to the compare value; if no trigger occurs within the expected time frame, the MCU is waked-up by the compare match event.

58.4.10 Waveform generation

Two 16-bit registers, the LPTIM_ARR (autoreload register) and LPTIM_CCRx (capture/compare register), are used to generate several different waveforms on LPTIM output

The timer can generate the following waveforms:

- The PWM mode: the LPTIM output is set as soon as the counter value in LPTIM_CNT exceeds the compare value in LPTIM_CCRx. The LPTIM output is reset as soon as a match occurs between the LPTIM_ARR and the LPTIM_CNT register. For more details see *Section 58.4.19: PWM mode*.
- The One-pulse mode: the output waveform is similar to the one of the PWM mode for the first pulse, then the output is permanently reset
- The Set-once mode: the output waveform is similar to the One-pulse mode except that the output is kept to the last signal level (depends on the output configured polarity).

The above described modes require the LPTIM_ARR register value to be strictly greater than the LPTIM_CCRx register value.

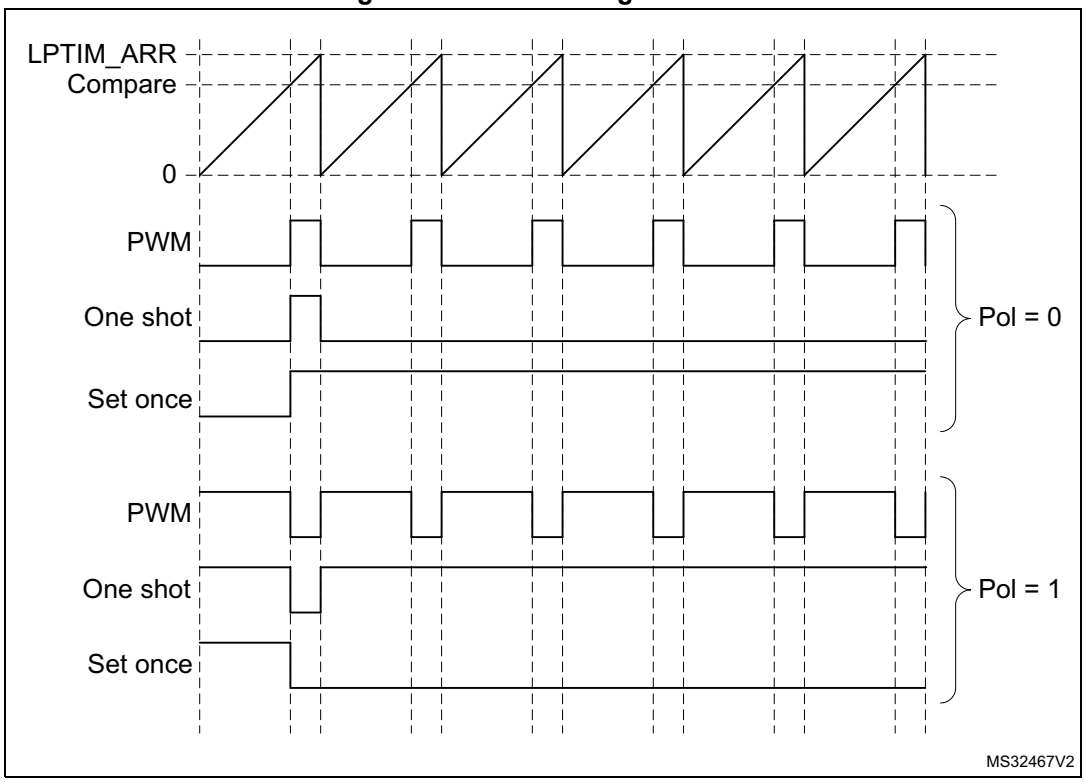
The LPTIM output waveform can be configured through the WAVE bit as follow:

- Resetting the WAVE bit to 0 forces the LPTIM to generate either a PWM waveform or a One pulse waveform depending on which bit is set: CNTSTRT or SNGSTRT.
- Setting the WAVE bit to 1 forces the LPTIM to generate a Set-once mode waveform.

The WAVPOL/CCxP bit controls the LPTIM output polarity. The change takes effect immediately, so the output default value changes immediately after the polarity is re-configured, even before the timer is enabled.

Signals with frequencies up to the LPTIM clock frequency divided by two can be generated. *Figure 763* below shows the three possible waveforms that can be generated on the LPTIM output. Also, it shows the effect of the polarity change using the WAVPOL/CCxP bit.

Figure 763. Waveform generation



58.4.11 Register update

The LPTIM_ARR register, the LPTIM_RCR register and the LPTIM_CCRx register are updated immediately after the APB bus write operation or in synchronization with the next LPTIM update event if the timer is already started.

The PRELOAD bit controls how the LPTIM_ARR, the LPTIM_RCR and the LPTIM_CCRx registers are updated:

- When the PRELOAD bit is reset to 0, the LPTIM_ARR, the LPTIM_RCR and the LPTIM_CCRx registers are immediately updated after any write access.
- When the PRELOAD bit is set to 1, the LPTIM_ARR, the LPTIM_RCR and the LPTIM_CCRx registers are updated at next LPTIM update event, if the timer has been already started.

The LPTIM APB interface and the LPTIM kernel logic use different clocks, so there is some latency between the APB write and the moment when these values are available to the counter comparator. Within this latency period, any additional write into these registers must be avoided.

The ARROK flag, the REPOK flag and the CMPxOK flag in the LPTIM_ISR register indicate when the write operation is completed to respectively the LPTIM_ARR register, the LPTIM_RCR register and the LPTIM_CCRx register.

After a write to the LPTIM_ARR, the LPTIM_RCR or the LPTIM_CCRx register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before respectively the ARROK flag, the REPOK flag or the CMPxOK flag be set, leads to unpredictable results.

58.4.12 Counter mode

The LPTIM counter can be used to count external events on the LPTIM input1 or it can be used to count internal clock cycles. The CKSEL and COUNTMODE bits control which source is used for updating the counter.

In case the LPTIM is configured to count external events on Input1, the counter can be updated following a rising edge, falling edge or both edges depending on the value written to the CKPOL[1:0] bits.

The count modes below can be selected, depending on CKSEL and COUNTMODE values:

- CKSEL = 0: the LPTIM is clocked by an internal clock source
  - COUNTMODE = 0

    The LPTIM is configured to be clocked by an internal clock source and the LPTIM counter is configured to be updated following each internal clock pulse.
  - COUNTMODE = 1

    The LPTIM external Input1 is sampled with the internal clock provided to the LPTIM.

    Consequently, in order not to miss any event, the frequency of the changes on the external Input1 signal must never exceed the frequency of the internal clock provided to the LPTIM. Also, the internal clock provided to the LPTIM must not be prescaled (PRESC[2:0] = 000).
- CKSEL = 1: the LPTIM is clocked by an external clock source

  COUNTMODE value is don't care.

  In this configuration, the LPTIM has no need for an internal clock source (except if the glitch filters are enabled). The signal injected on the LPTIM external input1 is used as system clock for the LPTIM. This configuration is suitable for operation modes where no embedded oscillator is enabled.

  For this configuration, the LPTIM counter can be updated either on rising edges or falling edges of the input1 clock signal but not on both rising and falling edges.

  Since the signal injected on the LPTIM external input1 is also used to clock the LPTIM kernel logic, there is some initial latency (after the LPTIM is enabled) before the counter is incremented. More precisely, the first five active edges on the LPTIM external Input1 (after LPTIM is enable) are lost.

58.4.13 Timer enable

The ENABLE bit located in the LPTIM_CR register is used to enable/disable the LPTIM kernel logic. After setting the ENABLE bit, a delay of two counter clock cycles is needed before the LPTIM is actually enabled.

The LPTIM_CFGR register must be modified only when the LPTIM is disabled.

58.4.14 Timer counter reset

In order to reset the content of LPTIM_CNT register, two reset mechanisms are implemented:

- The synchronous reset mechanism: the synchronous reset is controlled by the COUNTRST bit in the LPTIM_CR register. After setting the COUNTRST bitfield to '1', the reset signal is propagated in the LPTIM kernel clock domain. So it is important to note that a few clock pulses of the LPTIM kernel logic elapse before the reset is taken into account. This makes the LPTIM counter count few extra pluses between the time when the reset is triggered and it become effective. Since the COUNTRST bit is located in the APB clock domain and the LPTIM counter is located in the LPTIM kernel clock domain, a delay of 3 clock cycles of the kernel clock is needed to synchronize the reset signal issued by the APB clock domain when writing '1' to the COUNTRST bit.

Note: The software ensures that COUNRST bit is '0' before generating every synchronous reset.

- The asynchronous reset mechanism: the asynchronous reset is controlled by the RSTARE bit located in the LPTIM_CR register. When this bit is set to '1', any read access to the LPTIM_CNT register resets its content to zero. Asynchronous reset must be triggered within a timeframe in which no LPTIM core clock is provided. For example when LPTIM Input1 is used as external clock source, the asynchronous reset must be applied only when there is enough insurance that no toggle occurs on the LPTIM Input1.
  To read reliably the content of the LPTIM_CNT register two successive read accesses must be performed and compared. A read access can be considered reliable when the value of the two read accesses is equal. Unfortunately when asynchronous reset is enabled there is no possibility to read twice the LPTIM_CNT register.

Warning: There is no mechanism inside the LPTIM that prevents the two reset mechanisms from being used simultaneously. The developer must make sure that these two mechanisms are used exclusively.

58.4.15 Encoder mode

This mode allows handling signals from quadrature encoders used to detect angular position of rotary elements. Encoder interface mode acts simply as an external clock with direction selection. This means that the counter just counts continuously between 0 and the auto-reload value programmed into the LPTIM_ARR register (0 up to ARR or ARR down to 0 depending on the direction). Therefore LPTIM_ARR must be configured before starting the counter. From the two external input signals, Input1 and Input2, a clock signal is generated to clock the LPTIM counter. The phase between those two signals determines the counting direction.

The Encoder mode is only available when the LPTIM is clocked by an internal clock source. The signals frequency on both Input1 and Input2 inputs must not exceed the LPTIM internal clock frequency divided by 4. This is mandatory in order to guarantee a proper operation of the LPTIM.

Direction change is signalized by the two down and up flags in the LPTIM_ISR register. An interrupt can be generated for both direction change events if enabled through the DOWNIE bit.

To activate the Encoder mode the ENC bit has to be set to 1. The LPTIM must first be configured in Continuous mode.

When Encoder mode is active, the LPTIM counter is modified automatically following the speed and the direction of the incremental encoder. Therefore, its content always represents the encoder's position. The count direction, signaled by the Up and Down flags, correspond to the rotation direction of the encoder rotor.

According to the edge sensitivity configured using the CKPOL[1:0] bits, different counting scenarios are possible. The following table summarizes the possible combinations, assuming that Input1 and Input2 do not switch at the same time.

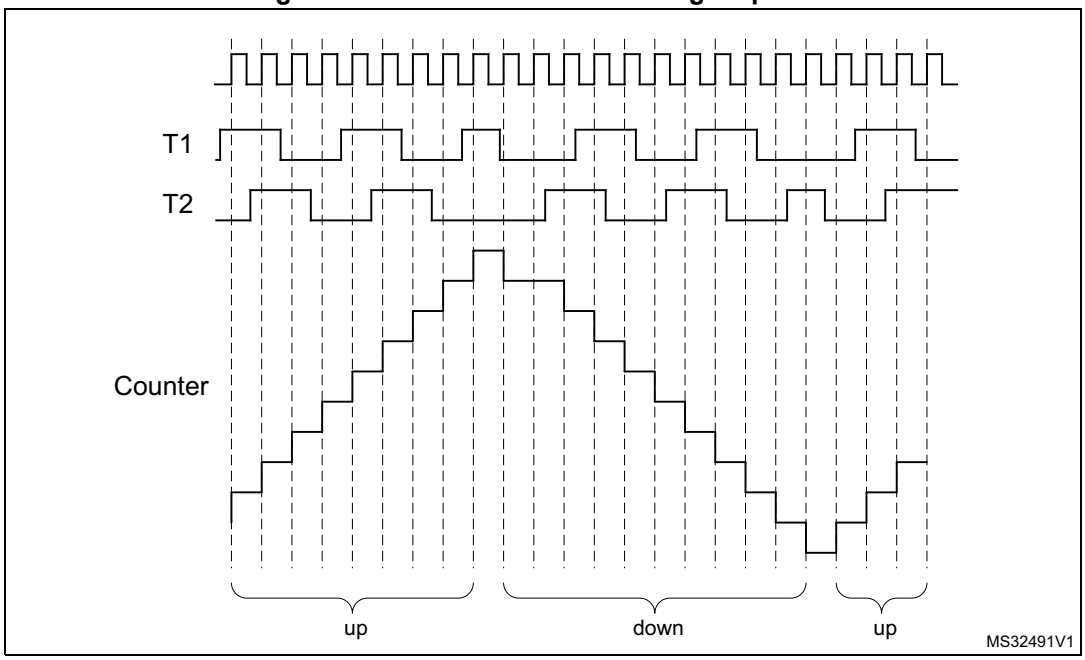
Table 567. Encoder counting scenarios

Active edge	Level on opposite signal (Input1 for Input2, Input2 for Input1)	Input1 signal		Input2 signal	
		Rising	Falling	Rising	Falling
Rising Edge (Encoder sub-mode 1)	High	Down	No count	Up	No count
	Low	Up	No count	Down	No count
Falling Edge (Encoder sub-mode 2)	High	No count	Up	No count	Down
	Low	No count	Down	No count	Up
Both Edges (Encoder sub-mode 3)	High	Down	Up	Up	Down
	Low	Up	Down	Down	Up

The following figure shows a counting sequence for Encoder mode where both-edge sensitivity is configured.

Caution: In this mode the LPTIM must be clocked by an internal clock source, so the CKSEL bit must be maintained to its reset value which is equal to 0. Also, the prescaler division ratio must be equal to its reset value which is 1 (PRESC[2:0] bits must be 000).

Figure 764. Encoder mode counting sequence



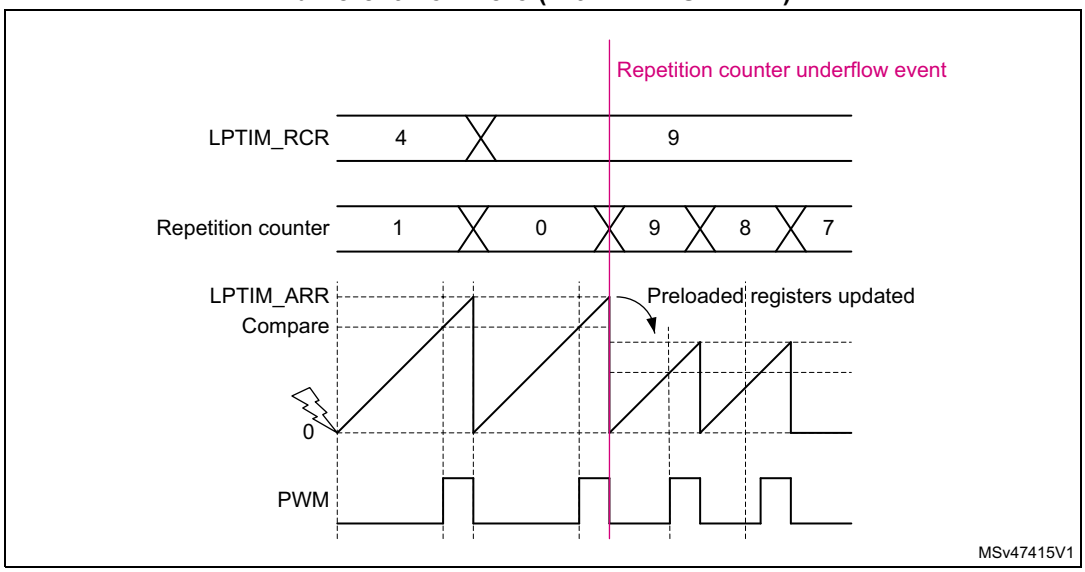
58.4.16 Repetition counter

The LPTIM features a repetition counter that decrements by 1 each time an LPTIM counter overflow event occurs. A repetition counter underflow event is generated when the repetition counter contains zero and the LPTIM counter overflows. Next to each repetition counter underflow event, the repetition counter gets loaded with the content of the REP[7:0] bitfield which belongs to the repetition register LPTIM_RCR.

A repetition underflow event is generated on each and every LPTIM counter overflow when the REP[7:0] register is set to 0.

When PRELOAD = 1, writing to the REP[7:0] bitfield has no effect on the content of the repetition counter until the next repetition underflow event occurs. The repetition counter continues to decrement each LPTIM counter overflow event and only when a repetition underflow event is generated, the new value written into REP[7:0] is loaded into the repetition counter. This behavior is depicted in *Figure 765*.

Figure 765. Continuous counting mode when repetition register LPTIM_RCR different from zero (with PRELOAD = 1)



A repetition counter underflow event is systematically associated with LPTIM preloaded registers update (refer to *Section 58.4.11: Register update* for more information).

Repetition counter underflow event is signaled to the software through the update event (UE) flag mapped into the LPTIM_ISR register. When set, the UE flag can trigger an LPTIM interrupt if its respective update event interrupt enable (UEIE) control bit, mapped to the LPTIM_DIER register, is set.

The repetition register LPTIM_RCR is located in the APB bus interface clock domain where the repetition counter itself is located in the LPTIM kernel clock domain. Each time a new value is written to the LPTIM_RCR register, this new content is propagated from the APB bus interface clock domain to the LPTIM kernel clock domain. The new written value is then loaded to the repetition counter immediately after a repetition counter underflow event. The synchronization delay for the new written content is four APB clock cycles plus three LPTIM kernel clock cycles and it is signaled by the REPOK flag located in the LPTIM_ISR register when it is elapsed. When the LPTIM kernel clock cycle is relatively slow, for instance when the LPTIM kernel is being clocked by the LSI clock source, it can be lengthy to keep polling on the REPOK flag by software to detect that the synchronization of the LPTIM_RCR register content is finished. For that reason, the REPOK flag, when set, can generate an interrupt if its associated REPOKIE control bit in the LPTIM_DIER register is set.

Note: After a write to the LPTIM_RCR register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive writes before the REPOK flag is set, lead to unpredictable results.

Caution: When using repetition counter with PRELOAD = 0, LPTIM_RCR register must be changed at least five counter cycles before the autoreload match event, otherwise an unpredictable behavior may occur.

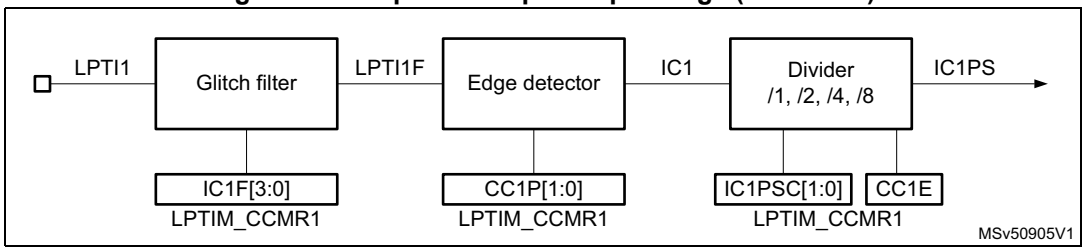
58.4.17 Capture/compare channels

Each capture/compare channel is built around a capture/compare register, an input stage for capture (with digital filter, multiplexing and prescaler) and an output stage (with comparator and output control) for PWM.

Input stage

The input stage samples the corresponding LPTIx input to generate a filtered signal LPTIxF. Then, an edge detector with polarity selection generates ICx signal used as the capture command. It is prescaled to generate the capture command signal (ICxPS).

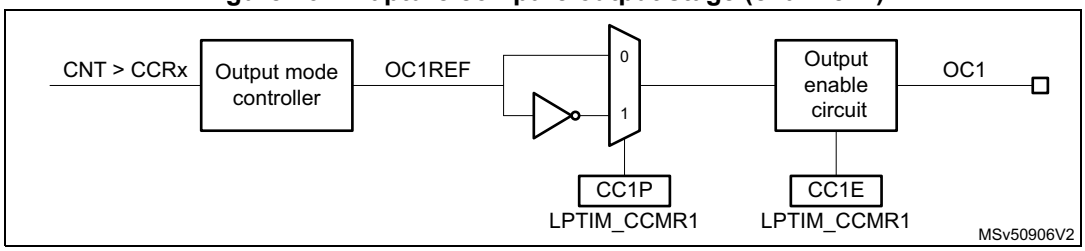
Figure 766. Capture/compare input stage (channel 1)



Output stage

The output stage generates an intermediate waveform which is then used for reference: OCxREF (active high). The polarity acts at the end of the chain.

Figure 767. Capture/compare output stage (channel 1)



58.4.18 Input capture mode

In Input capture mode, the capture/compare registers (LPTIM_CCRx) are used to latch the value of the counter after a transition detected by the corresponding ICx signal. Assuming input capture is enabled on a channel x (CCxE set) and when a capture occurs, the corresponding CCxIF flag (LPTIM_ISR register) is set and an interrupt or a DMA request can be sent if they are enabled. If a capture occurs while the CCxIF flag was already high, then the over-capture flag CCxOF (LPTIM_ISR register) is set. CCxIF can be cleared by software by writing the CCxICF to 1 or by reading the captured data stored in the LPTIM_CCRx register. CCxOF is cleared by writing CCxOCF to 1.

Note: In DMA mode, the input capture channel have to be enabled (set CCxE bit) the last, after enabling the DMA request and after starting the counter. This is in order to prevent generating an input capture DMA request when the counter is not started yet.

Input capture Glitch filter latency

When a trigger event arrives on channel x input (LPTIx) and depending on the configured glitch filter (ICxF[1:0] field in CCMRx register) and on the kernel clock prescaler value (PRESC[2:0] field in CFGR register), there is a variable latency that leads to a systematic offset (see *Table 568*) between the captured value stored in the CCRx register and the real value corresponding to the capture trigger.

This offset has no impact on pulse width measurement as it is systematic and compensated between two captures.

The real capture value corresponding to the input capture trigger can be calculated using the below formula:

Real capture value = captured(LPTIM_CCRx) - offset

The relevant offset must be used depending on the glitch filter and on the kernel clock prescaler value (PRESC field in CFGR register)

Example: determining the real capture value when PRESC[2:0] = 0x2 and ICxF = 0x3.

For this configuration (PRESC[2:0] = 0x2 and ICxF = 0x3) and according to the *Table 568*, the offset is 5.

Assuming that the captured value in CCRx is 9 (LPTIM_CNT = 9), this means that the capture trigger occurred when the LPTIM_CNT was equal to 9 - 5 = 4.

Table 568. Input capture Glitch filter latency (in counter step unit)

Prescaler PRESC[2:0]	ICxF[1:0]	Offset
0	0	2
	1	7
	2	9
	3	13
1	0	3
	1	5
	2	6
	3	8
2	0	2
	1	3
	2	4
	3	5
3	0	2
	1	2
	2	3
	3	3
4	0	2
	1	2
	2	2
	3	2
5	0	2
	1	2
	2	2
	3	2

Table 568. Input capture Glitch filter latency (in counter step unit) (continued)

Prescaler PRESC[2:0]	ICxF[1:0]	Offset
6	0	2
	1	2
	2	2
	3	2
7	0	2
	1	2
	2	2
	3	2

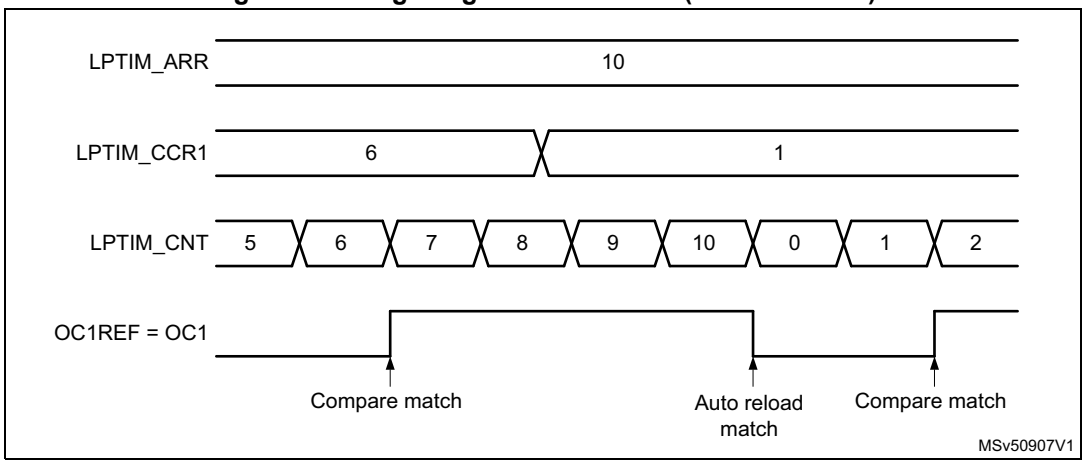
58.4.19 PWM mode

The PWM mode enables to generate a signal with a frequency determined by the value of the LPTIM_ARR register and a duty cycle determined by the value of the LPTIM_CCRx register. The LPTIM is able to generate PWM in edge-aligned mode.

OCx polarity is software programmable using the CCxP bit in the LPTIM_CCMRx register. It can be programmed as active high or active low. OCx output is enabled by the CCxE bit in the LPTIM_CCMRx register. Refer to the LPTIM_CCMRx register description for more details.

Figure 768 gives an example where the LPTIM channel 1 is configured in PWM mode with LPTIM_CCR1 = 6 then 1 and LPTIM_ARR=10.

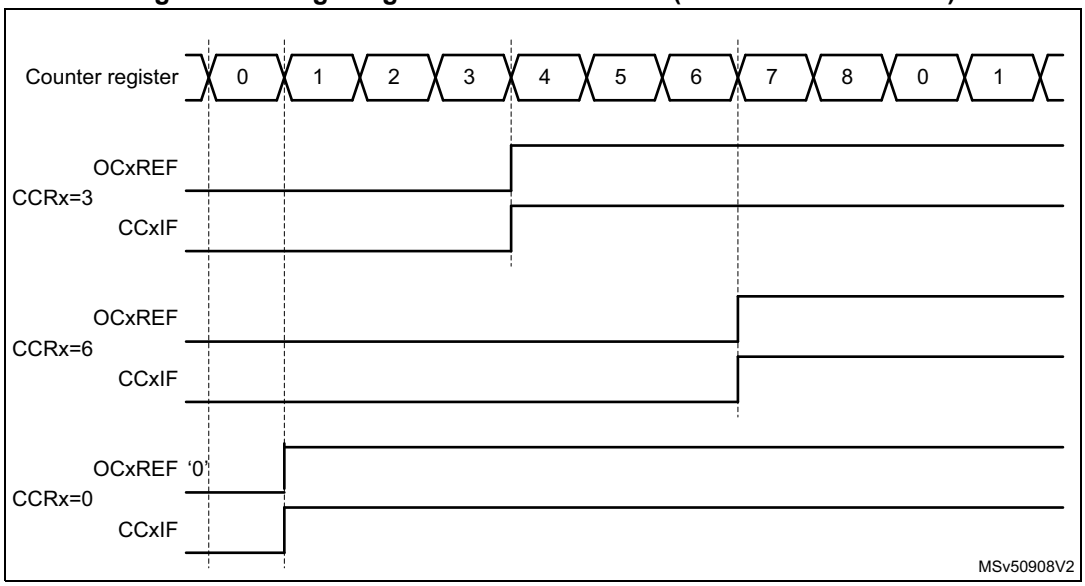
Figure 768. Edge-aligned PWM mode (PRELOAD = 1)



In the following example the reference PWM signal OCxREF is low as long as LPTIM_CNT ≤ LPTIM_CCRx else it becomes high.

Figure 769 shows some edge-aligned PWM waveforms in an example where LPTIM_ARR = 8.

Figure 769. Edge-aligned PWM waveforms (ARR=8 and CCxP = 0)



PWM mode with immediate update PRELOAD = 0

The PWM mode with PRELOAD = 0 enables the early change of the output level within the current PWM cycle. Based on the immediate update (PRELOAD = 0) of the LPTIM_CCRx register and on the continuous comparison of LPTIM_CNT and LPTIM_CCRx registers, it permits to have a new duty cycle value applied as soon as possible within the current PWM cycle, without having to wait for the completion of the current PWM period.

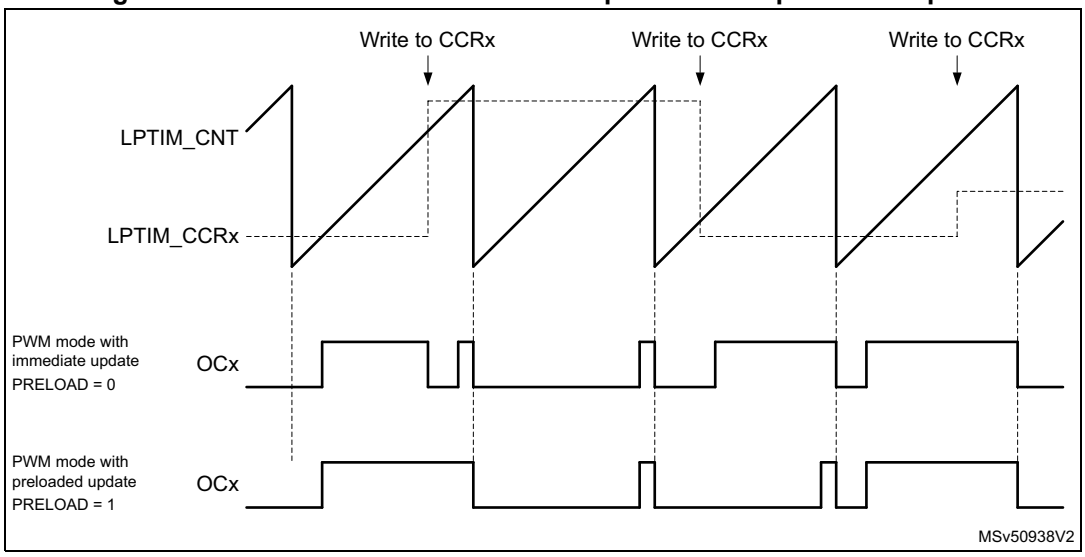
When the (PRELOAD = 0), the OCxREF signal level can be changed on-the-fly by software (or DMA) by updating the compare value in the LPTIM_CCRx register.

Depending on the written compare value and on the current counter and compare values, the OCxREF level is re-assigned as illustrated below:

- If the new compare value does not exceed the current counter value and the current compare value exceeds the counter, OCxREF level is re-assigned high as soon as the new compare value is written.
- If the new compare value exceeds the counter value and the current compare value does not exceed the counter, OCxREF level is re-assigned low as soon as the new compare value is written.

The output reference signal OCxREF level is left unchanged when none of the new compare value and the current compare value exceed the counter. *Figure 770* illustrates the behavior of the OCxREF signal level when PRELOAD = 0 and PRELOAD = 1.

Figure 770. PWM mode with immediate update versus preloaded update



Note: For both PWM modes, the compare match, auto-reload match, and the update event flags are set one LPTIM counter cycle later after the corresponding event, the OCxREF level is also changed one LPTIM counter cycle later after the corresponding event. For instance when the LPTIM_CCRx is set to 3 the CCxIF is set when the LPTIM_CNT = 4. Figure 768 illustrates this behavior.

58.4.20 DMA requests

The LPTIM can generate two categories of DMA requests:

- DMA requests used to retrieve the input-capture counter values
- DMA update requests used to re-program part of the LPTIM, multiple times, at regular intervals, without software overhead

Input capture DMA request

Each LPTIM channel has its dedicated input capture DMA request. A DMA request is generated (if CCxDE bit is set in LPTIM_DIER) and CCxIF is set each time a capture is ready in the LPTIM_CCRx register. The captured values in LPTIM_CCRx can then be transferred regularly by DMA to the desired memory destination. The CCxIF is automatically cleared by hardware when the captured value in LPTIM_CCRx register is read.

Note: The ICx DMA request signal lptim_icx_dma is reset in the following conditions:

- *if the corresponding DMA request is disabled (clear CCxDE bit in the LPTIM_DIER register)*
- *or if the channel x is disabled (clear CCxE bit)*
- *or if the LPTIM is disabled (clear the ENABLE bit in the LPTIM_CR register)*

Update event DMA request

A DMA request is generated (if UEDE is set in LPTIM_DIER) and the UE flag is set at each update event. DMA request can be used to regularly update the LPTIM_ARR, the LPTIM_RCR or the LPTIM_CCRx registers permitting to generate custom PWM waveforms.

The UE is automatically cleared by hardware upon any bus master (like CPU or DMA) write access to the LPTIM_ARR register.

Note: The UE DMA request signal lptim_ue_dma is reset in the following conditions:

- *if the corresponding DMA request is disabled (clear UEDE bit in the LPTIM_DIER register)*
- *or if the LPTIM is disabled (clear the ENABLE bit in the LPTIM_CR register)*
- *or if the channel x is disabled (clear CCxE bit) and all the other channels are already disabled*

58.4.21 Debug mode

When the microcontroller enters debug mode (core halted), the LPTIM counter either continues to work normally or stops, depending on the timer dedicated bit configuration in the debug support (DBG) peripheral.

For further details, refer to section debug support (DBG).

58.5 LPTIM low-power modes

Table 569. Effect of low-power modes on the LPTIM

Mode	Description
Sleep	No effect. LPTIM interrupts cause the device to exit Sleep mode.
Stop	If the LPTIM is clocked by an oscillator available in Stop mode, LPTIM is functional and the interrupts cause the device to exit the Stop mode.
Standby	The LPTIM peripheral is powered down and must be reinitialized after exiting Standby mode.

Note: All DMA requests must be disabled (reset UEDE and CCxDE bits) before entering Sleep, Stop and Standby modes.

58.6 LPTIM interrupts

The following events generate an interrupt/wake-up event, if they are enabled through the LPTIM_DIER register:

- Compare match
- Auto-reload match (whatever the direction if Encoder mode)
- External trigger event
- Autoreload register write completed
- Compare register write completed
- Direction change (Encoder mode), programmable (up / down / both).
- Update Event
- Repetition register update OK
- Input capture occurred
- Over-capture occurred
- Interrupt enable register update OK

Note: If any bit in the LPTIM_DIER register is set after that its corresponding flag in the LPTIM_ISR register (status register) is set, the interrupt is not asserted.

Table 570. Interrupt events

Interrupt vector	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop mode(1)
LPTIMx	Compare match	CCxIF	CCxIE	Write 1 to CCxCF	Yes	Yes
	Input capture(2)	CCxIF	CCxIE	Write 1 to CCxCF	Yes	Yes
	Over-capture(2)	CCxOF	CCxOIE	Write 1 to CCxOCF	Yes	Yes
	Auto-reload match	ARRM	ARRMIE	Write 1 to ARRMCF	Yes	Yes
	External trigger event	EXTTRIG	EXTTRIGIE	Write 1 to EXTTRIGCF	Yes	Yes
	Auto-reload register update OK	ARROK	ARROKIE	Write 1 to ARROKCF	Yes	Yes
	Capture/compare register update OK	CMPxOK	CMPxOKIE	Write 1 to CMPxOKCF	Yes	Yes
	Direction change to up(3)	UP	UPIE	Write 1 to UPCF	Yes	Yes
	Direction change to down(3)	DOWN	DOWNIE	Write 1 to DOWNCF	Yes	Yes
	Update event	UE	UEIE	Write 1 to UECF	Yes	Yes
	Repetition register update OK	REPOK	REPOKIE	Write 1 to REPOKCF	Yes	Yes

1. Each LPTIM event can wake up the device from Stop mode only if the LPTIM instance supports the wake-up from Stop mode feature. Refer to *Section 58.3: LPTIM implementation*.
2. If LPTIM does not implement any channel this event does not exist. Refer to *Section 58.3: LPTIM implementation*.
3. If LPTIM does not support Encoder mode feature, this event does not exist. Refer to *Section 58.3: LPTIM implementation*.

58.7 LPTIM registers

Refer to *Section 1.2: List of abbreviations for registers on page 156* for a list of abbreviations used in register descriptions.

The peripheral registers can only be accessed by words (32-bit).

58.7.1 LPTIMx interrupt and status register (LPTIMx_ISR)(x = 4 to 5)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							r								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPOK	UE	DOWN	UP	ARROK	CMP1OK	EXTTRIG	ARRM	CC1IF
							r	r	r	r	r	r	r	r	r

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROK:** Interrupt enable register update OK

DIEROK is set by hardware to inform application that the APB bus write operation to the LPTIM_DIER register has been successfully completed. DIEROK flag can be cleared by writing 1 to the DIEROKCF bit in the LPTIM_ICR register.

Bits 23:9 Reserved, must be kept at reset value.

Bit 8 **REPOK**: Repetition register update OK

REPOK is set by hardware to inform application that the APB bus write operation to the LPTIM_RCR register has been successfully completed. REPOK flag can be cleared by writing 1 to the REPOKCF bit in the LPTIM_ICR register.

Bit 7 **UE**: LPTIM update event occurred

UE is set by hardware to inform application that an update event was generated. UE flag can be cleared by writing 1 to the UECF bit in the LPTIM_ICR register.

Bit 6 **DOWN**: Counter direction change up to down

In Encoder mode, DOWN bit is set by hardware to inform application that the counter direction has changed from up to down. DOWN flag can be cleared by writing 1 to the DOWNCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 58.3: LPTIM implementation.

Bit 5 **UP**: Counter direction change down to up

In Encoder mode, UP bit is set by hardware to inform application that the counter direction has changed from down to up. UP flag can be cleared by writing 1 to the UPCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 58.3: LPTIM implementation.

Bit 4 **ARROK**: Autoreload register update OK

ARROK is set by hardware to inform application that the APB bus write operation to the LPTIM_ARR register has been successfully completed. ARROK flag can be cleared by writing 1 to the ARROKCF bit in the LPTIM_ICR register.

Bit 3 **CMP1OK**: Compare register 1 update OK

CMP1OK is set by hardware to inform application that the APB bus write operation to the LPTIM_CCR1 register has been successfully completed. CMP1OK flag can be cleared by writing 1 to the CMP1OKCF bit in the LPTIM_ICR register.

Bit 2 **EXTTRIG**: External trigger edge event

EXTTRIG is set by hardware to inform application that a valid edge on the selected external trigger input has occurred. If the trigger is ignored because the timer has already started, then this flag is not set. EXTTRIG flag can be cleared by writing 1 to the EXTTRIGCF bit in the LPTIM_ICR register.

Bit 1 **ARRM**: Autoreload match

ARRM is set by hardware to inform application that LPTIM_CNT register's value reached the LPTIM_ARR register's value. ARRM flag can be cleared by writing 1 to the ARRMCF bit in the LPTIM_ICR register.

Bit 0 **CC1IF**: Compare 1 interrupt flag

The CC1IF flag is set by hardware to inform application that LPTIM_CNT register value matches the compare register's value. The CC1IF flag can be cleared by writing 1 to the CC1CF bit in the LPTIM_ICR register.

0: No match

1: The content of the counter LPTIM_CNT register value has matched the LPTIM_CCR1 register's value

58.7.2 LPTIMx interrupt and status register [alternate] (LPTIMx_ISR) (x = 1 to 3)

This description of the register can only be used for output compare mode. See next section for input capture mode.

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIER OK	Res.	Res.	Res.	Res.	CMP2 OK	Res.	Res.	Res.
							r					r			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC2IF	REP OK	UE	DOWN	UP	ARR OK	CMP1 OK	EXT TRIG	ARRM	CC1IF
						r	r	r	r	r	r	r	r	r	r

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROK:** Interrupt enable register update OK

DIEROK is set by hardware to inform application that the APB bus write operation to the LPTIM_DIER register has been successfully completed. DIEROK flag can be cleared by writing 1 to the DIEROKCF bit in the LPTIM_ICR register.

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CMP2OK**: Compare register 2 update OK

CMP2OK is set by hardware to inform application that the APB bus write operation to the LPTIM_CCR2 register has been successfully completed. CMP2OK flag can be cleared by writing 1 to the CMP2OKCF bit in the LPTIM_ICR register.

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 58.3.

Bits 18:12 Reserved, must be kept at reset value.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2IF**: Compare 2 interrupt flag

If channel CC2 is configured as output:

The CC2IF flag is set by hardware to inform application that LPTIM_CNT register value matches the compare register's value. CC2IF flag can be cleared by writing 1 to the CC2CF bit in the LPTIM_ICR register.

0: No match

1: The content of the counter LPTIM_CNT register value has matched the LPTIM_CCR2 register's value

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 58.3.

Bit 8 **REPOK**: Repetition register update OK

REPOK is set by hardware to inform application that the APB bus write operation to the LPTIM_RCR register has been successfully completed. REPOK flag can be cleared by writing 1 to the REPOKCF bit in the LPTIM_ICR register.

Bit 7 **UE**: LPTIM update event occurred

UE is set by hardware to inform application that an update event was generated. The corresponding interrupt or DMA request is generated if enabled. UE flag can be cleared by writing 1 to the UECF bit in the LPTIM_ICR register. The UE flag is automatically cleared by hardware once the LPTIM_ARR register is written by any bus master like CPU or DMA.

Bit 6 **DOWN**: Counter direction change up to down

In Encoder mode, DOWN bit is set by hardware to inform application that the counter direction has changed from up to down. DOWN flag can be cleared by writing 1 to the DOWNCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 58.3.

Bit 5 **UP**: Counter direction change down to up

In Encoder mode, UP bit is set by hardware to inform application that the counter direction has changed from down to up. UP flag can be cleared by writing 1 to the UPCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 58.3.

Bit 4 **ARROK**: Autoreload register update OK

ARROK is set by hardware to inform application that the APB bus write operation to the LPTIM_ARR register has been successfully completed. ARROK flag can be cleared by writing 1 to the ARROKCF bit in the LPTIM_ICR register.

Bit 3 **CMP1OK**: Compare register 1 update OK

CMP1OK is set by hardware to inform application that the APB bus write operation to the LPTIM_CCR1 register has been successfully completed. CMP1OK flag can be cleared by writing 1 to the CMP1OKCF bit in the LPTIM_ICR register.

Bit 2 **EXTTRIG**: External trigger edge event

EXTTRIG is set by hardware to inform application that a valid edge on the selected external trigger input has occurred. If the trigger is ignored because the timer has already started, then this flag is not set. EXTTRIG flag can be cleared by writing 1 to the EXTTRIGCF bit in the LPTIM_ICR register.

Bit 1 **ARRM**: Autoreload match

ARRM is set by hardware to inform application that LPTIM_CNT register's value reached the LPTIM_ARR register's value. ARRM flag can be cleared by writing 1 to the ARRMCF bit in the LPTIM_ICR register.

Bit 0 **CC1IF**: Compare 1 interrupt flag

If channel CC1 is configured as output:

The CC1IF flag is set by hardware to inform application that LPTIM_CNT register value matches the compare register's value. CC1IF flag can be cleared by writing 1 to the CC1CF bit in the LPTIM_ICR register.

0: No match

1: The content of the counter LPTIM_CNT register value has matched the LPTIM_CCR1 register's value

58.7.3 LPTIMx interrupt and status register [alternate] (LPTIMx_ISR) (x = 1 to 3)

This description of the register can only be used for input capture mode. See previous section for output compare mode.

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIER OK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							r								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CC2 OF	CC1 OF	Res.	Res.	CC2IF	REP OK	UE	DOWN	UP	ARR OK	Res.	EXT TRIG	ARRM	CC1IF
		r	r			r	r	r	r	r	r		r	r	r

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROK:** Interrupt enable register update OK

DIEROK is set by hardware to inform application that the APB bus write operation to the LPTIM_DIER register has been successfully completed. DIEROK flag can be cleared by writing 1 to the DIEROKCF bit in the LPTIM_ICR register.

Bits 23:16 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC2OF**: Capture 2 over-capture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing 1 to the CC2OCF bit in the LPTIM_ICR register.

0: No over-capture has been detected.

1: The counter value has been captured in LPTIM_CCR2 register while CC2IF flag was already set.

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 58.3.

Bit 12 **CC1OF**: Capture 1 over-capture flag

This flag is set by hardware only when the corresponding channel is configured in input capture mode. It is cleared by software by writing 1 to the CC1OCF bit in the LPTIM_ICR register.

0: No over-capture has been detected.

1: The counter value has been captured in LPTIM_CCR1 register while CC1IF flag was already set.

Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 58.3.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2IF**: Capture 2 interrupt flag

If channel CC2 is configured as input:

CC2IF is set by hardware to inform application that the current value of the counter is captured in LPTIM_CCR2 register. The corresponding interrupt or DMA request is generated if enabled. The CC2OF flag is set if the CC2IF flag was already high.

0: No input capture occurred

1: The counter value has been captured in the LPTIM_CCR2 register. (An edge has been detected on IC2 which matches the selected polarity). The CC2IF flag is automatically cleared by hardware once the captured value is read (CPU or DMA). The CC2IF flag can be cleared by writing 1 to the CC2CF bit in the LPTIM_ICR register.

Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 58.3.

Bit 8 **REPOK**: Repetition register update OK

REPOK is set by hardware to inform application that the APB bus write operation to the LPTIM_RCR register has been successfully completed. REPOK flag can be cleared by writing 1 to the REPOKCF bit in the LPTIM_ICR register.

Bit 7 **UE**: LPTIM update event occurred

UE is set by hardware to inform application that an update event was generated. The corresponding interrupt or DMA request is generated if enabled. The UE flag can be cleared by writing 1 to the UECF bit in the LPTIM_ICR register. The UE flag is automatically cleared by hardware once the LPTIM_ARR register is written by any bus master like CPU or DMA.

Bit 6 **DOWN**: Counter direction change up to down

In Encoder mode, DOWN bit is set by hardware to inform application that the counter direction has changed from up to down. DOWN flag can be cleared by writing 1 to the DOWNCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 58.3.

Bit 5 **UP**: Counter direction change down to up

In Encoder mode, UP bit is set by hardware to inform application that the counter direction has changed from down to up. UP flag can be cleared by writing 1 to the UPCF bit in the LPTIM_ICR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 58.3.

Bit 4 **ARROK**: Autoreload register update OK

ARROK is set by hardware to inform application that the APB bus write operation to the LPTIM_ARR register has been successfully completed. ARROK flag can be cleared by writing 1 to the ARROKCF bit in the LPTIM_ICR register.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **EXTTRIG**: External trigger edge event

EXTTRIG is set by hardware to inform application that a valid edge on the selected external trigger input has occurred. If the trigger is ignored because the timer has already started, then this flag is not set. EXTTRIG flag can be cleared by writing 1 to the EXTTRIGCF bit in the LPTIM_ICR register.

Bit 1 **ARRM**: Autoreload match

ARRM is set by hardware to inform application that LPTIM_CNT register's value reached the LPTIM_ARR register's value. ARRM flag can be cleared by writing 1 to the ARRMCF bit in the LPTIM_ICR register.

Bit 0 **CC1IF**: capture 1 interrupt flag

If channel CC1 is configured as input:

CC1IF is set by hardware to inform application that the current value of the counter is captured in LPTIM_CCR1 register. The corresponding interrupt or DMA request is generated if enabled. The CC1OF flag is set if the CC1IF flag was already high.

0: No input capture occurred

1: The counter value has been captured in the LPTIM_CCR1 register. (An edge has been detected on IC1 which matches the selected polarity). The CC1IF flag is automatically cleared by hardware once the captured value is read (CPU or DMA).CC1IF flag can be cleared by writing 1 to the CC1CF bit in the LPTIM_ICR register.

58.7.4 LPTIMx interrupt clear register (LPTIMx_ICR)(x = 4 to 5)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROKCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							w								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPOKCF	UECF	DOWNCF	UPCF	ARROKCF	CMP1OKCF	EXTTRIGCF	ARRMCF	CC1CF
							w	w	w	w	w	w	w	w	w

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROKCF:** Interrupt enable register update OK clear flag

Writing 1 to this bit clears the DIEROK flag in the LPTIM_ISR register.

Bits 23:9 Reserved, must be kept at reset value.

Bit 8 **REPOKCF**: Repetition register update OK clear flag

Writing 1 to this bit clears the REPOK flag in the LPTIM_ISR register.

Bit 7 **UECF**: Update event clear flag

Writing 1 to this bit clear the UE flag in the LPTIM_ISR register.

Bit 6 **DOWNCF**: Direction change to down clear flag

Writing 1 to this bit clear the DOWN flag in the LPTIM_ISR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 58.3.

Bit 5 **UPCF**: Direction change to UP clear flag

Writing 1 to this bit clear the UP flag in the LPTIM_ISR register.

Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 58.3.

Bit 4 **ARROKCF**: Autoreload register update OK clear flag
Writing 1 to this bit clears the ARROK flag in the LPTIM_ISR register

Bit 3 **CMP1OKCF**: Compare register 1 update OK clear flag
Writing 1 to this bit clears the CMP1OK flag in the LPTIM_ISR register.

Bit 2 **EXTTRIGCF**: External trigger valid edge clear flag
Writing 1 to this bit clears the EXTTRIG flag in the LPTIM_ISR register

Bit 1 **ARRMCF**: Autoreload match clear flag
Writing 1 to this bit clears the ARRM flag in the LPTIM_ISR register

Bit 0 **CC1CF**: Capture/compare 1 clear flag
Writing 1 to this bit clears the CC1IF flag in the LPTIM_ISR register.

58.7.5 LPTIMx interrupt clear register [alternate] (LPTIMx_ICR) (x = 1 to 3)

This description of the register can only be used for output compare mode. See next section for input capture compare mode.

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIER OKCF	Res.	Res.	Res.	Res.	CMP2 OKCF	Res.	Res.	Res.
							w					w			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC2CF	REPOK CF	UECF	DOWN CF	UPCF	ARR OKCF	CMP1 OKCF	EXT TRIG CF	ARRM CF	CC1CF
						w	w	w	w	w	w	w	w	w	w

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROKCF:** Interrupt enable register update OK clear flag
Writing 1 to this bit clears the DIEROK flag in the LPTIM_ISR register.

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CMP2OKCF**: Compare register 2 update OK clear flag
Writing 1 to this bit clears the CMP2OK flag in the LPTIM_ISR register.
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 58.3.

Bits 18:12 Reserved, must be kept at reset value.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2CF**: Capture/compare 2 clear flag
Writing 1 to this bit clears the CC2IF flag in the LPTIM_ISR register.
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 58.3.

Bit 8 **REPOKCF**: Repetition register update OK clear flag
Writing 1 to this bit clears the REPOK flag in the LPTIM_ISR register.

Bit 7 **UECF**: Update event clear flag
Writing 1 to this bit clear the UE flag in the LPTIM_ISR register.

Bit 6 **DOWNCF**: Direction change to down clear flag
Writing 1 to this bit clear the DOWN flag in the LPTIM_ISR register.
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 58.3.

Bit 5 **UPCF**: Direction change to UP clear flag
Writing 1 to this bit clear the UP flag in the LPTIM_ISR register.
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 58.3.

Bit 4 **ARROKCF**: Autoreload register update OK clear flag
Writing 1 to this bit clears the ARROK flag in the LPTIM_ISR register

Bit 3 **CMP1OKCF**: Compare register 1 update OK clear flag
Writing 1 to this bit clears the CMP1OK flag in the LPTIM_ISR register.

Bit 2 **EXTTRIGCF**: External trigger valid edge clear flag
Writing 1 to this bit clears the EXTTRIG flag in the LPTIM_ISR register

Bit 1 **ARRMCF**: Autoreload match clear flag
Writing 1 to this bit clears the ARRM flag in the LPTIM_ISR register

Bit 0 **CC1CF**: Capture/compare 1 clear flag
Writing 1 to this bit clears the CC1IF flag in the LPTIM_ISR register.

58.7.6 LPTIMx interrupt clear register [alternate] (LPTIMx_ICR) (x = 1 to 3)

This description of the register can only be used for input capture mode. See previous section for output compare mode.

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROKCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							w								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CC2OCF	CC1OCF	Res.	Res.	CC2CF	REPOKCF	UECF	DOWNCF	UPCF	ARROKCF	Res.	EXTTRIGCF	ARRMCF	CC1CF
		w	w			w	w	w	w	w	w		w	w	w

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **DIEROKCF:** Interrupt enable register update OK clear flag
Writing 1 to this bit clears the DIEROK flag in the LPTIM_ISR register.

Bits 23:16 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC2OCF**: Capture/compare 2 over-capture clear flag
Writing 1 to this bit clears the CC2OF flag in the LPTIM_ISR register.
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 58.3.

Bit 12 **CC1OCF**: Capture/compare 1 over-capture clear flag
Writing 1 to this bit clears the CC1OF flag in the LPTIM_ISR register.
Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 58.3.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2CF**: Capture/compare 2 clear flag
Writing 1 to this bit clears the CC2IF flag in the LPTIM_ISR register.
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 58.3.

Bit 8 **REPOKCF**: Repetition register update OK clear flag
Writing 1 to this bit clears the REPOK flag in the LPTIM_ISR register.

Bit 7 **UECF**: Update event clear flag
Writing 1 to this bit clear the UE flag in the LPTIM_ISR register.

Bit 6 **DOWNCF**: Direction change to down clear flag
Writing 1 to this bit clear the DOWN flag in the LPTIM_ISR register.
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 58.3.

Bit 5 **UPCF**: Direction change to UP clear flag
Writing 1 to this bit clear the UP flag in the LPTIM_ISR register.
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 58.3.

Bit 4 **ARROKCF**: Autoreload register update OK clear flag
Writing 1 to this bit clears the ARROK flag in the LPTIM_ISR register

Bit 3 Reserved, must be kept at reset value.

Bit 2 **EXTTRIGCF**: External trigger valid edge clear flag
Writing 1 to this bit clears the EXTTRIG flag in the LPTIM_ISR register

Bit 1 **ARRMCF**: Autoreload match clear flag
Writing 1 to this bit clears the ARRM flag in the LPTIM_ISR register

Bit 0 **CC1CF**: Capture/compare 1 clear flag
Writing 1 to this bit clears the CC1IF flag in the LPTIM_ISR register.

58.7.7 LPTIMx interrupt enable register (LPTIMx_DIER)(x = 4 to 5)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPOKIE	UEIE	DOWNIE	UPIE	ARROKIE	CMP1OKIE	EXTTRIGIE	ARRMIE	CC1IE
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **REPOKIE**: Repetition register update OK interrupt Enable
0: Repetition register update OK interrupt disabled
1: Repetition register update OK interrupt enabled

Bit 7 **UEIE**: Update event interrupt enable
0: Update event interrupt disabled
1: Update event interrupt enabled

Bit 6 **DOWNIE**: Direction change to down Interrupt Enable
0: DOWN interrupt disabled
1: DOWN interrupt enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 58.3.

Bit 5 **UPIE**: Direction change to UP Interrupt Enable
0: UP interrupt disabled
1: UP interrupt enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 58.3.

Bit 4 **ARROKIE**: Autoreload register update OK Interrupt Enable
0: ARROK interrupt disabled
1: ARROK interrupt enabled

Bit 3 **CMP1OKIE**: Compare register 1 update OK interrupt enable
0: CMPOK register 1 interrupt disabled
1: CMPOK register 1 interrupt enabled

Bit 2 **EXTTRIGIE**: External trigger valid edge Interrupt Enable
0: EXTTRIG interrupt disabled
1: EXTTRIG interrupt enabled

Bit 1 **ARRMIE**: Autoreload match Interrupt Enable
0: ARRM interrupt disabled
1: ARRM interrupt enabled

Bit 0 **CC1IE**: Capture/compare 1 interrupt enable
0: Capture/compare 1 interrupt disabled
1: Capture/compare 1 interrupt enabled

Caution: The LPTIMx_DIER register must only be modified when the LPTIM is enabled (ENABLE bit set to 1). After a write to the LPTIMx_DIER register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before the DIEROK flag is set, leads to unpredictable results.

58.7.8 LPTIMx interrupt enable register [alternate] (LPTIMx_DIER) (x = 1 to 3)

This description of the register can only be used for output compare mode. See next section for input capture compare mode.

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UEDE	Res.	Res.	Res.	CMP2OKIE	Res.	Res.	Res.
								rw				rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CC2IE	REPOKIE	UEIE	DOWNIE	UPIE	ARROKIE	CMP1OKIE	EXTTRIGIE	ARRMIE	CC1IE
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **UEDE**: Update event DMA request enable
0: UE DMA request disabled. Writing '0' to the UEDE bit resets the associated ue_dma_req signal.
1: UE DMA request enabled
Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 58.3.

Bit 22 Reserved, must be kept at reset value.

Bit 21 Reserved, must be kept at reset value.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CMP2OKIE**: Compare register 2 update OK interrupt enable
0: CMPOK register 2 interrupt disabled
1: CMPOK register 2 interrupt enabled
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 58.3.

Bits 18:12 Reserved, must be kept at reset value.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2IE**: Capture/compare 2 interrupt enable
0: Capture/compare 2 interrupt disabled
1: Capture/compare 2 interrupt enabled
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 58.3.

Bit 8 **REPOKIE**: Repetition register update OK interrupt Enable
0: Repetition register update OK interrupt disabled
1: Repetition register update OK interrupt enabled

Bit 7 **UEIE**: Update event interrupt enable
0: Update event interrupt disabled
1: Update event interrupt enabled

Bit 6 **DOWNIE**: Direction change to down Interrupt Enable
0: DOWN interrupt disabled
1: DOWN interrupt enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 58.3.

Bit 5 **UPIE**: Direction change to UP Interrupt Enable
0: UP interrupt disabled
1: UP interrupt enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 58.3.

Bit 4 **ARROKIE**: Autoreload register update OK Interrupt Enable
0: ARROK interrupt disabled
1: ARROK interrupt enabled

Bit 3 **CMP1OKIE**: Compare register 1 update OK interrupt enable
0: CMPOK register 1 interrupt disabled
1: CMPOK register 1 interrupt enabled

Bit 2 **EXTTRIGIE**: External trigger valid edge Interrupt Enable
0: EXTTRIG interrupt disabled
1: EXTTRIG interrupt enabled

Bit 1 **ARRMIE**: Autoreload match Interrupt Enable
0: ARRM interrupt disabled
1: ARRM interrupt enabled

Bit 0 **CC1IE**: Capture/compare 1 interrupt enable
0: Capture/compare 1 interrupt disabled
1: Capture/compare 1 interrupt enabled

Caution: The LPTIMx_DIER register must only be modified when the LPTIM is enabled (ENABLE bit set to 1). After a write to the LPTIMx_DIER register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before the DIEROK flag is set, leads to unpredictable results.

58.7.9 LPTIMx interrupt enable register [alternate] (LPTIMx_DIER) (x = 1 to 3)

This description of the register can only be used for input capture mode. See previous section for output compare mode.

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	CC2DE	Res.	UEDE	Res.	Res.	Res.	Res.	Res.	Res.	CC1DE
						rw		rw							rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	CC2OIE	CC1OIE	Res.	Res.	CC2IE	REPOKIE	UEIE	DOWNIE	UPIE	ARROKIE	Res.	EXTTRIGIE	ARRMIE	CC1IE
		rw	rw			rw	rw	rw	rw	rw	rw		rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 Reserved, must be kept at reset value.

Bit 26 Reserved, must be kept at reset value.

Bit 25 **CC2DE**: Capture/compare 2 DMA request enable
0: CC2 DMA request disabled. Writing '0' to the CC2DE bit resets the associated ic2_dma_req signal.
1: CC2 DMA request enabled
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 58.3.

Bit 24 Reserved, must be kept at reset value.

Bit 23 **UEDE**: Update event DMA request enable
0: UE DMA request disabled. Writing '0' to the UEDE bit resets the associated ue_dma_req signal.
1: UE DMA request enabled
Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 58.3.

Bits 22:17 Reserved, must be kept at reset value.

Bit 16 **CC1DE**: Capture/compare 1 DMA request enable
0: CC1 DMA request disabled. Writing '0' to the CC1DE bit resets the associated ic1_dma_req signal.
1: CC1 DMA request enabled
Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 58.3.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **CC2OIE**: Capture/compare 2 over-capture interrupt enable
0: CC2 over-capture interrupt disabled
1: CC2 over-capture interrupt enabled
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 58.3.

Bit 12 **CC1OIE**: Capture/compare 1 over-capture interrupt enable
0: CC1 over-capture interrupt disabled
1: CC1 over-capture interrupt enabled
Note: If LPTIM does not implement at least 1 channel this bit is reserved. Refer to Section 58.3.

Bit 11 Reserved, must be kept at reset value.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CC2IE**: Capture/compare 2 interrupt enable
0: Capture/compare 2 interrupt disabled
1: Capture/compare 2 interrupt enabled
Note: If LPTIM does not implement at least 2 channels this bit is reserved. Refer to Section 58.3.

Bit 8 **REPOKIE**: Repetition register update OK interrupt Enable
0: Repetition register update OK interrupt disabled
1: Repetition register update OK interrupt enabled

Bit 7 **UEIE**: Update event interrupt enable
0: Update event interrupt disabled
1: Update event interrupt enabled

Bit 6 **DOWNIE**: Direction change to down Interrupt Enable
0: DOWN interrupt disabled
1: DOWN interrupt enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 58.3.

Bit 5 **UPIE**: Direction change to UP Interrupt Enable
0: UP interrupt disabled
1: UP interrupt enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 58.3.

Bit 4 **ARROKIE**: Autoreload register update OK Interrupt Enable
0: ARROK interrupt disabled
1: ARROK interrupt enabled

Bit 3 Reserved, must be kept at reset value.

Bit 2 **EXTTRIGIE**: External trigger valid edge Interrupt Enable
0: EXTTRIG interrupt disabled
1: EXTTRIG interrupt enabled

Bit 1 **ARRMIE**: Autoreload match Interrupt Enable
0: ARRM interrupt disabled
1: ARRM interrupt enabled

Bit 0 **CC1IE**: Capture/compare 1 interrupt enable
0: Capture/compare 1 interrupt disabled
1: Capture/compare 1 interrupt enabled

Caution: The LPTIMx_DIER register must only be modified when the LPTIM is enabled (ENABLE bit set to 1). After a write to the LPTIMx_DIER register, a new write operation to the same register can only be performed when the previous write operation is completed. Any successive write before the DIEROK flag is set, leads to unpredictable results.

58.7.10 LPTIM configuration register (LPTIM_CFGR)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENC	COUNT MODE	PRE LOAD	WAV POL	WAVE	TIMOUT	TRIGEN[1:0]		Res.
							rw	rw	rw	rw	rw	rw	rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRIGSEL[2:0]			Res.	PRESC[2:0]			Res.	TRGFLT[1:0]		Res.	CKFLT[1:0]		CKPOL[1:0]		CKSEL
rw	rw	rw		rw	rw	rw		rw	rw		rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 Reserved, must be kept at reset value.

Bits 28:25 Reserved, must be kept at reset value.

Bit 24 **ENC**: Encoder mode enable
The ENC bit controls the Encoder mode
0: Encoder mode disabled
1: Encoder mode enabled
Note: If the LPTIM does not support Encoder mode feature, this bit is reserved. Refer to Section 58.3.

Bit 23 **COUNTMODE**: counter mode enabled
The COUNTMODE bit selects which clock source is used by the LPTIM to clock the counter:
0: The counter is incremented following each internal clock pulse
1: The counter is incremented following each valid clock pulse on the LPTIM external Input1

Bit 22 **PRELOAD**: Registers update mode
The PRELOAD bit controls the LPTIM_ARR, LPTIM_RCR and the LPTIM_CCRx registers update modality
0: Registers are updated after each APB bus write access
1: Registers are updated at the end of the current LPTIM period

Bit 21 **WAVPOL**: Waveform shape polarity
The WAVPOL bit controls the output polarity
0: The LPTIM output reflects the compare results between LPTIM_CNT and LPTIM_CCRx registers
1: The LPTIM output reflects the inverse of the compare results between LPTIM_CNT and LPTIM_CCRx registers
Note: If the LPTIM implements at least one capture/compare channel, this bit is reserved. Refer to Section 58.3.

Bit 20 **WAVE**: Waveform shape
The WAVE bit controls the output shape
0: Deactivate Set-once mode
1: Activate the Set-once mode

Bit 19 **TIMOUT**: Timeout enable
The TIMOUT bit controls the Timeout feature
0: A trigger event arriving when the timer is already started is ignored
1: A trigger event arriving when the timer is already started resets and restarts the LPTIM counter and the repetition counter

Bits 18:17 **TRIGEN[1:0]**: Trigger enable and polarity
The TRIGEN bits controls whether the LPTIM counter is started by an external trigger or not. If the external trigger option is selected, three configurations are possible for the trigger active edge:
00: Software trigger (counting start is initiated by software)
01: Rising edge is the active edge
10: Falling edge is the active edge
11: Both edges are active edges

Bit 16 Reserved, must be kept at reset value.

Bits 15:13 **TRIGSEL[2:0]**: Trigger selector

The TRIGSEL bits select the trigger source that serves as a trigger event for the LPTIM among the below 8 available sources:

000: lptim_ext_trig0
001: lptim_ext_trig1
010: lptim_ext_trig2
011: lptim_ext_trig3
100: lptim_ext_trig4
101: lptim_ext_trig5
110: lptim_ext_trig6
111: lptim_ext_trig7

See *Section 58.4.3: LPTIM input and trigger mapping* for details.

Bit 12 Reserved, must be kept at reset value.

Bits 11:9 **PRESC[2:0]:** Clock prescaler

The PRESC bits configure the prescaler division factor. It can be one among the following division factors:

000: /1
001: /2
010: /4
011: /8
100: /16
101: /32
110: /64
111: /128

Bit 8 Reserved, must be kept at reset value.

Bits 7:6 **TRGFLT[1:0]:** Configurable digital filter for trigger

The TRGFLT value sets the number of consecutive equal samples that are detected when a level change occurs on an internal trigger before it is considered as a valid level transition. An internal clock source must be present to use this feature

00: Any trigger active level change is considered as a valid trigger
01: Trigger active level change must be stable for at least 2 clock periods before it is considered as valid trigger.
10: Trigger active level change must be stable for at least 4 clock periods before it is considered as valid trigger.
11: Trigger active level change must be stable for at least 8 clock periods before it is considered as valid trigger.

Bit 5 Reserved, must be kept at reset value.

Bits 4:3 **CKFLT[1:0]:** Configurable digital filter for external clock

The CKFLT value sets the number of consecutive equal samples that are detected when a level change occurs on an external clock signal before it is considered as a valid level transition. An internal clock source must be present to use this feature

00: Any external clock signal level change is considered as a valid transition
01: External clock signal level change must be stable for at least 2 clock periods before it is considered as valid transition.
10: External clock signal level change must be stable for at least 4 clock periods before it is considered as valid transition.
11: External clock signal level change must be stable for at least 8 clock periods before it is considered as valid transition.

Bits 2:1 **CKPOL[1:0]:** Clock polarity

When the LPTIM is clocked by an external clock source, CKPOL bits is used to configure the active edge or edges used by the counter:

00: The rising edge is the active edge used for counting.
If the LPTIM is configured in Encoder mode (ENC bit is set), the Encoder sub-mode 1 is active.

01: The falling edge is the active edge used for counting.
If the LPTIM is configured in Encoder mode (ENC bit is set), the Encoder sub-mode 2 is active.

10: Both edges are active edges. When both external clock signal edges are considered active ones, the LPTIM must also be clocked by an internal clock source with a frequency equal to at least four times the external clock frequency.
If the LPTIM is configured in Encoder mode (ENC bit is set), the Encoder sub-mode 3 is active.

11: Not allowed

Refer to *Section 58.4.15: Encoder mode* for more details about Encoder sub-modes.

Bit 0 **CKSEL:** Clock selector

The CKSEL bit selects which clock source the LPTIM uses:

0: LPTIM is clocked by internal clock source (APB clock or any of the embedded oscillators)

1: LPTIM is clocked by an external clock source through the LPTIM external Input1

Caution: The LPTIM_CFGR register must only be modified when the LPTIM is disabled (ENABLE bit reset to 0).

58.7.11 LPTIM control register (LPTIM_CR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTARE	COUNTRST	CNTSTRT	SNGSTRT	ENABLE
											rw	rs	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **RSTARE:** Reset after read enable

This bit is set and cleared by software. When RSTARE is set to '1', any read access to LPTIM_CNT register asynchronously resets LPTIM_CNT register content.
This bit can be set only when the LPTIM is enabled.

Bit 3 **COUNTRST:** Counter reset

This bit is set by software and cleared by hardware. When set to '1' this bit triggers a synchronous reset of the LPTIM_CNT counter register. Due to the synchronous nature of this reset, it only takes place after a synchronization delay of 3 LPTimer core clock cycles (LPTimer core clock can be different from APB clock).
This bit can be set only when the LPTIM is enabled. It is automatically reset by hardware.

Caution: COUNTRST must never be set to '1' by software before it is already cleared to '0' by hardware. Software must consequently check that COUNTRST bit is already cleared to '0' before attempting to set it to '1'.

Bit 2 **CNTSTRT:** Timer start in Continuous mode

This bit is set by software and cleared by hardware.

In case of software start (TRIGEN[1:0] = 00), setting this bit starts the LPTIM in Continuous mode.

If the software start is disabled (TRIGEN[1:0] different than 00), setting this bit starts the timer in Continuous mode as soon as an external trigger is detected.

If this bit is set when a single pulse mode counting is ongoing, then the timer does not stop at the next match between the LPTIM_ARR and LPTIM_CNT registers and the LPTIM counter keeps counting in Continuous mode.

This bit can be set only when the LPTIM is enabled. It is automatically reset by hardware.

Bit 1 **SNGSTRT:** LPTIM start in Single mode

This bit is set by software and cleared by hardware.

In case of software start (TRIGEN[1:0] = 00), setting this bit starts the LPTIM in single pulse mode.

If the software start is disabled (TRIGEN[1:0] different than 00), setting this bit starts the LPTIM in single pulse mode as soon as an external trigger is detected.

If this bit is set when the LPTIM is in continuous counting mode, then the LPTIM stops at the following match between LPTIM_ARR and LPTIM_CNT registers.

This bit can only be set when the LPTIM is enabled. It is automatically reset by hardware.

Bit 0 **ENABLE:** LPTIM enable

The ENABLE bit is set and cleared by software.

0: LPTIM is disabled. Writing '0' to the ENABLE bit resets all the DMA request signals (input capture and update event DMA requests).

1: LPTIM is enabled

58.7.12 LPTIM compare register 1 (LPTIM_CCR1)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CCR1[15:0]**: Capture/compare 1 value

If channel CC1 is configured as output:

CCR1 is the value to be loaded in the capture/compare 1 register.

Depending on the PRELOAD option, the CCR1 register is immediately updated if the PRELOAD bit is reset and updated at next LPTIM update event if PREOAD bit is reset.

The capture/compare register 1 contains the value to be compared to the counter LPTIM_CNT and signaled on OC1 output.

If channel CC1 is configured as input:

CCR1 becomes read-only, it contains the counter value transferred by the last input capture 1 event. The LPTIM_CCR1 register is read-only and cannot be programmed.

If LPTIM does not implement any channel:

The compare register 1 contains the value to be compared to the counter LPTIM_CNT and signaled on LPTIM output.

Caution: The LPTIM_CCR1 register must only be modified when the LPTIM is enabled (ENABLE bit set to 1).

58.7.13 LPTIM autoreload register (LPTIM_ARR)

Address offset: 0x018

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ARR[15:0]:** Auto reload value
ARR is the autoreload value for the LPTIM.
This value must be strictly greater than the CCRx[15:0] value.

Caution: The LPTIM_ARR register must only be modified when the LPTIM is enabled (ENABLE bit set to 1).

58.7.14 LPTIM counter register (LPTIM_CNT)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CNT[15:0]:** Counter value
When the LPTIM is running, reading the LPTIM_CNT register may return unreliable values. In this case it is necessary to perform consecutive reads until two returned values are identical.

58.7.15 LPTIM configuration register 2 (LPTIM_CFGR2)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2SEL[1:0]		Res.	Res.	IC1SEL[1:0]	
										rw	rw			rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IN2SEL[1:0]		Res.	Res.	IN1SEL[1:0]	
										rw	rw			rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:20 **IC2SEL[1:0]**: LPTIM input capture 2 selection

The IC2SEL bits control the LPTIM Input capture 2 multiplexer, which connects LPTIM Input capture 2 to one of the available inputs.
00: lptim_ic2_mux0
01: lptim_ic2_mux1
10: lptim_ic2_mux2
11: lptim_ic2_mux3
For connection details refer to *Section 58.4.3: LPTIM input and trigger mapping*.

Bits 19:18 Reserved, must be kept at reset value.

Bits 17:16 **IC1SEL[1:0]**: LPTIM input capture 1 selection

The IC1SEL bits control the LPTIM Input capture 1 multiplexer, which connects LPTIM Input capture 1 to one of the available inputs.
00: lptim_ic1_mux0
01: lptim_ic1_mux1
10: lptim_ic1_mux2
11: lptim_ic1_mux3
For connection details refer to *Section 58.4.3: LPTIM input and trigger mapping*.

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:4 **IN2SEL[1:0]**: LPTIM input 2 selection

The IN2SEL bits control the LPTIM input 2 multiplexer, which connects LPTIM input 2 to one of the available inputs.
00: lptim_in2_mux0
01: lptim_in2_mux1
10: lptim_in2_mux2
11: lptim_in2_mux3
For connection details refer to *Section 58.4.3: LPTIM input and trigger mapping*.

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **IN1SEL[1:0]**: LPTIM input 1 selection

The IN1SEL bits control the LPTIM input 1 multiplexer, which connects LPTIM input 1 to one of the available inputs.
00: lptim_in1_mux0
01: lptim_in1_mux1
10: lptim_in1_mux2
11: lptim_in1_mux3
For connection details refer to *Section 58.4.3: LPTIM input and trigger mapping*.

58.7.16 LPTIM repetition register (LPTIM_RCR)

Address offset: 0x028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **REP[7:0]**: Repetition register value
REP is the repetition value for the LPTIM.

Caution: The LPTIM_RCR register must only be modified when the LPTIM is enabled (ENABLE bit set to 1). When using repetition counter with PRELOAD = 0, LPTIM_RCR register must be changed at least five counter cycles before the auto reload match event, otherwise an unpredictable behavior may occur.

58.7.17 LPTIM capture/compare mode register 1 (LPTIM_CCMR1)

Address offset: 0x02C

Reset value: 0x0000 0000

The channels can be used in input (capture mode) or in output (PWM mode). The direction of a channel is defined by configuring the corresponding CCxSEL bits.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	IC2F[1:0]		Res.	Res.	IC2PSC[1:0]		Res.	Res.	Res.	Res.	CC2P[1:0]		CC2E	CC2 SEL
		rw	rw			rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	IC1F[1:0]		Res.	Res.	IC1PSC[1:0]		Res.	Res.	Res.	Res.	CC1P[1:0]		CC1E	CC1 SEL
		rw	rw			rw	rw					rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **IC2F[1:0]**: Input capture 2 filter

This bitfield defines the number of consecutive equal samples that are detected when a level change occurs on an external input capture signal before it is considered as a valid level transition. An internal clock source must be present to use this feature.

00: Any external input capture signal level change is considered as a valid transition

01: External input capture signal level change must be stable for at least 2 clock periods before it is considered as valid transition.

10: External input capture signal level change must be stable for at least 4 clock periods before it is considered as valid transition.

11: External input capture signal level change must be stable for at least 8 clock periods before it is considered as valid transition.

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:24 **IC2PSC[1:0]**: Input capture 2 prescaler

This bitfield defines the ratio of the prescaler acting on the CC2 input (IC2).

00: No prescaler, capture is done each time an edge is detected on the capture input

01: Capture is done once every 2 events

10: Capture is done once every 4 events

11: Capture is done once every 8 events

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:18 **CC2P[1:0]**: Capture/compare 2 output polarity.

Condition: CC2 as output

Only bit2 is used to set polarity when output mode is enabled, bit3 is don't care.

0: OC2 active high

1: OC2 active low

Condition: CC2 as input

This field is used to select the IC2 polarity for capture operations.

00: Rising edge, circuit is sensitive to IC2 rising edge

01: Falling edge, circuit is sensitive to IC2 falling edge

10: Reserved, do not use this configuration.

11: Both edges, circuit is sensitive to both IC2 rising and falling edges.

Bit 17 **CC2E**: Capture/compare 2 output enable.

Condition: CC2 as output

0: Off - OC2 is not active. Writing '0' to the CC2E bit resets the ue_dma_req signal only if all the other LPTIM channels are disabled.

1: On - OC2 signal is output on the corresponding output pin

Condition: CC2 as input

This bit determines if a capture of the counter value can actually be done into the input capture/compare register 2 (LPTIM_CCR2) or not.

0: Capture disabled. Writing '0' to the CC2E bit resets the associated ic2_dma_req signal.

1: Capture enabled.

Bit 16 **CC2SEL**: Capture/compare 2 selection

This bitfield defines the direction of the channel, input (capture) or output mode.

0: CC2 channel is configured in output PWM mode

1: CC2 channel is configured in input capture mode

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:12 **IC1F[1:0]**: Input capture 1 filter

This bitfield defines the number of consecutive equal samples that are detected when a level change occurs on an external input capture signal before it is considered as a valid level transition. An internal clock source must be present to use this feature.

00: Any external input capture signal level change is considered as a valid transition

01: External input capture signal level change must be stable for at least 2 clock periods before it is considered as valid transition.

10: External input capture signal level change must be stable for at least 4 clock periods before it is considered as valid transition.

11: External input capture signal level change must be stable for at least 8 clock periods before it is considered as valid transition.

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:8 **IC1PSC[1:0]**: Input capture 1 prescaler

This bitfield defines the ratio of the prescaler acting on the CC1 input (IC1).

00: No prescaler, capture is done each time an edge is detected on the capture input

01: Capture is done once every 2 events

10: Capture is done once every 4 events

11: Capture is done once every 8 events

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:2 **CC1P[1:0]**: Capture/compare 1 output polarity.

Condition: CC1 as output

Only bit2 is used to set polarity when output mode is enabled, bit3 is don't care.

0: OC1 active high, the LPTIM output reflects the compare results between LPTIM_ARR and LPTIM_CCRx registers

1: OC1 active low, the LPTIM output reflects the inverse of the compare results between LPTIM_ARR and LPTIM_CCRx registers

Condition: CC1 as input

This field is used to select the IC1 polarity for capture operations.

00: Rising edge, circuit is sensitive to IC1 rising edge

01: Falling edge, circuit is sensitive to IC1 falling edge

10: Reserved, do not use this configuration.

11: Both edges, circuit is sensitive to both IC1 rising and falling edges.

Bit 1 **CC1E**: Capture/compare 1 output enable.

Condition: CC1 as output

0: Off - OC1 is not active. Writing '0' to the CC1E bit resets the ue_dma_req signal only if all the other LPTIM channels are disabled.

1: On - OC1 signal is output on the corresponding output pin

Condition: CC1 as input

This bit determines if a capture of the counter value can actually be done into the input capture/compare register 1 (LPTIM_CCR1) or not.

0: Capture disabled. Writing '0' to the CC1E bit resets the associated ic1_dma_req signal.

1: Capture enabled.

Bit 0 **CC1SEL**: Capture/compare 1 selection

This bitfield defines the direction of the channel input (capture) or output mode.

0: CC1 channel is configured in output PWM mode

1: CC1 channel is configured in input capture mode

Caution: After a write to the LPTIM_CCMRx register, a new write operation to the same register can only be performed after a delay that must be equal or greater than the value of (PRESC × 3)

kernel clock cycles, PRESC[2:0] being the clock decimal division factor (1, 2, 4,..128). Any successive write violating this delay, leads to unpredictable results.

Caution: The CCxSEL, ICxF[1:0], CCxP[1:0] and ICxPSC[1:0] fields must only be modified when the channel x is disabled (CCxE bit reset to 0).

If LPTIM does not implement any channel this register is reserved. Refer to Section 58.3.

58.7.18 LPTIM compare register 2 (LPTIM_CCR2)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCR2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CCR2[15:0]**: Capture/compare 2 value

If channel CC2 is configured as output:

CCR2 is the value to be loaded in the capture/compare 2 register.

Depending on the PRELOAD option, the CCR2 register is immediately updated if the PRELOAD bit is reset and updated at next LPTIM update event if PREOAD bit is reset.

The capture/compare register 2 contains the value to be compared to the counter LPTIM_CNT and signaled on OC2 output.

If channel CC2 is configured as input:

CCR2 becomes read-only, it contains the counter value transferred by the last input capture 2 event. The LPTIM_CCR2 register is read-only and cannot be programmed.

Caution: The LPTIM_CCR2 register must only be modified when the LPTIM is enabled (ENABLE bit set to 1).

Note: If the LPTIM implements less than 2 channels this register is reserved. Refer to *Section 58.3: LPTIM implementation*.

58.7.19 LPTIM register map

The following table summarizes the LPTIM registers.

Table 571. LPTIM register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	LPTIMx_ISR (x = 4 to 5)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPOK	UE	DOWN(2)	UP(2)	ARROK	CMP1OK	EXTTRIG	ARRM	CC1IF
	Reset value								0																0	0	0	0	0	0	0	0	0

Table 571. LPTIM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	LPTIMx_ISR (x = 1 to 3) Output compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROK	Res.	Res.	Res.	Res.	CMP2OK(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2IF(1)	REPOK	UE	DOWN(2)	UP(2)	ARROK	CMP1OK	EXTTRIG	ARRM	CC1IF
	Reset value								0					0										0	0	0	0	0	0	0	0	0	0
	LPTIMx_ISR (x = 1 to 3) Input capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2OF(1)	CC1OF	Res.	Res.	CC2IF(1)	REPOK	UE	DOWN(2)	UP(2)	ARROK	Res.	EXTTRIG	ARRM	CC1IF
	Reset value								0											0	0			0	0	0	0	0	0		0	0	0
0x004	LPTIMx_ICR (x = 4 to 5)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROKCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPOKCF	UECF	DOWNCF(2)	UPCF(2)	ARROKCF	CMP1OKCF	EXTTRIGCF	ARRMCF	CC1CF
	Reset value								0															0	0	0	0	0	0	0	0	0	0
0x004	LPTIMx_ICR (x = 1 to 3) Output compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROKCF	Res.	Res.	Res.	Res.	CMP2OKCF(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2CF(1)	REPOKCF	UECF	DOWNCF(2)	UPCF(2)	ARROKCF	CMP1OKCF	EXTTRIGCF	ARRMCF	CC1CF
	Reset value								0					0										0	0	0	0	0	0	0	0	0	0
	LPTIMx_ICR (x = 1 to 3) Input capture mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIEROKCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2OCF(1)	CC1OCF	Res.	Res.	CC2CF(1)	REPOKCF	UECF	DOWNCF(2)	UPCF(2)	ARROKCF	Res.	EXTTRIGCF	ARRMCF	CC1CF
	Reset value								0											0	0			0	0	0	0	0	0		0	0	0
0x008	LPTIMx_DIER (x = 4 to 5)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REPOKIE	UEIE	DOWNIE(2)	UPIE(2)	ARROKIE	CMP1OKIE	EXTTRIGIE	ARRMIE	CC1IE
	Reset value																								0	0	0	0	0	0	0	0	0
0x008	LPTIMx_DIER (x = 1 to 3) Output compare mode	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UEDE	Res.	Res.	Res.	CMP2OKIE(1)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2IE(1)	REPOKIE	UEIE	DOWNIE(2)	UPIE(2)	ARROKIE	CMP1OKIE	EXTTRIGIE	ARRMIE	CC1IE
	Reset value									0				0										0	0	0	0	0	0	0	0	0	0
	LPTIMx_DIER (x = 1 to 3) Input capture mode	Res.	Res.	Res.	Res.	Res.	Res.	CC2DE(1)	Res.	UEDE	Res.	Res.	Res.	Res.	Res.	Res.	CC1DE	Res.	Res.	CC2OIE(1)	CC1OIE	Res.	Res.	CC2IE(1)	REPOKIE	UEIE	DOWNIE(2)	UPIE(2)	ARROKIE	Res.	EXTTRIGIE	ARRMIE	CC1IE
	Reset value							0		0							0			0	0			0	0	0	0	0	0		0	0	0
0x00C	LPTIM_CFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ENC(2)	COUNTMODE	PRELOAD	WAVPOL(3)	WAVE	TIMOUT	TRIGEN		Res.	TRIGSEL[2:0]		Res.	PRESC			Res.	TRGFLT		CKFLT		CKPOL		CKSEL		
	Reset value								0	0	0	0	0	0	0	0		0	0	0		0	0	0	0	0		0	0	0	0	0	0
0x010	LPTIM_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTARE	COUNTRST	CNTSTRT	SNGSTRT	ENABLE
	Reset value																												0	0	0	0	0
0x014	LPTIM_CCR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR1[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 571. LPTIM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x018	LPTIM_ARR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x01C	LPTIM_CNT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	LPTIM_ CFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IC2SEL[1:0]		Res.	Res.	IC1SEL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IN2SEL[1:0]		Res.	Res.	IN1SEL[1:0]	
	Reset value											0	0			0	0											0	0			0	0
0x028	LPTIM_RCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REP[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x02C	LPTIM_CCMR1(4)	Res.	Res.	IC2F[1:0]		Res.	Res.	IC2PSC[1:0]		Res.	Res.	Res.	Res.	CC2P[1:0]		CC2E	CC2SEL	Res.	Res.	IC1F[1:0]		Res.	Res.	IC1PSC[1:0]		Res.	Res.	Res.	Res.	CC1P[1:0]		CC1E	CC1SEL
	Reset value			0	0			0	0					0	0	0	0			0	0			0	0					0	0	0	0
0x034	LPTIM_CCR2(5)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCR2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1. If LPTIM does not implement at least 2 channels this bit is reserved. Refer to *Section 58.3: LPTIM implementation*.
2. If LPTIM does not support Encoder mode feature, this bit is reserved. Refer to *Section 58.3: LPTIM implementation*.
3. If the LPTIM implements at least one capture/compare channel, this bit is reserved. Refer to *Section 58.3: LPTIM implementation*.
4. If LPTIM does not implement any channel this register is reserved. Refer to *Section 58.3: LPTIM implementation*.
5. If the LPTIM implements less than 2 channels this register is reserved. Refer to *Section 58.3: LPTIM implementation*.

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

59 Independent watchdog (IWDG)

59.1 IWDG introduction

The independent watchdog (IWDG) peripheral offers a high safety level, thanks to its capability to detect malfunctions due to software or hardware failures.

The IWDG is clocked by an independent clock, and stays active even if the main clock fails.

In addition, the watchdog function is performed in the V_{DD} voltage domain, allowing the IWDG to remain functional even in low power modes. Refer to *Section 59.3* to check the capability of the IWDG in this product.

The IWDG is best suited for applications that require the watchdog to run as a totally independent process outside the main application, making it very reliable to detect any unexpected behavior.

59.2 IWDG main features

- 12-bit down-counter
- Dual voltage domain, thus enabling operation in low power modes
- Independent clock
- Early wake-up interrupt generation
- Reset generation
  - In case of timeout
  - In case of refresh outside the expected window

59.3 IWDG implementation

Table 572. IWDG features (1)

IWDG modes/features	IWDG
LSI used as IWDG kernel clock (iwdg_ker_ck)	X
Window function	X
Early wake-up interrupt generation	X
System reset generation(2)	X
Capability to work in system Stop	X
Capability to work in system Standby	X
Capability to generate an interrupt in system Stop	X
Capability to generate an interrupt in system Standby	-
Capability to be frozen when the microcontroller enters in Debug mode(3)	X
Option bytes to control the activity in Stop mode(4)	X
Option bytes to control the activity in Standby mode(5)	X
Option bytes to control the Hardware mode(6)	X

1. 'X' = supported, '-' = not supported.
2. Refer to the RCC section for additional information.
3. Controlled via DBG_IWDG_STOP in DBG section.

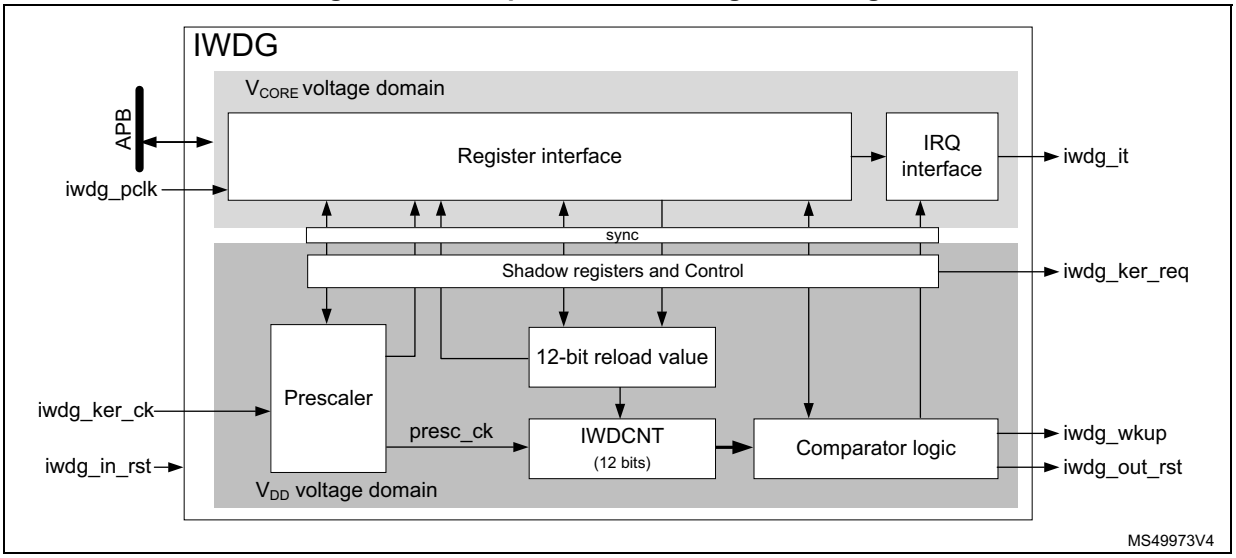
4. Controlled via the option byte IWDG_STOP in FLASH section.
5. Controlled via the option byte IWDG_STDBY in FLASH section.
6. Controlled via the option byte IWDG_SW in FLASH section.

59.4 IWDG functional description

59.4.1 IWDG block diagram

Figure 771 shows the functional blocks of the independent watchdog module.

Figure 771. Independent watchdog block diagram



The register and IRQ interfaces are located into the V_{CORE} voltage domain. The watchdog function itself is located into the V_{DD} voltage domain to remain functional in low power modes. See *Section 59.3* for IWDG capabilities.

The register and IRQ interfaces are mainly clocked by the APB clock (iwdg_pclk), while the watchdog function is clocked by a dedicated kernel clock (iwdg_ker_ck). A synchronization mechanism makes the data exchange between the two domains possible. Note that most of the registers located in the register interface are shadowed into the V_{DD} voltage domain.

The IWDG down-counter (IWDCNT) is clocked by the prescaled clock (presc_ck). The prescaled clock is generated from the kernel clock iwdg_ker_ck divided by the prescaler, according to PR[3:0] bitfield.

The table below gives the timing delays according to the actions performed on the IWDG: changing the prescaler, the timeout, the window or the early wake-up comparator values or doing a refresh.

Table 573. IWDG delays versus actions (1)

TD_{RVU}, TD_{PVU}		TD_{WVU}		TD_{EWU}		$TD_{Refresh}$		$TD_{RefAuto}$	
Min	**Max**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**
5 T_k	6 T_k	-	6 T_k	-	6 T_k	2 T_k	2 T_k + T_p	-	T_p

1. Tk represents a period of the kernel clock input, Tp represents a period of presc_ck.

59.4.2 IWDG internal signals

The list of IWDG internal signals is detailed in *Table 574*.

Table 574. IWDG internal input/output signals

Signal name	Signal type	Description
iwdg_ker_ck	Input	IWDG kernel clock
iwdg_ker_req	Input	IWDG kernel clock request
iwdg_pclk	Input	IWDG APB clock
iwdg_out_rst	Output	IWDG reset output
iwdg_in_rst	Input	IWDG reset input
iwdg_wkup	Output	IWDG wake-up event
iwdg_it	Output	IWDG early wake-up interrupt

59.4.3 Software and hardware watchdog modes

The watchdog modes allow the application to select the way the IWDG is enabled, either by software commands (Software watchdog mode), or automatically (Hardware watchdog mode). All other functions work similarly for both Software and Hardware modes.

The software watchdog mode is the default working mode. The independent watchdog is started by writing the value 0x0000 CCCC into the *IWDG key register (IWDG_KR)*, and the IWDCNT starts counting down from the reset value (0xFFF).

In the hardware watchdog mode the independent watchdog is started automatically at power-on, or every time it is reset (via iwdg_in_rst). The IWDCNT down-counter starts counting down from the reset value 0xFFF. The hardware watchdog mode feature is enabled through the device option bits, see *Section 59.3* for details.

When the IWDG is enabled the ONF flag is set to 1.

When the IWDCNT reaches 0x000, a reset signal is generated (iwdg_out_rst asserted).

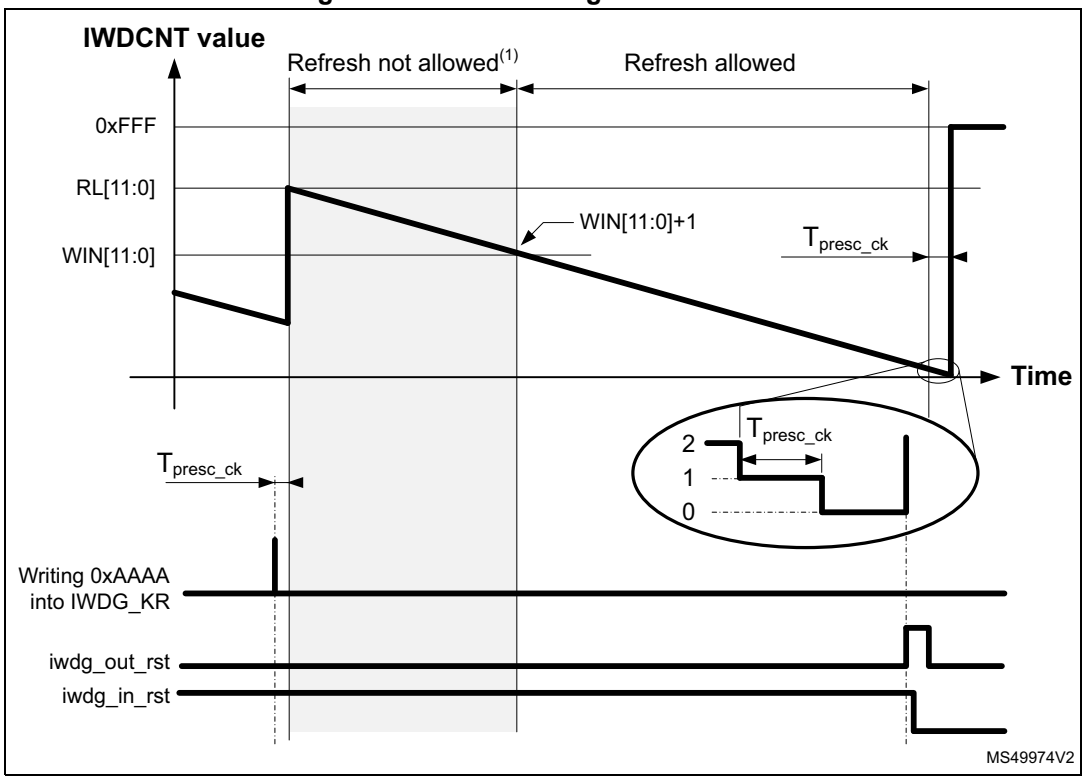
Whenever the key value 0x0000 AAAA is written in the *IWDG key register (IWDG_KR)*, the IWDG_RLR value is reloaded into the IWDCNT, and the watchdog reset is prevented.

Due to re-synchronization delays, the IWDG must be refreshed before the IWDCNT down-counter reaches 1.

Once started, the IWDG can be stopped only when it is reset (iwdg_in_rst asserted).

As shown in *Figure 772*, when the refresh command is executed, one period of presc_ck later, the IWDCNT is reloaded with the content of RL[11:0].

Figure 772. Reset timing due to timeout



1. If window option activated.

If the IWDG is not refreshed before the IWDCNT reaches 1, the IWDG generates a reset (iwdg_out_rst is asserted). In return, the RCC resets the IWDG (assertion of iwdg_in_rst) to clear the reset source.

59.4.4 Window option

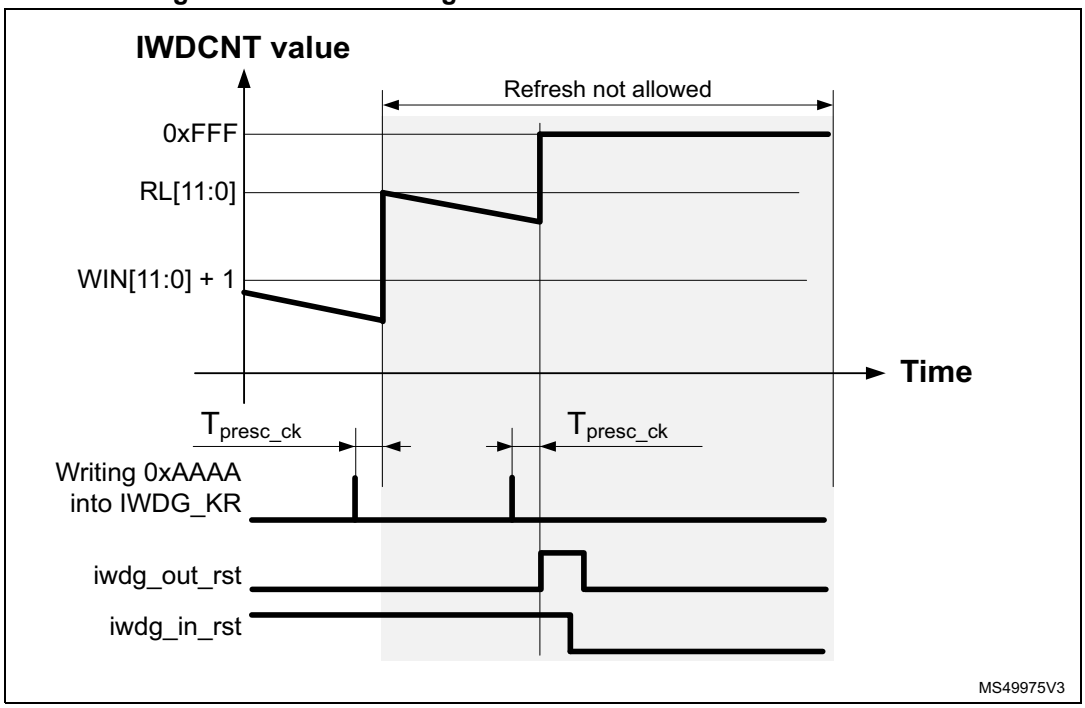
The IWDG can also work as a window watchdog, by setting the appropriate window in the *IWDG window register (IWDG_WINR)*.

If the reload operation is performed while the counter is greater than WIN[11:0] + 1, a reset is generated. WIN[11:0] is located in the *IWDG window register (IWDG_WINR)*. As shown in *Figure 773*, the reset is generated one period of presc_ck after the unexpected refresh command.

The default value of the *IWDG window register (IWDG_WINR)* is 0x0000 0FFF, so, if not updated, the window option is disabled.

As soon as the window value changes, the down-counter (IWDCNT) is reloaded with the RL[11:0] value, to ease the estimation for where the next refresh must take place.

Figure 773. Reset timing due to refresh in the not allowed area



Configuring the IWDG when the window option is enabled

1. Enable the IWDG by writing 0x0000 CCCC in the *IWDG key register (IWDG_KR)*.
2. Enable register access by writing 0x0000 5555 in the *IWDG key register (IWDG_KR)*.
3. Write the IWDG prescaler by programming *IWDG prescaler register (IWDG_PR)*.
4. Write the *IWDG reload register (IWDG_RLR)*.
5. If needed, enable the early wake-up interrupt, and program the early wake-up comparator, by writing the proper values into the *IWDG early wake-up interrupt register (IWDG_EWCR)*.
6. Write to the *IWDG window register (IWDG_WINR)*. This automatically reloads the IWDCNT down-counter with the RL[11:0] value.
7. Wait for the registers to be updated (IWDG_SR = 0x0000 0000).
8. Write 0x0000 0000 into *IWDG key register (IWDG_KR)* to write-protect registers.

Note: Step 7 can be skipped if the application does not intend to disable the APB clock after the completion of this sequence.

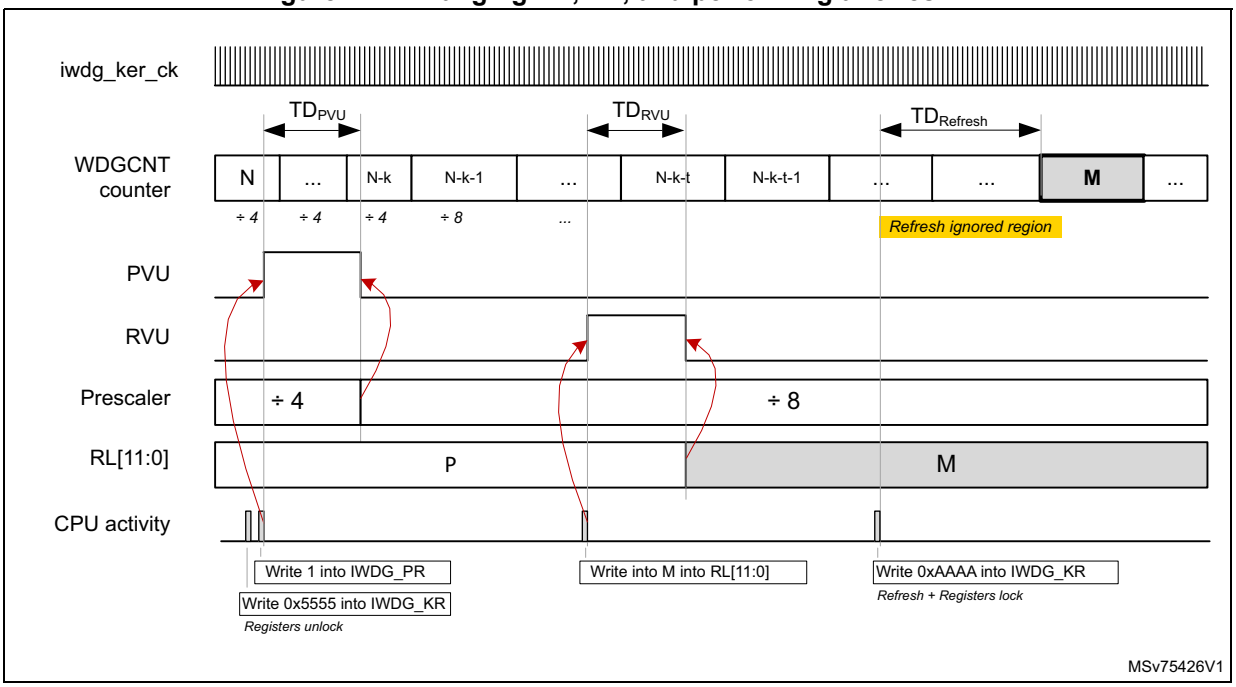
Configuring the IWDG when the window option is disabled

When the window option it is not used, the IWDG can be configured as follows:

1. Enable the IWDG by writing 0x0000 CCCC in the *IWDG key register (IWDG_KR)*.
2. Enable register access by writing 0x0000 5555 in the *IWDG key register (IWDG_KR)*.
3. Write the prescaler by programming the *IWDG prescaler register (IWDG_PR)*.
4. Write the *IWDG reload register (IWDG_RLR)*.
5. If needed, enable the early wake-up interrupt, and program the early wake-up comparator, by writing the proper values into the *IWDG early wake-up interrupt register (IWDG_EWCR)*.
6. Wait for the registers to be updated (IWDG_SR = 0x0000 0000).
7. Refresh the counter with RL[11:0] value, and write-protect registers by writing 0x0000 AAAA into *IWDG key register (IWDG_KR)*.

The figure below shows a sequence example changing the prescaler, the reload value, and then performing a refresh.

Figure 774. Changing PR, RL, and performing a refresh(1)



1. Refer to *Table 573: IWDG delays versus actions* for details on timing values.

Note: When the new prescaler value is accepted by the IWDG (falling edge of PVU), the new division ratio is effective when the current division sequence is completed (N-k in the drawing).

Note: The timeout delay between the refresh (write 0xAAAA into IWDG_KR) and the watchdog reset is increased by TDRefresh.

If a refresh command is sent while the previous refresh command is not yet completed, this last refresh command is ignored.

Updating the window comparator

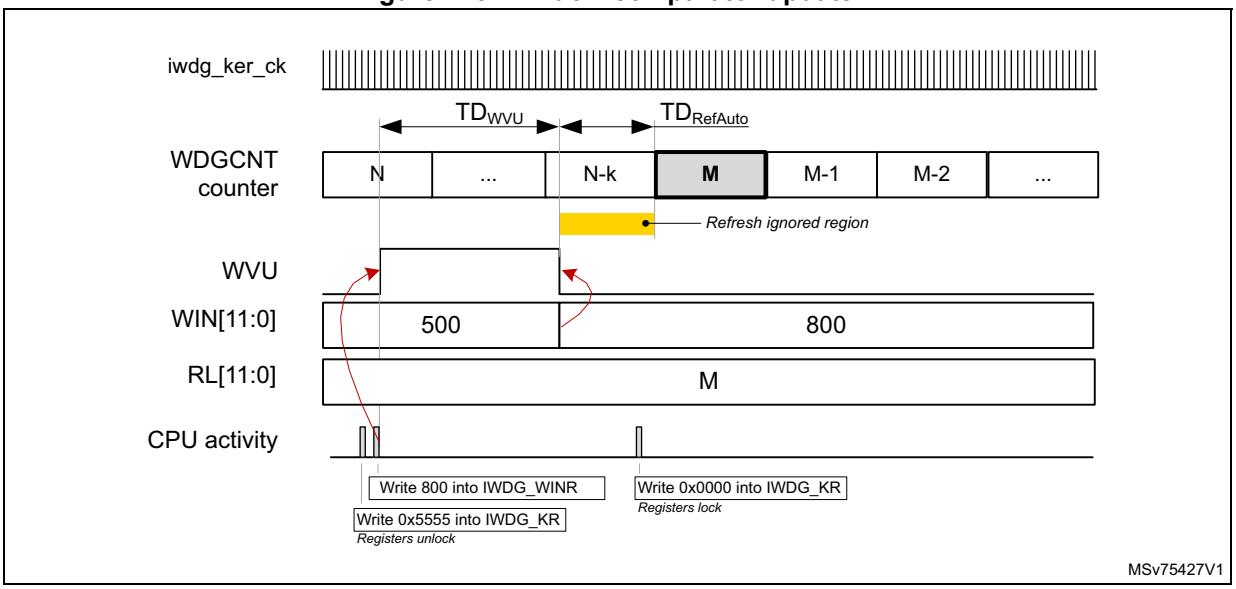
It is possible to update the window comparator when the IWDG is already running. The IWDCNT is reloaded as well. The following sequence can be performed to update the window comparator:

1. Enable register access by writing 0x0000 5555 in the IWDG key register (IWDG_KR).
2. Write to the IWDG window register (IWDG_WINR). This automatically reloads the IWDCNT down-counter with RL[11:0] value.
3. Wait for WVU = 0
4. Lock registers by writing IWDG_KR to 0x0000 0000

Step *3* can be skipped if the application does not intend to disable the APB clock after the completion of this sequence.

Figure 775 shows this sequence. As soon as the IWDG_WINR register is written, the WVU flag goes high for TDwvu. After a window comparator update, a refresh is automatically performed. The refresh is effective in the worst case, on the next prescaler clock active edge after the falling edge of WVU (TDRefAuto)

Figure 775. Window comparator update(1)



1. Refer to *Table 573: IWDG delays versus actions* for details on timing values.

59.4.5 Debug

When the processor enters into Debug mode (core halted), the IWDCNT down-counter either continues to work normally or stops, depending on debug capability of the product. Refer to *Section 59.3* for details on the capabilities of this product.

59.4.6 Register access protection

Write accesses to *IWDG prescaler register (IWDG_PR)*, *IWDG reload register (IWDG_RLR)*, *IWDG early wake-up interrupt register (IWDG_EWCR)* and *IWDG window register (IWDG_WINR)* are protected. To modify them, first write 0x0000 5555 in the *IWDG key register (IWDG_KR)*. A write access to this register with a different value breaks the

sequence and register access is protected again. This is the case of the reload operation (writing 0x0000 AAAA).

A status register is available to indicate that an update of the prescaler or the down-counter reload value or the window value is ongoing.

59.5 IWDG low power modes

Depending on option bytes configuration, the IWDG can continue counting or not during the low power modes. Refer to *Section 59.3* for details.

Table 575. Effect of low power modes on IWDG

Mode	Description
Sleep	No effect. IWDG interrupts cause the device to exit from the mode.
Stop	The IWDG remains active or not, depending on option bytes configuration. Refer to *Section 59.3* for details. IWDG interrupts cause the device exit the Stop mode.
Standby	The IWDG remains active or not, depending on option bytes configuration. Refer to *Section 59.3* for details. IWDG interrupts do not make the device to exit from Standby mode.
Shutdown	The IWDG is not working.

59.6 IWDG interrupts

The IWDG offers the possibility to generate an early interrupt depending on the value of the down-counter. The early interrupt is enabled by setting the EWIE bit of the *IWDG early wake-up interrupt register (IWDG_EWCR)* to 1.

A comparator value (EWIT[11:0]) allows the application to define the position where the early interrupt must be generated.

When the IWDCNT down-counter reaches the value of EWIT[11:0] - 1, the iwdg_wkup is activated, making it possible for the system to exit from low power modes, if needed.

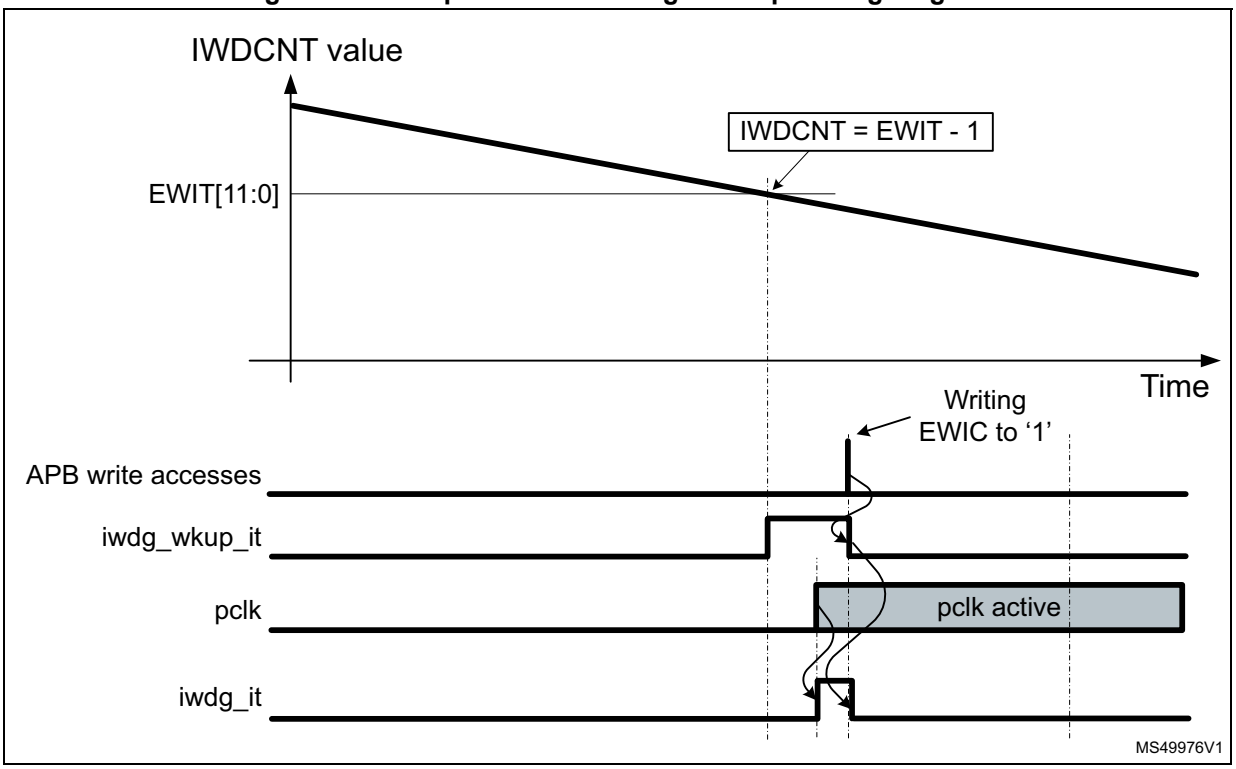
When the APB clock is available, the iwdg_it is activated as well.

In addition, the flag EWIF of the *IWDG status register (IWDG_SR)* is set to 1.

The EWI interrupt is acknowledged by writing 1 to the EWIC bit in the *IWDG interrupt clear register (IWDG_ICR)*.

Writing into the IWDG_EWCR register also triggers a refresh of the down-counter (IWDCNT) with the reload value RL[11:0].

Figure 776. Independent watchdog interrupt timing diagram



The early wake-up interrupt (EWI) can be used if specific safety operations or data logging must be performed before the watchdog reset is generated.

Changing the early wake-up comparator value

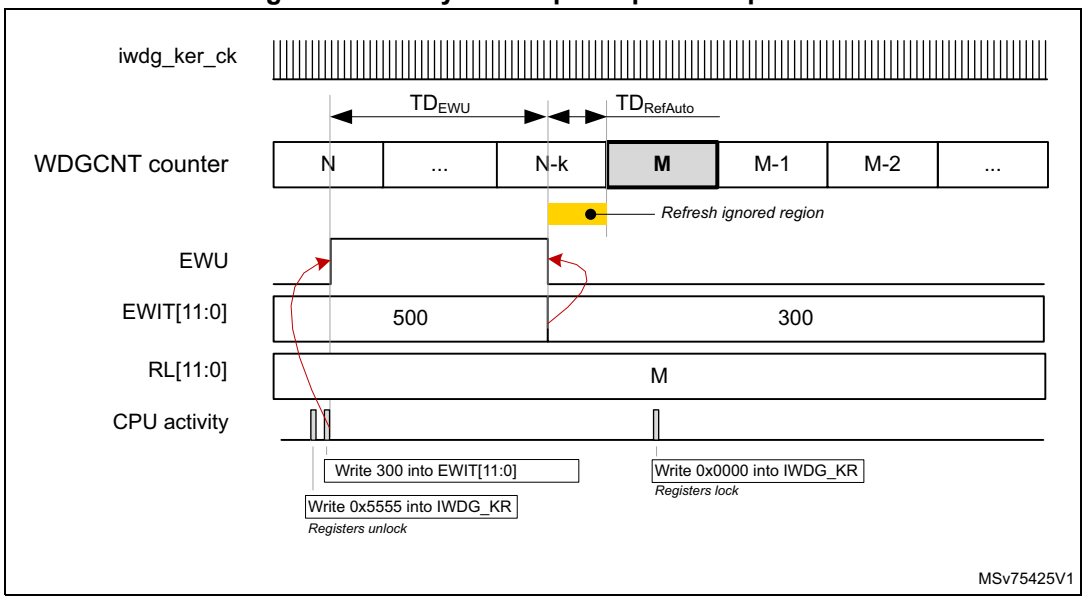
It is possible to change the early wake-up comparator value or to enable/disable the interrupt generation at any time, by performing the following sequence:

1. Enable register access by writing 0x0000 5555 in the *IWDG key register (IWDG_KR)*.
2. Enable or disable the early wake-up interrupt, and/or program the early wake-up comparator, by writing the proper values into the *IWDG early wake-up interrupt register (IWDG_EWCR)*.
3. Wait for EWU = 0, EWU is located into the *IWDG status register (IWDG_SR)*.
4. Write-protect registers by writing 0x0000 0000 to *IWDG key register (IWDG_KR)*.

Step *3* can be skipped if the application does not intend to disable the APB clock after the completion of this sequence.

Figure 777 shows this sequence. During the early wake-up comparator update operation the flag EWU remains to 1 for TDewu. A refresh is automatically performed. The refresh is effective on the worst case, on the next prescaler clock active edge after the falling edge of EWU (TDRefAuto).

Figure 777. Early wake-up comparator update(1)



1. Refer to *Table 573: IWDG delays versus actions* for details on timing values.

Table 576 summarizes the IWDG interrupt request.

Table 576. IWDG interrupt request

Interrupt event	Event flag	Interrupt clear method	Interrupt enable control bit	Activated interrupt	
				iwdg_it	iwdg_wkup_it
IWDCNT reaches EWIT value	EWIF	Writing EWIC to 1	EWIE	Y(1)	Y(2)

1. Generated when a clock is present on iwdg_pclk input.
2. Generated when a clock is present on iwdg_ker_ck input.

59.7 IWDG registers

Refer to *Section 1.2 on page 156* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by half-words (16-bit) or words (32-bit).

Most of the registers located into the register interface are shadowed into the V_{DD} voltage domain. When the iwdg_in_rst is asserted, the watchdog logic and the shadow registers located into the V_{DD} voltage domain are reset.

When the application reads back a watchdog register, the hardware transfers the value of the corresponding shadow register to the register interface.

When the application writes a watchdog register, the hardware updates the corresponding shadow register.

59.7.1 IWDG key register (IWDG_KR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **KEY[15:0]:** Key value (write only, read 0x0000)

These bits can be used for several functions, depending upon the value written by the application:

- 0xAAAA: reloads the RL[11:0] value into the IWDCNT down-counter (watchdog refresh), and write-protects registers. This value must be written by software at regular intervals, otherwise the watchdog generates a reset when the counter reaches 0.
- 0x5555: enables write-accesses to the registers.
- 0xCCCC: enables the watchdog (except if the hardware watchdog option is selected) and write-protects registers.
- values different from 0x5555: write-protects registers.

Note that only IWDG_PR, IWDG_RLR, IWDG_EWCR and IWDG_WINR registers have a write-protection mechanism.

59.7.2 IWDG prescaler register (IWDG_PR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PR[3:0]:** Prescaler divider

These bits are write access protected, see *Section 59.4.6*. They are written by software to select the prescaler divider feeding the counter clock. PVU bit of the *IWDG status register (IWDG_SR)* must be reset to be able to change the prescaler divider.

0000: divider / 4
0001: divider / 8
0010: divider / 16
0011: divider / 32
0100: divider / 64
0101: divider / 128
0110: divider / 256
0111: divider / 512
Others: divider / 1024

Note: Reading this register returns the prescaler value from the V_{DD} voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the PVU bit in the IWDG status register (IWDG_SR) is reset.

59.7.3 IWDG reload register (IWDG_RLR)

Address offset: 0x08

Reset value: 0x0000 0FFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	RL[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **RL[11:0]:** Watchdog counter reload value

These bits are write access protected, see *Section 59.4.6*. They are written by software to define the value to be loaded in the watchdog counter each time the value 0xAAAA is written in the *IWDG key register (IWDG_KR)*. The watchdog counter counts down from this value. The timeout period is a function of this value and the prescaler.clock. It is not recommended to set RL[11:0] to a value lower than 2.

The RVU bit in the *IWDG status register (IWDG_SR)* must be reset to be able to change the reload value.

Note: Reading this register returns the reload value from the V_{DD} voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing, hence the value read from this register is valid only when the RVU bit in the IWDG status register (IWDG_SR) is reset.

59.7.4 IWDG status register (IWDG_SR)

Address offset: 0x0C

Reset value: 0x0000 0000 (0xFFFF FEFF)

This register contains various status flags. Note that the mask value between parenthesis means that the reset value of ONF bit is not defined. When the IWDG is configured in

software mode, the reset value of ONF bit is 0, when the IWDG is configured in hardware mode, the reset value of ONF bit is 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EWIF	Res.	Res.	Res.	Res.	Res.	Res.	ONF	Res.	Res.	Res.	Res.	EWU	WVU	RVU	PVU
r							r					r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **EWIF**: Watchdog early interrupt flag

This bit is set to ‘1’ by hardware in order to indicate that an early interrupt is pending. This bit must be cleared by the software by writing the bit EWIC of IWDG_ICR register to ‘1’.

Bits 14:9 Reserved, must be kept at reset value.

Bit 8 **ONF**: Watchdog enable status bit

Set to ‘1’ by hardware as soon as the IWDG is started. In software mode, it remains to '1' until the IWDG is reset. In hardware mode, this bit is always set to '1'.

0: The IWDG is not activated

1: The IWDG is activated and needs to be refreshed regularly by the application

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **EWU**: Watchdog interrupt comparator value update

This bit is set by hardware to indicate that an update of the interrupt comparator value (EWIT[11:0]) or an update of the EWIE is ongoing. It is reset by hardware when the update operation is completed in the V_{DD} voltage domain. Refer to *Table 573: IWDG delays versus actions* for delay values.

The EWIT[11:0] and EWIE fields can be updated only when EWU bit is reset.

Bit 2 **WVU:** Watchdog counter window value update

This bit is set by hardware to indicate that an update of the window value is ongoing. It is reset by hardware when the reload value update operation is completed in the V_{DD} voltage domain. Refer to *Table 573: IWDG delays versus actions* for delay values.

The window value can be updated only when WVU bit is reset.

This bit is generated only if generic “window” = 1.

Bit 1 **RVU:** Watchdog counter reload value update

This bit is set by hardware to indicate that an update of the reload value is ongoing. It is reset by hardware when the reload value update operation is completed in the V_{DD} voltage domain. Refer to *Table 573: IWDG delays versus actions* for delay values.

The reload value can be updated only when RVU bit is reset.

Bit 0 **PVU:** Watchdog prescaler value update

This bit is set by hardware to indicate that an update of the prescaler value is ongoing. It is reset by hardware when the prescaler update operation is completed in the V_{DD} voltage domain. Refer to *Table 573: IWDG delays versus actions* for delay values.

The prescaler value can be updated only when PVU bit is reset.

Note: If several reload, prescaler, early interrupt position or window values are used by the application, it is mandatory to wait until RVU bit is reset before changing the reload value, to wait until PVU bit is reset before changing the prescaler value, to wait until WVU bit is reset before changing the window value, and to wait until EWU bit is reset before changing the

early interrupt position value. After updating the prescaler and/or the reload/window/early interrupt value, it is not necessary to wait until RVU or PVU or WVU or EWU is reset before continuing code execution, except in case of low power mode entry.

59.7.5 IWDG window register (IWDG_WINR)

Address offset: 0x10

Reset value: 0x0000 0FFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	WIN[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **WIN[11:0]:** Watchdog counter window value

These bits are write access protected, see *Section 59.4.6*.They contain the high limit of the window value to be compared with the downcounter.

To prevent a reset, the IWDCNT downcounter must be reloaded when its value is lower than WIN[11:0] + 1 and greater than 1.

The WVU bit in the *IWDG status register (IWDG_SR)* must be reset to be able to change the reload value.

Note: Reading this register returns the reload value from the V_{DD} voltage domain. This value may not be valid if a write operation to this register is ongoing. For this reason the value read from this register is valid only when the WVU bit in the IWDG status register (IWDG_SR) is reset.

59.7.6 IWDG early wake-up interrupt register (IWDG_EWCR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EWIE	Res.	Res.	Res.	EWIT[11:0]											
rw				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **EWIE**: Watchdog early interrupt enable

Set and reset by software.

0: The early interrupt interface is disabled.

1: The early interrupt interface is enabled.

The EWU bit in the *IWDG status register (IWDG_SR)* must be reset to be able to change the value of this bit.

Bits 14:12 Reserved, must be kept at reset value.

Bits 11:0 **EWIT[11:0]:** Watchdog counter early wake-up interrupt value

These bits are write access protected (see *Section 59.4.6*). They are written by software to define at which position of the IWDCNT down-counter the early wake-up interrupt must be generated. The early interrupt is generated when the IWDCNT is lower or equal to EWIT[11:0] - 1.

EWIT[11:0] must be bigger than 1.

An interrupt is generated only if EWIE = 1.

The EWU bit in the *IWDG status register (IWDG_SR)* must be reset to be able to change the reload value.

Note: Reading this register returns the Early wake-up comparator value and the Interrupt enable bit from the V_{DD} voltage domain. This value may not be up to date/valid if a write operation to this register is ongoing, hence the value read from this register is valid only when the EWU bit in the IWDG status register (IWDG_SR) is reset.

59.7.7 IWDG interrupt clear register (IWDG_ICR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EWIC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **EWIC**: Watchdog early interrupt acknowledge

The software must write a 1 into this bit in order to acknowledge the early wake-up interrupt and to clear the EWIF flag. Writing 0 has not effect, reading this flag returns a 0.

Bits 14:0 Reserved, must be kept at reset value.

59.7.8 IWDG register map

Table 577. IWDG register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	IWDG_KR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	IWDG_PR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PR[3:0]			
	Reset value																													0	0	0	0
0x08	IWDG_RLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RL[11:0]											
	Reset value																					1	1	1	1	1	1	1	1	1	1	1	1
0x0C	IWDG_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIF	Res.	Res.	Res.	Res.	Res.	Res.	ONF	Res.	Res.	Res.	Res.	EWU	WVU	RVU	PVU
	Reset value																	0							x					0	0	0	0
0x10	IWDG_WINR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WIN[11:0]											
	Reset value																					1	1	1	1	1	1	1	1	1	1	1	1
0x14	IWDG_EWCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIE	Res.	Res.	Res.	EWIT[11:0]											
	Reset value																	0				0	0	0	0	0	0	0	0	0	0	0	0
0x18	IWDG_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																	0															

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

60 System window watchdog (WWDG)

60.1 WWDG introduction

The system window watchdog (WWDG) is used to detect the occurrence of a software fault, usually generated by external interference or by unforeseen logical conditions, which causes the application program to abandon its normal sequence.

The watchdog circuit generates a reset on expiry of a programmed time period, unless the program refreshes the contents of the down-counter before the T6 bit is cleared. A reset is also generated if the 7-bit down-counter value (in the control register) is refreshed before the down-counter reaches the window register value. This implies that the counter must be refreshed in a limited window.

The WWDG clock is prescaled from the APB clock and has a configurable time window that can be programmed to detect abnormally late or early application behavior.

The WWDG is best suited for applications requiring the watchdog to react within an accurate timing window.

60.2 WWDG main features

- Programmable free-running down-counter
- Conditional reset
  - Reset (if watchdog activated) when the down-counter value becomes lower than 0x40
  - Reset (if watchdog activated) if the down-counter is reloaded outside the window (see *Figure 779*)
- Early wake-up interrupt (EWI): triggered (if enabled and the watchdog activated) when the down-counter is equal to 0x40

60.3 WWDG implementation

Table 578. WWDG features(1)

WWDG mode / feature	WWDG
Window function	X
Early wake-up interrupt generation	X
System reset generation(2)	X
Capability to work in system Stop	-
Capability to work in system Standby	-
Capability to be frozen when the microcontroller enters in Debug mode(3)	X
Option bytes to control the hardware mode(4)	X

1. “X” = supported, “-” = not supported.
2. Refer to the RCC section for additional information.
3. Controlled via WWDG_STOP in DBG block.

4. Controlled via the option byte WWDG_SW. When WWDG_SW is set in HW mode the WWDG is running as soon as the CPU is in Run or Sleep modes.

60.4 WWDG functional description

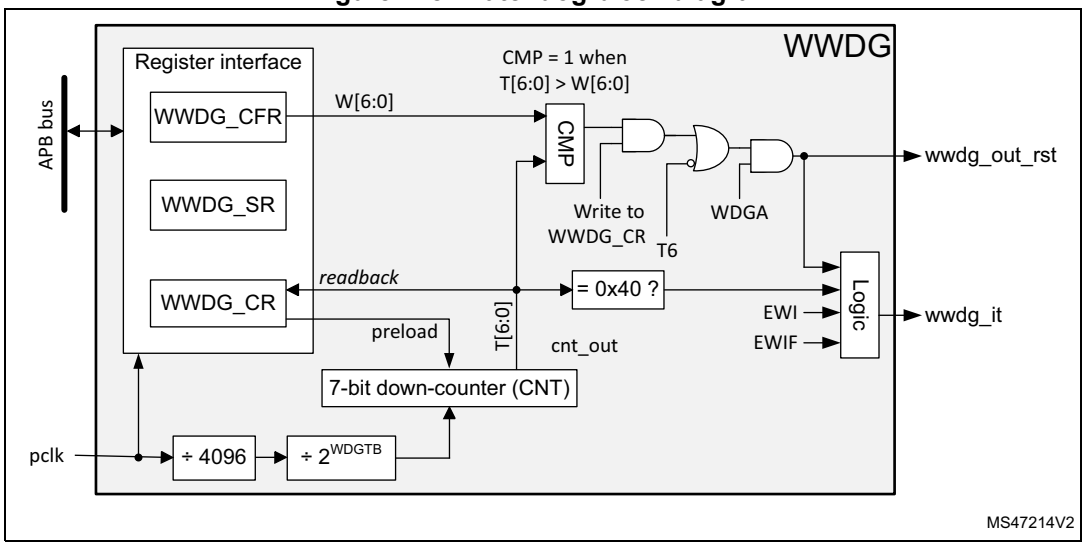
If the watchdog is activated (the WDGA bit is set in the WWDG_CR register), and when the 7-bit down-counter (T[6:0] bits) is decremented from 0x40 to 0x3F (T6 becomes cleared), it initiates a reset. If the software reloads the counter while the counter is greater than the value stored in the window register, then a reset is generated.

The application program must write in the WWDG_CR register at regular intervals during normal operation to prevent a reset. This operation can take place only when the counter value is lower than or equal to the window register value, and higher than 0x3F. The value to be stored in the WWDG_CR register must be between 0xFF and 0xC0.

Refer to *Figure 778* for the WWDG block diagram.

60.4.1 WWDG block diagram

Figure 778. Watchdog block diagram



60.4.2 WWDG internal signals

Table 579 gives the list of WWDG internal signals.

Table 579. WWDG internal input/output signals

Signal name	Signal type	Description
pclk	Digital input	APB bus clock
wwdg_out_rst	Digital output	WWDG reset signal output
wwdg_it	Digital output	WWDG early interrupt output

60.4.3 Enabling the watchdog

When the user option WWDG_SW selects “Software window watchdog”, the watchdog is always disabled after a reset. It is enabled by setting the WDGA bit in the WWDG_CR register, then it cannot be disabled again, except by a reset.

When the user option WWDG_SW selects “Hardware window watchdog”, the watchdog is always enabled after a reset, it cannot be disabled.

60.4.4 Controlling the down-counter

This down-counter is free-running, counting down even if the watchdog is disabled. When the watchdog is enabled, the T6 bit must be set to prevent generating an immediate reset.

The T[5:0] bits contain the number of increments that represent the time delay before the watchdog produces a reset. The timing varies between a minimum and a maximum value, due to the unknown status of the prescaler when writing to the WWDG_CR register (see *Figure 779*). The *WWDG configuration register (WWDG_CFR)* contains the high limit of the window: to prevent a reset, the down-counter must be reloaded when its value is lower than or equal to the window register value, and greater than 0x3F. *Figure 779* describes the window watchdog process.

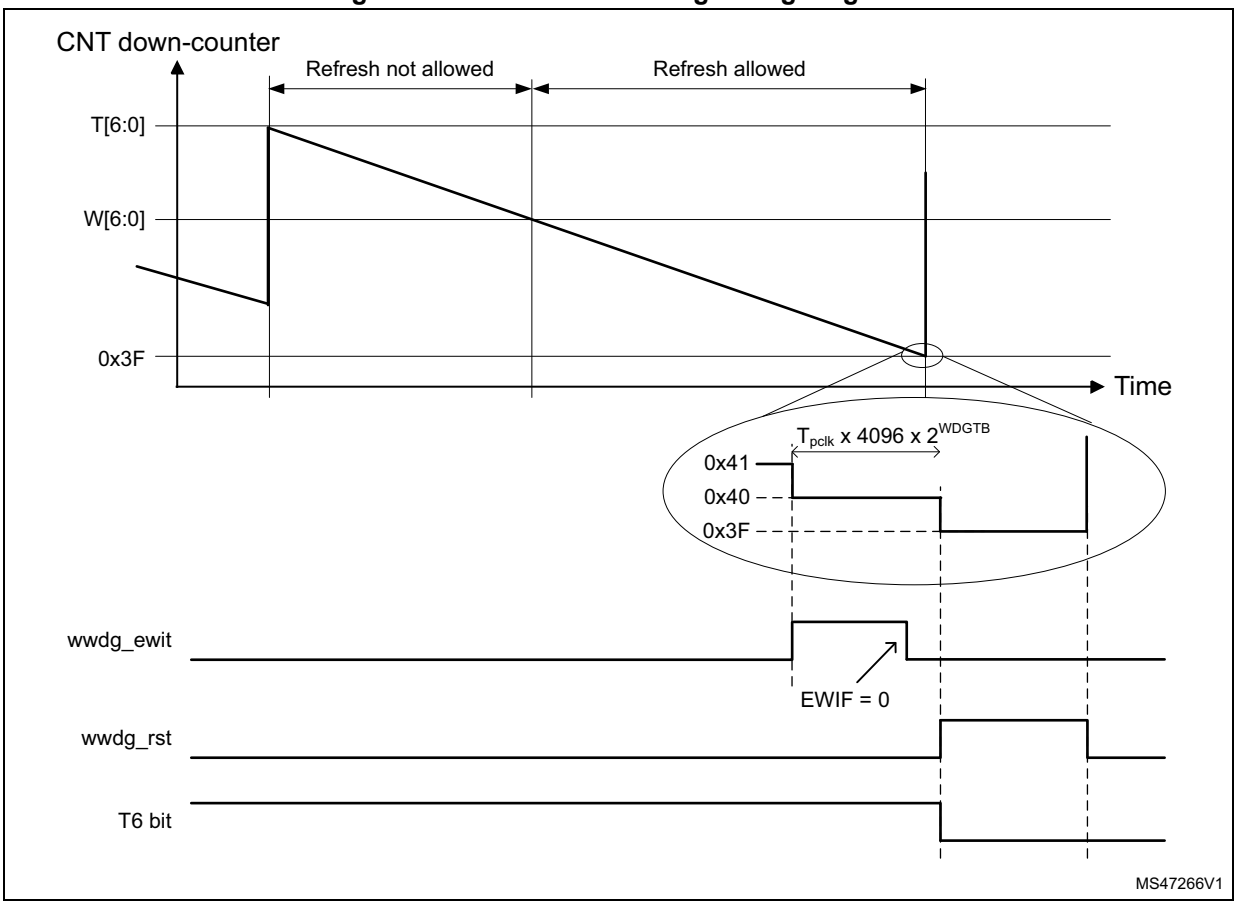
Note: The T6 bit can be used to generate a software reset (the WDGA bit is set and the T6 bit is cleared).

60.4.5 How to program the watchdog timeout

Use the formula in *Figure 779* to calculate the WWDG timeout.

Warning: When writing to the WWDG_CR register, always write 1 in the T6 bit to avoid generating an immediate reset.

Figure 779. Window watchdog timing diagram



The formula to calculate the timeout value is given by:

$$t_{WWDG} = t_{PCLK} \times 4096 \times 2^{WDGTB[2:0]} \times (T[5:0] + 1) \quad (ms)$$

where:

- t_{WWDG}: WWDG timeout
- t_{PCLK}: APB clock period measured in ms
- 4096: value corresponding to internal divider

As an example, if APB frequency is 48 MHz, WDGTB[2:0] is set to 3, and T[5:0] is set to 63:

$$t_{WWDG} = (1/\ 48000) \times 4096 \times 2^{3} \times (63 + 1) = 43.69ms$$

Refer to the datasheet for the minimum and maximum values of t_{WWDG}.

60.4.6 Debug mode

When the device enters debug mode (processor halted), the WWDG counter either continues to work normally or stops, depending on the configuration bit in DBG module. For more details, refer to *Section 40: Debug support (DBG)*.

60.5 WWDG interrupts

The early wake-up interrupt (EWI) can be used if specific safety operations or data logging must be performed before the reset is generated. To enable the early wake-up interrupt, the application must:

- Write EWIF bit of WWDG_SR register to 0, to clear unwanted pending interrupt
- Write EWI bit of WWDG_CFR register to 1, to enable interrupt

When the down-counter reaches the value 0x40, a watchdog interrupt is generated, and the corresponding interrupt service routine (ISR) can be used to trigger specific actions (such as communications or data logging), before resetting the device.

In some applications, the EWI interrupt can be used to manage a software system check and/or system recovery/graceful degradation, without generating a WWDG reset. In this case the corresponding ISR must reload the WWDG counter to avoid the WWDG reset, then trigger the required actions.

The watchdog interrupt is cleared by writing 0 to the EWIF bit in the WWDG_SR register.

Note: When the watchdog interrupt cannot be served (for example due to a system lock in a higher priority task), the WWDG reset is eventually generated.

Table 580. WWDG interrupt requests

Interrupt		Event flag	Enable control bit	Interrupt clearing method	Exit from mode		
Vector	Event				Sleep	Stop(1)	Standby(1)
WWDG(2)	Early wake-up interrupt	EWIF	EWI	Write EWIF flag to 0	Yes	No	No

1. The WWDG interrupt can have additional capabilities, refer to *Section 60.3* for details.
2. WWDG vector corresponds to the assertion of the wwdg_it signal.

60.6 WWDG registers

Refer to *Section 1.2: List of abbreviations for registers* for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by halfwords (16-bit) or words (32-bit).

60.6.1 WWDG control register (WWDG_CR)

Address offset: 0x000

Reset value: 0x0000 007F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDGA	T[6:0]						
								rs	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **WDGA:** Activation bit

This bit is set by software and only cleared by hardware after a reset. When WDGA = 1, the watchdog can generate a reset.

0: Watchdog disabled

1: Watchdog enabled

Bits 6:0 **T[6:0]:** 7-bit counter (MSB to LSB)

These bits contain the value of the watchdog counter, decremented every (4096 x $2^{WDGTB[2:0]}$) PCLK cycles. A reset is produced when it is decremented from 0x40 to 0x3F (T6 becomes cleared).

60.6.2 WWDG configuration register (WWDG_CFR)

Address offset: 0x004

Reset value: 0x0000 007F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	WDGTB[2:0]			Res.	EWI	Res.	Res.	W[6:0]						
		rw	rw	rw		rs			rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:11 **WDGTB[2:0]:** Timer base

The timebase of the prescaler can be modified as follows:

000: CK counter clock (PCLK div 4096) div 1

001: CK counter clock (PCLK div 4096) div 2

010: CK counter clock (PCLK div 4096) div 4

011: CK counter clock (PCLK div 4096) div 8

100: CK counter clock (PCLK div 4096) div 16

101: CK counter clock (PCLK div 4096) div 32

110: CK counter clock (PCLK div 4096) div 64

111: CK counter clock (PCLK div 4096) div 128

Bit 10 Reserved, must be kept at reset value.

Bit 9 **EWI:** Early wake-up interrupt enable

Set by software and cleared by hardware after a reset. When set, an interrupt occurs whenever the counter reaches the value 0x40.

Bits 8:7 Reserved, must be kept at reset value.

Bits 6:0 **W[6:0]:** 7-bit window value

These bits contain the window value to be compared with the down-counter.

60.6.3 WWDG status register (WWDG_SR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIF
															rc_w0

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **EWIF:** Early wake-up interrupt flag

This bit is set by hardware when the counter has reached the value 0x40. It must be cleared by software by writing 0. Writing 1 has no effect. This bit is also set if the interrupt is not enabled.

60.6.4 WWDG register map

The following table gives the WWDG register map and reset values.

Table 581. WWDG register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	WWDG_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDGA	T[6:0]						
	Reset value																									0	1	1	1	1	1	1	1
0x004	WWDG_CFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDGTB [2:0]			Res.	EWI	Res.	Res.	W[6:0]						
	Reset value																			0	0	0		0			1	1	1	1	1	1	1
0x008	WWDG_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EWIF
	Reset value																																0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

61 Real-time clock (RTC)

61.1 RTC introduction

The RTC provides an automatic wake-up to manage all low-power modes.

The real-time clock (RTC) is an independent BCD timer/counter. The RTC provides a time-of-day clock/calendar with programmable alarm interrupts.

As long as the supply voltage remains in the operating range, the RTC never stops, regardless of the device status (Run mode, low-power mode or under reset).

The RTC is functional in V_{BAT} mode.

61.2 RTC main features

The RTC supports the following features (see *Figure 780: RTC block diagram*):

- Calendar with subseconds, seconds, minutes, hours (12 or 24 format), week day, date, month, year, in BCD (binary-coded decimal) format.
- Binary mode with 32-bit free-running counter.
- Automatic correction for 28, 29 (leap year), 30, and 31 days of the month.
- Two programmable alarms.
- On-the-fly correction from 1 to 32767 RTC clock pulses. This can be used to synchronize it with a master clock.
- Reference clock detection: a more precise second source clock (50 or 60 Hz) can be used to enhance the calendar precision.
- Digital calibration circuit with 0.95 ppm resolution, to compensate for quartz crystal inaccuracy.
- Timestamp feature which can be used to save the calendar content. This function can be triggered by an event on the timestamp pin, or by a tamper event, or by a switch to V_{BAT} mode.
- 17-bit auto-reload wake-up timer (WUT) for periodic events with programmable resolution and period.
- TrustZone support:
  - RTC fully securable
  - Alarm A, alarm B, wake-up Timer and timestamp individual secure or nonsecure configuration
- Alarm A, alarm B, wake-up Timer and timestamp individual privilege protection

The RTC is supplied through a switch that takes power either from the V_{DD} supply when present or from the VBAT pin.

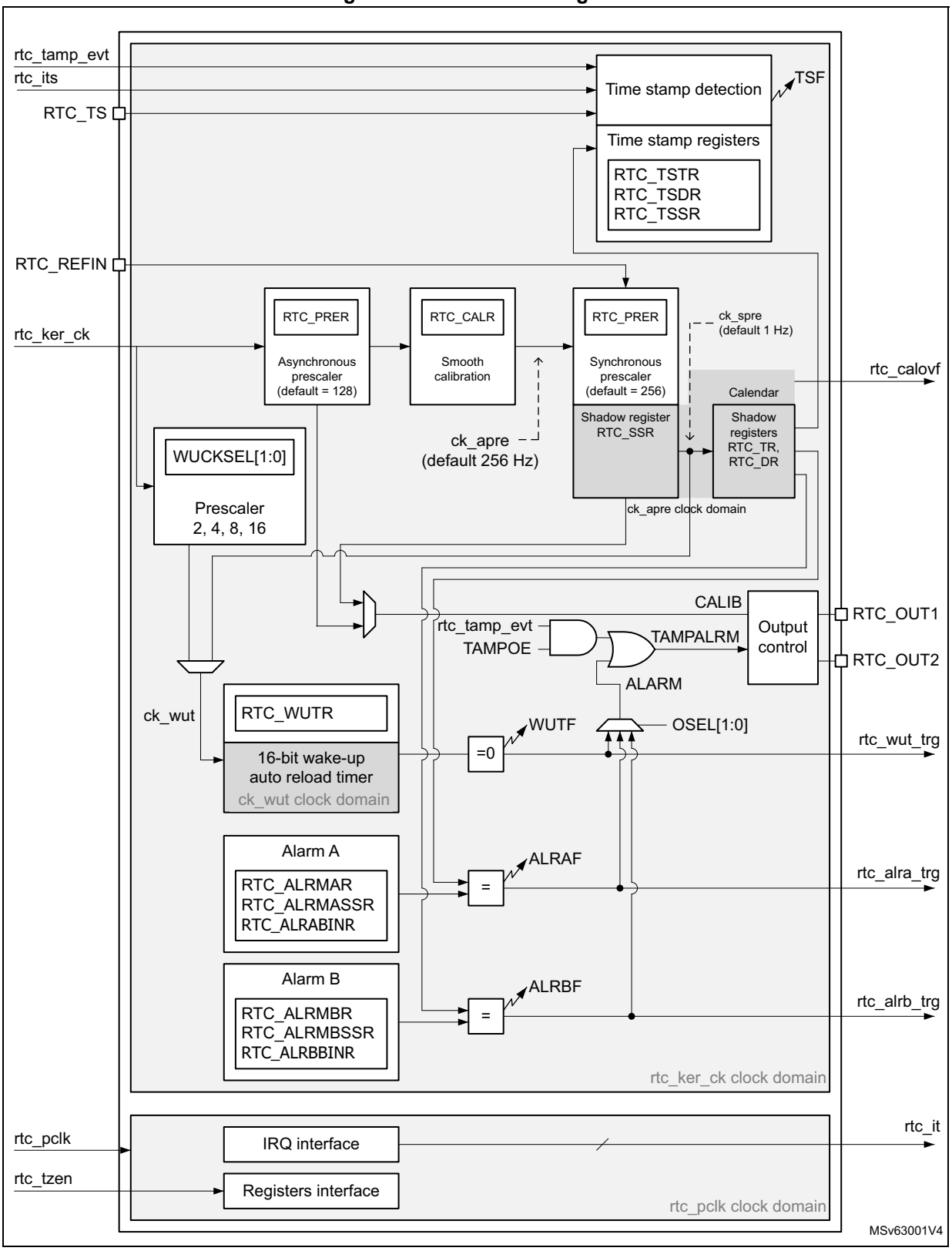
The RTC is functional in V_{BAT} mode and in all low-power modes when it is clocked by the LSE.

All RTC events (Alarm, wake-up Timer, Timestamp) can generate an interrupt and wake-up the device from the low-power modes.

61.3 RTC functional description

61.3.1 RTC block diagram

Figure 780. RTC block diagram



61.3.2 RTC pins and internal signals

Table 582. RTC input/output pins

Pin name	Signal type	Description
RTC_TS	Input	RTC timestamp input
RTC_REFIN	Input	RTC 50 or 60 Hz reference clock input
RTC_OUT1	Output	RTC output 1
RTC_OUT2	Output	RTC output 2

RTC_OUT1 and RTC_OUT2 which select one of the following two outputs:

- CALIB: 512 Hz or 1 Hz clock output (with an LSE frequency of 32.768 kHz). This output is enabled by setting the COE bit in the RTC_CR register.
- TAMPALRM: This output is the OR between rtc_tamp_evt and ALARM signals.

ALARM is enabled by configuring the OSEL[1:0] bits in the RTC_CR register which select the alarm A, alarm B or wake-up outputs. rtc_tamp_evt is enabled by setting the TAMPOE bit in the RTC_CR register which selects the tamper event outputs.

Table 583. RTC internal input/output signals

Internal signal name	Signal type	Description
rtc_ker_ck	Input	RTC kernel clock, also named RTCCLK in this document
rtc_pclk	Input	RTC APB clock
rtc_its	Input	RTC internal timestamp event
rtc_tamp_evt	Input	Tamper event (internal or external) detected in TAMP peripheral
rtc_tzen	Input	RTC TrustZone enabled
rtc_it	Output	RTC interrupts (refer to *Section 61.5: RTC interrupts* for details)
rtc_alra_trg	Output	RTC alarm A event detection trigger
rtc_alrb_trg	Output	RTC alarm B event detection trigger
rtc_wut_trg	Output	RTC wake-up timer event detection trigger
rtc_calovf	Output	RTC calendar overflow: this signal is generated when the RTC calendar reaches its maximum value, on the 31st of December 99, at 23:59:59. The calendar is then frozen and cannot overflow.

The RTC kernel clock is usually the LSE at 32.768 kHz although it is possible to select other clock sources in the RCC (refer to RCC for more details). Some functions are not available in some low-power modes or V_{BAT} when the selected clock is not LSE. Refer to *Section 61.4: RTC low-power modes* for more details.

Table 584. RTC interconnection

Signal name	Source/destination
rtc_its	From power controller (PWR): main power loss/switch to V_{BAT} detection output
rtc_tamp_evt	From TAMP peripheral: tamp_evt
rtc_tzen	Tied to 1
rtc_calovf	To TAMP peripheral: tamp_itamp5

The triggers outputs can be used as triggers for other peripherals.

61.3.3 GPIOs controlled by the RTC and TAMP

The GPIOs included in the Battery Backup Domain (V_{BAT}) are directly controlled by the peripherals providing functions on these I/Os, whatever the GPIO configuration.

RTC_OUT1, RTC_TS, TAMP_IN1 and TAMP_OUT2 are mapped on the same pin (PC13). The RTC and TAMP functions mapped on PC13 are available in all low-power modes and in V_{BAT} mode.

The output mechanism follows the priority order shown in *Table 585*.

Table 585. RTC pin PC13 configuration(1)

PC13 pin function	OSEL[1:0] (ALARM output enable)	TAMPOE (TAMPER output enable)	COE (CALIB output enable)	OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	TAMP2E=TAMP2AM=1 with ATOSHARE=0, or TAMPxE=TAMPxAM=1 with ATOSHARE=1 and ATOSELx=1	TAMP1E (TAMP_IN1 input enable)	TSE (RTC_TS input enable)
TAMPALRM output Push-Pull	01 or 10 or 11	0	Don't care	Don't care	0	0	Don't care	Don't care	Don't care
	00	1							
	01 or 10 or 11	1							

Table 585. RTC pin PC13 configuration(1) (continued)

PC13 pin function		OSEL[1:0] (ALARM output enable)	TAMPOE (TAMPER output enable)	COE (CALIB output enable)	OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	TAMP2E=TAMP2AM=1 with ATOSHARE=0, or TAMPxE=TAMPxAM=1 with ATOSHARE=1 and ATOSELx=1	TAMP1E (TAMP_IN1 input enable)	TSE (RTC_TS input enable)
TAMPALRM output Open-Drain(2)	No pull	01 or 10 or 11	0	Don't care	Don't care	1	0	Don't care	Don't care	Don't care
		00	1							
		01 or 10 or 11	1							
	Internal pull-up	01 or 10 or 11	0	Don't care	Don't care	1	1	Don't care	Don't care	Don't care
		00	1							
		01 or 10 or 11	1							
CALIB output PP		00	0	1	0	Don't care	Don't care	Don't care	Don't care	Don't care
TAMP_OUT2 output PP		00	0	0	0	Don't care	Don't care	1	Don't care	Don't care
TAMP_IN1 input floating		00	0	0	Don't care	Don't care	Don't care	0	1	0
		00	0	1	1					
		Don't care	Don't care	0						
RTC_TS and TAMP_IN1 input floating		00	0	0	Don't care	Don't care	Don't care	0	1	1
		00	0	1	1			0		
		Don't care	Don't care	0				0		

Table 585. RTC pin PC13 configuration(1) (continued)

PC13 pin function	OSEL[1:0] (ALARM output enable)	TAMPOE (TAMPER output enable)	COE (CALIB output enable)	OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	TAMP2E=TAMP2AM=1 with ATOSHARE=0, or TAMPxE=TAMPxAM=1 with ATOSHARE=1 and ATOSELx=1	TAMP1E (TAMP_IN1 input enable)	TSE (RTC_TS input enable)
RTC_TS input floating	00	0	0	Don't care	Don't care	Don't care	0	0	1
	00	0	1	1			0		
	Don't care	Don't care	0				0		
Wake-up pin or Standard GPIO	00	0	0	Don't care	Don't care	Don't care	0	0	0
	00	0	1	1			0		
	Don't care	Don't care	0				0		

1. OD: open drain; PP: push-pull.
2. In this configuration the GPIO must be configured in input.

In addition, it is possible to output RTC_OUT2 on PB2 pin thanks to OUT2EN bit. The different functions are mapped on RTC_OUT1 or on RTC_OUT2 depending on OSEL, COE and OUT2EN configuration, as shown in table *Table 586*.

Table 586. RTC_OUT mapping

OSEL[1:0] bits ALARM output enable)	COE bit (CALIB output enable)	OUT2EN bit	RTC_OUT1 on PC13	RTC_OUT2 on PB2
00	0	0	-	-
00	1		CALIB	-
01 or 10 or 11	Don't care		TAMPALRM	-
00	0	1	-	-
00	1		-	CALIB
01 or 10 or 11	0		-	TAMPALRM
01 or 10 or 11	1		TAMPALRM	CALIB

61.3.4 RTC secure protection modes

By default after a backup domain power-on reset, all RTC registers can be read or written in both secure and nonsecure modes, except for the RTC secure configuration register (RTC_SECCFGR) which can be written in secure mode only. The RTC protection configuration is not affected by a system reset.

When the SEC bit is set in the RTC_SECCFGR register:

- Writing the RTC registers is possible only in secure mode.
- Reading RTC_SECCFGR, RTC_PRIVCFGR, RTC_MISR, RTC_TR, RTC_DR, RTC_SSR, RTC_PRER and RTC_CALR is always possible in secure and nonsecure modes. All the other RTC registers can be read only in secure mode.

When the SEC bit is cleared, it is still possible to protect some of the registers by setting dedicated INITSEC, CALSEC, TSSEC, WUTSEC, ALRASEC or ALRBSEC control bits. If all these bits are also clear, all the RTC registers can be read and written in secure and nonsecure mode.

- When INITSEC is set:
  - RTC_TR, RTC_DR, RTC_PRER registers, plus INIT, BIN and BCDU in RTC_ICSR, FMT control bits in RTC_CR and INITPRIV in the RTC_PRIVCFGR can be written only in secure mode.
  - These registers and control bits can be read in secure and nonsecure mode.
- When CALSEC is set:
  - RTC_SHIFTR and RTC_CALR registers, plus ADD1H, SUB1H and REFCKON control bits in the RTC_CR and CALPRIV in the RTC_PRIVCFGR can be written only in secure mode.
  - These registers and control bits can be read in secure and nonsecure mode.
- When ALRASEC is set:
  - RTC_ALRMAR, RTC_ALRMASSR and RTC_ALRABINR registers, plus ALRAE, ALRAFCLR, ALRAIE and SSRUIE in the RTC_CR, CALRAF and CSSRUF in the RTC_SCR, ALRAF and SSRUF in RTC_SR, and ALRAMF and SSRUMF in RTC_SMISR can be read and written only in secure mode.
  - ALRAPRIV in the RTC_PRIVCFGR can be written only in secure mode
- When ALRBSEC is set:
  - RTC_ALRMBR, RTC_ALRMBSSR and RTC_ALRBBINR registers, plus ALRBE ALRBFCLR, ALRBIE in the RTC_CR, CALRBF in the RTC_SCR, ALRBF in RTC_SR, and ALRBMF in RTC_SMISR can be read and written only in secure mode.
  - ALRBPRIV in the RTC_PRIVCFGR can be written only in secure mode.
- When WUTSEC is set:
  - RTC_WUTR register, plus WUTE, WUTIE and WUCKSEL control bits in the RTC_CR, CWUTF in the RTC_SCR, WUTF in RTC_SR, and WUTMF in RTC_SMISR can be read and written only in secure mode.
  - WUTPRIV in the RTC_PRIVCFGR can be written only in secure mode
- When TSSEC is set:
  - RTC_TSTR, RTC_TSDR and RTC_TSSSR registers, plus TAMPTS, ITSE, TSE, TSIE, TSEDGE control bits in the RTC_CR, CITSF, CTSOVF and CTSF bits in the

RTC_SCR, TSF, TSOVF and ITSF in RTC_SR, and TSMF, TSOVMF and ITSMF in RTC_SMISR can be read and written only in secure mode.

– TSPRIV in the RTC_PRIVCFGR can be written only in secure mode

A nonsecure access to a secure-protected register is denied:

- There is no bus error generated.
- When write protected, the bits are not written.
- When read protected they are read as 0.

As soon as at least one function is configured to be secured, the RTC reset and clock control is also secured in the RCC.

61.3.5 RTC privilege protection modes

By default after a backup domain power-on reset, all RTC registers can be read or written in both privileged and unprivileged modes, except for the RTC privilege mode control register (RTC_PRIVCFGR) which can be written in privilege mode only. The RTC protection configuration is not affected by a system reset.

When the PRIV bit is set in the RTC_PRIVCFGR register:

- Writing the RTC registers is possible only in privileged mode.
- Reading the RTC_SECCFGR, RTC_PRIVCFGR, RTC_TR, RTC_DR, RTC_SSR, RTC_PRER and RTC_CALR is always possible in privilege and unprivilege modes. All the other RTC registers can be read only in privilege mode.

When the PRIV bit is cleared, it is still possible to protect some of the registers by setting dedicated INITPRIV, CALPRIV, TSPRIV, WUTPRIV, ALRAPRV or ALRBPRIV control bits. If all these bits are also cleared, all the RTC registers can be read or written in privilege and unprivilege modes.

- When INITPRIV is set:
  - RTC_TR, RTC_DR, RTC_PRER registers, plus INIT, BIN and BCDU in RTC_ICSR and FMT control bits in RTC_CR, plus INITSEC in the RTC_SECCFGR can be written only in privilege mode.
  - These registers and control bits can be read in privilege and unprivilege mode.
- When CALPRIV is set:
  - RTC_SHIFTR and RTC_CALR registers, plus ADD1H, SUB1H and REFCKON control bits in the RTC_CR, plus CALDSEC in the RTC_SECCFGR can be written only in privilege mode.
  - These registers and control bits can be read in privilege and unprivilege mode.
- When ALRAPRIV is set:
  - RTC_ALRMAR, RTC_ALRMASSR and RTC_ALRABINR registers, plus ALRAE, ALRAFCLR, ALRAIE and SSRUIE in the RTC_CR, and CALRAF and CSSRUF in the RTC_SCR, ALRAF and SSRUF in RTC_SR, and ALRAMF and SSRUMF in RTC_MISR and RTC_SMISR can be read and written only in privilege mode.
  - ALRASEC in the RTC_SECCFGR can be written only in privilege mode.
- When ALRBPRIV is set:
  - RTC_ALRMBR, RTC_ALRMBSSR and RTC_ALRBBINR registers, plus ALRBE, ALRBFCLR, ALRBIE in the RTC_CR, and CALRBF in the RTC_SCR, ALRBF in

RTC_SR, and ALRBMF in RTC_MISR and RTC_SMISR can be read and written only in privilege mode.

  - ALRBSEC in the RTC_SECCFGR can be written only in privilege mode.
- When WUTPRIV is set:
  - RTC_WUTR register, plus WUTE, WUTIE and WUCKSEL control bits in the RTC_CR, and CWUTF in the RTC_SCR, WUTF in RTC_SR, and WUTMF in RTC_MISR and RTC_SMISR can be read and written only in privilege mode.
  - WUTSEC in the RTC_SECCFGR can be written only in privilege mode.
- When TSPRIV is set:
  - RTC_TSTR, RTC_TSDR and RTC_TSSSR registers, plus TAMPTS, ITSE, TSE, TSIE, TSEDGE control bits in the RTC_CR, CITSF, CTSOVF and CTSF bits in the RTC_SCR, TSF, TSOVF and ITSF in RTC_SR, and TSMF, TSOVMF and ITSMF in RTC_MISR and RTC_SMISR can be read and written only in privilege mode.
  - TSSEC in the RTC_SECCFGR can be written only in privilege mode.

A unprivileged access to a privileged-protected register is denied:

- There is no bus error generated.
- When write protected, the bits are not written.
- When read protected they are read as 0.

61.3.6 Clock and prescalers

The RTC clocks must respect this ratio: frequency(PCLK) ≥ 2 × frequency(RTCCLK).

For more information on the RTC clock (RTCCLK) source configuration, refer to “Reset and clock control (RCC)”.

BCD mode (BIN=00)

A programmable prescaler stage generates a 1 Hz clock which is used to update the calendar. To minimize power consumption, the prescaler is split into 2 programmable prescalers (see *Figure 780: RTC block diagram*):

- A 7-bit asynchronous prescaler configured through the PREDIV_A bits of the RTC_PRER register.
- A 15-bit synchronous prescaler configured through the PREDIV_S bits of the RTC_PRER register.

Note: When both prescalers are used, it is recommended to configure the asynchronous prescaler to a high value to minimize consumption.

The asynchronous prescaler division factor is set to 128, and the synchronous division factor to 256, to obtain an internal clock frequency of 1 Hz (ck_spre) with an LSE frequency of 32.768 kHz.

The minimum division factor is 1 and the maximum division factor is 2^{22}.

This corresponds to a maximum input frequency of around 4 MHz.

f_{ck_apre} is given by the following formula:

$$f_{CK_APRE} = \frac{f_{RTCCLK}}{PREDIV_A + 1}$$

The ck_apre clock is used to clock the binary RTC_SSR subsecond downcounter. When it reaches 0, RTC_SSR is reloaded with the content of PREDIV_S.

f_{ck_spre} is given by the following formula:

$$f_{CK_SPRE} = \frac{f_{RTCCLK}}{(PREDIV_S + 1) \times (PREDIV_A + 1)}$$

The ck_spre clock can be used either to update the calendar or as timebase for the 16-bit wake-up auto-reload timer. To obtain short timeout periods, the 16-bit wake-up auto-reload timer can also run with the RTCCLK divided by the programmable 4-bit asynchronous prescaler (see *Section 61.3.10: Periodic auto-wake-up* for details).

Binary mode (BIN=01)

The SSR binary down-counter is extended to 32-bit length and is free running. The time and date calendar BCD registers are not functional.

This down-counter is clocked by ck_apre: the output of the 7-bit asynchronous prescaler configured through the PREDIV_A bits of the RTC_PRER register.

PREDIV_S value is don't care.

Mixed mode (BIN=10 or 11)

The SSR binary down-counter is extended to 32-bit length and is free running. The time and date calendar BCD registers are also available.

This down-counter is clocked by ck_apre: the output of the 7-bit asynchronous prescaler configured through the PREDIV_A bits of the RTC_PRER register. The bits BCDU[2:0] are used to define when the calendar is incremented by 1 second, using the SSR least significant bits.

61.3.7 Real-time clock and calendar

The RTC calendar time and date registers are accessed through shadow registers which are synchronized with PCLK (APB clock). They can also be accessed directly in order to avoid waiting for the synchronization duration.

- RTC_SSR for the subseconds
- RTC_TR for the time
- RTC_DR for the date

Every RTCCLK periods, the current calendar value is copied into the shadow registers, and the RSF bit of RTC_ICSR register is set (see *Section 61.6.12: RTC shift control register (RTC_SHIFTR)*). The copy is not performed in Stop and Standby mode. When exiting these modes, the shadow registers are updated after up to 4 RTCCLK periods.

When the application reads the calendar registers, it accesses the content of the shadow registers. It is possible to make a direct access to the calendar registers by setting the BYPSHAD control bit in the RTC_CR register. By default, this bit is cleared, and the user accesses the shadow registers.

When reading the RTC_SSR, RTC_TR or RTC_DR registers in BYPSHAD = 0 mode, the frequency of the APB clock (f_{APB}) must be at least 7 times the frequency of the RTC clock (f_{RTCCLK}).

The shadow registers are reset by system reset.

61.3.8 Calendar ultra-low power mode

It is possible to reduce drastically the RTC power consumption by setting the LPCAL bit in the RTC_CALR register. In this configuration, the whole RTC is clocked by ck_apre only instead of both RTCCLK and ck_apre. Consequently, some flags delays are longer, and the calibration window is longer (refer to *Section : RTC ultra-low-power mode*).

The LPCAL bit is ignored (assumed to be 0) when asynchronous prescaler division factor (PREDIV_A+1) is not a power of 2.

Switching from LPCAL=0 to LPCAL=1 or from LPCAL=1 to LPCAL=0 is not immediate and requires a few ck_apre periods to complete.

61.3.9 Programmable alarms

The RTC unit provides programmable alarm: alarm A and alarm B. The description below is given for alarm A, but can be translated in the same way for alarm B.

The programmable alarm function is enabled through the ALRAE bit in the RTC_CR register.

The ALRAF is set to 1 if the calendar subseconds, seconds, minutes, hours, date or day match the values programmed in the alarm registers RTC_ALRMASSR and RTC_ALRMAR. Each calendar field can be independently selected through the MSKx bits of the RTC_ALRMAR register, and through the MASKSSx bits of the RTC_ALRMASSR register.

When the binary mode is used, the subsecond field can be programmed in the alarm binary register RTC_ALRABINR.

The alarm interrupt is enabled through the ALRAIE bit in the RTC_CR register.

In case the Alarm is used to generate a trigger event for another peripheral, the ALRAF can be automatically cleared by hardware by configuring the ALRAFCLR bit at 1 in the RTC_CR register. In this configuration there is no need for software intervention if the only purpose is clearing the ALRAF flag.

Caution: If the seconds field is selected (MSK1 bit reset in RTC_ALRMAR), the synchronous prescaler division factor set in the RTC_PRER register must be at least 3 to ensure correct behavior.

Alarm A and alarm B (if enabled by bits OSEL[1:0] in RTC_CR register) can be routed to the TAMPALRM output. TAMPALRM output polarity can be configured through bit POL the RTC_CR register.

61.3.10 Periodic auto-wake-up

The periodic wake-up flag is generated by a 16-bit programmable auto-reload down-counter. The wake-up timer range can be extended to 17 bits.

The wake-up function is enabled through the WUTE bit in the RTC_CR register.

The wake-up timer clock input ck_wut can be:

- RTC clock (RTCCLK) divided by 2, 4, 8, or 16.

  When RTCCLK is LSE (32.768 kHz), this permits the wake-up interrupt period to be configured from 122 µs to 32 s, with a resolution down to 61 µs.
- ck_spre (usually 1 Hz internal clock) in BCD mode, or the clock used to update the calendar as defined by BCDU in binary or mixed (BCD-binary) modes.

  When ck_spre frequency is 1 Hz, a wake-up time from 1 s to around 36 hours can be achieved with one-second resolution. This large programmable time range is divided in 2 parts:
  - from 1 s to 18 hours when WUCKSEL [2:1] = 10
  - and from around 18 h to 36 h when WUCKSEL[2:1] = 11. In this last case 2^{16} is added to the 16-bit counter current value. When the initialization sequence is complete (see *Programming the wake-up timer on page 3018*), the timer starts counting down. When the wake-up function is enabled, the down-counting remains active in low-power modes. In addition, when it reaches 0, the WUTF flag is set in the RTC_SR register, and the wake-up counter is automatically reloaded with its reload value (RTC_WUTR register value).

Depending on WUTOCLR in the RTC_WUTR register, the WUTF flag must either be cleared by software (WUTOCLR = 0x0000), or the WUTF is automatically cleared by hardware when the auto-reload down counter reaches WUTOCLR value (0x0000 < WUTOCLR ≤ WUT).

The wake-up flag is output on an internal signal rtc_wut that can be used by other peripherals (refer to section *Section 61.3.1: RTC block diagram*).

When the periodic wake-up interrupt is enabled by setting the WUTIE bit in the RTC_CR register, it can exit the device from low-power modes.

The periodic wake-up flag can be routed to the TAMPALRM output provided it has been enabled through bits OSEL[1:0] of RTC_CR register. TAMPALRM output polarity can be configured through the POL bit in the RTC_CR register.

System reset, as well as low-power modes (Sleep, Stop, and Standby) have no influence on the wake-up timer.

61.3.11 RTC initialization and configuration

RTC Binary, BCD or Mixed mode

By default the RTC is in BCD mode (BIN = 00 in the RTC_ICSR register): the RTC_SSR register contains the subsecond field SS[15:0], clocked by ck_apre, allowing to generate a 1 Hz clock to update the calendar registers in BCD format (RTC_TR and RTC_DR).

When the RTC is configured in binary mode (BIN = 01 in the RTC_ICSR register): the RTC_SSR register contains the binary counter SS[31:0], clocked by ck_apre. The calendar registers in BCD format (RTC_TR and RTC_DR) are not used.

When the RTC is configured in mixed mode (BIN = 10 or 11 in the RTC_ICSR register): the RTC_SSR register contains the binary counter SS[31:0], clocked by ck_apre. The calendar is updated (1 second increment) each time the SSR[BCDU+7:0] reaches 0.

RTC register write protection

After a system reset, the RTC registers are protected against parasitic write access by the DBP bit in the power control peripheral (refer to the PWR power control section). DBP bit must be set to enable RTC registers write access.

After Backup domain reset, the RTC registers related to RTC initialization and RTC calibration are write-protected: Writing the unlock sequence in RTC_WPR is required to gain write access to these registers.

The following steps are required to unlock the write protection on the registers related to RTC initialization that is RTC_TR, RTC_DR, RTC_PRER, the bits INIT, BIN and BCDU in RTC_ICSR, and the bit FMT in RTC_CR:

1. Write 0xCA into the RTC_WPR register.
2. Write 0x53 into the RTC_WPR register.

Those two write accesses must be performed with the privilege and security mode defined by PRIV, INITPRIV, SEC, INITSEC configuration

Writing an incorrect key reactivates the write protection.

Any write access to RTC_WPR with incorrect privilege, security settings has no effect and is ignored without any illegal access notification.

The following steps are required to unlock the write protection of the registers related to RTC calibration, that is RTC_CALR, RTC_SHIFTR, and the bits SUB1H, ADD1H, REFCKON in RTC_CR:

1. Write 0xCA into the RTC_WPR register.
2. Write 0x53 into the RTC_WPR register.

Those two write accesses must be done with the privilege and security mode defined by PRIV, CALPRIV, SEC, CALSEC configuration.

Writing an incorrect key reactivates the write protection.

Any write access to RTC_WPR with the incorrect privilege, security settings has no effect and is ignored without any illegal access notification.

Calendar initialization and configuration

To program the initial time and date calendar values, including the time format and the prescaler configuration, the following sequence is required:

1. Set INIT bit to 1 in the RTC_ICSR register to enter initialization mode. In this mode, the calendar counter is stopped and its value can be updated.
2. Poll INITF bit of in the RTC_ICSR register. The initialization phase mode is entered when INITF is set to 1.
   – If LPCAL=0: INITF is set around 2 RTCCLK cycles after INIT bit is set.
   – If LPCAL=1: INITF is set up to 2 ck_apre cycle after INIT bit is set.
3. To generate a 1 Hz clock for the calendar counter, program both the prescaler factors in RTC_PRER register, plus BIN and BCDU in the RTC_ICSR register.
4. Load the initial time and date values in the shadow registers (RTC_TR and RTC_DR), and configure the time format (12 or 24 hours) through the FMT bit in the RTC_CR register.
5. Exit the initialization mode by clearing the INIT bit. The actual calendar counter value is then automatically loaded.
   – If LPCAL=0: the counting restarts after 4 RTCCLK clock cycles.
   – If LPCAL=1: the counting restarts after up to 2 RTCCLK + 1 ck_apre.

When the initialization sequence is complete, the calendar starts counting. The RTC_SSR content is initialized with:

- PREDIV_S in BCD mode (BIN=00)
- 0xFFFF FFFF in binary or mixed (BCD-binary) modes (BIN=01, 10 or 11).

In BCD mode, RTC_SSR contains the value of the synchronous prescaler counter. This enables one to calculate the exact time being maintained by the RTC down to a resolution of 1 / (PREDIV_S + 1) seconds. As a consequence, the resolution can be improved by increasing the synchronous prescaler value (PREDIV_S[14:0]). The maximum resolution allowed (30.52 µs with a 32768 Hz clock) is obtained with PREDIV_S set to 0x7FFF.

However, increasing PREDIV_S means that PREDIV_A must be decreased in order to maintain the synchronous prescaler output at 1 Hz. In this way, the frequency of the asynchronous prescaler output increases, which may increase the RTC dynamic consumption. The RTC dynamic consumption is optimized for PREDIV_A+1 being a power of 2.

Note: There is no hardware protection against incorrect calendar value initialization. Software must ensure that the initialization is performed correctly using allowed calendar BCD values.

Note: After a system reset, the application can read the INITS flag in the RTC_ICSR register to check if the calendar has been initialized or not. If this flag equals 0, the calendar has not been initialized since the year field is set at its Backup domain reset default value (0x00).

Note: To read the calendar after initialization, the software must first check that the RSF flag is set in the RTC_ICSR register.

Daylight saving time

The daylight saving time management is performed through bits SUB1H, ADD1H, and BKP of the RTC_CR register.

Using SUB1H or ADD1H, the software can subtract or add one hour to the calendar in one single operation without going through the initialization procedure.

In addition, the software can use the BKP bit to memorize this operation.

Programming the alarm

A similar procedure must be followed to program or update the programmable alarms. The procedure below is given for alarm A but can be translated in the same way for alarm B.

1. Clear ALRAE in RTC_CR to disable alarm A.
2. Program the alarm A registers (RTC_ALRMASSR/RTC_ALRMAR or RTC_ALRABINR).
3. Set ALRAE in the RTC_CR register to enable alarm A again.

Note: Each change of the RTC_CR register is taken into account after around 2 RTCCLK clock cycles due to clock synchronization.

Programming the wake-up timer

The following sequence is required to configure or change the wake-up timer auto-reload value (WUT[15:0] in RTC_WUTR):

1. Clear WUTE in RTC_CR to disable the wake-up timer.
2. Poll WUTWF until it is set in RTC_ICSR to make sure the access to wake-up auto-reload counter and to WUCKSEL[2:0] bits is allowed.
   - If WUCKSEL[2] = 0: WUTWF is set around 1 ck_wut + 1 RTCCLK cycles after WUTE bit is cleared.
   - If WUCKSEL[2] = 1: WUTWF is set up to 1 ck_apre + 1 RTCCLK cycles after WUTE bit is cleared.
3. Program the wake-up auto-reload value WUT[15:0], WUTOCLR[15:0] and the wake-up clock selection (WUCKSEL[2:0] bits in RTC_CR). Set WUTE in RTC_CR to enable the timer again. The wake-up timer restarts down-counting.
   - If WUCKSEL[2] = 0: WUTWF is cleared around 1 ck_wut + 1 RTCCLK cycles after WUTE bit is set.
   - If WUCKSEL[2] = 1: WUTWF is cleared up to 1 ck_apre + 1 RTCCLK cycles after WUTE bit is set.

61.3.12 Reading the calendar

When BYPSHAD control bit is cleared in the RTC_CR register

To read the RTC calendar registers (RTC_SSR, RTC_TR and RTC_DR) properly, the APB clock frequency (f_{PCLK}) must be equal to or greater than seven times the RTC clock frequency (f_{RTCCLK}). This ensures a secure behavior of the synchronization mechanism.

If the APB clock frequency is less than seven times the RTC clock frequency, the software must read the calendar time and date registers twice. If the second read of the RTC_TR gives the same result as the first read, this ensures that the data is correct. Otherwise a third read access must be done. In any case the APB clock frequency must never be lower than two times the RTC clock frequency.

The RSF bit is set in RTC_ICSR register each time the calendar registers are copied into the RTC_SSR, RTC_TR and RTC_DR shadow registers. The copy is performed every RTCCLK cycle. To ensure consistency between the 3 values, reading either RTC_SSR or RTC_TR locks the values in the higher-order calendar shadow registers until RTC_DR is read. In case the software makes read accesses to the calendar in a time interval smaller

than 1 RTCCLK periods: RSF must be cleared by software after the first calendar read, and then the software must wait until RSF is set before reading again the RTC_SSR, RTC_TR and RTC_DR registers.

After waking up from low-power mode (Stop or Standby), RSF must be cleared by software. The software must then wait until it is set again before reading the RTC_SSR, RTC_TR and RTC_DR registers.

The RSF bit must be cleared after wake-up and not before entering low-power mode.

After a system reset, the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers. Indeed, a system reset resets the shadow registers to their default values.

After an initialization (refer to *Calendar initialization and configuration on page 3017*): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

After synchronization (refer to *Section 61.3.14: RTC synchronization*): the software must wait until RSF is set before reading the RTC_SSR, RTC_TR and RTC_DR registers.

When the BYPSHAD control bit is set in the RTC_CR register (bypass shadow registers)

Reading the calendar registers gives the values from the calendar counters directly, thus eliminating the need to wait for the RSF bit to be set. This is especially useful after exiting from low-power modes (Stop or Standby), since the shadow registers are not updated during these modes.

When the BYPSHAD bit is set to 1, the results of the different registers might not be coherent with each other if an RTCCLK edge occurs between two read accesses to the registers. Additionally, the value of one of the registers may be incorrect if an RTCCLK edge occurs during the read operation. The software must read all the registers twice, and then compare the two readings. If the values differ, repeat the process until matching results are obtained. Alternatively, the software can just compare the two results of the least-significant calendar register.

Note: While BYPSHAD = 1, instructions which read the calendar registers require one extra APB cycle to complete.

61.3.13 Resetting the RTC

The calendar shadow registers (RTC_SSR, RTC_TR and RTC_DR) and some bits of the RTC status register (RTC_ICSR) are reset to their default values by all available system reset sources.

On the contrary, the following registers are reset to their default values by a Backup domain reset and are not affected by a system reset: the RTC current calendar registers, the RTC control register (RTC_CR), the prescaler register (RTC_PRER), the RTC calibration register (RTC_CALR), the RTC shift register (RTC_SHIFTR), the RTC timestamp registers (RTC_TSSSR, RTC_TSTR and RTC_TSDR), the wake-up timer register (RTC_WUTR), the alarm A and alarm B registers (RTC_ALRMASSR/RTC_ALRMAR/RTC_ALRABINR and RTC_ALRMBSSR/RTC_ALRMBR/RTC_ALRBBINR).

In addition, when clocked by LSE, the RTC keeps on running under system reset if the reset source is different from the Backup domain reset one (refer to RCC for details about RTC

clock sources not affected by system reset). When a Backup domain reset occurs, the RTC is stopped and all the RTC registers are set to their reset values.

61.3.14 RTC synchronization

The RTC can be synchronized to a remote clock with a high degree of precision. After reading the subsecond field (RTC_SSR or RTC_TSSSR), a calculation can be made of the precise offset between the times being maintained by the remote clock and the RTC. The RTC can then be adjusted to eliminate this offset by “shifting” its clock by a fraction of a second using RTC_SHIFTR.

The RTC can be finely adjusted using the RTC shift control register (RTC_SHIFTR). Writing to RTC_SHIFTR can shift (either delay or advance) the clock with a resolution of 1 ck_apre period.

The shift operation consists in adding the SUBFS[14:0] value to the synchronous prescaler counter SS[15:0]: this delays the clock.

If at the same time the ADD1S bit is set in BCD or mixed mode, this results in adding one second and at the same time subtracting a fraction of second, so this advances the clock. ADD1S has no effect in binary mode.

As soon as a shift operation is initiated by a write to the RTC_SHIFTR register, the SHPF flag is set by hardware to indicate that a shift operation is pending. This bit is cleared by hardware as soon as the shift operation has completed.

Caution: In mixed mode (BIN=10 or 11), the SUBFS[14:BCDU+8] must be written with 0.

Caution: Before initiating a shift operation in BCD mode, the user must check that SS[15] = 0 in order to ensure that no overflow occurs. In mixed mode, the user must check that the bit SS[BCDU+8] = 0.

Caution: This synchronization feature is not compatible with the reference clock detection feature: firmware must not write to RTC_SHIFTR when REFCKON = 1.

61.3.15 RTC reference clock detection

This feature is available only in BCD mode (BIN=00).

The update of the RTC calendar can be synchronized to a reference clock, RTC_REFIN, which is usually the mains frequency (50 or 60 Hz). The precision of the RTC_REFIN reference clock should be higher than the 32.768 kHz LSE clock. When the RTC_REFIN detection is enabled (REFCKON bit of RTC_CR set to 1), the calendar is still clocked by the LSE, and RTC_REFIN is used to compensate for the imprecision of the calendar update frequency (1 Hz).

Each 1 Hz clock edge is compared to the nearest RTC_REFIN clock edge (if one is found within a given time window). In most cases, the two clock edges are properly aligned. When the 1 Hz clock becomes misaligned due to the imprecision of the LSE clock, the RTC shifts the 1 Hz clock a bit so that future 1 Hz clock edges are aligned. Thanks to this mechanism, the calendar becomes as precise as the reference clock.

The RTC detects if the reference clock source is present by using the 256 Hz clock (ck_apre) generated from the 32.768 kHz quartz. The detection is performed during a time window around each of the calendar updates (every 1 s). The window equals 7 ck_apre periods when detecting the first reference clock edge. A smaller window of 3 ck_apre periods is used for subsequent calendar updates.

Each time the reference clock is detected in the window, the asynchronous prescaler which outputs the ck_spre clock is forced to reload. This has no effect when the reference clock and the 1 Hz clock are aligned because the prescaler is being reloaded at the same moment. When the clocks are not aligned, the reload shifts future 1 Hz clock edges a little for them to be aligned with the reference clock.

If the reference clock halts (no reference clock edge occurred during the 3 ck_apre window), the calendar is updated continuously based solely on the LSE clock. The RTC then waits for the reference clock using a large 7 ck_apre period detection window centered on the ck_spre edge.

When the RTC_REFIN detection is enabled, PREDIV_A and PREDIV_S must be set to their default values:

- PREDIV_A = 0x007F
- PREVID_S = 0x00FF

Note: RTC_REFIN clock detection is not available in Standby mode.

61.3.16 RTC smooth digital calibration

The RTC frequency can be digitally calibrated with a resolution of about 0.954 ppm with a range from -487.1 ppm to +488.5 ppm. The correction of the frequency is performed using series of small adjustments (adding and/or subtracting individual ck_cal pulses).

If LPCAL=0: ck_cal = RTCCLK

If LPCAL=1: ck_cal = ck_apre

These adjustments are fairly well distributed so that the RTC is well calibrated even when observed over short durations of time.

RTC ultra-low-power mode

The RTC consumption can be reduced by setting the LPCAL bit in the RTC calibration register (RTC_CALR). In this case, the calibration mechanism is applied on ck_apre instead of RTCCLK. The resulting accuracy is the same, but the calibration is performed during a calibration cycle of about 2^{20} x PREDIV_A x RTCCLK pulses instead of 2^{20} RTCCLK pulses when LPCAL=0.

Smooth calibration mechanism

The smooth calibration register (RTC_CALR) specifies the number of ck_cal clock cycles to be masked during the calibration cycle:

- Setting the bit CALM[0] to 1 causes exactly one pulse to be masked during the calibration cycle.
- Setting CALM[1] to 1 causes two additional cycles to be masked
- Setting CALM[2] to 1 causes four additional cycles to be masked
- and so on up to CALM[8] set to 1 which causes 256 clocks to be masked.

Note: CALM[8:0] (RTC_CALR) specifies the number of ck_cal pulses to be masked during the calibration cycle. Setting the bit CALM[0] to 1 causes exactly one pulse to be masked during the calibration cycle at the moment when cal_cnt[19:0] is 0x80000; CALM[1] = 1 causes two other cycles to be masked (when cal_cnt is 0x40000 and 0xC0000); CALM[2] = 1 causes four other cycles to be masked (cal_cnt = 0x20000/0x60000/0xA0000/ 0xE0000); and so on up to CALM[8] = 1 which causes 256 clocks to be masked (cal_cnt = 0xXX800).

While CALM permits the RTC frequency to be reduced by up to 487.1 ppm with fine resolution, the bit CALP can be used to increase the frequency by 488.5 ppm. Setting CALP to 1 effectively inserts an extra ck_cal pulse every 2^{11} ck_cal cycles, which means that 512 clocks are added during every calibration cycle.

Using CALM together with CALP, an offset ranging from -511 to +512 ck_cal cycles can be added during the calibration cycle, which translates to a calibration range of -487.1 ppm to +488.5 ppm with a resolution of about 0.954 ppm.

The formula to calculate the effective calibrated frequency (F_{CAL}) given the input frequency (F_{RTCCLK}) is as follows:

$$F_{CAL} = F_{RTCCLK} \times [1 + (CALP \times 512 - CALM) / (2^{20} + CALM - CALP \times 512)]$$

Caution: PREDIV_A must be greater or equal to 3.

Calibration when PREDIV_A < 3

The CALP bit can not be set to 1 when the asynchronous prescaler value (PREDIV_A bits in RTC_PRER register) is less than 3. If CALP was already set to 1 and PREDIV_A bits are set to a value less than 3, CALP is ignored and the calibration operates as if CALP was equal to 0.

It is however possible to perform a calibration with PREDIV_A less than 3 in BCD mode, the synchronous prescaler value (PREDIV_S) should be reduced so that each second is accelerated by 8 ck_cal clock cycles, which is equivalent to adding 256 clock cycles every calibration cycle. As a result, between 255 and 256 clock pulses (corresponding to a calibration range from 243.3 to 244.1 ppm) can effectively be added during each calibration cycle using only the CALM bits.

With a nominal RTCCLK frequency of 32768 Hz, when PREDIV_A equals 1 (division factor of 2), PREDIV_S should be set to 16379 rather than 16383 (4 less). The only other interesting case is when PREDIV_A equals 0, PREDIV_S should be set to 32759 rather than 32767 (8 less).

If PREDIV_S is reduced in this way, the formula given the effective frequency of the calibrated input clock is as follows:

$$F_{CAL} = F_{RTCCLK} \times [1 + (256 - CALM) / (2^{20} + CALM - 256)]$$

In this case, CALM[7:0] equals 0x100 (the midpoint of the CALM range) is the correct setting if RTCCLK is exactly 32768.00 Hz.

Verifying the RTC calibration

It is recommended to verify the RTC calibration with LPCAL = 0, in order to have a 32-second calibration cycle.

RTC precision is ensured by measuring the precise frequency of RTCCLK and calculating the correct CALM value and CALP values. An optional 1 Hz output is provided to allow applications to measure and verify the RTC precision.

Measuring the precise frequency of the RTC over a limited interval can result in a measurement error of up to 2 RTCCLK clock cycles over the measurement period, depending on how the digital calibration cycle is aligned with the measurement period.

However, this measurement error can be eliminated if the measurement period is the same length as the calibration cycle period. In this case, the only error observed is the error due to the resolution of the digital calibration.

- By default, the calibration cycle period is 32 seconds.

  Using this mode and measuring the accuracy of the 1 Hz output over exactly 32 seconds guarantees that the measure is within 0.477 ppm (0.5 RTCCLK cycles over 32 seconds, due to the limitation of the calibration resolution).
- CALW16 bit of the RTC_CALR register can be set to 1 to force a 16- second calibration cycle period.

  In this case, the RTC precision can be measured during 16 seconds with a maximum error of 0.954 ppm (0.5 RTCCLK cycles over 16 seconds). However, since the calibration resolution is reduced, the long term RTC precision is also reduced to 0.954 ppm: CALM[0] bit is stuck at 0 when CALW16 is set to 1.
- CALW8 bit of the RTC_CALR register can be set to 1 to force a 8-second calibration cycle period.

  In this case, the RTC precision can be measured during 8 seconds with a maximum error of 1.907 ppm (0.5 RTCCLK cycles over 8 s). The long term RTC precision is also reduced to 1.907 ppm: CALM[1:0] bits are stuck at 00 when CALW8 is set to 1.

Re-calibration on-the-fly

The calibration register (RTC_CALR) can be updated on-the-fly while RTC_ICSR/INITF = 0, by using the follow process:

1. Poll the RTC_ICSR/RECALPF (re-calibration pending flag).
2. If it is set to 0, write a new value to RTC_CALR, if necessary. RECALPF is then automatically set to 1
3. Within three ck_apre cycles after the write operation to RTC_CALR, the new calibration settings take effect.

61.3.17 Timestamp function

Timestamp is enabled by setting the TSE or ITSE bits of RTC_CR register to 1.

When TSE is set:

The calendar is saved in the timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when a timestamp event is detected on the RTC_TS pin.

When TAMPTS is set:

The calendar is saved in the timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when an internal or external tamper event is detected. Refer to *RTC control register (RTC_CR)* and refer to *Section : Timestamp on tamper event*.

When ITSE is set:

The calendar is saved in the timestamp registers (RTC_TSSSR, RTC_TSTR, RTC_TSDR) when an internal timestamp event is detected. The internal timestamp event is generated by the switch to the V_{BAT} supply.

When a timestamp event occurs, due to internal or external event, the timestamp flag bit (TSF) in RTC_SR register is set. In case the event is internal, the ITSF flag is also set in RTC_SR register.

By setting the TSIE bit in the RTC_CR register, an interrupt is generated when a timestamp event occurs.

If a new timestamp event is detected while the timestamp flag (TSF) is already set, the timestamp overflow flag (TSOVF) flag is set and the timestamp registers (RTC_TSTR and RTC_TSDR) maintain the results of the previous event.

Note: TSF is set up to 2 ck_apre cycles after the timestamp event from RTC_TS pin or from rtc_its internal signal occurs, due to synchronization process. TSF is set up to 3 ck_apre cycles after tamper flags.

TSOVF is set up to only 1 ck_apre cycle after the event occurs. This means that if two timestamp events are close together, TSOVF can be seen as '1' while TSF is still '0'. As a consequence, it is recommended to poll TSOVF only after TSF has been set.

Caution: If a timestamp event occurs immediately after the TSF bit is supposed to be cleared, then both TSF and TSOVF bits are set. To avoid masking a timestamp event occurring at the same moment, the application must not write 0 into TSF bit unless it has already read it to 1.

61.3.18 Calibration clock output

When the COE bit is set to 1 in the RTC_CR register, a reference clock is provided on the CALIB device output.

If the COSEL bit in the RTC_CR register is reset and PREDIV_A = 0x7F, the CALIB frequency is $f_{RTCCLK/64}$. This corresponds to a calibration output at 512 Hz for an RTCCLK frequency at 32.768 kHz. The CALIB duty cycle is irregular: there is a light jitter on falling edges. It is therefore recommended to use rising edges.

When COSEL is set and “PREDIV_S+1” is a non-zero multiple of 256 (i.e: PREDIV_S[7:0] = 0xFF), the CALIB frequency is $f_{RTCCLK}/(256 * (PREDIV_A+1))$. This corresponds to a calibration output at 1 Hz for prescaler default values (PREDIV_A = Ox7F, PREDIV_S = 0xFF), with an RTCCLK frequency at 32.768 kHz.

Note: When COSEL is cleared, the CALIB output is the output of the 6^{th} stage of the asynchronous prescaler. If LPCAL is changed from 0 to 1, the output can be irregular (glitch...) during the LPCAL switch. If LPCAL = 1 this output is always available. If LPCAL = 0, no output is present if PREDIV_A is < 0x20.

When COSEL is set, the CALIB output is the output of the 8^{th} stage of the synchronous prescaler.

61.3.19 Tamper and alarm output

The OSEL[1:0] control bits in the RTC_CR register are used to activate the alarm output TAMPALRM, and to select the function which is output. These functions reflect the contents of the corresponding flags in the RTC_SR register.

When the TAMPOE control bit is set in the RTC_CR, all external and internal tamper flags are ORed and routed to the TAMPALRM output. If OSEL = 00 the TAMPALRM output reflects only the tampers flags. If OSEL ≠ 00, the signal on TAMPALRM provides both tamper flags and alarm A, B, or wake-up flag.

The polarity of the TAMPALRM output is determined by the POL control bit in RTC_CR so that the opposite of the selected flags bit is output when POL is set to 1.

TAMPALRM output

The TAMPALRM pin can be configured in output open drain or output push-pull using the control bit TAMPALRM_TYPE in the RTC_CR register. It is possible to apply the internal pull-up in output mode thanks to TAMPALRM_PU in the RTC_CR.

Note: Once the TAMPALRM *output is enabled, it has priority over CALIB on RTC_OUT1.*

61.4 RTC low-power modes

Table 587. Effect of low-power modes on RTC

Mode	Description
Sleep	No effect RTC interrupts cause the device to exit the Sleep mode.
Stop	The RTC remains active when the RTC clock source is LSE or LSI. RTC interrupts cause the device to exit the Stop mode.
Standby	The RTC remains active when the RTC clock source is LSE or LSI. RTC interrupts cause the device to exit the Standby mode.

The table below summarizes the RTC pins and functions capability in all modes.

Table 588. RTC pins functionality over modes

Functions	Functional in all low-power modes except Standby mode	Functional in Standby mode	Functional in V_{BAT} mode
RTC_TS	Yes	Yes	Yes
RTC_REFIN	Yes	No	No
RTC_OUT1	Yes	Yes	Yes
RTC_OUT2	Yes	No	No

61.5 RTC interrupts

The interrupt channel is set in the masked interrupt status register or in the secure masked interrupt status register depending on its security mode configuration. The nonsecure interrupt output or the secure interrupt output is also activated.

Table 589. Nonsecure interrupt requests

Interrupt acronym	Interrupt event	Event flag(1)	Enable control bit(2)	Interrupt clear method	Exit from low-power modes
RTC	Alarm A	ALRAF	ALRAIE and (ALRASEC=0 and SEC=0)	write 1 in CALRAF	Yes(3)
	Alarm B	ALRBF	ALRBIE and (ALRBSEC=0 and SEC=0)	write 1 in CALRBF	Yes(3)
	Timestamp	TSF	TSIE and (TSSEC=0 and SEC=0)	write 1 in CTSF	Yes(3)
	Wake-up timer	WUTF	WUTIE and (WUTSEC=0 and SEC=0)	write 1 in CWUTF	Yes(3)
	SSR underflow (reload)	SSRUF	SSRUIE and (ALRASEC=0 and SEC=0)	write 1 in CSSRUF	Yes(3)

1. The event flags are in the RTC_SR register.
2. The interrupt masked flags (resulting from event flags AND enable control bits) are in the RTC_MISR register.
3. When the RTC is clocked by an oscillator functional in the low-power mode.

Table 590. Secure interrupt requests

Interrupt acronym	Interrupt event	Event flag(1)	Enable control bit(2)	Interrupt clear method	Exit from low-power modes
RTC_S	Alarm A	ALRAF	ALRAIE and (ALRASEC=1 or SEC=1)	write 1 in CALRAF	Yes(3)
	Alarm B	ALRBF	ALRBIE and (ALRBSEC=1 or SEC=1)	write 1 in CALRBF	Yes(3)
	Timestamp	TSF	TSIE and (TSSEC=1 or SEC=1)	write 1 in CTSF	Yes(3)
	Wake-up timer	WUTF	WUTIE and (WUTSEC=1 or SEC=1)	write 1 in CWUTF	Yes(3)
	SSR underflow (reload)	SSRUF	SSRUIE and (ALRASEC=1 or SEC=1)	write 1 in CSSRUF	Yes(3)

1. The event flags are in the RTC_SR register.
2. The interrupt masked flags (resulting from event flags AND enable control bits) are in the RTC_SMISR register.
3. When the RTC is clocked by an oscillator functional in the low-power mode.

61.6 RTC registers

Refer to *Section 1.2* of the reference manual for a list of abbreviations used in register descriptions.

The peripheral registers can be accessed by words (32-bit).

61.6.1 RTC time register (RTC_TR)

The RTC_TR is the calendar time shadow register. This register must be written in initialization mode only. Refer to *Calendar initialization and configuration on page 3017* and *Reading the calendar on page 3018*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 3016*.

This register can be write-protected against nonsecure access. Refer to *Section 61.3.4: RTC secure protection modes*.

This register can be write-protected against unprivileged access. Refer to *Section 61.3.5: RTC privilege protection modes*.

Address offset: 0x00

Backup domain reset value: 0x0000 0000

System reset value: 0x0000 0000 (when BYPSHAD = 0, not affected when BYPSHAD = 1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]			
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **PM**: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bits 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bits 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ST[2:0]**: Second tens in BCD format

Bits 3:0 **SU[3:0]**: Second units in BCD format

61.6.2 RTC date register (RTC_DR)

The RTC_DR is the calendar date shadow register. This register must be written in initialization mode only. Refer to *Calendar initialization and configuration on page 3017* and *Reading the calendar on page 3018*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 3016*.

This register can be write-protected against nonsecure access. Refer to *Section 61.3.4: RTC secure protection modes*.

This register can be write-protected against unprivileged access. Refer to *Section 61.3.5: RTC privilege protection modes*.

Address offset: 0x04

Backup domain reset value: 0x0000 2101

System reset value: 0x0000 2101 (when BYPSHAD = 0, not affected when BYPSHAD = 1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	YT[3:0]				YU[3:0]			
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT[1:0]		DU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:20 **YT[3:0]**: Year tens in BCD format

Bits 19:16 **YU[3:0]**: Year units in BCD format

Bits 15:13 **WDU[2:0]**: Week day units
000: forbidden
001: Monday
...
111: Sunday

Bit 12 **MT**: Month tens in BCD format

Bits 11:8 **MU[3:0]**: Month units in BCD format

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **DT[1:0]**: Date tens in BCD format

Bits 3:0 **DU[3:0]**: Date units in BCD format

Note: The calendar is frozen when reaching the maximum value, and can't roll over.

61.6.3 RTC subsecond register (RTC_SSR)

Address offset: 0x08

Backup domain reset value: 0x0000 0000

System reset value: 0x0000 0000 (when BYPSHAD = 0, not affected when BYPSHAD = 1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SS[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **SS[31:0]**: Synchronous binary counter

SS[31:16]: Synchronous binary counter MSB values

When Binary or Mixed mode is selected (BIN = 01 or 10 or 11):

SS[31:16] are the 16 MSB of the SS[31:0] free-running down-counter.

When BCD mode is selected (BIN=00):

SS[31:16] are forced by hardware to 0x0000.

SS[15:0]: Subsecond value/synchronous binary counter LSB values

When Binary mode is selected (BIN = 01 or 10 or 11):

SS[15:0] are the 16 LSB of the SS[31:0] free-running down-counter.

When BCD mode is selected (BIN=00):

SS[15:0] is the value in the synchronous prescaler counter. The fraction of a second is given by the formula below:

Second fraction = (PREDIV_S - SS) / (PREDIV_S + 1)

SS can be larger than PREDIV_S only after a shift operation. In that case, the correct time/date is one second less than as indicated by RTC_TR/RTC_DR.

61.6.4 RTC initialization control and status register (RTC_ICSR)

This register is write protected. The write access procedure is described in *RTC register write protection on page 3016*.

This register can be globally protected, or each bit of this register can be individually protected against nonsecure access. Refer to *Section 61.3.4: RTC secure protection modes*.

This register can be globally protected, or each bit of this register can be individually protected against unprivileged access. Refer to *Section 61.3.5: RTC privilege protection modes*.

Address offset: 0x0C

Backup domain reset value: 0x0000 0007

System reset: not affected except INIT, INITF, and RSF bits which are cleared to 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RECALPF
															r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	BCDU[2:0]			BIN[1:0]		INIT	INITF	RSF	INITS	SHPF	WUTWF	Res.	Res.
			rw	rw	rw	rw	rw	rw	r	rc_w0	r	r	r		

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **RECALPF**: Recalibration pending Flag

The RECALPF status flag is automatically set to 1 when software writes to the RTC_CALR register, indicating that the RTC_CALR register is blocked. When the new calibration settings are taken into account, this bit returns to 0. Refer to *Re-calibration on-the-fly*.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:10 **BCDU[2:0]**: BCD update (BIN = 10 or 11)

In mixed mode when both BCD calendar and binary extended counter are used (BIN = 10 or 11), the calendar second is incremented using the SSR Least Significant Bits.

0x0: 1s calendar increment is generated each time SS[7:0] = 0
0x1: 1s calendar increment is generated each time SS[8:0] = 0
0x2: 1s calendar increment is generated each time SS[9:0] = 0
0x3: 1s calendar increment is generated each time SS[10:0] = 0
0x4: 1s calendar increment is generated each time SS[11:0] = 0
0x5: 1s calendar increment is generated each time SS[12:0] = 0
0x6: 1s calendar increment is generated each time SS[13:0] = 0
0x7: 1s calendar increment is generated each time SS[14:0] = 0

Bits 9:8 **BIN[1:0]**: Binary mode

00: Free running BCD calendar mode (Binary mode disabled).
01: Free running Binary mode (BCD mode disabled)
10: Free running BCD calendar and Binary modes
11: Free running BCD calendar and Binary modes

Bit 7 **INIT**: Initialization mode

0: Free running mode

1: Initialization mode used to program time and date register (RTC_TR and RTC_DR), and prescaler register (RTC_PRER), plus BIN and BCDU fields. Counters are stopped and start counting from the new value when INIT is reset.

Bit 6 **INITF**: Initialization flag

When this bit is set to 1, the RTC is in initialization state, and the time, date and prescaler registers can be updated.

0: Calendar registers update is not allowed

1: Calendar registers update is allowed

Bit 5 **RSF**: Registers synchronization flag

This bit is set by hardware each time the calendar registers are copied into the shadow registers (RTC_SSR, RTC_TR and RTC_DR). This bit is cleared by hardware in initialization mode, while a shift operation is pending (SHPF = 1), or when in bypass shadow register mode (BYPSHAD = 1). This bit can also be cleared by software.

It is cleared either by software or by hardware in initialization mode.

0: Calendar shadow registers not yet synchronized

1: Calendar shadow registers synchronized

Note: RSF is not set when the frequency of the APB clock is less than twice the frequency of RTCCLK.

Bit 4 **INITS**: Initialization status flag

This bit is set by hardware when the calendar year field is different from 0 (Backup domain reset state).

0: Calendar has not been initialized

1: Calendar has been initialized

Bit 3 **SHPF**: Shift operation pending

This flag is set by hardware as soon as a shift operation is initiated by a write to the RTC_SHIFTR register. It is cleared by hardware when the corresponding shift operation has been executed. Writing to the SHPF bit has no effect.

0: No shift operation is pending

1: A shift operation is pending

Bit 2 **WUTWF**: Wake-up timer write flag

This bit is set by hardware when WUT value can be changed, after the WUTE bit has been set to 0 in RTC_CR.

0: Wake-up timer configuration update not allowed

1: Wake-up timer configuration update allowed

Bits 1:0 Reserved, must be kept at reset value.

61.6.5 RTC prescaler register (RTC_PRER)

This register must be written in initialization mode only. The initialization must be performed in two separate write accesses. Refer to *Calendar initialization and configuration on page 3017*.

This register is write protected. The write access procedure is described in *RTC register write protection on page 3016*.

This register can be write-protected against nonsecure access. Refer to *Section 61.3.4: RTC secure protection modes*.

This register can be write-protected against unprivileged access. Refer to *Section 61.3.5: RTC privilege protection modes*.

Address offset: 0x10

Backup domain reset value: 0x007F 00FF

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREDIV_A[6:0]						
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PREDIV_S[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **PREDIV_A[6:0]**: Asynchronous prescaler factor

This is the asynchronous division factor:
ck_apre frequency = RTCCLK frequency/(PREDIV_A+1)

Bit 15 Reserved, must be kept at reset value.

Bits 14:0 **PREDIV_S[14:0]**: Synchronous prescaler factor

This is the synchronous division factor:
ck_spre frequency = ck_apre frequency/(PREDIV_S+1)

61.6.6 RTC wake-up timer register (RTC_WUTR)

This register can be written only when WUTWF is set to 1 in RTC_ICSR.

This register can be protected against nonsecure access. Refer to *Section 61.3.4: RTC secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 61.3.5: RTC privilege protection modes*.

Address offset: 0x14

Backup domain reset value: 0x0000 FFFF

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WUTOCLR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WUT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **WUTOCLR[15:0]**: Wake-up auto-reload output clear value

When WUTOCLR[15:0] is different from 0x0000, WUTF is set by hardware when the auto-reload down-counter reaches 0 and is cleared by hardware when the auto-reload downcounter reaches WUTOCLR[15:0].

When WUTOCLR[15:0] = 0x0000, WUTF is set by hardware when the WUT down-counter reaches 0 and is cleared by software.

Bits 15:0 **WUT[15:0]**: Wake-up auto-reload value bits

When the wake-up timer is enabled (WUTE set to 1), the WUTF flag is set every (WUT[15:0] + 1) ck_wut cycles. The ck_wut period is selected through WUCKSEL[2:0] bits of the RTC_CR register.

When WUCKSEL[2] = 1, the wake-up timer becomes 17-bits and WUCKSEL[1] effectively becomes WUT[16] the most-significant bit to be reloaded into the timer.

The first assertion of WUTF occurs between WUT and (WUT + 2) ck_wut cycles after WUTE is set. Setting WUT[15:0] to 0x0000 with WUCKSEL[2:0] = 011 (RTCCLK/2) is forbidden.

61.6.7 RTC control register (RTC_CR)

This register is write protected. The write access procedure is described in RTC register write protection on page 3016.

This register can be globally protected, or each bit of this register can be individually protected against nonsecure access. Refer to *Section 61.3.4: RTC secure protection modes*.

This register can be globally protected, or each bit of this register can be individually protected against unprivileged access. Refer to *Section 61.3.5: RTC privilege protection modes*.

Address offset: 0x18

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	ALRBFCLR	ALRAFCLR	TAMPOE	TAMPTS	ITSE	COE	OSEL[1:0]		POL	COSEL	BKP	SUB1H	ADD1H
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSIE	WUTIE	ALRBIE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	SSRUIE	FMT	BYPSHAD	REFCKON	TSEDGE	WUCKSEL[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **OUT2EN**: RTC_OUT2 output enable

With this bit set, the RTC outputs can be remapped on RTC_OUT2 as follows:
OUT2EN = 0: RTC output 2 disable
If OSEL ≠ 00 or TAMPOE = 1: TAMPALRM is output on RTC_OUT1
If OSEL = 00 and TAMPOE = 0 and COE = 1: CALIB is output on RTC_OUT1
OUT2EN = 1: RTC output 2 enable
If (OSEL ≠ 00 or TAMPOE = 1) and COE = 0: TAMPALRM is output on RTC_OUT2
If OSEL = 00 and TAMPOE = 0 and COE = 1: CALIB is output on RTC_OUT2
If (OSEL ≠ 00 or TAMPOE = 1) and COE = 1: CALIB is output on RTC_OUT2 and TAMPALRM is output on RTC_OUT1.

Bit 30 **TAMPALRM_TYPE**: TAMPALRM output type

0: TAMPALRM is push-pull output
1: TAMPALRM is open-drain output

Bit 29 **TAMPALRM_PU**: TAMPALRM pull-up enable

0: No pull-up is applied on TAMPALRM output
1: A pull-up is applied on TAMPALRM output

Bit 28 **ALRBFCLR**: Alarm B flag automatic clear

0: Alarm B event generates a trigger event and ALRBF must be cleared by software to allow next alarm event.
1: Alarm B event generates a trigger event. ALRBF is automatically cleared by hardware after 1 ck_apre cycle.

Bit 27 **ALRAFCLR**: Alarm A flag automatic clear

0: Alarm A event generates a trigger event and ALRAF must be cleared by software to allow next alarm event.
1: Alarm A event generates a trigger event. ALRAF is automatically cleared by hardware after 1 ck_apre cycle.

Bit 26 **TAMPOE**: Tamper detection output enable on TAMPALRM
0: The tamper flag is not routed on TAMPALRM
1: The tamper flag is routed on TAMPALRM, combined with the signal provided by OSEL and with the polarity provided by POL.

Bit 25 **TAMPTS**: Activate timestamp on tamper detection event
0: Tamper detection event does not cause a RTC timestamp to be saved
1: Save RTC timestamp on tamper detection event
TAMPTS is valid even if TSE = 0 in the RTC_CR register. Timestamp flag is set up to 3 ck_apre cycles after the tamper flags.
Note: TAMPTS must be cleared before entering RTC initialization mode.

Bit 24 **ITSE**: timestamp on internal event enable
0: internal event timestamp disabled
1: internal event timestamp enabled

Bit 23 **COE**: Calibration output enable
This bit enables the CALIB output
0: Calibration output disabled
1: Calibration output enabled

Bits 22:21 **OSEL[1:0]**: Output selection
These bits are used to select the flag to be routed to TAMPALRM output.
00: Output disabled
01: Alarm A output enabled
10: Alarm B output enabled
11: Wake-up output enabled

Bit 20 **POL**: Output polarity
This bit is used to configure the polarity of TAMPALRM output.
0: The pin is high when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0]), or when a TAMPxF/ITAMPxF is asserted (if TAMPOE = 1).
1: The pin is low when ALRAF/ALRBF/WUTF is asserted (depending on OSEL[1:0]), or when a TAMPxF/ITAMPxF is asserted (if TAMPOE = 1).

Bit 19 **COSEL**: Calibration output selection
When COE = 1, this bit selects which signal is output on CALIB.
0: Calibration output is 512 Hz
1: Calibration output is 1 Hz
These frequencies are valid for RTCCLK at 32.768 kHz and prescalers at their default values (PREDIV_A = 127 and PREDIV_S = 255). Refer to *Section 61.3.18: Calibration clock output*.

Bit 18 **BKP**: Backup
This bit can be written by the user to memorize whether the daylight saving time change has been performed or not.

Bit 17 **SUB1H**: Subtract 1 hour (winter time change)
When this bit is set outside initialization mode, 1 hour is subtracted to the calendar time if the current hour is not 0. This bit is always read as 0.
Setting this bit has no effect when current hour is 0.
0: No effect
1: Subtracts 1 hour to the current time. This can be used for winter time change.

Bit 16 **ADD1H**: Add 1 hour (summer time change)
When this bit is set outside initialization mode, 1 hour is added to the calendar time. This bit is always read as 0.
0: No effect
1: Adds 1 hour to the current time. This can be used for summer time change
Note: ADD1H is ignored when SUB1H is set.

Bit 15 **TSIE**: Timestamp interrupt enable
0: Timestamp interrupt disable
1: Timestamp interrupt enable

Bit 14 **WUTIE**: Wake-up timer interrupt enable
0: Wake-up timer interrupt disabled
1: Wake-up timer interrupt enabled

Bit 13 **ALRBIE**: Alarm B interrupt enable
0: Alarm B interrupt disable
1: Alarm B interrupt enable

Bit 12 **ALRAIE**: Alarm A interrupt enable
0: Alarm A interrupt disabled
1: Alarm A interrupt enabled

Bit 11 **TSE**: timestamp enable
0: timestamp disable
1: timestamp enable

Bit 10 **WUTE**: Wake-up timer enable
0: Wake-up timer disabled
1: Wake-up timer enabled
Note: When the wake-up timer is disabled, wait for WUTWF = 1 before enabling it again.

Bit 9 **ALRBE**: Alarm B enable
0: Alarm B disabled
1: Alarm B enabled

Bit 8 **ALRAE**: Alarm A enable
0: Alarm A disabled
1: Alarm A enabled

Bit 7 **SSRUIE**: SSR underflow interrupt enable
0: SSR underflow interrupt disabled
1: SSR underflow interrupt enabled

Bit 6 **FMT**: Hour format
0: 24 hour/day format
1: AM/PM hour format

Bit 5 **BYPSHAD**: Bypass the shadow registers
0: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken from the shadow registers, which are updated once every two RTCCLK cycles.
1: Calendar values (when reading from RTC_SSR, RTC_TR, and RTC_DR) are taken directly from the calendar counters.
Note: If the frequency of the APB clock is less than seven times the frequency of RTCCLK, BYPSHAD must be set to 1.

Bit 4 **REFCKON**: RTC_REFIN reference clock detection enable (50 or 60 Hz)
0: RTC_REFIN detection disabled
1: RTC_REFIN detection enabled
Note: BIN must be 0x00 and PREDIV_S must be 0x00FF.

Bit 3 **TSEDGE**: Timestamp event active edge
0: RTC_TS input rising edge generates a timestamp event
1: RTC_TS input falling edge generates a timestamp event
TSE must be reset when TSEDGE is changed to avoid unwanted TSF setting.

Bits 2:0 **WUCKSEL[2:0]**: ck_wut wake-up clock selection
000: RTC/16 clock is selected
001: RTC/8 clock is selected
010: RTC/4 clock is selected
011: RTC/2 clock is selected
10x: ck_spre (usually 1 Hz) clock is selected in BCD mode. In binary or mixed mode, this is the clock selected by BCDU.
11x: ck_spre (usually 1 Hz) clock is selected in BCD mode. In binary or mixed mode, this is the clock selected by BCDU. Furthermore, 2^{16} is added to the WUT counter value.

Note: Bits 6 and 4 of this register can be written in initialization mode only (RTC_ICSR/INITF = 1).

WUT = wake-up unit counter value. WUT = (0x0000 to 0xFFFF) + 0x10000 added when WUCKSEL[2:1 = 11].

Bits 2 to 0 of this register can be written only when RTC_CR WUTE bit = 0 and RTC_ICSR WUTWF bit = 1.

It is recommended not to change the hour during the calendar hour increment as it may mask the incrementation of the calendar hour.

ADD1H and SUB1H changes are effective in the next second.

61.6.8 RTC privilege mode control register (RTC_PRIVCFGR)

This register can be written only when the APB access is privileged. This register can be write-protected, or each bit of this register can be individually write-protected against nonsecure access depending on the RTC_SECCFGR configuration (refer to *Section 61.3.5: RTC privilege protection modes*).

Address offset: 0x1C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRIV	INIT PRIV	CAL PRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS PRIV	WUT PRIV	ALRB PRIV	ALRA PRIV
rw	rw	rw										rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **PRIV**: RTC privilege protection

0: All RTC registers can be written when the APB access is privileged or unprivileged, except the registers protected by other privilege protection bits.

1: All RTC registers can be written only when the APB access is privileged.

Bit 14 **INITPRIV**: Initialization privilege protection

0: RTC Initialization mode, calendar and prescalers registers can be written when the APB access is privileged or unprivileged.

1: RTC Initialization mode, calendar and prescalers registers can be written only when the APB access is privileged.

Bit 13 **CALPRIV**: Shift register, daylight saving, calibration and reference clock privilege protection

0: Shift register, daylight saving, calibration and reference clock can be written when the APB access is privileged or unprivileged.

1: Shift register, daylight saving, calibration and reference clock can be written only when the APB access is privileged.

Bits 12:4 Reserved, must be kept at reset value.

Bit 3 **TSPRIV**: Timestamp privilege protection

0: RTC Timestamp configuration and interrupt clear can be written when the APB access is privileged or unprivileged.

1: RTC Timestamp configuration and interrupt clear can be written only when the APB access is privileged.

Bit 2 **WUTPRIV**: Wake-up timer privilege protection

0: RTC wake-up timer configuration and interrupt clear can be written when the APB access is privileged or unprivileged.

1: RTC wake-up timer configuration and interrupt clear can be written only when the APB access is privileged.

Bit 1 **ALRBPRIV**: Alarm B privilege protection

0: RTC Alarm B configuration and interrupt clear can be written when the APB access is privileged or unprivileged.

1: RTC Alarm B configuration and interrupt clear can be written only when the APB access is privileged.

Bit 0 **ALRAPRIV**: Alarm A and SSR underflow privilege protection

0: RTC Alarm A and SSR underflow configuration and interrupt clear can be written when the APB access is privileged or unprivileged.

1: RTC Alarm A and SSR underflow configuration and interrupt clear can be written only when the APB access is privileged.

Note: Refer to Section 61.3.5: RTC privilege protection modes for details on the read protection.

61.6.9 RTC secure configuration register (RTC_SECCFGR)

This register can be written only when the APB access is secure.

This register can be globally write-protected, or each bit of this register can be individually write-protected against unprivileged access depending on the RTC_PRIVCFGR configuration (refer to *Section 61.3.5: RTC privilege protection modes*).

Address offset: 0x20

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEC	INIT SEC	CAL SEC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TS SEC	WUT SEC	ALRB SEC	ALRA SEC
rw	rw	rw										rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **SEC**: RTC global protection

0: All RTC registers can be written when the APB access is secure or nonsecure, except the registers protected by other secure protection bits.

1: All RTC registers can be written only when the APB access is secure.

Bit 14 **INITSEC**: Initialization protection

0: RTC Initialization mode, calendar and prescalers registers can be written when the APB access is secure or nonsecure.

1: RTC Initialization mode, calendar and prescalers registers can be written only when the APB access is secure.

Bit 13 **CALSEC**: Shift register, daylight saving, calibration and reference clock protection

0: Shift register, daylight saving, calibration and reference clock can be written when the APB access is secure or nonsecure.

1: Shift register, daylight saving, calibration and reference clock can be written only when the APB access is secure.

Bits 12:4 Reserved, must be kept at reset value.

Bit 3 **TSSEC**: Timestamp protection

0: RTC timestamp configuration and interrupt clear can be written when the APB access is secure or nonsecure.

1: RTC timestamp configuration and interrupt clear can be written only when the APB access is secure.

Bit 2 **WUTSEC**: Wake-up timer protection

0: RTC wake-up timer configuration and interrupt clear can be written when the APB access is secure or nonsecure.

1: RTC wake-up timer configuration and interrupt clear can be written only when the APB access is secure.

Bit 1 **ALRBSEC**: Alarm B protection

0: RTC alarm B configuration and interrupt clear can be written when the APB access is secure or nonsecure.

1: RTC alarm B configuration and interrupt clear can be written only when the APB access is secure.

Bit 0 **ALRASEC**: Alarm A and SSR underflow protection

0: RTC alarm A and SSR underflow configuration and interrupt clear can be written when the APB access is secure or nonsecure.

1: RTC alarm A and SSR underflow configuration and interrupt clear can be written only when the APB access is secure.

Note: Refer to Section 61.3.4: RTC secure protection modes for details on the read protection.

61.6.10 RTC write protection register (RTC_WPR)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY[7:0]							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **KEY[7:0]**: Write protection key

This byte is written by software.

Reading this byte always returns 0x00.

Refer to *RTC register write protection* for a description of how to unlock RTC register write protection.

61.6.11 RTC calibration register (RTC_CALR)

This register is write protected. The write access procedure is described in *RTC register write protection on page 3016*.

This register can be write-protected against nonsecure access. Refer to *Section 61.3.4: RTC secure protection modes*.

This register can be write-protected against unprivileged access. Refer to *Section 61.3.5: RTC privilege protection modes*.

Address offset: 0x28

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CALP	CALW8	CALW 16	LPCAL	Res.	Res.	Res.	CALM[8:0]								
rw	rw	rw	rw				rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **CALP**: Increase frequency of RTC by 488.5 ppm
0: No RTCCLK pulses are added.
1: One RTCCLK pulse is effectively inserted every 2^{11} pulses (frequency increased by 488.5 ppm).
This feature is intended to be used in conjunction with CALM, which lowers the frequency of the calendar with a fine resolution. if the input frequency is 32768 Hz, the number of RTCCLK pulses added during a 32-second window is calculated as follows:
(512 × CALP) - CALM.
Refer to *Section 61.3.16: RTC smooth digital calibration*.

Bit 14 **CALW8**: Use an 8-second calibration cycle period
When CALW8 is set to 1, the 8-second calibration cycle period is selected.
Note: CALM[1:0] are stuck at 00 when CALW8 = 1. Refer to Section 61.3.16: RTC smooth digital calibration.

Bit 13 **CALW16**: Use a 16-second calibration cycle period
When CALW16 is set to 1, the 16-second calibration cycle period is selected. This bit must not be set to 1 if CALW8 = 1.
Note: CALM[0] is stuck at 0 when CALW16 = 1. Refer to Section 61.3.16: RTC smooth digital calibration.

Bit 12 **LPCAL**: RTC low-power mode

0: Calibration window is 2^{20} RTCCLK, which is a high-consumption mode. This mode must be set only when less than 32s calibration window is required.

1: Calibration window is 2^{20} ck_apre, which is the required configuration for ultra-low consumption mode.

Bits 11:9 Reserved, must be kept at reset value.

Bits 8:0 **CALM[8:0]**: Calibration minus

The frequency of the calendar is reduced by masking CALM out of 2^{20} RTCCLK pulses (32 seconds if the input frequency is 32768 Hz). This decreases the frequency of the calendar with a resolution of 0.9537 ppm.

To increase the frequency of the calendar, this feature should be used in conjunction with CALP. See *Section 61.3.16: RTC smooth digital calibration on page 3021*.

61.6.12 RTC shift control register (RTC_SHIFTR)

This register is write protected. The write access procedure is described in *RTC register write protection on page 3016*.

This register can be protected against nonsecure access. Refer to *Section 61.3.4: RTC secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 61.3.5: RTC privilege protection modes*.

Address offset: 0x2C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SUBFS[14:0]														
	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit 31 **ADD1S**: Add one second

0: No effect

1: Add one second to the clock/calendar

This bit is write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF = 1, in RTC_ICSR).

This function is intended to be used with SUBFS (see description below) in order to effectively add a fraction of a second to the clock in an atomic operation.

Bits 30:15 Reserved, must be kept at reset value.

Bits 14:0 **SUBFS[14:0]**: Subtract a fraction of a second

These bits are write only and is always read as zero. Writing to this bit has no effect when a shift operation is pending (when SHPF = 1, in RTC_ICSR).

The value which is written to SUBFS is added to the synchronous prescaler counter. Since this counter counts down, this operation effectively subtracts from (delays) the clock by:

Delay (seconds) = SUBFS / (PREDIV_S + 1)

A fraction of a second can effectively be added to the clock (advancing the clock) when the ADD1S function is used in conjunction with SUBFS, effectively advancing the clock by:

Advance (seconds) = (1 - (SUBFS / (PREDIV_S + 1))).

In mixed BCD-binary mode (BIN=10 or 11), the SUBFS[14:BCDU+8] must be written with 0.

Note: Writing to SUBFS causes RSF to be cleared. Software can then wait until RSF = 1 to be sure that the shadow registers have been updated with the shifted time.

61.6.13 RTC timestamp time register (RTC_TSTR)

The content of this register is valid only when TSF is set to 1 in RTC_SR. It is cleared when TSF bit is reset.

This register can be protected against nonsecure access. Refer to *Section 61.3.4: RTC secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 61.3.5: RTC privilege protection modes*.

Address offset: 0x30

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]			
									r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	r	r	r	r	r	r	r		r	r	r	r	r	r	r

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **PM**: AM/PM notation

0: AM or 24-hour format

1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format.

Bits 19:16 **HU[3:0]**: Hour units in BCD format.

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format.

Bits 11:8 **MNU[3:0]**: Minute units in BCD format.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **ST[2:0]**: Second tens in BCD format.

Bits 3:0 **SU[3:0]**: Second units in BCD format.

61.6.14 RTC timestamp date register (RTC_TSDR)

The content of this register is valid only when TSF is set to 1 in RTC_SR. It is cleared when TSF bit is reset.

This register can be protected against nonsecure access. Refer to *Section 61.3.4: RTC secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 61.3.5: RTC privilege protection modes*.

Address offset: 0x34

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT[1:0]		DU[3:0]			
r	r	r	r	r	r	r	r			r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:13 **WDU[2:0]**: Week day units

Bit 12 **MT**: Month tens in BCD format

Bits 11:8 **MU[3:0]**: Month units in BCD format

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **DT[1:0]**: Date tens in BCD format

Bits 3:0 **DU[3:0]**: Date units in BCD format

61.6.15 RTC timestamp subsecond register (RTC_TSSSR)

The content of this register is valid only when TSF is set to 1 in RTC_SR. It is cleared when the TSF bit is reset.

This register can be protected against nonsecure access. Refer to *Section 61.3.4: RTC secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 61.3.5: RTC privilege protection modes*.

Address offset: 0x38

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SS[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 SS[31:0]: Subsecond value/synchronous binary counter values

SS[31:0] is the value of the synchronous prescaler counter when the timestamp event occurred.

61.6.16 RTC alarm A register (RTC_ALRMAR)

This register can be written only when ALRAE is reset in RTC_CR register.

This register can be protected against nonsecure access. Refer to *Section 61.3.4: RTC secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 61.3.5: RTC privilege protection modes*.

Address offset: 0x40

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSK4	WDSEL	DT[1:0]		DU[3:0]				MSK3	PM	HT[1:0]		HU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **MSK4**: Alarm A date mask
0: Alarm A set if the date/day match
1: Date/day don’t care in alarm A comparison

Bit 30 **WDSEL**: Week day selection
0: DU[3:0] represents the date units
1: DU[3:0] represents the week day. DT[1:0] is don’t care.

Bits 29:28 **DT[1:0]**: Date tens in BCD format

Bits 27:24 **DU[3:0]**: Date units or day in BCD format

Bit 23 **MSK3**: Alarm A hours mask
0: Alarm A set if the hours match
1: Hours don’t care in alarm A comparison

Bit 22 **PM**: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bits 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 **MSK2**: Alarm A minutes mask
0: Alarm A set if the minutes match
1: Minutes don’t care in alarm A comparison

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bits 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 **MSK1**: Alarm A seconds mask
0: Alarm A set if the seconds match
1: Seconds don’t care in alarm A comparison

Bits 6:4 **ST[2:0]**: Second tens in BCD format.

Bits 3:0 **SU[3:0]**: Second units in BCD format.

61.6.17 RTC alarm A subsecond register (RTC_ALRMASSR)

This register can be written only when ALRAE is reset in RTC_CR register.

This register can be protected against nonsecure access. Refer to *Section 61.3.4: RTC secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 61.3.5: RTC privilege protection modes*.

Address offset: 0x44

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SSCLR	Res.	MASKSS[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw		rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SS[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw

Bit 31 **SSCLR**: Clear synchronous counter on alarm (Binary mode only)

0: The synchronous binary counter (SS[31:0] in RTC_SSR) is free-running.

1: The synchronous binary counter (SS[31:0] in RTC_SSR) is running from 0xFFFF FFFF to RTC_ALRABINR.SS[31:0] value and is automatically reloaded with 0xFFFF FFFF one ck_apre cycle after reaching RTC_ALRABINR.SS[31:0].

Note: SSCLR must be kept to 0 when BCD or mixed mode is used (BIN = 00, 10 or 11).

Bit 30 Reserved, must be kept at reset value.

Bits 29:24 **MASKSS[5:0]**: Mask the most-significant bits starting at this bit

0: No comparison on subseconds for Alarm A. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).

1: SS[31:1] are don't care in Alarm A comparison. Only SS[0] is compared.

2: SS[31:2] are don't care in Alarm A comparison. Only SS[1:0] are compared.

...

31: SS[31] is don't care in Alarm A comparison. Only SS[30:0] are compared.

From 32 to 63: All 32 SS bits are compared and must match to activate alarm.

Note: In BCD mode (BIN=00) the overflow bits of the synchronous counter (bits 31:15) are never compared. These bits can be different from 0 only after a shift operation.

Bits 23:15 Reserved, must be kept at reset value.

Bits 14:0 **SS[14:0]**: Subseconds value

This value is compared with the contents of the synchronous prescaler counter to determine if alarm A is to be activated. Only bits 0 up MASKSS-1 are compared.

This field is the mirror of SS[14:0] in the RTC_ALRABINR, and so can also be read or written through RTC_ALRABINR.

61.6.18 RTC alarm B register (RTC_ALRMBR)

This register can be written only when ALRBE is reset in RTC_CR register.

This register can be protected against nonsecure access. Refer to *Section 61.3.4: RTC secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 61.3.5: RTC privilege protection modes*.

Address offset: 0x48

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MSK4	WD SEL	DT[1:0]		DU[3:0]				MSK3	PM	HT[1:0]		HU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **MSK4**: Alarm B date mask
0: Alarm B set if the date and day match
1: Date and day don’t care in alarm B comparison

Bit 30 **WDSEL**: Week day selection
0: DU[3:0] represents the date units
1: DU[3:0] represents the week day. DT[1:0] is don’t care.

Bits 29:28 **DT[1:0]**: Date tens in BCD format

Bits 27:24 **DU[3:0]**: Date units or day in BCD format

Bit 23 **MSK3**: Alarm B hours mask
0: Alarm B set if the hours match
1: Hours don’t care in alarm B comparison

Bit 22 **PM**: AM/PM notation
0: AM or 24-hour format
1: PM

Bits 21:20 **HT[1:0]**: Hour tens in BCD format

Bits 19:16 **HU[3:0]**: Hour units in BCD format

Bit 15 **MSK2**: Alarm B minutes mask
0: Alarm B set if the minutes match
1: Minutes don’t care in alarm B comparison

Bits 14:12 **MNT[2:0]**: Minute tens in BCD format

Bits 11:8 **MNU[3:0]**: Minute units in BCD format

Bit 7 **MSK1**: Alarm B seconds mask
0: Alarm B set if the seconds match
1: Seconds don't care in alarm B comparison

Bits 6:4 **ST[2:0]**: Second tens in BCD format

Bits 3:0 **SU[3:0]**: Second units in BCD format

61.6.19 RTC alarm B subsecond register (RTC_ALRMBSSR)

This register can be written only when ALRBE is reset in RTC_CR register.

This register can be protected against nonsecure access. Refer to *Section 61.3.4: RTC secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 61.3.5: RTC privilege protection modes*.

Address offset: 0x4C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SSCLR	Res.	MASKSS[5:4]		MASKSS[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw		rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	SS[14:0]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw

Bit 31 **SSCLR**: Clear synchronous counter on alarm (Binary mode only)
0: The synchronous binary counter (SS[31:0] in RTC_SSR) is free-running.
1: The synchronous binary counter (SS[31:0] in RTC_SSR) is running from 0xFFFF FFFF to RTC_ALRBBINR.SS[31:0] value and is automatically reloaded with 0xFFFF FFFF one ck_apre cycle after reaching RTC_ALRBBINR.SS[31:0].
Note: SSCLR must be kept to 0 when BCD or mixed mode is used (BIN = 00, 10 or 11).

Bit 30 Reserved, must be kept at reset value.

Bits 29:24 **MASKSS[5:0]**: Mask the most-significant bits starting at this bit

0: No comparison on subseconds for Alarm B. The alarm is set when the seconds unit is incremented (assuming that the rest of the fields match).

1: SS[31:1] are don't care in Alarm B comparison. Only SS[0] is compared.

2: SS[31:2] are don't care in Alarm B comparison. Only SS[1:0] are compared.

...

31: SS[31] is don't care in Alarm B comparison. Only SS[30:0] are compared.

From 32 to 63: All 32 SS bits are compared and must match to activate alarm.

Note: In BCD mode (BIN=00)The overflow bits of the synchronous counter (bits 15) is never compared. This bit can be different from 0 only after a shift operation.

Bits 23:15 Reserved, must be kept at reset value.

Bits 14:0 **SS[14:0]**: Subseconds value

This value is compared with the contents of the synchronous prescaler counter to determine if alarm B is to be activated. Only bits 0 up to MASKSS-1 are compared.

This field is the mirror of SS[14:0] in the RTC_ALRBBINR, and so can also be read or written through RTC_ALRBBINR.

61.6.20 RTC status register (RTC_SR)

This register can be globally protected, or each bit of this register can be individually protected against nonsecure access. Refer to *Section 61.3.4: RTC secure protection modes*.

This register can be globally protected, or each bit of this register can be individually protected against unprivileged access. Refer to *Section 61.3.5: RTC privilege protection modes*.

Address offset: 0x50

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSRUF	ITSF	TSOVF	TSF	WUTF	ALRBF	ALRAF
									r	r	r	r	r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **SSRUF**: SSR underflow flag

This flag is set by hardware when the SSR is reloaded with 0xFFFF FFFF after reaching 0. SSRUF is not set when SSCLR = 1.

Note: SSRUF is not an error event as SSR counter is a free-running down-counter with automatic reload. SSRUF is set only in binary and mixed modes. It is not set in BCD mode.

Bit 5 **ITSF**: Internal timestamp flag

This flag is set by hardware when a timestamp on the internal event occurs.

Bit 4 **TSOVF**: Timestamp overflow flag

This flag is set by hardware when a timestamp event occurs while TSF is already set.

It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **TSF**: Timestamp flag

This flag is set by hardware when a timestamp event occurs.

If ITSF flag is set, TSF must be cleared together with ITSF.

Note: TSF is not set if TAMPTS = 1 and the tamper flag is read during the 3 ck_apre cycles following tamper event. Refer to Timestamp on tamper event for more details.

Bit 2 **WUTF**: Wake-up timer flag

This flag is set by hardware when the wake-up auto-reload counter reaches 0.

If WUTOCLR[15:0] is different from 0x0000, WUTF is cleared by hardware when the wake-up auto-reload counter reaches WUTOCLR value.

If WUTOCLR[15:0] is 0x0000, WUTF must be cleared by software.

This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

Bit 1 **ALRBF**: Alarm B flag

This flag is set by hardware when the time/date registers (RTC_TR and RTC_DR) match the alarm B register (RTC_ALRMBR).

Bit 0 **ALRAF**: Alarm A flag

This flag is set by hardware when the time/date registers (RTC_TR and RTC_DR) match the alarm A register (RTC_ALRMAR).

Note: The bits of this register are cleared few APB clock cycles after setting their corresponding clear bit in the RTC_SCR register. After clearing the flag, read it until it is read at 0 before leaving the interrupt routine.

61.6.21 RTC nonsecure masked interrupt status register (RTC_MISR)

This register can be globally protected, or each bit of this register can be individually protected against unprivileged access. Refer to *Section 61.3.5: RTC privilege protection modes*.

Address offset: 0x54

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSRUMF	ITSMF	TSOVMF	TSMF	WUTMF	ALRBMF	ALRAMF
									r	r	r	r	r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **SSRUMF**: SSR underflow nonsecure masked flag

This flag is set by hardware when the SSR underflow nonsecure interrupt occurs.

Bit 5 **ITSMF**: Internal timestamp nonsecure masked flag
This flag is set by hardware when a timestamp on the internal event occurs and timestamp nonsecure interrupt is raised.

Bit 4 **TSOVMF**: Timestamp overflow nonsecure masked flag
This flag is set by hardware when a timestamp interrupt occurs while TSMF is already set.
It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **TSMF**: Timestamp nonsecure masked flag
This flag is set by hardware when a timestamp nonsecure interrupt occurs.
If ITSF flag is set, TSF must be cleared together with ITSF.

Bit 2 **WUTMF**: Wake-up timer nonsecure masked flag
This flag is set by hardware when the wake-up timer nonsecure interrupt occurs.
This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

Bit 1 **ALRBMF**: Alarm B nonsecure masked flag
This flag is set by hardware when the alarm B nonsecure interrupt occurs.

Bit 0 **ALRAMF**: Alarm A masked flag
This flag is set by hardware when the alarm A nonsecure interrupt occurs.

Note: The bits of this register are cleared few APB clock cycles after setting their corresponding clear bit in the RTC_SCR register. After clearing the flag, read it until it is read at 0 before leaving the interrupt routine.

61.6.22 RTC secure masked interrupt status register (RTC_SMISR)

This register can be globally protected, or each bit of this register can be individually protected against nonsecure access. Refer to *Section 61.3.4: RTC secure protection modes*.

This register can be globally protected, or each bit of this register can be individually protected against unprivileged access. Refer to *Section 61.3.5: RTC privilege protection modes*.

Address offset: 0x58

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSRUMF	ITSMF	TSOVMF	TSMF	WUTMF	ALRBMF	ALRAMF
									r	r	r	r	r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **SSRUMF**: SSR underflow secure masked flag
This flag is set by hardware when the SSR underflow secure interrupt occurs.

Bit 5 **ITSMF**: Internal timestamp interrupt secure masked flag
This flag is set by hardware when a timestamp on the internal event occurs and timestamp secure interrupt is raised.

Bit 4 **TSOVMF**: Timestamp overflow interrupt secure masked flag
This flag is set by hardware when a timestamp secure interrupt occurs while TSMF is already set.
It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **TSMF**: Timestamp interrupt secure masked flag
This flag is set by hardware when a timestamp secure interrupt occurs.
If ITSF flag is set, TSF must be cleared together with ITSF.

Bit 2 **WUTMF**: Wake-up timer interrupt secure masked flag
This flag is set by hardware when the wake-up timer secure interrupt occurs.
This flag must be cleared by software at least 1.5 RTCCLK periods before WUTF is set to 1 again.

Bit 1 **ALRBMF**: Alarm B interrupt secure masked flag
This flag is set by hardware when the alarm B secure interrupt occurs.

Bit 0 **ALRAMF**: Alarm A interrupt secure masked flag
This flag is set by hardware when the alarm A secure interrupt occurs.

Note: The bits of this register are cleared few APB clock cycles after setting their corresponding clear bit in the RTC_SCR register. After clearing the flag, read it until it is read at 0 before leaving the interrupt routine.

61.6.23 RTC status clear register (RTC_SCR)

This register can be globally protected, or each bit of this register can be individually protected against nonsecure access. Refer to *Section 61.3.4: RTC secure protection modes*.

This register can be globally protected, or each bit of this register can be individually protected against unprivileged access. Refer to *Section 61.3.5: RTC privilege protection modes*.

Address offset: 0x5C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSSRUF	CITSF	CTSOVF	CTSF	CWUTF	CALRBF	CALRAF
									w	w	w	w	w	w	w

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CSSRUF**: Clear SSR underflow flag
Writing '1' in this bit clears the SSRUF in the RTC_SR register.

Bit 5 **CITSF**: Clear internal timestamp flag
Writing 1 in this bit clears the ITSF bit in the RTC_SR register.

Bit 4 **CTSOVF**: Clear timestamp overflow flag
Writing 1 in this bit clears the TSOVF bit in the RTC_SR register.
It is recommended to check and then clear TSOVF only after clearing the TSF bit. Otherwise, an overflow might not be noticed if a timestamp event occurs immediately before the TSF bit is cleared.

Bit 3 **CTSF**: Clear timestamp flag
Writing 1 in this bit clears the TSF bit in the RTC_SR register.
If ITSF flag is set, TSF must be cleared together with ITSF by setting CRSF and CITSF.

Bit 2 **CWUTF**: Clear wake-up timer flag
Writing 1 in this bit clears the WUTF bit in the RTC_SR register.

Bit 1 **CALRBF**: Clear alarm B flag
Writing 1 in this bit clears the ALRBF bit in the RTC_SR register.

Bit 0 **CALRAF**: Clear alarm A flag
Writing 1 in this bit clears the ALRAF bit in the RTC_SR register.

61.6.24 RTC alarm A binary mode register (RTC_ALRABINR)

This register can be written only when ALRAE is reset in RTC_CR register.

This register can be protected against nonsecure access. Refer to *Section 61.3.4: RTC secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 61.3.5: RTC privilege protection modes*.

Address offset: 0x70

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SS[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SS[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SS[31:0]**: Synchronous counter alarm value
This value is compared with the contents of the synchronous counter to determine if Alarm A is to be activated. Only bits 0 up MASKSS-1 are compared.
SS[14:0] is the mirror of SS[14:0] in the RTC_ALRMASSR, and so can also be read or written through RTC_ALRMASSR.

61.6.25 RTC alarm B binary mode register (RTC_ALRBBINR)

This register can be written only when ALRBE is reset in RTC_CR register.

This register can be protected against nonsecure access. Refer to *Section 61.3.4: RTC secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 61.3.5: RTC privilege protection modes*.

Address offset: 0x74

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SS[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SS[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SS[31:0]**: Synchronous counter alarm value

This value is compared with the contents of the synchronous counter to determine if Alarm Bis to be activated. Only bits 0 up MASKSS-1 are compared.
SS[14:0] is the mirror of SS[14:0] in the RTC_ALRMBSSR, and so can also be read or written through RTC_ALRMBSSR.

61.6.26 RTC register map

Table 591. RTC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	RTC_TR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT [1:0]		HU[3:0]				Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	Reset value										0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x04	RTC_DR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	YT[3:0]				YU[3:0]				WDU[2:0]			MT	MU[3:0]				Res.	Res.	DT [1:0]		DU[3:0]			
	Reset value									0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1			0	0	0	0	0	1
0x08	RTC_SSR	SS[31:16]																SS[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C	RTC_ICSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RECALPF	Res.	Res.	Res.	BCDU [2:0]			BIN [1:0]		INIT	INITF	RSF	INITS	SHPF	WUT WF	Res.	Res.
	Reset value																0				0	0	0	0	0	0	0	0	0	0	1		
0x10	RTC_PRER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREDIV_A[6:0]							PREDIV_S[14:0]															
	Reset value										1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0x14	RTC_WUTR	WUTOCLR[15:0]																WUT[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x18	RTC_CR	OUT2EN	TAMPALRM_TYPE	TAMPALRM_PU	ALRBFCLR	ALRAFCLR	TAMPOE	TAMPTS	ITSE	COE	O SEL [1:0]		POL	COSEL	BKP	SUB1H	ADD1H	TSIE	WUTIE	ALRBIE	ALRAIE	TSE	WUTE	ALRBE	ALRAE	SSRUIE	FMT	BYPSHAD	REFCKON	TSEDGE	WUCK SEL[2:0]		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1C	RTC_ PRIVCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIV	INITPRIV	CALPRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSPRIV	WUTPRIV	ALRBPRIV	ALRAPRIV
	Reset value																	0	0	0										0	0	0	0
0x20	RTC_ SECCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEC	INITSEC	CALSEC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSSEC	WUTSEC	ALRBSEC	ALRASEC
	Reset value																	0	0	0										0	0	0	0
0x24	RTC_WPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	KEY[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x28	RTC_ CALR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CALP	CALW8	CALW16	LPCAL	Res.	Res.	Res.	CALM[8:0]								
	Reset value																	0	0	0	0				0	0	0	0	0	0	0	0	0
0x2C	RTC_SHIFTR	ADD1S	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBFS[14:0]														
	Reset value	0																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 591. RTC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x30	RTC_TSTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PM	HT[1:0]		HU[3:0]				Res.	MNT[2:0]			MNU[3:0]				Res.	ST[2:0]			SU[3:0]			
	Reset value										0	0	0	0	0	0	0		0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x34	RTC_TSDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		WDU[2:0]		MT	MU[3:0]				Res.	Res.	DT [1:0]		DU[3:0]			
	Reset value																		0	0	0	0	0	0	0	0		0	0	0	0	0	0
0x38	RTC_TSSSR	SS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x40	RTC_ALRMAR	MSK4	WDSEL	DT [1:0]		DU[3:0]				MSK3	PM	HT [1:0]		HU[3:0]				MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	RTC_ ALRMASSR	SSCLR	Res.	MASKSS [5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[14:0]														
	Reset value	0		0	0	0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x48	RTC_ALRMBR	MSK4	WDSEL	DT [1:0]		DU[3:0]				MSK3	PM	HT [1:0]		HU[3:0]				MSK2	MNT[2:0]			MNU[3:0]				MSK1	ST[2:0]			SU[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4C	RTC_ ALRMBSSR	SSCLR	Res.	MASKSS [5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SS[14:0]														
	Reset value	0		0	0	0	0	0	0										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	RTC_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSRUF	ITSF	TSOVF	TSF	WUTF	ALRBF	ALRAF
	Reset value																										0	0	0	0	0	0	0
0x54	RTC_MISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSRUMF	ITSMF	TSOVMF	TSMF	WUTMF	ALRBMF	ALRAMF
	Reset value																										0	0	0	0	0	0	0
0x58	RTC_SMISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSRUMF	ITSMF	TSOVMF	TSMF	WUTMF	ALRBMF	ALRAMF
	Reset value																										0	0	0	0	0	0	0
0x5C	RTC_SCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSSRUF	CITSF	CTSOVF	CTSF	CWUTF	CALRBF	CALRAF
	Reset value																										0	0	0	0	0	0	0
0x70	RTC_ ALRABINR	SS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x74	RTC_ ALRBBINR	SS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

62 Tamper and backup registers (TAMP)

62.1 TAMP introduction

The anti-tamper detection circuit is used to protect sensitive data from external attacks. 32 32-bit backup registers are retained in all low-power modes and also in V_{BAT} mode. The backup registers, as well as other secrets in the device, are protected by this anti-tamper detection circuit with 7 tamper pins and 10 internal tampers. The external tamper pins can be configured for edge detection, or level detection with or without filtering, or active tamper which increases the security level by auto checking that the tamper pins are not externally opened or shorted.

62.2 TAMP main features

- A tamper detection can optionally erase the backup registers, backup SRAM, AHB SRAM2, cryptographic peripherals. The device resources protected by tamper are named “device secrets”.
- 32×32-bit backup registers:
  - The backup registers (TAMP_BKPxR) are implemented in the backup domain that remains powered-on by V_{BAT} when the V_{DD} power is switched off.
- Up to 7 tamper pins for 7 external tamper detection events:
  - Active tamper mode: continuous comparison between tamper output and input to protect from physical open-short attacks.
  - Flexible active tamper I/O management: from 3 meshes (each input associated to its own exclusive output) to 6 meshes (single output shared for up to 6 tamper inputs)
  - Passive tampers: ultra-low power edge or level detection with internal pull-up hardware management.
  - Configurable digital filter.
- 10 internal tamper events to protect against transient or environmental perturbation attacks
- Each tamper can be configured in two modes:
  - Confirmed mode: immediate erase of secrets on tamper detection, including backup registers erase
  - Potential mode: most of the secrets erase following a tamper detection are launched by software
- Any tamper detection can generate a RTC timestamp event.
- TrustZone support:
  - Tamper secure or nonsecure configuration.
  - Backup registers configuration in 3 configurable-size areas:

    1 read/write secure area.

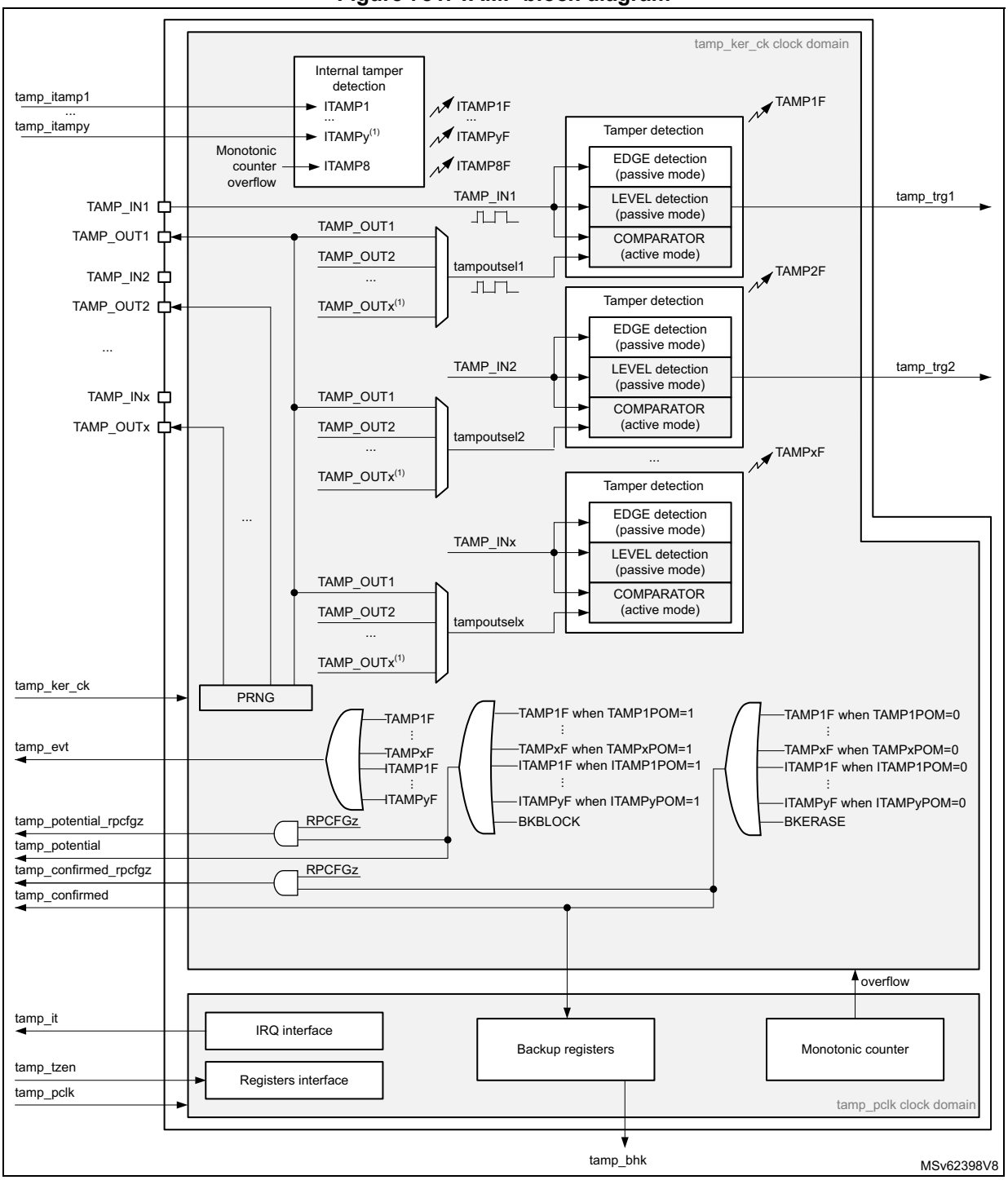
    1 write secure/read nonsecure area.

    1 read/write nonsecure area.
  - Boot hardware key for secure AES, stored in backup registers, protected against read and write access.
- Tamper configuration and backup registers privilege protection
- Monotonic counter.

62.3 TAMP functional description

62.3.1 TAMP block diagram

Figure 781. TAMP block diagram



1. The number of external and internal tampers depends on products.

62.3.2 TAMP pins and internal signals

Table 592. TAMP input/output pins

Pin name	Signal type	Description
TAMP_INx (x = pin index)	Input	Tamper input pin
TAMP_OUTx (x = pin index)	Output	Tamper output pin (active mode only)

Table 593. TAMP internal input/output signals

Internal signal name	Signal type	Description
tamp_ker_ck	Input	TAMP kernel clock, connected to rtc_ker_ck and also named RTCCLK in this document
tamp_pclk	Input	TAMP APB clock, connected to rtc_pclk
tamp_itamp[y] (y = signal index)	Inputs	Internal tamper event sources
tamp_tzen	Input	TAMP TrustZone enabled
tamp_evt	Output	Tamper event detection flag (internal or external tamper), whatever confirmed or potential mode configuration.
tamp_potential	Output	Potential tamper detection signal, used for device secrets(1) protection. This signal is active when: – a tamper event detection flag (internal or external tamper), is generated in potential mode. – or a software request is done by writing BKBLOCK to 1
tamp_confirmed	Output	Confirmed tamper detection signal, used for device secrets(1) protection. This signal is active when: – a tamper event detection flag (internal or external tamper), is generated in confirmed mode. – or a software request is done by writing BKERASE to 1
tamp_potential_rpcfgz (z = signal index)	Output	Potential tamper detection signal generated only when RPCFGz = 1. This signal is active when: – a tamper event detection flag (internal or external tamper), is generated in potential mode. – or a software request is done by writing BKBLOCK to 1

Table 593. TAMP internal input/output signals (continued)

Internal signal name	Signal type	Description
tamp_confirmed_rpcfgz (z = signal index)	Output	Confirmed tamper detection signal generated only when RPCFGz = 1. This signal is active when: – a tamper event detection flag (internal or external tamper), is generated in confirmed mode. – or a software request is done by writing BKERASE to 1
tamp_it	Output	TAMP interrupt (refer to *Section 62.5: TAMP interrupts* for details)
tamp_trg[x] (x = signal index)	Output	Tamper detection trigger
tamp_bhk	Output	Tamper boot hardware key bus

1. Refer to *Table 594: TAMP interconnection*.

The TAMP kernel clock is usually the LSE at 32.768 kHz although it is possible to select other clock sources in the RCC (refer to RCC for more details). The TAMP kernel clock is required only for external tamper inputs level with filtering, and for external active tamper detection modes. Internal tampers detection and external tampers inputs edge detection are functional without requiring any kernel clock.

Read and write access to backup registers and all other TAMP registers are also functional without any kernel clock (only APB clock is needed).

Some detections modes are not available in some low-power modes or V_{BAT} depending on the selected clock (refer to for more details.

Table 594. TAMP interconnection

Signal name	Source/Destination
tamp_tzen	Tied to 1
tamp_evt	rtc_tamp_evt used to generate a timestamp event
tamp_potential	The tamp_potential signal is used to block the read and write accesses to the backup registers. RHUK (root hardware unique key) in system Flash memory and BHK (boot hardware key) hardware buses to SAES are blocked. The tamp_potential signal is used to erase the device secrets listed hereafter: – SAES, AES - aka CRYP, HASH peripherals – PKA SRAM The device secrets access is blocked when erase is on-going.
tamp_confirmed	The tamp_confirmed signal is used to erase the backup registers and the AHB SRAM2. The device secrets access is blocked when erase is on-going. RHUK in system Flash memory (root hardware unique key) hardware bus to SAES is blocked.

Table 594. TAMP interconnection (continued)

Signal name	Source/Destination
tamp_potential_rpcfg0	When the bit RPCFG0 is set in the TAMP_RPCFGR, the tamp_potential_rpcfg0 signal is used to block the read and write accesses to the device secrets listed hereafter: – Backup SRAM
tamp_confirmed_rpcfg0	When the bit RPCFG0 is set in the TAMP_RPCFGR, the tamp_confirmed_rpcfg0 signal is used to erase the device secrets listed hereafter: – Backup SRAM The device secrets access is blocked when erase is on-going.
tamp_itamp1	Backup domain voltage threshold monitoring(1)
tamp_itamp2	Temperature monitoring(1)
tamp_itamp3	LSE monitoring (LSECSS)
tamp_itamp4	HSE monitoring (rcc_hsecss_fail)
tamp_itamp5	RTC calendar overflow (rtc_calovf)
tamp_itamp6	JTAG/SWD access
tamp_itamp7	V_{DDCORE} monitoring under/over voltage(2)
tamp_itamp8(3)	Monotonic counter 1 overflow
tamp_itamp9	Cryptographic peripherals fault (SAES or AES or PKA or TRNG)
tamp_itamp11	IWDG reset when tamper flag is set (potential tamper timeout)
tamp_bhk	saes_bhk. This bus is used to load the boot hardware key in the secure AES co-processor.

1. This monitoring must be enabled by setting MONEN in *PWR backup domain control register 1 (PWR_BDCR1)*.
2. This monitoring must be enabled by setting the bit 0 of TAMP_OR.
3. This signal is generated in the TAMP peripheral.

62.3.3 GPIOs controlled by the RTC and TAMP

Refer to *Section 61.3.3: GPIOs controlled by the RTC and TAMP*.

62.3.4 TAMP register write protection

After system reset, the TAMP registers (including backup registers) are protected against parasitic write access by the DBP bit in the power control peripheral (refer to the PWR power control section). DBP bit must be set in order to enable TAMP registers write access.

62.3.5 TAMP secure protection modes

By default after a backup domain power-on reset, all TAMP registers can be read or written in both secure and nonsecure modes, except for the TAMP secure configuration register

(TAMP_SECCFGR) which can be written in secure mode only. The TAMP protection configuration is not affected by a system reset.

- When the TAMPSEC bit is set in the TAMP_SECCFGR register:
  - Writing the TAMP registers is possible only in secure mode, except for the backup registers and monotonic counters which have their own protection setting.
  - Reading TAMP_SECCFGR, TAMP_PRIVCFGR and TAMP_MISR is always possible in secure and nonsecure modes. All the other TAMP registers can be read only in secure mode, except for the backup registers and monotonic counters which have their own protection setting.
- When the CNT1SEC bit is set in the TAMP_SECCFGR register: the TAMP_COUNT1R can be read and written only in secure mode, and CNT1PRIV can be written only in secure mode.
- When the BKPRWSEC is set in the TAMP_SECCFGR register: the BKPRWPRIV can be written only in secure mode.
- When the BKPWSEC is set in the TAMP_SECCFGR register: the BKPRWPRIV and BKPWPRIV can be written only in secure mode.

A nonsecure access to a secure-protected register is denied:

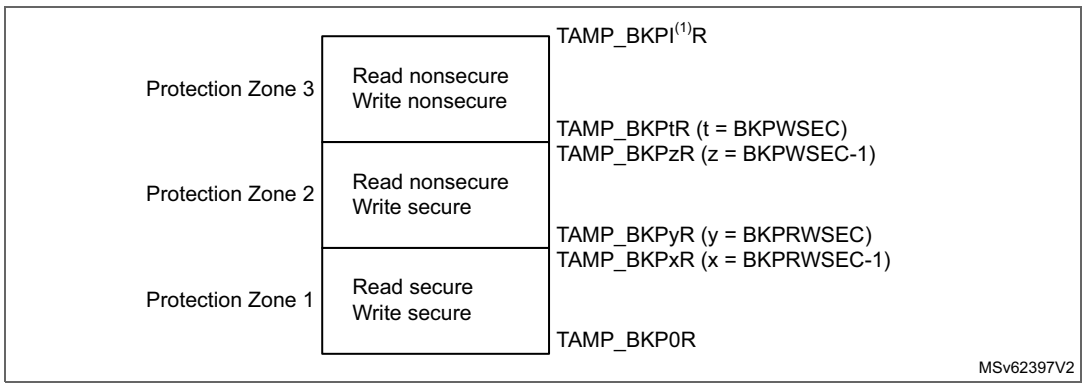
- There is no bus error generated.
- When write protected, the bits are not written.
- When read protected they are read as 0.

As soon as at least one function is configured to be secured, the TAMP reset and clock control is also secured in the RCC.

62.3.6 Backup registers protection zones

The backup registers protection is configured thanks to BKPRWSEC[7:0] and BKPWSEC[7:0] (refer to the figure below):

Figure 782. Backup registers protection zones



1. I = last backup register index

62.3.7 TAMP privilege protection modes

By default after a backup domain power-on reset, all TAMP registers can be read or written in both privileged and unprivileged modes, except for the TAMP privilege configuration

register (TAMP_PRIVCFGR) which can be written in privilege mode only. The TAMP protection configuration is not affected by a system reset.

- When the TAMPPRIV bit is set in the TAMP_PRIVCFGR register:
  - Writing the TAMP registers is possible only in privilege mode, except for the backup registers and the monotonic counter which have their own protection setting.
  - Reading TAMP_SECCFGR, TAMP_PRIVCFGR is always possible in privilege and unprivilege modes. All the other TAMP registers can be read only in privileged mode, except for the backup registers and the monotonic counter which have their own protection setting.
- When the CNT1PRIV bit is set in the TAMP_PRIVCFGR register: the TAMP_COUNT1R can be read and written only in privilege mode, and CNT1SEC can be written only in privilege mode.
- When the BKPRWPRIV is set in the TAMP_PRIVCFGR register: the BKPWSEC can be written only in privilege mode.
- When the BKPWPRIV is set in the TAMP_PRIVCFGR register: the BKPRWSEC and BKPWSEC can be written only in privilege mode.

The backup registers protection is configured thanks to BKPRWSEC[7:0] and BKPRWPRIV for the protection zone 1, and thanks to BKPRWSEC[7:0], BKPWSEC[7:0] and BKPWPRIV for the protection zone 2 (refer to *Figure 782*). The BHKLOCK bit can be written only in privileged mode when the BKPRWPRIV bit is set.

A unprivileged access to a privileged-protected register is denied:

- There is no bus error generated.
- When write protected, the bits are not written.
- When read protected they are read as 0.

62.3.8 Boot hardware key (BHK)

The eight first backup registers from TAMP_BKP0R to TAMP_BKP7R can be used to store a boot hardware key for the secure AES.

For this purpose, these registers must belong to the Protection Zone 1: BKPRWSEC must be greater or equal to 8.

Once the backup registers are written with the boot hardware key, the BHKLOCK bit must be set in the TAMP_SECCFGR register. Once BHKLOCK is set, the 8 backup registers cannot be accessed anymore by software: they are read as 0 and write to these registers is ignored. BHKLOCK cannot be cleared by software, and is cleared by hardware following a tamper event. It is also cleared with BKERASE command (in all cases the backup registers are also erased).

Refer to section secure AES co-processor (SAES) for details on procedure to download the boot hardware key in the SAES.

62.3.9 Tamper detection

The tamper detection main purpose is to protect the device secrets from device external attacks. The detection is made on events on TAMP_INx (x = pin index) I/Os, or on internal monitors detecting out-of-range device conditions.

The tamper detection can be configured for the following purposes:

- erase the backup registers and other device secrets stored in SRAMs or peripherals listed in *Table 594: TAMP interconnection*. The device secrets list is configurable thanks to *TAMP resources protection configuration register (TAMP_RPCFGR)*.
- block the read/write access to the backup registers and other device secrets stored in SRAMs or peripherals listed in *Table 594: TAMP interconnection*. The device secrets list is configurable thanks to TAMP_RPCFGR.
- generate an interrupt, capable to wake-up from low-power modes
- generate a hardware trigger for the low-power timers, or a RTC timestamp event

The external I/Os tamper detection supports 2 main configurations:

- Passive mode: TAMP_INx I/Os are monitored and a tamper is detected either on edge or on level.
- Active mode: TAMP_INx (x = pin index) is continuously compared with TAMP_OUTy (y = pin index) allowing open-short detection.

A digital filter can be applied on external tamper detection to avoid false detection. In addition, it is possible to configure each tamper source in potential mode, so that the secrets erase is not launched by hardware on tamper detection. The secrets erase can then be launched by software after software checks.

62.3.10 TAMP backup registers and other device secrets erase

The backup registers (TAMP_BKPxR) are not reset by system reset or when the device wakes up from Standby mode.

The backup registers and the other device secrets are not reset when the corresponding mask is set (TAMPxMSK=1 in the TAMP_CR2 register).

Tamper detection – confirmed mode

The confirmed mode is selected for TAMPx (external tamper x) when TAMPxPOM = 0 in the TAMP_CR2 register. The confirmed mode is selected for ITAMPx (internal tamper x) when ITAMPxPOM = 0 in the TAMP_CR3 register. The effects of a tamper detection in confirmed mode are described with tamp_confirmed and tamp_confirmed_rpcfgx signals in the *Table 594: TAMP interconnection*.

This mode is selected to erase automatically the device secrets when the tamper is detected.

Tamper detection – potential mode

The potential mode is selected for TAMPx (external tamper x) when TAMPxPOM = 1 in the TAMP_CR2 register. The potential tamper mode is selected for ITAMPx (internal tamper x) when ITAMPxPOM = 1 in the TAMP_CR3 register. The effects of a tamper detection in potential mode are described with tamp_potential and tamp_potential_rpcfgx signals in the *Table 594: TAMP interconnection*.

This mode is selected to avoid irreversible erasure of some device secrets when the tamper is detected. In this mode, some device secrets are not erased when the corresponding tamper event is detected. In addition, the read and write accesses to these device secrets are blocked as soon as the tamper detection flag is set in potential mode, until this flag is cleared by setting the corresponding clear flag in the TAMP_SCR register. Therefore the software can perform some checks to discriminate false from true tampers, and decide to

launch secrets erase only in case of the potential tamper is confirmed to be a true tamper. The device secrets are erased by software by setting the BKERASE bit in the TAMP_CR2 register.

Potential tamper to confirmed tamper timeout

Some internal tampers generate a tamper event if the independent watchdog reset occurs when another tamper flag is set (refer to *Table 594: TAMP interconnection*). The IWDG tamper must be configured with ITAMPxPOM = 0. This permits the erasure of device secrets to be forced by hardware after a timeout, in case the previous tamper event was in potential mode. This is equivalent to change the “potential tamper” into “confirmed tamper” if a watchdog reset occurs before any software decision following the potential tamper event.

Device resources protection configuration

Some device resources can be configured in order to be included to the list of the device secrets protected by tamper detection.

When RPCFGz = 0 in the TAMP_RPCFGR, the device resource associated to RPCFGz is not protected by the TAMP peripheral:

- It is not affected by tamper detection (whatever confirmed or potential mode)
- It is not affected by BKERASE software command
- It is not affected by BKBLOCK software command

When RPCFGz = 1 in the TAMP_RPCFGR, the device resource associated to RPCFGz is protected by the TAMP peripheral:

- It is affected by confirmed tamper detection and BKERASE software command, as described with tamp_confirmed_rpcfgz signal in *Table 594: TAMP interconnection*
- It is affected by potential tamper detection and BKBLOCK software command, as described with tamp_potential_rpcfgz signal in *Table 594: TAMP interconnection*

Table 595. Device resource x tamper protection

-	Potential tamper or BKBLOCK	Confirmed tamper or BKERASE
RPCFGx = 0	No effect on device resource x	No effect on device resource x
RPCFGx = 1	Device secret x protected as described by tamp_potential_rpcfgx(1)	Device secret x protected as described by tamp_confirmed_rpcfgx(1)

1. Refer to *Table 594: TAMP interconnection*.

Device secrets access blocked by software

By default, the device secrets can be accessed by the application, except if a tamper event flag is detected: the device secrets access is not possible as long as a tamper flag is set.

It is possible to block the access to the device secrets by software, by setting the BKBLOCK bit of the TAMP_CR2 register. The device secrets access is possible only when BKBLOCK = 0 and no tamper flag is set.

62.3.11 Tamper detection configuration and initialization

Each input can be enabled by setting the corresponding TAMPxE bits to 1 in the TAMP_CR register.

Each TAMP_INx tamper detection input is associated with a flag TAMPxF in the TAMP_SR register.

By setting the TAMPxIE bit in the TAMP_IER register, an interrupt is generated when a tamper detection event occurs (when TAMPxF is set). Setting TAMPxIE is not allowed when the corresponding TAMPxMSK is set.

Trigger output generation on tamper event

The tamper event detection can be used as trigger input by the low-power timers.

When TAMPxMSK bit is cleared in TAMP_CR register, the TAMPxF flag must be cleared by software in order to allow a new tamper detection on the same pin.

When TAMPxMSK bit is set, the TAMPxF flag is masked, and kept cleared in TAMP_SR register. This configuration permits the low-power timers to be triggered automatically in Stop mode, without requiring the system wake-up to perform the TAMPxF clearing. In this case, the backup registers are not cleared.

This feature is available only when the tamper is configured in level detection with filtering mode (TAMPFLT ≠ 00 and active mode is not selected). Refer to *Section : Level detection with filtering on tamper inputs (passive mode)*.

Timestamp on tamper event

With TAMPTS set to 1 in the RTC_CR, any internal or external tamper event causes a timestamp to occur. In case a timestamp occurs due to tamper event, either the TSF bit or the TSOVF bit is set in RTC_SR, in the same manner as if a normal timestamp event occurs.

Note: TSF is set up to 3 ck_apre cycles after TAMPxF flags. TSF is not set if RTCCLK is stopped (it is set when RTCCLK restarts).

Note: If TAMPxF is cleared before the expected rise of TSF, TSF is not set. Consequently, in case TAMPTS = 1, the software should either wait for timestamp flag before clearing the tamper flag, or should read the RTC counters values in the TAMP interrupt routine.

Edge detection on tamper inputs (passive mode)

If the TAMPFLT bits are 00, the TAMP_INx pins generate tamper detection events when either a rising edge or a falling edge is observed depending on the corresponding TAMPxTRG bit. The internal pull-up resistors on the TAMP_INx inputs are deactivated when edge detection is selected.

Caution: When TAMPFLT = 00 and TAMPxTRG = 0 (rising edge detection), a tamper event may be detected by hardware if the tamper input is already at high level before enabling the tamper detection.

After a tamper event has been detected and cleared, the TAMP_INx should be disabled and then re-enabled (TAMPxE set to 1) before re-programming the backup registers (TAMP_BKPxR). This prevents the application from writing to the backup registers while the TAMP_INx input value still indicates a tamper detection. This is equivalent to a level detection on the TAMP_INx input.

Note: Tamper detection is still active when V_{DD} power is switched off. To avoid unwanted resetting of the backup registers, the pin to which the TAMPx is mapped should be externally tied to the correct level.

Level detection with filtering on tamper inputs (passive mode)

Level detection with filtering is performed by setting TAMPFLT to a non-zero value. A tamper detection event is generated when either 2, 4, or 8 (depending on TAMPFLT) consecutive samples are observed at the level designated by the TAMPxTRG bits.

The TAMP_INx inputs are precharged through the I/O internal pull-up resistance before its state is sampled, unless disabled by setting TAMPPUDIS to 1. The duration of the precharge is determined by the TAMPPRCH bits, allowing for larger capacitances on the TAMP_INx inputs.

The trade-off between tamper detection latency and power consumption through the pull-up can be optimized by using TAMPFREQ to determine the frequency of the sampling for level detection. The TAMP_IN I/O schmitt trigger is enabled only during the precharge duration to avoid any extra consumption if the tamper switch is open (floating state).

Figure 783. Tamper sampling with precharge pulse

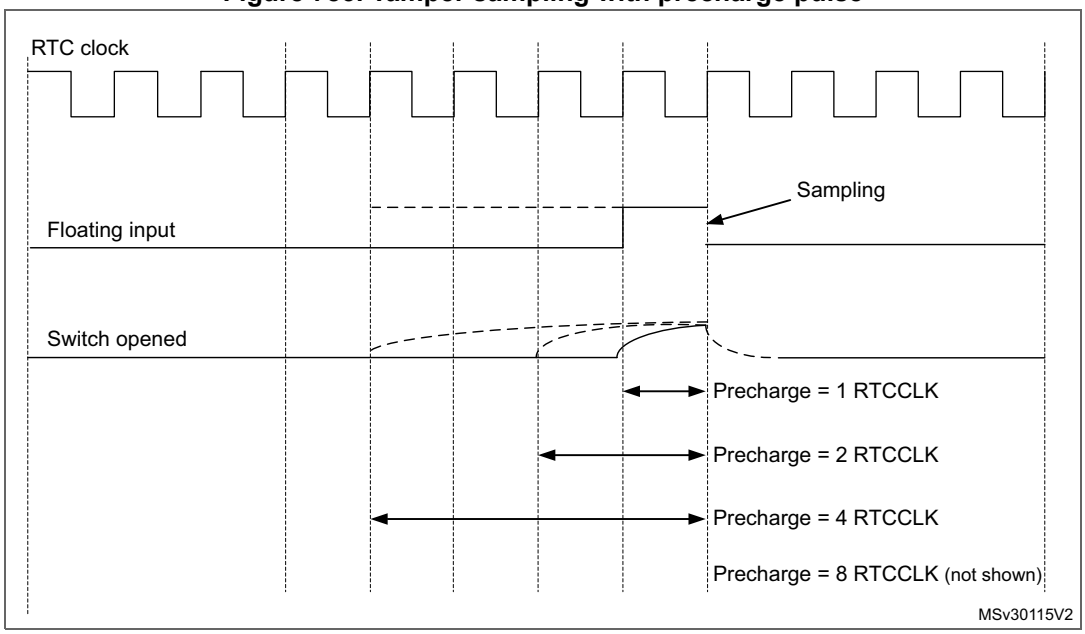
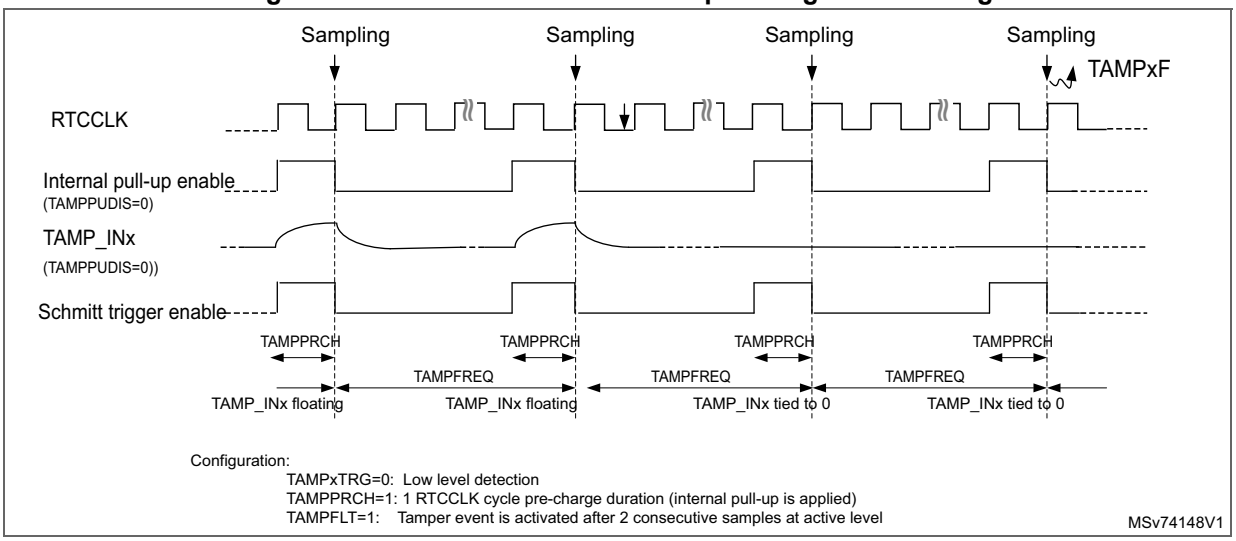


Figure 784. Low level detection with precharge and filtering



Note: Refer to the microcontroller datasheet for the electrical characteristics of the pull-up resistors.

Active tamper detection

When the TAMPxAM bit is set in the TAMP_ATCR, the tamper events are configured in active mode, which is based on a comparison between a TAMP_OUTy pin and a TAMP_INx pin. By default (ATOSHARE = 0) the comparison is made between TAMP_INx and TAMP_OUTx (y = x). When ATOSHARE bit is set, the same output can be used for several tamper inputs. The TAMP_OUTy function is enabled on the I/O as soon as it is selected for comparison with an active tamper input TAMP_INx (TAMPxEN = TAMPxAM = 1), thanks to ATOSHARE and ATOSELx bits. Refer to ATOSHARE and ATOSEL bits descriptions in the TAMP_ATCRx (x = 1, 2) registers.

Every two CK_ATPER cycles (CK_ATPER = 2^{ATPER} × CK_ATPRE), TAMP_OUTy output pin provides a value provided by a pseudo random number generator (PRNG). After outputting this value, the TAMP_OUTy pin outputs its opposite value one CK_ATPER cycle after.

Table 596. Active tamper output change period

ATCKSEL[3:0]	CK_ATPRE frequency	ATPER[2:0]	Tamper output change (CK_ATPER) frequency	Tamper output change period(1) (ms)
0x0	f_{RTCCLK}	0x0	f_{RTCCLK}	0.030
		0x1	$f_{RTCCLK}/2$	0.061
		0x2	$f_{RTCCLK}/4$	0.122
		0x3	$f_{RTCCLK}/8$	0.244
		0x4	$f_{RTCCLK}/16$	0.488
		0x5	$f_{RTCCLK}/32$	0.977
		0x6	$f_{RTCCLK}/64$	1.953
		0x7	$f_{RTCCLK}/128$	3.906
…	…	…	…	…
0x7	$f_{RTCCLK}/128$	0x0	$f_{RTCCLK}/128$	3.906
		0x1	$f_{RTCCLK}/256$	7.8125
		0x2	$f_{RTCCLK}/512$	15.625
		0x3	$f_{RTCCLK}/1024$	31.250
		0x4	$f_{RTCCLK}/2048$	62.5
		0x5	$f_{RTCCLK}/4096$	125
		0x6	$f_{RTCCLK}/8192$	250
		0x7	$f_{RTCCLK}/16384$	500
0x8	$f_{RTCCLK}/256$	0x0	$f_{RTCCLK}/256$	7.8125
		0x1	$f_{RTCCLK}/512$	15.625
		0x2	$f_{RTCCLK}/1024$	31.250
		0x3	$f_{RTCCLK}/2048$	62.5
		0x4	$f_{RTCCLK}/4096$	125
		0x5	$f_{RTCCLK}/8192$	250
		0x6	$f_{RTCCLK}/16384$	500
		0x7	$f_{RTCCLK}/32768$	1000
…	…	…	…	…

Table 596. Active tamper output change period (continued)

ATCKSEL[3:0]	CK_ATPRE frequency	ATPER[2:0]	Tamper output change (CK_ATPER) frequency	Tamper output change period(1) (ms)
0xF	f_{RTCCLK}/32768	0x0	f_{RTCCLK}/32768	1000
		0x1	f_{RTCCLK}/65536	2000
		0x2	f_{RTCCLK}/131072	4000
		0x3	f_{RTCCLK}/262144	8000
		0x4	f_{RTCCLK}/524288	16000
		0x5	f_{RTCCLK}/1048576	32000
		0x6	f_{RTCCLK}/2097152	64000
		0x7	f_{RTCCLK}/4194304	128000

1. Assuming f_{RTCCLK} = 32768 Hz.

PRNG is consumed by the selected tamper outputs at a different frequency depending on the number of selected tamper outputs. The number of selected outputs depends on TAMPxAM, TAMPxE, ATOSEL and ATOSHARE.

- When only 1 output is selected: PRNG is consumed every 16 CK_ATPER periods.
- When 2 outputs are selected: PRNG is consumed every 8 CK_ATPER periods.
- When 3 or 4 outputs are selected: PRNG is consumed every 4 CK_ATPER periods.
- When 5 or more outputs are selected: PRNG is consumed every 2 CK_ATPER periods

The PRNG needs minimum 9 CK_ATPRE cycles to output a new value. Consequently the minimum ATPER values for correct functionality are provided in the table below:

Table 597. Minimum ATPER value

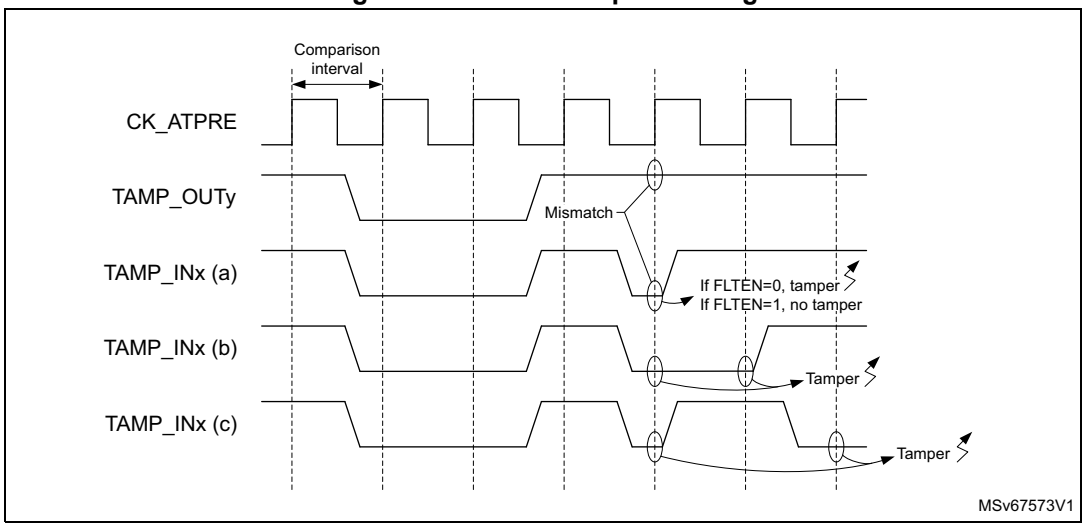
Number of selected outputs	Minimum ATPER
1	0
2	1
3 or 4	2
5 or more	3

The TAMP_INx pin is externally connected to TAMP_OUTy pin. The comparison is made between TAMP_OUTy output value and TAMP_INx received value, every CK_ATPRE cycle. In case a comparison mismatch occurs, the TAMPxF bit is set in the TAMP_SR register.

As an example, TAMP_OUT1 can be used for comparison with TAMP_IN1 and TAMP_IN2 by configuring and enabling both TAMP1 and TAMP2 in active mode, with ATOSHARE = 1, ATOSEL1 = 000 and ATOSEL2 = 000.

The active tamper can be combined with input filtering when FLTEN = 1. In this case, the tamper is detected only when 2 comparisons are false, in 4 consecutive comparison samples.

Figure 785. Active tamper filtering



As illustrated in *Figure 785*, if FLTEN = 0, any mismatch between the TAMP_OUTy output and the associated TAMP_INx input when the latter is sampled generates a tamper. This is the case in all three examples (a), (b) and (c).

If FLTEN = 1, example (a) does not generate a tamper, since only one mismatch is detected in four consecutive comparisons. In example (b), a tamper is generated since two successive mismatches are detected. Example (c) also generates a tamper, since two mismatches occur in four consecutive comparisons, even though the mismatches do not occur on successive samples.

Setting FLTEN = 1 avoids unwanted detection of tampers due to glitches, bounce or transitory states on the TAMP_INx inputs, by ignoring single pulses which are shorter than one period of CK_ATPRE, programmed in the ATCKSEL field of the TAMP_ATCR1 register. The minimum filtered pulse width is listed in *Table 598* for each possible setting of ATCKSEL, assuming f_{RTCCLK} = 32.768 kHz.

Table 598. Active tamper filtered pulse duration

ATCKSEL[3:0]	CK_ATPRE frequency	Minimum filtered pulse width (ms)
0x0	f_{RTCCLK}	0.030
0x1	$f_{RTCCLK}/2$	0.061
0x2	$f_{RTCCLK}/4$	0.122
0x3	$f_{RTCCLK}/8$	0.244
0x4	$f_{RTCCLK}/16$	0.488
0x5	$f_{RTCCLK}/32$	0.977
0x6	$f_{RTCCLK}/64$	1.953
0x7	$f_{RTCCLK}/128$	3.906
0x8	$f_{RTCCLK}/256$	7.812
0x9	$f_{RTCCLK}/512$	15.625
0xA	$f_{RTCCLK}/1024$	31.25
0xB	$f_{RTCCLK}/2048$	62.5

Table 598. Active tamper filtered pulse duration (continued)

ATCKSEL[3:0]	CK_ATPRE frequency	Minimum filtered pulse width (ms)
0xC	f_{RTCCLK}/4096	125
0xD	f_{RTCCLK}/8192	250
0xE	f_{RTCCLK}/16384	500
0xF	f_{RTCCLK}/32768	1000

Note: Multiple pulses which are shorter than one CK_ATPRE period may nevertheless cause a tamper if they result in two mismatches in four consecutive comparisons.

The pseudo-random generator must be initialized with a seed. This is done by writing consecutively four 32-bit random values in the TAMP_ATSEEDR register. Programming the seed automatically sends it to the PRNG. As long as the new seed is transferred and elaborated by the PRNG, the SEEDF bit is set in the TAMP_ATOR and it is not allowed to switch off the TAMP APB clock. The duration of the elaboration is up to 184 APB clock cycles after the forth seed is written. Consequently, after writing a new seed, the user must wait until SEEDF is cleared before entering low-power modes.

The active tamper outputs are activated only after the first seed is written and the elaboration is completed. Then new seeds can be written and elaborated during active tamper activity.

Active tamper initialization

Here is the software procedure to initialize the active tampers after system reset:

Read INITS in TAMP_ATOR register.

- If INITS = 0x0 (initialization was not done):
  a) Write TAMP_ATCR to configure Active tamper clock, filter and output sharing if any, and active mode.
  b) Write TAMP_CR1 to enable tampers (all the needed tampers must be enabled in the same write access).
  c) Write SEED by writing four times in the TAMP_ATSEEDR.
  d) Wait until SEEDF = 0 in TAMP_ATOR. Backup registers are then protected by active tamper.
- If INITS = 0x1 (initialization already done):

  No initialization. To increase randomness a new SEED should be provided regularly. When a new SEED is provided, wait until SEEDF = 0 before entering a low-power mode which switches off the TAMP APB clock.
- In case the tampers are disabled by software, and re-enabled afterwards, the SEED must be written after enabling tampers:
  a) Write TAMP_CR1 to enable tampers (all the needed tampers must be enabled in the same write access).
  b) Write SEED by writing four times in the TAMP_ATSEEDR.
  c) Wait until SEEDF = 0 in TAMP_ATOR. Backup registers are then protected by active tamper.

62.4 TAMP low-power modes

Table 599. Effect of low-power modes on TAMP

Mode	Description
Sleep	No effect. TAMP interrupts cause the device to exit the Sleep mode.
Stop	No effect on all features, except for level detection with filtering and active tamper modes which remain active only when the clock source is LSE or LSI. TAMP interrupts cause the device to exit the Stop mode.
Standby	No effect on all features, except for level detection with filtering and active tamper modes which remain active only when the clock source is LSE or LSI. TAMP interrupts cause the device to exit the Standby mode.

Table 600. TAMP pins functionality over modes

Pin name	Functional in all low-power modes	Functional in V_{BAT} mode
TAMP_INx (x = 1, 2)	Yes	Yes
TAMP_INx (x = 3, 4)	Yes(1)	Yes(1)
TAMP_INx (x = 5, 6, 7)	No	No
TAMP_OUTx (x = 1, 2)	Yes	Yes
TAMP_OUTx (x = 3, 4)	Yes	No
TAMP_OUTx (x = 5, 6, 8)	No	No

1. Through noPad IO to support implementation of active tamper.

62.5 TAMP interrupts

The interrupt channel is set in the masked interrupt status register or in the secure masked interrupt status register depending on its security mode configuration (TAMPSEC).

Table 601. Interrupt requests

Interrupt acronym	Interrupt event	Event flag(1)	Enable control bit	Interrupt clear method	Exit from low-power modes
TAMP	Tamper x(2)	TAMPxF	TAMPxIE	Write 1 in CTAMPxF	Yes(3)
	Internal tamper y(2)	ITAMPyF	ITAMPyIE	Write 1 in CITAMPyF	Yes(3)

1. The event flags are in the TAMP_SR register.
2. The number of tampers and internal tampers events depend on products.
3. Refer to *Table 599: Effect of low-power modes on TAMP* for more details about available features in the low-power modes.

62.6 TAMP registers

Refer to *Section 1.2* of the reference manual for a list of abbreviations used in register descriptions. The peripheral registers can be accessed by words (32-bit).

62.6.1 TAMP control register 1 (TAMP_CR1)

This register can be protected against nonsecure access. Refer to *Section 62.3.5: TAMP secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 62.3.7: TAMP privilege protection modes*.

Address offset: 0x00

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	ITAMP11E	Res.	ITAMP9E	ITAMP8E	ITAMP7E	ITAMP6E	ITAMP5E	ITAMP4E	ITAMP3E	ITAMP2E	ITAMP1E
					rw		rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP7E	TAMP6E	TAMP5E	TAMP4E	TAMP3E	TAMP2E	TAMP1E
									rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 Reserved, must be kept at reset value.

Bit 29 Reserved, must be kept at reset value.

Bit 28 Reserved, must be kept at reset value.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **ITAMP11E**: Internal tamper 11 enable
0: Internal tamper 11 disabled.
1: Internal tamper 11 enabled.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **ITAMP9E**: Internal tamper 9 enable
0: Internal tamper 9 disabled.
1: Internal tamper 9 enabled.

Bit 23 **ITAMP8E**: Internal tamper 8 enable
0: Internal tamper 8 disabled.
1: Internal tamper 8 enabled.

Bit 22 **ITAMP7E**: Internal tamper 7 enable
0: Internal tamper 7 disabled.
1: Internal tamper 7 enabled

Bit 21 **ITAMP6E**: Internal tamper 6 enable
0: Internal tamper 6 disabled.
1: Internal tamper 6 enabled.

Bit 20 **ITAMP5E**: Internal tamper 5 enable
0: Internal tamper 5 disabled.
1: Internal tamper 5 enabled.

Bit 19 **ITAMP4E**: Internal tamper 4 enable
0: Internal tamper 4 disabled.
1: Internal tamper 4 enabled.

Bit 18 **ITAMP3E**: Internal tamper 3 enable
0: Internal tamper 3 disabled.
1: Internal tamper 3 enabled.

Bit 17 **ITAMP2E**: Internal tamper 2 enable
0: Internal tamper 2 disabled.
1: Internal tamper 2 enabled.

Bit 16 **ITAMP1E**: Internal tamper 1 enable
0: Internal tamper 1 disabled.
1: Internal tamper 1 enabled.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TAMP7E**: Tamper detection on TAMP_IN7 enable(1)
0: Tamper detection on TAMP_IN7 is disabled.
1: Tamper detection on TAMP_IN7 is enabled.

Bit 5 **TAMP6E**: Tamper detection on TAMP_IN6 enable(1)
0: Tamper detection on TAMP_IN6 is disabled.
1: Tamper detection on TAMP_IN6 is enabled.

Bit 4 **TAMP5E**: Tamper detection on TAMP_IN5 enable(1)
0: Tamper detection on TAMP_IN5 is disabled.
1: Tamper detection on TAMP_IN5 is enabled.

Bit 3 **TAMP4E**: Tamper detection on TAMP_IN4 enable(1)
0: Tamper detection on TAMP_IN4 is disabled.
1: Tamper detection on TAMP_IN4 is enabled.

Bit 2 **TAMP3E**: Tamper detection on TAMP_IN3 enable(1)
0: Tamper detection on TAMP_IN3 is disabled.
1: Tamper detection on TAMP_IN3 is enabled.

Bit 1 **TAMP2E**: Tamper detection on TAMP_IN2 enable(1)
0: Tamper detection on TAMP_IN2 is disabled.
1: Tamper detection on TAMP_IN2 is enabled.

Bit 0 **TAMP1E**: Tamper detection on TAMP_IN1 enable(1)
0: Tamper detection on TAMP_IN1 is disabled.
1: Tamper detection on TAMP_IN1 is enabled.

1. Tamper detection mode (selected with TAMP_FLTCR, TAMP_ATCR1, TAMP_ATCR2 registers and TAMPxTRG bits in TAMP_CR2), must be configured before enabling the tamper detection.

62.6.2 TAMP control register 2 (TAMP_CR2)

This register can be protected against nonsecure access. Refer to *Section 62.3.5: TAMP*

secure protection modes.

This register can be protected against unprivileged access. Refer to *Section 62.3.7: TAMP privilege protection modes*.

Address offset: 0x04

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TAMP7 TRG	TAMP6 TRG	TAMP5 TRG	TAMP4 TRG	TAMP3 TRG	TAMP2 TRG	TAMP1 TRG	BK ERASE	BK BLOCK	Res.	Res.	Res.	TAMP3 MSK	TAMP2 MSK	TAMP1 MSK
	rw	rw	rw	rw	rw	rw	rw	w	rw				rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP7 POM	TAMP6 POM	TAMP5 POM	TAMP4 POM	TAMP3 POM	TAMP2 POM	TAMP1 POM
									rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **TAMP7TRG**: Active level for tamper 7 input (active mode disabled)
- 0: If TAMPFLT ≠ 00 tamper 7 input staying low triggers a tamper detection event.
  If TAMPFLT = 00 tamper 7 input rising edge triggers a tamper detection event.
- 1: If TAMPFLT ≠ 00 tamper 7 input staying high triggers a tamper detection event.
  If TAMPFLT = 00 tamper 7 input falling edge triggers a tamper detection event.

Bit 29 **TAMP6TRG**: Active level for tamper 6 input (active mode disabled)
- 0: If TAMPFLT ≠ 00 tamper 6 input staying low triggers a tamper detection event.
  If TAMPFLT = 00 tamper 6 input rising edge triggers a tamper detection event.
- 1: If TAMPFLT ≠ 00 tamper 6 input staying high triggers a tamper detection event.
  If TAMPFLT = 00 tamper 6 input falling edge triggers a tamper detection event.

Bit 28 **TAMP5TRG**: Active level for tamper 5 input (active mode disabled)
- 0: If TAMPFLT ≠ 00 tamper 5 input staying low triggers a tamper detection event.
  If TAMPFLT = 00 tamper 5 input rising edge triggers a tamper detection event.
- 1: If TAMPFLT ≠ 00 tamper 5 input staying high triggers a tamper detection event.
  If TAMPFLT = 00 tamper 5 input falling edge triggers a tamper detection event.

Bit 27 **TAMP4TRG**: Active level for tamper 4 input (active mode disabled)
- 0: If TAMPFLT ≠ 00 tamper 4 input staying low triggers a tamper detection event.
  If TAMPFLT = 00 tamper 4 input rising edge triggers a tamper detection event.
- 1: If TAMPFLT ≠ 00 tamper 4 input staying high triggers a tamper detection event.
  If TAMPFLT = 00 tamper 4 input falling edge triggers a tamper detection event.

Bit 26 **TAMP3TRG**: Active level for tamper 3 input
- 0: If TAMPFLT ≠ 00 tamper 3 input staying low triggers a tamper detection event.
  If TAMPFLT = 00 tamper 3 input rising edge triggers a tamper detection event.
- 1: If TAMPFLT ≠ 00 tamper 3 input staying high triggers a tamper detection event.
  If TAMPFLT = 00 tamper 3 input falling edge triggers a tamper detection event.

Bit 25 **TAMP2TRG**: Active level for tamper 2 input
- 0: If TAMPFLT ≠ 00 tamper 2 input staying low triggers a tamper detection event.
  If TAMPFLT = 00 tamper 2 input rising edge triggers a tamper detection event.
- 1: If TAMPFLT ≠ 00 tamper 2 input staying high triggers a tamper detection event.
  If TAMPFLT = 00 tamper 2 input falling edge triggers a tamper detection event.

Bit 24 **TAMP1TRG**: Active level for tamper 1 input
0: If TAMPFLT ≠ 00 tamper 1 input staying low triggers a tamper detection event.
If TAMPFLT = 00 tamper 1 input rising edge triggers a tamper detection event.
1: If TAMPFLT ≠ 00 tamper 1 input staying high triggers a tamper detection event.
If TAMPFLT = 00 tamper 1 input falling edge triggers a tamper detection event.

Bit 23 **BKERASE**: Backup registers and device secrets(1) erase
Writing '1' to this bit reset the backup registers and device secrets(1). Writing 0 has no effect. This bit is always read as 0.

Bit 22 **BKBLOCK**: Backup registers and device secrets(1) access blocked
0: backup registers and device secrets(1) can be accessed if no tamper flag is set
1: backup registers and device secrets(1) cannot be accessed

Bits 21:19 Reserved, must be kept at reset value.

Bit 18 **TAMP3MSK**: Tamper 3 mask
0: Tamper 3 event generates a trigger event and TAMP3F must be cleared by software to allow next tamper event detection.
1: Tamper 3 event generates a trigger event. TAMP3F is masked and internally cleared by hardware. The backup registers and device secrets(1) are not erased.
The tamper 3 interrupt must not be enabled when TAMP3MSK is set.

Bit 17 **TAMP2MSK**: Tamper 2 mask
0: Tamper 2 event generates a trigger event and TAMP2F must be cleared by software to allow next tamper event detection.
1: Tamper 2 event generates a trigger event. TAMP2F is masked and internally cleared by hardware. The backup registers and device secrets(1) are not erased.
The tamper 2 interrupt must not be enabled when TAMP2MSK is set.

Bit 16 **TAMP1MSK**: Tamper 1 mask
0: Tamper 1 event generates a trigger event and TAMP1F must be cleared by software to allow next tamper event detection.
1: Tamper 1 event generates a trigger event. TAMP1F is masked and internally cleared by hardware. The backup registers and device secrets(1) are not erased.
The tamper 1 interrupt must not be enabled when TAMP1MSK is set.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TAMP7POM**: Tamper 7 potential mode
0: Tamper 7 event detection is in confirmed mode(1).
1: Tamper 7 event detection is in potential mode(2).

Bit 5 **TAMP6POM**: Tamper 6 potential mode
0: Tamper 6 event detection is in confirmed mode(1).
1: Tamper 6 event detection is in potential mode(2).

Bit 4 **TAMP5POM**: Tamper 5 potential mode
0: Tamper 5 event detection is in confirmed mode(1).
1: Tamper 5 event detection is in potential mode(2).

Bit 3 **TAMP4POM**: Tamper 4 potential mode
0: Tamper 4 event detection is in confirmed mode(1).
1: Tamper 4 event detection is in potential mode(2).

Bit 2 **TAMP3POM**: Tamper 3 potential mode
0: Tamper 3 event detection is in confirmed mode(1).
1: Tamper 3 event detection is in potential mode(2).

Bit 1 **TAMP2POM**: Tamper 2 potential mode
0: Tamper 2 event detection is in confirmed mode(1).
1: Tamper 2 event detection is in potential mode(2).

Bit 0 **TAMP1POM**: Tamper 1 potential mode
0: Tamper 1 event detection is in confirmed mode(1).
1: Tamper 1 event detection is in potential mode(2).

1. The effects of tamper detection in confirmed mode is described with tamp_confirmed and tamp_confirmed_rpcfgx signals in *Table 594: TAMP interconnection*.
2. The effects of tamper detection in potential mode is described with tamp_potential and tamp_potential_rpcfgx signals in *Table 594: TAMP interconnection*.

62.6.3 TAMP control register 3 (TAMP_CR3)

This register can be protected against nonsecure access. Refer to *Section 62.3.5: TAMP secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 62.3.7: TAMP privilege protection modes*.

Address offset: 0x08

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	ITAMP 11POM	Res.	ITAMP9 POM	ITAMP8 POM	ITAMP7 POM	ITAMP6 POM	ITAMP5 POM	ITAMP4 POM	ITAMP3 POM	ITAMP2 POM	ITAMP1 POM
					rw		rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 Reserved, must be kept at reset value.

Bit 12 Reserved, must be kept at reset value.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **ITAMP11POM**: Internal tamper 11 potential mode
0: Internal tamper 11 event detection is in confirmed mode(1).
1: Internal tamper 11 event detection is in potential mode(2).

Bit 9 Reserved, must be kept at reset value.

Bit 8 **ITAMP9POM**: Internal tamper 9 potential mode
0: Internal tamper 9 event detection is in confirmed mode(1).
1: Internal tamper 9 event detection is in potential mode(2).

Bit 7 **ITAMP8POM**: Internal tamper 8 potential mode
0: Internal tamper 8 event detection is in confirmed mode(1).
1: Internal tamper 8 event detection is in potential mode(2).

Bit 6 **ITAMP7POM**: Internal tamper 7 potential mode
0: Internal tamper 7 event detection is in confirmed mode(1).
1: Internal tamper 7 event detection is in potential mode(2).

Bit 5 **ITAMP6POM**: Internal tamper 6 potential mode
0: Internal tamper 6 event detection is in confirmed mode(1).
1: Internal tamper 6 event detection is in potential mode(2).

Bit 4 **ITAMP5POM**: Internal tamper 5 potential mode
0: Internal tamper 5 event detection is in confirmed mode(1).
1: Internal tamper 5 event detection is in potential mode(2).

Bit 3 **ITAMP4POM**: Internal tamper 4 potential mode
0: Internal tamper 4 event detection is in confirmed mode(1).
1: Internal tamper 4 event detection is in potential mode(2).

Bit 2 **ITAMP3POM**: Internal tamper 3 potential mode
0: Internal tamper 3 event detection is in confirmed mode(1).
1: Internal tamper 3 event detection is in potential mode(2).

Bit 1 **ITAMP2POM**: Internal tamper 2 potential mode
0: Internal tamper 2 event detection is in confirmed mode(1).
1: Internal tamper 2 event detection is in potential mode(2).

Bit 0 **ITAMP1POM**: Internal tamper 1 potential mode
0: Internal tamper 1 event detection is in confirmed mode(1).
1: Internal tamper 1 event detection is in potential mode(2).

1. The effects of internal tamper detection in confirmed mode is described with tamp_confirmed and tamp_confirmed_rpcfgx signals in *Table 594: TAMP interconnection*
2. The effects of internal tamper detection in potential mode is described with tamp_potential and tamp_potential_rpcfgx signals in *Table 594: TAMP interconnection*.

62.6.4 TAMP filter control register (TAMP_FLTCR)

This register can be protected against nonsecure access. Refer to *Section 62.3.5: TAMP secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 62.3.7: TAMP*

privilege protection modes.

Address offset: 0x0C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP PUDIS	TAMPPRCH [1:0]		TAMPFLT [1:0]		TAMPFREQ [2:0]		
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **TAMPPUDIS**: TAMP_INx pull-up disable

This bit determines if each of the TAMPx pins are precharged before each sample.
0: Precharge TAMP_INx pins before sampling (enable internal pull-up)
1: Disable precharge of TAMP_INx pins.

Bits 6:5 **TAMPPRCH[1:0]**: TAMP_INx precharge duration

These bit determines the duration of time during which the pull-up/is activated before each sample. TAMPPRCH is valid for each of the TAMP_INx inputs.
0x0: 1 RTCCLK cycle
0x1: 2 RTCCLK cycles
0x2: 4 RTCCLK cycles
0x3: 8 RTCCLK cycles

Bits 4:3 **TAMPFLT[1:0]**: TAMP_INx filter count

These bits determines the number of consecutive samples at the specified level (TAMP*TRG) needed to activate a tamper event. TAMPFLT is valid for each of the TAMP_INx inputs.
0x0: Tamper event is activated on edge of TAMP_INx input transitions to the active level (no internal pull-up on TAMP_INx input).
0x1: Tamper event is activated after 2 consecutive samples at the active level.
0x2: Tamper event is activated after 4 consecutive samples at the active level.
0x3: Tamper event is activated after 8 consecutive samples at the active level.

Bits 2:0 **TAMPFREQ[2:0]**: Tamper sampling frequency

Determines the frequency at which each of the TAMP_INx inputs are sampled.
0x0: RTCCLK / 32768 (1 Hz when RTCCLK = 32768 Hz)
0x1: RTCCLK / 16384 (2 Hz when RTCCLK = 32768 Hz)
0x2: RTCCLK / 8192 (4 Hz when RTCCLK = 32768 Hz)
0x3: RTCCLK / 4096 (8 Hz when RTCCLK = 32768 Hz)
0x4: RTCCLK / 2048 (16 Hz when RTCCLK = 32768 Hz)
0x5: RTCCLK / 1024 (32 Hz when RTCCLK = 32768 Hz)
0x6: RTCCLK / 512 (64 Hz when RTCCLK = 32768 Hz)
0x7: RTCCLK / 256 (128 Hz when RTCCLK = 32768 Hz)

Note: This register concerns only the tamper inputs in passive mode.

62.6.5 TAMP active tamper control register 1 (TAMP_ATCR1)

This register can be protected against nonsecure access. Refer to *Section 62.3.5: TAMP secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 62.3.7: TAMP privilege protection modes*.

Address offset: 0x10

Backup domain reset value: 0x0007 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FLTEN	ATO SHARE	Res.	Res.	Res.	ATPER[2:0]			Res.	Res.	Res.	Res.	ATCKSEL[3:0]			
rw	rw				rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ATOSEL4[1:0]		ATOSEL3[1:0]		ATOSEL2[1:0]		ATOSEL1[1:0]		Res.	TAMP7 AM	TAMP6 AM	TAMP5 AM	TAMP4 AM	TAMP3 AM	TAMP2 AM	TAMP1 AM
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bit 31 **FLTEN**: Active tamper filter enable

0: Active tamper filtering disable

1: Active tamper filtering enable: a tamper event is detected when 2 comparison mismatches occur out of 4 consecutive samples.

Bit 30 **ATOSHARE**: Active tamper output sharing

0: Each active tamper input TAMP_INi is compared with its dedicated output TAMP_OUTi

1: Each active tamper input TAMP_INi is compared with TAMPOUTSELi defined by ATOSELi bits.

Bits 29:27 Reserved, must be kept at reset value.

Bits 26:24 **ATPER[2:0]**: Active tamper output change period

The tamper output is changed every CK_ATPER = (2^{ATPER} x CK_ATPRE) cycles. Refer to *Table 597: Minimum ATPER value*.

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **ATCKSEL[3:0]**: Active tamper RTC asynchronous prescaler clock selection

These bits selects the RTC asynchronous prescaler stage output. The selected clock is CK_ATPRE = RTCCLK/$2^{ATCKSEL}$.

Note: These bits can be written only when all active tampers are disabled. The write protection remains for up to 1.5 CK_ATPRE cycles after all the active tampers are disable.

Bits 15:14 **ATOSEL4[1:0]**: Active tamper shared output 4 selection

00: TAMPOUTSEL4 = TAMP_OUT1

01: TAMPOUTSEL4 = TAMP_OUT2

10: TAMPOUTSEL4 = TAMP_OUT3

11: TAMPOUTSEL4 = TAMP_OUT4

If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.

Bits 13:12 **ATOSEL3[1:0]**: Active tamper shared output 3 selection
00: TAMPOUTSEL3 = TAMP_OUT1
01: TAMPOUTSEL3 = TAMP_OUT2
10: TAMPOUTSEL3 = TAMP_OUT3
11: TAMPOUTSEL3 = TAMP_OUT4
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.

Bits 11:10 **ATOSEL2[1:0]**: Active tamper shared output 2 selection
00: TAMPOUTSEL2 = TAMP_OUT1
01: TAMPOUTSEL2 = TAMP_OUT2
10: TAMPOUTSEL2 = TAMP_OUT3
11: TAMPOUTSEL2 = TAMP_OUT4
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.

Bits 9:8 **ATOSEL1[1:0]**: Active tamper shared output 1 selection
00: TAMPOUTSEL1 = TAMP_OUT1
01: TAMPOUTSEL1 = TAMP_OUT2
10: TAMPOUTSEL1 = TAMP_OUT3
11: TAMPOUTSEL1 = TAMP_OUT4
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TAMP7AM**: Tamper 7 active mode
0: Tamper 7 detection mode is passive.
1: Tamper 7 detection mode is active.

Bit 5 **TAMP6AM**: Tamper 6 active mode
0: Tamper 6 detection mode is passive.
1: Tamper 6 detection mode is active.

Bit 4 **TAMP5AM**: Tamper 5 active mode
0: Tamper 5 detection mode is passive.
1: Tamper 5 detection mode is active.

Bit 3 **TAMP4AM**: Tamper 4 active mode
0: Tamper 4 detection mode is passive.
1: Tamper 4 detection mode is active.

Bit 2 **TAMP3AM**: Tamper 3 active mode
0: Tamper 3 detection mode is passive.
1: Tamper 3 detection mode is active.

Bit 1 **TAMP2AM**: Tamper 2 active mode
0: Tamper 2 detection mode is passive.
1: Tamper 2 detection mode is active.

Bit 0 **TAMP1AM**: Tamper 1 active mode
0: Tamper 1 detection mode is passive.
1: Tamper 1 detection mode is active.

Note: Changing the active tampers configuration in this register is not allowed when a TAMPxAM bit is set, unless the corresponding TAMPxE bits are all cleared in the TAMP_CR1 register.

All tampers configured in active mode must be enabled at the same time (by setting all related TAMPxE in the same TAMP_CR1 write).

All tampers configured in active mode must be disabled at the same time (by clearing all related TAMPxE in the same TAMP_CR1 write).

A minimum duration of 1 CK_ATPRE period must be waited for after disabling the active tampers and before re-enabling them.

62.6.6 TAMP active tamper seed register (TAMP_ATSEEDR)

This register can be protected against nonsecure access. Refer to *Section 62.3.5: TAMP secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 62.3.7: TAMP privilege protection modes*.

Address offset: 0x14

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEED[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEED[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **SEED[31:0]**: Pseudo-random generator seed value

This register must be written four times with 32-bit values to provide the 128-bit seed to the PRNG. Writing to this register automatically sends the seed value to the PRNG.

62.6.7 TAMP active tamper output register (TAMP_ATOR)

This register can be protected against nonsecure access. Refer to *Section 62.3.5: TAMP secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 62.3.7: TAMP privilege protection modes*.

Address offset: 0x18

Backup domain reset value: 0x0000 0000

System reset: not affected, except for SEEDF which is reset to 0.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INITS	SEEDF	Res.	Res.	Res.	Res.	Res.	Res.	PRNG[7:0]							
r	r							r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **INITS**: Active tamper initialization status

This flag is set by hardware when the PRNG has absorbed the first 128-bit seed, meaning that the enabled active tampers are functional. This flag is cleared when the active tampers are disabled.

Bit 14 **SEEDF**: Seed running flag

This flag is set by hardware when a new seed is written in the TAMP_ATSEEDR. It is cleared by hardware when the PRNG has absorbed this new seed, and by system reset. The TAMP APB clock must not be switched off as long as SEEDF is set.

Bits 13:8 Reserved, must be kept at reset value.

Bits 7:0 **PRNG[7:0]**: Pseudo-random generator value

This field provides the values of the PRNG output. Because of potential inconsistencies due to synchronization delays, PRNG must be read at least twice. The read value is correct if it is equal to previous read value.

This field can only be read when the APB is in secure mode.

62.6.8 TAMP active tamper control register 2 (TAMP_ATCR2)

This register can be protected against nonsecure access. Refer to *Section 62.3.5: TAMP secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 62.3.7: TAMP privilege protection modes*.

Address offset: 0x1C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	ATOSEL7[2:0]			ATOSEL6[2:0]			ATOSEL5[2:0]			ATOSEL4[2:0]			ATOSEL3[2]
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ATOSEL3[1:0]		ATOSEL2[2:0]			ATOSEL1[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw								

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:26 **ATOSEL7[2:0]**: Active tamper shared output 7 selection
000: TAMPOUTSEL7 = TAMP_OUT1
001: TAMPOUTSEL7 = TAMP_OUT2
010: TAMPOUTSEL7 = TAMP_OUT3
011: TAMPOUTSEL7 = TAMP_OUT4
100: TAMPOUTSEL7 = TAMP_OUT5
101: TAMPOUTSEL7 = TAMP_OUT6
110: TAMPOUTSEL7 = TAMP_OUT7
111: TAMPOUTSEL7 = TAMP_OUT8
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.

Bits 25:23 **ATOSEL6[2:0]**: Active tamper shared output 6 selection
000: TAMPOUTSEL6 = TAMP_OUT1
001: TAMPOUTSEL6 = TAMP_OUT2
010: TAMPOUTSEL6 = TAMP_OUT3
011: TAMPOUTSEL6 = TAMP_OUT4
100: TAMPOUTSEL6 = TAMP_OUT5
101: TAMPOUTSEL6 = TAMP_OUT6
110: TAMPOUTSEL6 = TAMP_OUT7
111: TAMPOUTSEL6 = TAMP_OUT8
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.

Bits 22:20 **ATOSEL5[2:0]**: Active tamper shared output 5 selection
000: TAMPOUTSEL5 = TAMP_OUT1
001: TAMPOUTSEL5 = TAMP_OUT2
010: TAMPOUTSEL5 = TAMP_OUT3
011: TAMPOUTSEL5 = TAMP_OUT4
100: TAMPOUTSEL5 = TAMP_OUT5
101: TAMPOUTSEL5 = TAMP_OUT6
110: TAMPOUTSEL5 = TAMP_OUT7
111: TAMPOUTSEL5 = TAMP_OUT8
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.

Bits 19:17 **ATOSEL4[2:0]**: Active tamper shared output 4 selection
000: TAMPOUTSEL4 = TAMP_OUT1
001: TAMPOUTSEL4 = TAMP_OUT2
010: TAMPOUTSEL4 = TAMP_OUT3
011: TAMPOUTSEL4 = TAMP_OUT4
100: TAMPOUTSEL4 = TAMP_OUT5
101: TAMPOUTSEL4 = TAMP_OUT6
110: TAMPOUTSEL4 = TAMP_OUT7
111: TAMPOUTSEL4 = TAMP_OUT8
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.
Bits 18:17 are the mirror of ATOSEL4[1:0] in the TAMP_ATCR1, and so can also be read or written through TAMP_ATCR1.

Bits 16:14 **ATOSEL3[2:0]**: Active tamper shared output 3 selection
000: TAMPOUTSEL3 = TAMP_OUT1
001: TAMPOUTSEL3 = TAMP_OUT2
010: TAMPOUTSEL3 = TAMP_OUT3
011: TAMPOUTSEL3 = TAMP_OUT4
100: TAMPOUTSEL3 = TAMP_OUT5
101: TAMPOUTSEL3 = TAMP_OUT6
110: TAMPOUTSEL3 = TAMP_OUT7
111: TAMPOUTSEL3 = TAMP_OUT8
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.
Bits 15:14 are the mirror of ATOSEL3[1:0] in the TAMP_ATCR1, and so can also be read or written through TAMP_ATCR1.

Bits 13:11 **ATOSEL2[2:0]**: Active tamper shared output 2 selection
000: TAMPOUTSEL2 = TAMP_OUT1
001: TAMPOUTSEL2 = TAMP_OUT2
010: TAMPOUTSEL2 = TAMP_OUT3
011: TAMPOUTSEL2 = TAMP_OUT4
100: TAMPOUTSEL2 = TAMP_OUT5
101: TAMPOUTSEL2 = TAMP_OUT6
110: TAMPOUTSEL2 = TAMP_OUT7
111: TAMPOUTSEL2 = TAMP_OUT8
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.
Bits 12:11 are the mirror of ATOSEL2[1:0] in the TAMP_ATCR1, and so can also be read or written through TAMP_ATCR1.

Bits 10:8 **ATOSEL1[2:0]**: Active tamper shared output 1 selection
000: TAMPOUTSEL1 = TAMP_OUT1
001: TAMPOUTSEL1 = TAMP_OUT2
010: TAMPOUTSEL1 = TAMP_OUT3
011: TAMPOUTSEL1 = TAMP_OUT4
100: TAMPOUTSEL1 = TAMP_OUT5
101: TAMPOUTSEL1 = TAMP_OUT6
110: TAMPOUTSEL1 = TAMP_OUT7
111: TAMPOUTSEL1 = TAMP_OUT8
If the TAMP_OUTx output is not available in the package pinout, the ouput selection value is reserved and must not be used.
Bits 9:8 are the mirror of ATOSEL1[1:0] in the TAMP_ATCR1, and so can also be read or written through TAMP_ATCR1.

Bits 7:0 Reserved, must be kept at reset value.

Note: Changing the active tampers configuration in this register is not allowed when a TAMPxAM bit is set, unless the corresponding TAMPxE bits are all cleared in the TAMP_CR1 register.

All tampers configured in active mode must be enabled at the same time (by setting all related TAMPxE in the same TAMP_CR1 write).

All tampers configured in active mode must be disabled at the same time (by clearing all related TAMPxE in the same TAMP_CR1 write).

A minimum duration of 1 CK_ATPRE period must be waited for after disabling the active tampers and before re-enabling them.

62.6.9 TAMP secure configuration register (TAMP_SECCFGR)

If TZEN = 1, this register can be written only when the APB access is secure. If TZEN = 0, BKPRWSEC[7:0], BKPWSEC[7:0] and BHKLOCK can be written with nonsecure APB access, and TAMPSEC, CNT1SEC cannot be written.

This register can be globally write-protected, or each bit of this register can be individually write-protected against unprivileged access depending on the TAMP_PRIVCFGR configuration (refer to *Section 62.3.7: TAMP privilege protection modes*).

Address offset: 0x20

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TAMPSEC	BHKLOCK	Res.	Res.	Res.	Res.	Res.	Res.	BKPWSEC[7:0]							
rw	rs							rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT1SEC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPRWSEC[7:0]							
rw								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TAMPSEC**: Tamper protection (excluding monotonic counters and backup registers)

0: Tamper configuration and interrupt can be written when the APB access is secure or nonsecure.

1: Tamper configuration and interrupt can be written only when the APB access is secure.

Note: Refer to Section 62.3.5: TAMP secure protection modes for details on the read protection.

Bit 30 **BHKLOCK**: Boot hardware key lock

This bit can be read and can only be written to 1 by software. It is cleared by hardware together with the backup registers following a tamper detection event.

0: The Backup registers from TAMP_BKP0R to TAMP_BKP7R can be accessed according to the Protection zone they belong to.

1: The backup registers from TAMP_BKP0R to TAMP_BKP7R cannot be accessed neither in read nor in write (they are read as 0 and write ignore).

Bits 29:24 Reserved, must be kept at reset value.

Bits 23:16 **BKPWSEC[7:0]**: Backup registers write protection offset
BKPWSEC value must be from 0 to 32.

Protection zone 2 is defined for backup registers from TAMP_BKPyR (y = BKPRWSEC) to TAMP_BKPzR (z = BKPWSEC-1, with BKPWSEC > BKPRWSEC):
– if TZEN=1, these backup registers can be written only with secure access.
– They can be read with secure or nonsecure access.
If BKPWSEC = 0 or if BKPWSEC ≤ BKPRWSEC: there is no protection zone 2.

Protection zone 3 is defined for backup registers from TAMP_BKPtR (t = BKPWSEC if BKPWSEC ≥ BKPRWSEC, else t = BKPRWSEC).
– They can be read or written with secure or nonsecure access.
If BKPWSEC = 32: there is no protection zone 3.

Refer to *Figure 782: Backup registers protection zones*.

Note: If TZEN=0: the protection zone 2 can be read and written with nonsecure access.
Note: If BKPWPRIV is set, BKPWSEC[7:0] can be written only in privileged mode.

Bit 15 **CNT1SEC**: Monotonic counter 1 secure protection
0: Monotonic counter 1 (TAMP_COUNT1R) can be read and written when the APB access is secure or nonsecure.
1: Monotonic counter 1 (TAMP_COUNT1R) can be read and written only when the APB access is secure.

Bits 14:8 Reserved, must be kept at reset value.

Bits 7:0 **BKPRWSEC[7:0]**: Backup registers read/write protection offset
BKPRWSEC value must be from 0 to 32.

Protection zone 1 is defined for backup registers from TAMP_BKP0R to TAMP_BKPxR (x = BKPRWSEC-1, with BKPRWSEC ≥ 1).
– if TZEN=1, these backup registers can be read and written only with secure access.
If BKPRWSEC = 0: there is no protection zone 1.

Refer to *Figure 782: Backup registers protection zones*.

Note: If TZEN=0: the protection zone 1 can be read and written with nonsecure access.
Note: If BKPRWPRIV or BKPWPRIV is set, BKPRWSEC[7:0] can be written only in privileged mode.

62.6.10 TAMP privilege configuration register (TAMP_PRIVCFGR)

This register can be written only when the APB access is privileged.

When TZEN = 1, this register can be write-protected, or each bit of this register can be individually write-protected against nonsecure access depending on the TAMP_SECCFGR configuration (refer to *Section 62.3.5: TAMP secure protection modes*).

Address offset: 0x24

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TAMP PRIV	BKP WPRIV	BKPR WPRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNT1 PRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															

Bit 31 **TAMPPRIV**: Tamper privilege protection (excluding monotonic counters and backup registers)

0: Tamper configuration and interrupt can be written with privileged or unprivileged access.

1: Tamper configuration and interrupt can be written only with privileged access.

Note: Refer to Section 62.3.7: TAMP privilege protection modes for details on the read protection.

Bit 30 **BKPWPRIV**: Backup registers zone 2 privilege protection

0: Backup registers zone 2 can be written with privileged or unprivileged access.

1: Backup registers zone 2 can be written only with privileged access.

Bit 29 **BKPRWPRIV**: Backup registers zone 1 privilege protection

0: Backup registers zone 1 can be read and written with privileged or unprivileged access.

1: Backup registers zone 1 can be read and written only with privileged access

Bits 28:16 Reserved, must be kept at reset value.

Bit 15 **CNT1PRIV**: Monotonic counter 1 privilege protection

0: Monotonic counter 1 (TAMP_COUNT1R) can be read and written when the APB access is privileged or unprivileged.

1: Monotonic counter 1 (TAMP_COUNT1R) can be read and written only when the APB access is privileged.

Bits 14:0 Reserved, must be kept at reset value.

62.6.11 TAMP interrupt enable register (TAMP_IER)

This register can be protected against nonsecure access. Refer to *Section 62.3.5: TAMP secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 62.3.7: TAMP*

privilege protection modes.

Address offset: 0x2C

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	ITAMP11IE	Res.	ITAMP9IE	ITAMP8IE	ITAMP7IE	ITAMP6IE	ITAMP5IE	ITAMP4IE	ITAMP3IE	ITAMP2IE	ITAMP1IE
					rw		rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP7IE	TAMP6IE	TAMP5IE	TAMP4IE	TAMP3IE	TAMP2IE	TAMP1IE
									rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 Reserved, must be kept at reset value.

Bit 29 Reserved, must be kept at reset value.

Bit 28 Reserved, must be kept at reset value.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **ITAMP11IE**: Internal tamper 11 interrupt enable
0: Internal tamper 11 interrupt disabled.
1: Internal tamper 11 interrupt enabled.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **ITAMP9IE**: Internal tamper 9 interrupt enable
0: Internal tamper 9 interrupt disabled.
1: Internal tamper 9 interrupt enabled.

Bit 23 **ITAMP8IE**: Internal tamper 8 interrupt enable
0: Internal tamper 8 interrupt disabled.
1: Internal tamper 8 interrupt enabled.

Bit 22 **ITAMP7IE**: Internal tamper 7 interrupt enable
0: Internal tamper 7 interrupt disabled.
1: Internal tamper 7 interrupt enabled.

Bit 21 **ITAMP6IE**: Internal tamper 6 interrupt enable
0: Internal tamper 6 interrupt disabled.
1: Internal tamper 6 interrupt enabled.

Bit 20 **ITAMP5IE**: Internal tamper 5 interrupt enable
0: Internal tamper 5 interrupt disabled.
1: Internal tamper 5 interrupt enabled.

Bit 19 **ITAMP4IE**: Internal tamper 4 interrupt enable
0: Internal tamper 4 interrupt disabled.
1: Internal tamper 4 interrupt enabled.

Bit 18 **ITAMP3IE**: Internal tamper 3 interrupt enable
0: Internal tamper 3 interrupt disabled.
1: Internal tamper 3 interrupt enabled.

Bit 17 **ITAMP2IE**: Internal tamper 2 interrupt enable
0: Internal tamper 2 interrupt disabled.
1: Internal tamper 2 interrupt enabled.

Bit 16 **ITAMP1IE**: Internal tamper 1 interrupt enable
0: Internal tamper 1 interrupt disabled.
1: Internal tamper 1 interrupt enabled

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TAMP7IE**: Tamper 7interrupt enable
0: Tamper 7 interrupt disabled.
1: Tamper 7interrupt enabled.

Bit 5 **TAMP6IE**: Tamper 6 interrupt enable
0: Tamper 6 interrupt disabled.
1: Tamper 6 interrupt enabled.

Bit 4 **TAMP5IE**: Tamper 5 interrupt enable
0: Tamper 5 interrupt disabled.
1: Tamper 5 interrupt enabled.

Bit 3 **TAMP4IE**: Tamper 4 interrupt enable
0: Tamper 4 interrupt disabled.
1: Tamper 4 interrupt enabled.

Bit 2 **TAMP3IE**: Tamper 3 interrupt enable
0: Tamper 3 interrupt disabled.
1: Tamper 3 interrupt enabled..

Bit 1 **TAMP2IE**: Tamper 2 interrupt enable
0: Tamper 2 interrupt disabled.
1: Tamper 2 interrupt enabled.

Bit 0 **TAMP1IE**: Tamper 1 interrupt enable
0: Tamper 1 interrupt disabled.
1: Tamper 1 interrupt enabled.

62.6.12 TAMP status register (TAMP_SR)

This register can be protected against nonsecure access. Refer to *Section 62.3.5: TAMP secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 62.3.7: TAMP*

privilege protection modes.

Address offset: 0x30

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	ITAMP11F	Res.	ITAMP9F	ITAMP8F	ITAMP7F	ITAMP6F	ITAMP5F	ITAMP4F	ITAMP3F	ITAMP2F	ITAMP1F
					r		r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP7F	TAMP6F	TAMP5F	TAMP4F	TAMP3F	TAMP2F	TAMP1F
									r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bit 30 Reserved, must be kept at reset value.

Bit 29 Reserved, must be kept at reset value.

Bit 28 Reserved, must be kept at reset value.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **ITAMP11F**: Internal tamper 11 flag

This flag is set by hardware when a tamper detection event is detected on the internal tamper 11.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **ITAMP9F**: Internal tamper 9 flag

This flag is set by hardware when a tamper detection event is detected on the internal tamper 9.

Bit 23 **ITAMP8F**: Internal tamper 8 flag

This flag is set by hardware when a tamper detection event is detected on the internal tamper 8.

Bit 22 **ITAMP7F**: Internal tamper 7 flag

This flag is set by hardware when a tamper detection event is detected on the internal tamper 7.

Bit 21 **ITAMP6F**: Internal tamper 6 flag

This flag is set by hardware when a tamper detection event is detected on the internal tamper 6.

Bit 20 **ITAMP5F**: Internal tamper 5 flag

This flag is set by hardware when a tamper detection event is detected on the internal tamper 5.

Bit 19 **ITAMP4F**: Internal tamper 4 flag

This flag is set by hardware when a tamper detection event is detected on the internal tamper 4.

Bit 18 **ITAMP3F**: Internal tamper 3 flag

This flag is set by hardware when a tamper detection event is detected on the internal tamper 3.

Bit 17 **ITAMP2F**: Internal tamper 2 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 2.

Bit 16 **ITAMP1F**: Internal tamper 1 flag
This flag is set by hardware when a tamper detection event is detected on the internal tamper 1.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TAMP7F**: TAMP7 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP7 input.

Bit 5 **TAMP6F**: TAMP6 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP6 input.

Bit 4 **TAMP5F**: TAMP5 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP5 input.

Bit 3 **TAMP4F**: TAMP4 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP4 input.

Bit 2 **TAMP3F**: TAMP3 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP3 input.

Bit 1 **TAMP2F**: TAMP2 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP2 input.

Bit 0 **TAMP1F**: TAMP1 detection flag
This flag is set by hardware when a tamper detection event is detected on the TAMP1 input.

62.6.13 TAMP nonsecure masked interrupt status register (TAMP_MISR)

This register can be protected against unprivileged access. Refer to *Section 62.3.7: TAMP privilege protection modes*.

Address offset: 0x34

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	ITAMP11MF	Res.	ITAMP9MF	ITAMP8MF	ITAMP7MF	ITAMP6MF	ITAMP5MF	ITAMP4MF	ITAMP3MF	ITAMP2MF	ITAMP1MF
					r		r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP7MF	TAMP6MF	TAMP5MF	TAMP4MF	TAMP3MF	TAMP2MF	TAMP1MF
									r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bit 30 Reserved, must be kept at reset value.

Bit 29 Reserved, must be kept at reset value.

Bit 28 Reserved, must be kept at reset value.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **ITAMP11MF**: internal tamper 11 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 11 nonsecure interrupt is raised.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **ITAMP9MF**: internal tamper 9 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 9 nonsecure interrupt is raised.

Bit 23 **ITAMP8MF**: Internal tamper 8 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 8 nonsecure interrupt is raised.

Bit 22 **ITAMP7MF**: Internal tamper 7 tamper nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 7 nonsecure interrupt is raised.

Bit 21 **ITAMP6MF**: Internal tamper 6 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 6 nonsecure interrupt is raised.

Bit 20 **ITAMP5MF**: Internal tamper 5 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 5 nonsecure interrupt is raised.

Bit 19 **ITAMP4MF**: Internal tamper 4 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 4 nonsecure interrupt is raised.

Bit 18 **ITAMP3MF**: Internal tamper 3 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 3 nonsecure interrupt is raised.

Bit 17 **ITAMP2MF**: Internal tamper 2 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 2 nonsecure interrupt is raised.

Bit 16 **ITAMP1MF**:Internal tamper 1 nonsecure interrupt masked flag
This flag is set by hardware when the internal tamper 1 nonsecure interrupt is raised.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TAMP7MF**: TAMP7 nonsecure interrupt masked flag
This flag is set by hardware when the tamper 7 nonsecure interrupt is raised.

Bit 5 **TAMP6MF**: TAMP6 nonsecure interrupt masked flag
This flag is set by hardware when the tamper 6 nonsecure interrupt is raised.

Bit 4 **TAMP5MF**: TAMP5 nonsecure interrupt masked flag
This flag is set by hardware when the tamper 5 nonsecure interrupt is raised.

Bit 3 **TAMP4MF**: TAMP4 nonsecure interrupt masked flag
This flag is set by hardware when the tamper 4 nonsecure interrupt is raised.

Bit 2 **TAMP3MF**: TAMP3 nonsecure interrupt masked flag
This flag is set by hardware when the tamper 3 nonsecure interrupt is raised.

Bit 1 **TAMP2MF**: TAMP2 nonsecure interrupt masked flag
This flag is set by hardware when the tamper 2 nonsecure interrupt is raised.

Bit 0 **TAMP1MF**: TAMP1 nonsecure interrupt masked flag
This flag is set by hardware when the tamper 1 nonsecure interrupt is raised.

62.6.14 TAMP secure masked interrupt status register (TAMP_SMISR)

This register can be protected against nonsecure access. Refer to *Section 62.3.5: TAMP secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 62.3.7: TAMP privilege protection modes*.

Address offset: 0x38

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	ITAMP11MF	Res.	ITAMP9MF	ITAMP8MF	ITAMP7MF	ITAMP6MF	ITAMP5MF	ITAMP4MF	ITAMP3MF	ITAMP2MF	ITAMP1MF
					r		r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP7MF	TAMP6MF	TAMP5MF	TAMP4MF	TAMP3MF	TAMP2MF	TAMP1MF
									r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bit 30 Reserved, must be kept at reset value.

Bit 29 Reserved, must be kept at reset value.

Bit 28 Reserved, must be kept at reset value.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **ITAMP11MF**: internal tamper 11 secure interrupt masked flag
This flag is set by hardware when the internal tamper 11 secure interrupt is raised.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **ITAMP9MF**: internal tamper 9 secure interrupt masked flag
This flag is set by hardware when the internal tamper 9 secure interrupt is raised.

Bit 23 **ITAMP8MF**: Internal tamper 8 secure interrupt masked flag
This flag is set by hardware when the internal tamper 8 secure interrupt is raised.

Bit 22 **ITAMP7MF**: Internal tamper 7 secure interrupt masked flag
This flag is set by hardware when the internal tamper 7 secure interrupt is raised.

Bit 21 **ITAMP6MF**: Internal tamper 6 secure interrupt masked flag
This flag is set by hardware when the internal tamper 6 secure interrupt is raised.

Bit 20 **ITAMP5MF**: Internal tamper 5 secure interrupt masked flag
This flag is set by hardware when the internal tamper 5 secure interrupt is raised.

Bit 19 **ITAMP4MF**: Internal tamper 4 secure interrupt masked flag
This flag is set by hardware when the internal tamper 4 secure interrupt is raised.

Bit 18 **ITAMP3MF**: Internal tamper 3 secure interrupt masked flag
This flag is set by hardware when the internal tamper 3 secure interrupt is raised.

Bit 17 **ITAMP2MF**: Internal tamper 2 secure interrupt masked flag
This flag is set by hardware when the internal tamper 2 secure interrupt is raised.

Bit 16 **ITAMP1MF**: Internal tamper 1 secure interrupt masked flag
This flag is set by hardware when the internal tamper 1 secure interrupt is raised.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TAMP7MF**: TAMP7 secure interrupt masked flag
This flag is set by hardware when the tamper 7 secure interrupt is raised.

Bit 5 **TAMP6MF**: TAMP6 secure interrupt masked flag
This flag is set by hardware when the tamper 6 secure interrupt is raised.

Bit 4 **TAMP5MF**: TAMP5 secure interrupt masked flag
This flag is set by hardware when the tamper 5 secure interrupt is raised.

Bit 3 **TAMP4MF**: TAMP4 secure interrupt masked flag
This flag is set by hardware when the tamper 4 secure interrupt is raised.

Bit 2 **TAMP3MF**: TAMP3 secure interrupt masked flag
This flag is set by hardware when the tamper 3 secure interrupt is raised.

Bit 1 **TAMP2MF**: TAMP2 secure interrupt masked flag
This flag is set by hardware when the tamper 2 secure interrupt is raised.

Bit 0 **TAMP1MF**: TAMP1 secure interrupt masked flag
This flag is set by hardware when the tamper 1 secure interrupt is raised.

62.6.15 TAMP status clear register (TAMP_SCR)

This register can be protected against nonsecure access. Refer to *Section 62.3.5: TAMP secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 62.3.7: TAMP privilege protection modes*.

Address offset: 0x3C

System reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	CITAMP11F	Res.	CITAMP9F	CITAMP8F	CITAMP7F	CITAMP6F	CITAMP5F	CITAMP4F	CITAMP3F	CITAMP2F	CITAMP1F
					w		w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTAMP7F	CTAMP6F	CTAMP5F	CTAMP4F	CTAMP3F	CTAMP2F	CTAMP1F
									w	w	w	w	w	w	w

Bit 31 Reserved, must be kept at reset value.

Bit 30 Reserved, must be kept at reset value.

Bit 29 Reserved, must be kept at reset value.

Bit 28 Reserved, must be kept at reset value.

Bit 27 Reserved, must be kept at reset value.

Bit 26 **CITAMP11F**: Clear ITAMP11 detection flag
Writing 1 in this bit clears the ITAMP11F bit in the TAMP_SR register.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **CITAMP9F**: Clear ITAMP9 detection flag
Writing 1 in this bit clears the ITAMP9F bit in the TAMP_SR register.

Bit 23 **CITAMP8F**: Clear ITAMP8 detection flag
Writing 1 in this bit clears the ITAMP8F bit in the TAMP_SR register.

Bit 22 **CITAMP7F**: Clear ITAMP7 detection flag
Writing 1 in this bit clears the ITAMP7F bit in the TAMP_SR register.

Bit 21 **CITAMP6F**: Clear ITAMP6 detection flag
Writing 1 in this bit clears the ITAMP6F bit in the TAMP_SR register.

Bit 20 **CITAMP5F**: Clear ITAMP5 detection flag
Writing 1 in this bit clears the ITAMP5F bit in the TAMP_SR register.

Bit 19 **CITAMP4F**: Clear ITAMP4 detection flag
Writing 1 in this bit clears the ITAMP4F bit in the TAMP_SR register.

Bit 18 **CITAMP3F**: Clear ITAMP3 detection flag
Writing 1 in this bit clears the ITAMP3F bit in the TAMP_SR register.

Bit 17 **CITAMP2F**: Clear ITAMP2 detection flag
Writing 1 in this bit clears the ITAMP2F bit in the TAMP_SR register.

Bit 16 **CITAMP1F**: Clear ITAMP1 detection flag
Writing 1 in this bit clears the ITAMP1F bit in the TAMP_SR register.

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **CTAMP7F**: Clear TAMP7 detection flag
Writing 1 in this bit clears the TAMP7F bit in the TAMP_SR register.

Bit 5 **CTAMP6F**: Clear TAMP6 detection flag
Writing 1 in this bit clears the TAMP6F bit in the TAMP_SR register.

Bit 4 **CTAMP5F**: Clear TAMP5 detection flag
Writing 1 in this bit clears the TAMP5F bit in the TAMP_SR register.

Bit 3 **CTAMP4F**: Clear TAMP4 detection flag
Writing 1 in this bit clears the TAMP4F bit in the TAMP_SR register.

Bit 2 **CTAMP3F**: Clear TAMP3 detection flag
Writing 1 in this bit clears the TAMP3F bit in the TAMP_SR register.

Bit 1 **CTAMP2F**: Clear TAMP2 detection flag
Writing 1 in this bit clears the TAMP2F bit in the TAMP_SR register.

Bit 0 **CTAMP1F**: Clear TAMP1 detection flag
Writing 1 in this bit clears the TAMP1F bit in the TAMP_SR register.

62.6.16 TAMP monotonic counter 1 register (TAMP_COUNT1R)

This register can be protected against nonsecure access. Refer to *Section 62.3.5: TAMP secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 62.3.7: TAMP privilege protection modes*.

Address offset: 0x040

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COUNT[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COUNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **COUNT[31:0]:** Monotonic counter

The monotonic counter increments by one when a write access to this register occurs, regardless of the written value. Reading this register returns the monotonic counter value. This register does not roll-over and freezes upon reaching its maximum value.

62.6.17 TAMP option register (TAMP_OR)

This register can be protected against nonsecure access. Refer to *Section 62.3.5: TAMP secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 62.3.7: TAMP privilege protection modes*.

Address offset: 0x50

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VCOREMEN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **VCOREMEN**: V_{CORE} monitoring

0: V_{CORE} monitoring disable

1: V_{CORE} monitoring enable

62.6.18 TAMP resources protection configuration register (TAMP_RPCFGR)

This register can be protected against nonsecure access. Refer to *Section 62.3.5: TAMP secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 62.3.7: TAMP privilege protection modes*.

Address offset: 0x54

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RP CFG0
															rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bit 5 Reserved, must be kept at reset value.

Bit 4 Reserved, must be kept at reset value.

Bit 3 Reserved, must be kept at reset value.

Bit 2 Reserved, must be kept at reset value.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **RPCFG0**: Configurable resource 0 protection(1)

0: Resource 0 is not included in the device secrets protected by TAMP peripheral

1: Resource 0 is included in the device secrets protected by TAMP peripheral

1. Refer to tamp_confirmed_rpcfg0 and tamp_potential_rpcfg0 signals in *Table 592: TAMP input/output pins* and *Table 594: TAMP interconnection*.

62.6.19 TAMP backup x register (TAMP_BKPxR)

This register can be protected against nonsecure access. Refer to *Section 62.3.5: TAMP secure protection modes*.

This register can be protected against unprivileged access. Refer to *Section 62.3.7: TAMP privilege protection modes*.

Address offset: 0x100 + 0x04 * x, (x = 0 to 31)

Backup domain reset value: 0x0000 0000

System reset: not affected

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BKP[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BKP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw

Bits 31:0 **BKP[31:0]:**

The application can write or read data to and from these registers.

In the default (ERASE) configuration this register is reset on a tamper detection event. It is forced to reset value as long as there is at least one internal or external tamper flag being set.

62.6.20 TAMP register map

Table 602. TAMP register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	TAMP_CR1	Res.	Res.	Res.	Res.	Res.	ITAMP11E	Res.	ITAMP9E	ITAMP8E	ITAMP7E	ITAMP6E	ITAMP5E	ITAMP4E	ITAMP3E	ITAMP2E	ITAMP1E	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP7E	TAMP6E	TAMP5E	TAMP4E	TAMP3E	TAMP2E	TAMP1E
	Reset value						0		0	0	0	0	0	0	0	0	0										0	0	0	0	0	0	0
0x04	TAMP_CR2	Res.	TAMP7TRG	TAMP6TRG	TAMP5TRG	TAMP4TRG	TAMP3TRG	TAMP2TRG	TAMP1TRG	BKERASE	BKBLOCK	Res.	Res.	Res.	TAMP3MSK	TAMP2MSK	TAMP1MSK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP7POM	TAMP6POM	TAMP5POM	TAMP4POM	TAMP3POM	TAMP2POM	TAMP1POM
	Reset value		0	0	0	0	0	0	0	0	0				0	0	0										0	0	0	0	0	0	0
0x08	TAMP_CR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ITAMP11POM	Res.	ITAMP9POM	ITAMP8POM	ITAMP7POM	ITAMP6POM	ITAMP5POM	ITAMP4POM	ITAMP3POM	ITAMP2POM	ITAMP1POM
	Reset value																						0		0	0	0	0	0	0	0	0	0
0x0C	TAMP_FLTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMPPUDIS	TAMPPRCH[1:0]		TAMPFLT[1:0]		TAMPFREQ[2:0]		
	Reset value																									0	0	0	0	0	0	0	0
0x10	TAMP_ATCR1	FLTEN	ATOSHARE	Res.	Res.	Res.	AT PER[2:0]			Res.	Res.	Res.	Res.	ATCK SEL[3:0]				ATO SEL4 [1:0]		ATO SEL3 [1:0]		ATO SEL2 [1:0]		ATO SEL1 [1:0]		Res.	TAMP7AM	TAMP6AM	TAMP5AM	TAMP4AM	TAMP3AM	TAMP2AM	TAMP1AM
	Reset value	0	0				0	0	0					0	1	1	1	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x14	TAMP_ ATSEEDR	SEED[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	TAMP_ATOR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INITS	SEEDF	Res.	Res.	Res.	Res.	Res.	Res.	PRNG[7:0]							
	Reset value																	0	0							0	0	0	0	0	0	0	0
0x1C	TAMP_ATCR2	Res.	Res.	Res.	ATO SEL7 [2:0]			ATO SEL6 [2:0]			ATO SEL5 [2:0]			ATO SEL4 [2:0]				ATO SEL3 [2:0]			ATO SEL2 [2:0]			ATO SEL1 [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								
0x20	TAMP_SEC CFGR	TAMPSEC	BHKLOCK	Res.	Res.	Res.	Res.	Res.	Res.	BKPWSEC[7:0]								CNT1SEC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BKPRWSEC[7:0]							
	Reset value	0	0							0	0	0	0	0	0	0	0	0								0	0	0	0	0	0	0	0
0x24	TAMP_ PRIVCFGR	TAMPPRIV	BKPWPRIV	BKPRWPRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNT1PRIV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0														0															

Table 602. TAMP register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x2C	TAMP_IER	Res.	Res.	Res.	Res.	Res.	ITAMP11IE	Res.	ITAMP9IE	ITAMP8IE	ITAMP7IF	ITAMP6IE	ITAMP5IE	ITAMP4IE	ITAMP3IE	ITAMP2IE	ITAMP1IE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP7IE	TAMP6IE	TAMP5IE	TAMP4IE	TAMP3IE	TAMP2IE	TAMP1IE
	Reset value						0		0	0	0	0	0	0	0	0	0										0	0	0	0	0	0	0
0x30	TAMP_SR	Res.	Res.	Res.	Res.	Res.	ITAMP11F	Res.	ITAMP9F	ITAMP8F	ITAMP7F	ITAMP6F	ITAMP5F	ITAMP4F	ITAMP3F	ITAMP2F	ITAMP1F	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP7F	TAMP6F	TAMP5F	TAMP4F	TAMP3F	TAMP2F	TAMP1F
	Reset value						0		0	0	0	0	0	0	0	0	0										0	0	0	0	0	0	0
0x34	TAMP_MISR	Res.	Res.	Res.	Res.	Res.	ITAMP11MF	Res.	ITAMP9MF	ITAMP8MF	ITAMP7MF	ITAMP6MF	ITAMP5MF	ITAMP4MF	ITAMP3MF	ITAMP2MF	ITAMP1MF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP7MF	TAMP6MF	TAMP5MF	TAMP4MF	TAMP3MF	TAMP2MF	TAMP1MF
	Reset value						0		0	0	0	0	0	0	0	0	0										0	0	0	0	0	0	0
0x38	TAMP_SMISR	Res.	Res.	Res.	Res.	Res.	ITAMP11MF	Res.	ITAMP9MF	ITAMP8MF	ITAMP7MF	ITAMP6MF	ITAMP5MF	ITAMP4MF	ITAMP3MF	ITAMP2MF	ITAMP1MF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TAMP7MF	TAMP6MF	TAMP5MF	TAMP4MF	TAMP3MF	TAMP2MF	TAMP1MF
	Reset value						0		0	0	0	0	0	0	0	0	0										0	0	0	0	0	0	0
0x3C	TAMP_SCR	Res.	Res.	Res.	Res.	Res.	CITAMP11F	Res.	CITAMP9F	CITAMP8F	CITAMP7F	CITAMP6F	CITAMP5F	CITAMP4F	CITAMP3F	CITAMP2F	CITAMP1F	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTAMP7F	CTAMP6F	CTAMP5F	CTAMP4F	CTAMP3F	CTAMP2F	CTAMP1F
	Reset value						0		0	0	0	0	0	0	0	0	0										0	0	0	0	0	0	0
0x40	TAMP_ COUNT1R	COUNT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x50	TAMP_OR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VCOREMEN
	Reset value																																0
0x54	TAMP_ RPCFGR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RPCFG0
	Reset value																																0
0x100 + 0x04*x, (x= 0 to 31)	TAMP_BKPxR	BKP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

63 Inter-integrated circuit interface (I2C)

63.1 I2C introduction

The I2C peripheral handles the interface between the device and the serial I²C (inter-integrated circuit) bus. It provides multicontroller capability, and controls all I²C-bus-specific sequencing, protocol, arbitration and timing. It supports Standard-mode (Sm), Fast-mode (Fm) and Fast-mode Plus (Fm+).

The I2C peripheral is also SMBus (system management bus) and PMBus® (power management bus) compatible.

It can use DMA to reduce the CPU load.

63.2 I2C main features

- I²C-bus specification rev03 compatibility:
  - Target and controller modes
  - Multicontroller capability
  - Standard-mode (up to 100 kHz)
  - Fast-mode (up to 400 kHz)
  - Fast-mode Plus (up to 1 MHz)
  - 7-bit and 10-bit addressing mode
  - Multiple 7-bit target addresses (2 addresses, 1 with configurable mask)
  - All 7-bit-addresses acknowledge mode
  - General call
  - Programmable setup and hold times
  - Easy-to-use event management
  - Clock stretching (optional)
- 1-byte buffer with DMA capability
- Programmable analog and digital noise filters
- SMBus specification rev 3.0 compatibility[(a)]:
  - Hardware PEC (packet error checking) generation and verification with ACK control
  - Command and data acknowledge control
  - Address resolution protocol (ARP) support
  - Host and device support
  - SMBus alert
  - Timeouts and idle condition detection
- PMBus rev 1.3 standard compatibility
- Independent clock

a. To check the compliance of the GPIOs selected for SMBus with the specified logical levels, refer to the product datasheet.

- Wake-up from Stop mode on address match

For information on I2C instantiation, refer to *Section 63.3: I2C implementation*.

63.3 I2C implementation

This section provides an implementation overview with respect to the I2C instantiation.

Table 603. I2C implementation

I2C features[1]	I2C1	I2C2	I2C3	I2C4
7-bit addressing mode	X	X	X	X
10-bit addressing mode	X	X	X	X
Standard-mode (up to 100 kbit/s)	X	X	X	X
Fast-mode (up to 400 kbit/s)	X	X	X	X
Fast-mode Plus with 20 mA output drive I/Os (up to 1 Mbit/s)	X	X	X	X
Independent clock	X	X	X	X
Wake-up from Stop mode	X[2]	X[2]	X[2]	X[2]
SMBus/PMBus	X	X	X	X

1. X = supported.
2. Supported only from Stop SVOS high.

63.4 I2C functional description

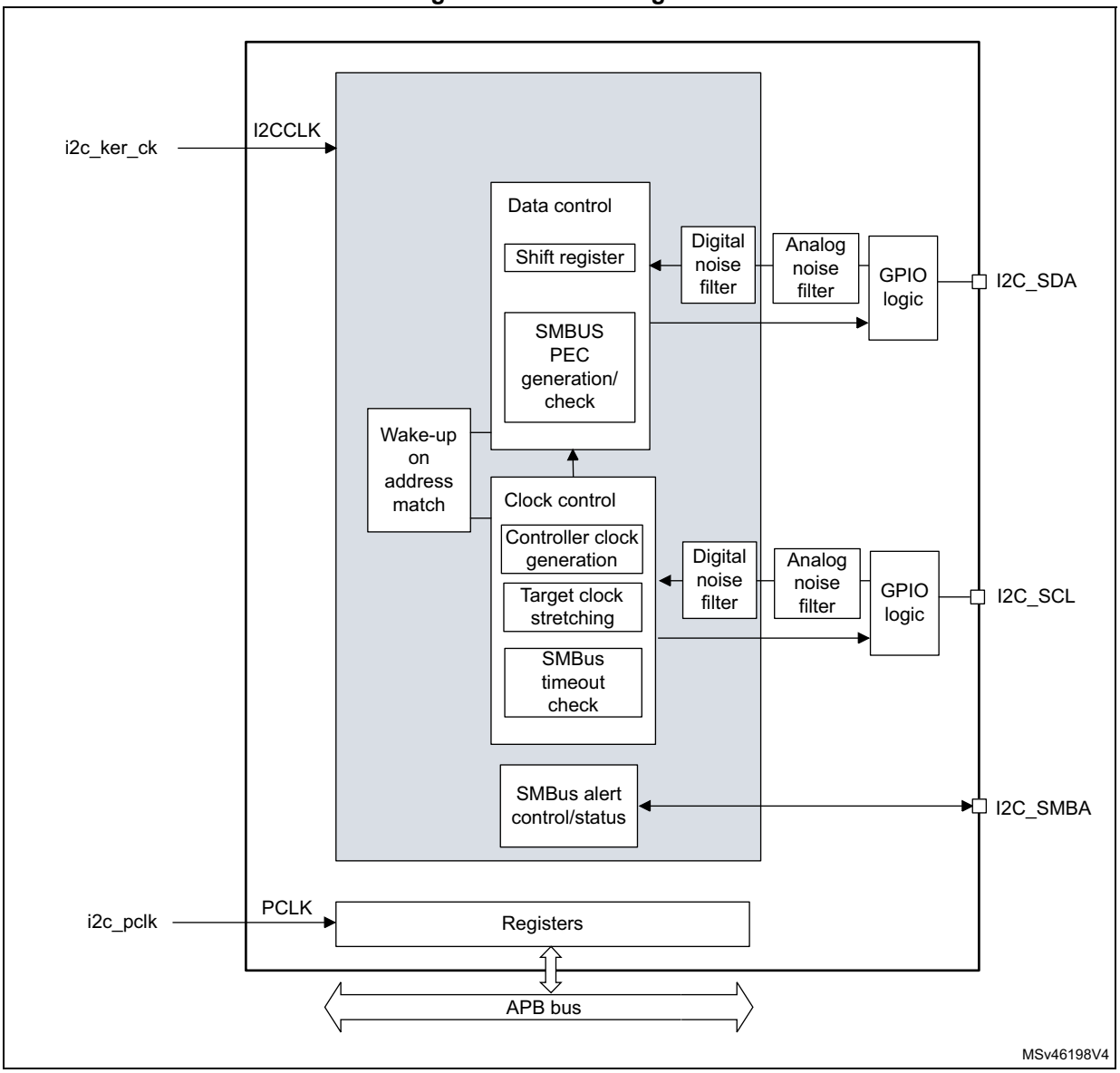
In addition to receiving and transmitting data, the peripheral converts them from serial to parallel format and vice versa. The interrupts are enabled or disabled by software. The peripheral is connected to the I²C-bus through a data pin (SDA) and a clock pin (SCL). It supports Standard-mode (up to 100 kHz), Fast-mode (up to 400 kHz), and Fast-mode Plus (up to 1 MHz) I²C-bus.

The peripheral can also be connected to an SMBus, through the data pin (SDA), the clock pin (SCL), and an optional SMBus alert pin (SMBA).

The independent clock function allows the I2C communication speed to be independent of the i2c_pclk frequency.

63.4.1 I2C block diagram

Figure 786. Block diagram



63.4.2 I2C pins and internal signals

Table 604. I2C input/output pins

Pin name	Signal type	Description
I2C_SDA	Bidirectional	I²C-bus data
I2C_SCL	Bidirectional	I²C-bus clock
I2C_SMBA	Bidirectional	SMBus alert

Table 605. I2C internal input/output signals

Internal signal name	Signal type	Description
i2c_ker_ck	Input	I2C kernel clock, also named I2CCLK in this document
i2c_pclk	Input	I2C APB clock
i2c_it	Output	I2C interrupts, refer to *Table 618* for the list of interrupt sources
i2c_rx_dma	Output	I2C receive data DMA request (I2C_RX)
i2c_tx_dma	Output	I2C transmit data DMA request (I2C_TX)

63.4.3 I2C clock requirements

The I2C kernel is clocked by i2c_ker_ck.

The i2c_ker_ck period t_{I2CCLK} must respect the following conditions:

$t_{I2CCLK} < (t_{LOW} - t_{filters}) / 4$

$t_{I2CCLK} < t_{HIGH}$

where t_{LOW} is the SCL low time, t_{HIGH} is the SCL high time, and $t_{filters}$ is the sum of the analog and digital filter delays (when enabled).

The digital filter delay is DNF[3:0] x t_{I2CCLK}.

The i2c_pclk clock period t_{PCLK} must respect the condition $t_{PCLK} < 4/3\ t_{SCL}$, where t_{SCL} is the SCL period.

Caution: When the I2C kernel is clocked by i2c_pclk, this clock must respect the conditions for t_{I2CCLK}.

63.4.4 I2C mode selection

The peripheral can operate as:

- Target transmitter
- Target receiver
- Controller transmitter
- Controller receiver

By default, the peripheral operates in target mode. It automatically switches from target to controller mode upon generating START condition, and from controller to target mode upon arbitration loss or upon generating STOP condition. This allows the use of the I2C peripheral in a multicontroller I²C-bus environment.

Communication flow

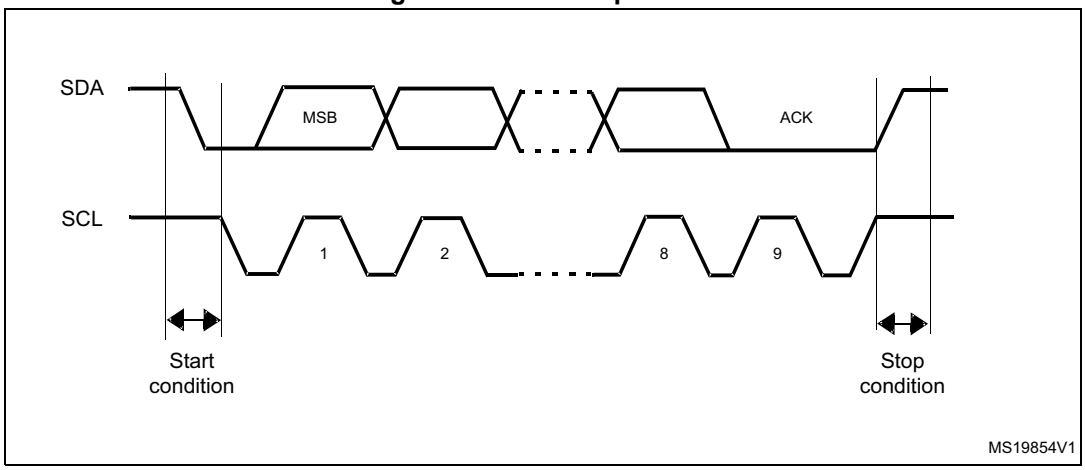
In controller mode, the I2C peripheral initiates a data transfer and generates the clock signal. Serial data transfers always begin with a START condition and end with a STOP condition. Both START and STOP conditions are generated in controller mode by software.

In target mode, the peripheral recognizes its own 7-bit or 10-bit address, and the general call address. The general call address detection can be enabled or disabled by software. The reserved SMBus addresses can also be enabled by software.

Data and addresses are transferred as 8-bit bytes, MSB first. The address is contained in the first byte (7-bit addressing) or in the first two bytes (10-bit addressing) following the START condition. The address is always transmitted in controller mode.

The following figure shows the transmission of a single byte. The controller generates nine SCL pulses. The transmitter sends the eight data bits to the receiver with the SCL pulses 1 to 8. Then the receiver sends the acknowledge bit to the transmitter with the ninth SCL pulse.

Figure 787. I²C-bus protocol



The acknowledge can be enabled or disabled by software. The own addresses of the I2C peripheral can be selected by software.

63.4.5 I2C initialization

Enabling and disabling the peripheral

Before enabling the I2C peripheral, configure and enable its clock through the RCC, and initialize its control registers.

The I2C peripheral can then be enabled by setting the PE bit of the I2C_CR1 register.

Disabling the I2C peripheral by clearing the PE bit resets the I2C peripheral. Refer to *Section 63.4.6* for more details.

Noise filters

Before enabling the I2C peripheral by setting the PE bit of the I2C_CR1 register, the user must configure the analog and/or digital noise filters, as required.

The analog noise filter on the SDA and SCL inputs complies with the I²C-bus specification which requires, in Fast-mode and Fast-mode Plus, the suppression of spikes shorter than 50 ns. Enabled by default, it can be disabled by setting the ANFOFF bit.

The digital filter is controlled through the DNF[3:0] bitfield of the I2C_CR1 register. When it is enabled, the internal SCL and SDA signals only take the level of their corresponding I²C-bus line when remaining stable for more than DNF[3:0] periods of i2c_ker_ck. This allows suppressing spikes shorter than the filtering capacity period programmable from one to fifteen i2c_ker_ck periods.

The following table compares the two filters.

Table 606. Comparison of analog and digital filters

Item	Analog filter	Digital filter
Filtering capacity(1)	≥ 50 ns	One to fifteen i2c_ker_ck periods
Benefits	Available in Stop mode	– Programmable filtering capacity – Extra filtering capability versus I²C-bus specification requirements – Stable filtering capacity
Drawbacks	Filtering capacity variation with temperature, voltage, and silicon process	Wake-up from Stop mode on address match not supported when the digital filter is enabled

1. Maximum duration of spikes that the filter can suppress

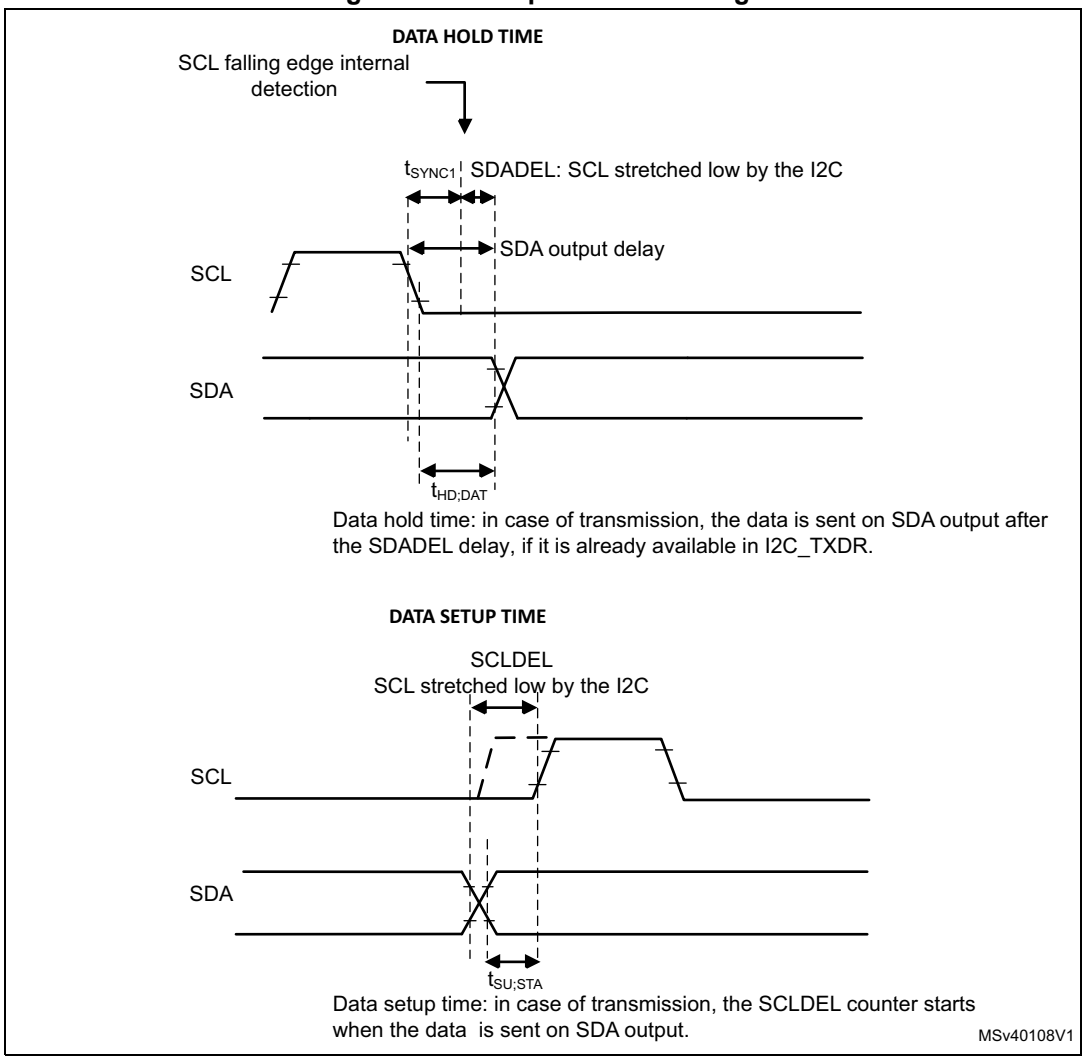
Caution: The filter configuration cannot be changed when the I2C peripheral is enabled.

I2C timings

To ensure correct data hold and setup times, the corresponding timings must be configured through the PRESC[3:0], SCLDEL[3:0], and SDADEL[3:0] bitfields of the I2C_TIMINGR register.

The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the *I2C configuration* window.

Figure 788. Setup and hold timings



When the SCL falling edge is internally detected, the delay t_{SDADEL} (impacting the hold time $t_{HD;DAT}$) is inserted before sending SDA output:

t_{SDADEL} = SDADEL x t_{PRESC} + t_{I2CCLK}, where t_{PRESC} = (PRESC + 1) x t_{I2CCLK}.

The total SDA output delay is:

t_{SYNC1} + {[SDADEL x (PRESC + 1) + 1] x t_{I2CCLK}}

The t_{SYNC1} duration depends upon:

- SCL falling slope
- input delay $t_{AF(min)} < t_{AF} < t_{AF(max)}$ introduced by the analog filter (if enabled)
- input delay t_{DNF} = DNF x t_{I2CCLK} introduced by the digital filter (if enabled)
- delay due to SCL synchronization to i2c_ker_ck clock (two to three i2c_ker_ck periods)

To bridge the undefined region of the SCL falling edge, the user must set SDADEL[3:0] so as to fulfill the following condition:

$\{t_{f(max)} + t_{HD;DAT(min)} - t_{AF(min)} - [(DNF + 3) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\} \leq SDADEL$

$SDADEL \leq \{t_{HD;DAT\,(max)} - t_{AF(max)} - [(DNF + 4) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\}$

Note: $t_{AF(min)}$ and $t_{AF(max)}$ are only part of the condition when the analog filter is enabled. Refer to the device datasheet for t_{AF} values.

The $t_{HD;DAT}$ time can at maximum be 3.45 µs for Standard-mode, 0.9 µs for Fast-mode, and 0.45 µs for Fast-mode Plus. It must be lower than the maximum of $t_{VD;DAT}$ by a transition time. This maximum must only be met if the device does not stretch the LOW period (t_{LOW}) of the SCL signal. When it stretches SCL, the data must be valid by the set-up time before it releases the clock.

The SDA rising edge is usually the worst case. The previous condition then becomes:

$$SDADEL \leq \{t_{VD;DAT\ (max)} - t_{r\ (max)} - t_{AF\ (max)} - [(DNF + 4) \times t_{I2CCLK}]\} / \{(PRESC + 1) \times t_{I2CCLK}\}$$

Note: This condition can be violated when NOSTRETCH = 0, because the device stretches SCL low to guarantee the set-up time, according to the SCLDEL[3:0] value.

After t_{SDADEL}, or after sending SDA output when the target had to stretch the clock because the data was not yet written in I2C_TXDR register, the SCL line is kept at low level during the setup time. This setup time is $t_{SCLDEL} = (SCLDEL + 1) \times t_{PRESC}$, where $t_{PRESC} = (PRESC + 1) \times t_{I2CCLK}$. t_{SCLDEL} impacts the setup time $t_{SU;DAT}$.

To bridge the undefined region of the SDA transition (rising edge usually worst case), the user must program SCLDEL[3:0] so as to fulfill the following condition:

$$\{[t_{r\ (max)} + t_{SU;DAT\ (min)}] / [(PRESC + 1) \times t_{I2CCLK}]\} - 1 \leq SCLDEL$$

Refer to the following table for t_f, t_r, $t_{HD;DAT}$, $t_{VD;DAT}$, and $t_{SU;DAT}$ standard values.

Use the SDA and SCL real transition time values measured in the application to widen the scope of allowed SDADEL[3:0] and SCLDEL[3:0] values. Use the maximum SDA and SCL transition time values defined in the standard to make the device work reliably regardless of the application.

Note: At every clock pulse, after SCL falling edge detection, I2C operating as controller or target stretches SCL low during at least [(SDADEL + SCLDEL + 1) x (PRESC + 1) + 1] x t_{I2CCLK}, in both transmission and reception modes. In transmission mode, if the data is not yet written in I2C_TXDR when SDA delay elapses, the I2C peripheral keeps stretching SCL low until the next data is written. Then new data MSB is sent on SDA output, and SCLDEL counter starts, continuing stretching SCL low to guarantee the data setup time.

When the NOSTRETCH bit is set in target mode, the SCL is not stretched. The SDADEL[3:0] must then be programmed so that it ensures a sufficient setup time.

Table 607. I²C-bus and SMBus specification data setup and hold times

Symbol	Parameter	Standard-mode (Sm)		Fast-mode (Fm)		Fast-mode Plus (Fm+)		SMBus		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	
$t_{HD;DAT}$	Data hold time	0	-	0	-	0	-	0.3	-	µs
$t_{VD;DAT}$	Data valid time	-	3.45	-	0.9	-	0.45	-	-	
$t_{SU;DAT}$	Data setup time	250	-	100	-	50	-	250	-	ns
t_r	Rise time of both SDA and SCL signals	-	1000	-	300	-	120	-	1000	
t_f	Fall time of both SDA and SCL signals	-	300	-	300	-	120	-	300	

Additionally, in controller mode, the SCL clock high and low levels must be configured by programming the PRESC[3:0], SCLH[7:0], and SCLL[7:0] bitfields of the I2C_TIMINGR register.

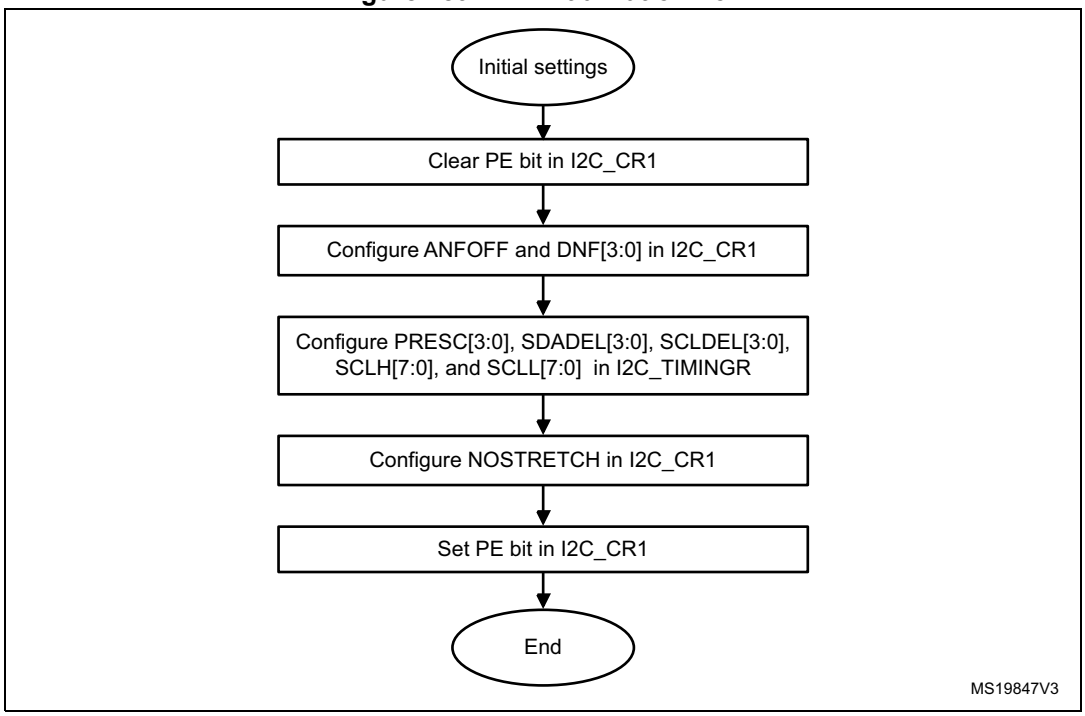
When the SCL falling edge is internally detected, the I2C peripheral releasing the SCL output after the delay t_{SCLL} = (SCLL + 1) x t_{PRESC}, where t_{PRESC} = (PRESC + 1) x t_{I2CCLK}. The t_{SCLL} delay impacts the SCL low time t_{LOW}.

When the SCL rising edge is internally detected, the I2C peripheral forces the SCL output to low level after the delay t_{SCLH} = (SCLH + 1) x t_{PRESC}, where t_{PRESC} = (PRESC + 1) x t_{I2CCLK}. The t_{SCLH} impacts the SCL high time t_{HIGH}.

Refer to *I2C controller initialization* for more details.

Caution: Changing the timing configuration and the NOSTRETCH configuration is not allowed when the I2C peripheral is enabled. Like the timing settings, the target NOSTRETCH settings must also be done before enabling the peripheral. Refer to *I2C target initialization* for more details.

Figure 789. I2C initialization flow



63.4.6 I2C reset

The reset of the I2C peripheral is performed by clearing the PE bit of the I2C_CR1 register. It has the effect of releasing the SCL and SDA lines. Internal state machines are reset and the communication control bits and the status bits revert to their reset values. This reset does not impact the configuration registers.

The impacted register bits are:

1. I2C_CR2 register: START, STOP, PECBYTE, and NACK
2. I2C_ISR register: BUSY, TXE, TXIS, RXNE, ADDR, NACKF, TCR, TC, STOPF, BERR, ARLO, PECERR, TIMEOUT, ALERT, and OVR

PE must be kept low during at least three APB clock cycles to perform the I2C reset. To ensure this, perform the following software sequence:

1. Write PE = 0
2. Check PE = 0
3. Write PE = 1

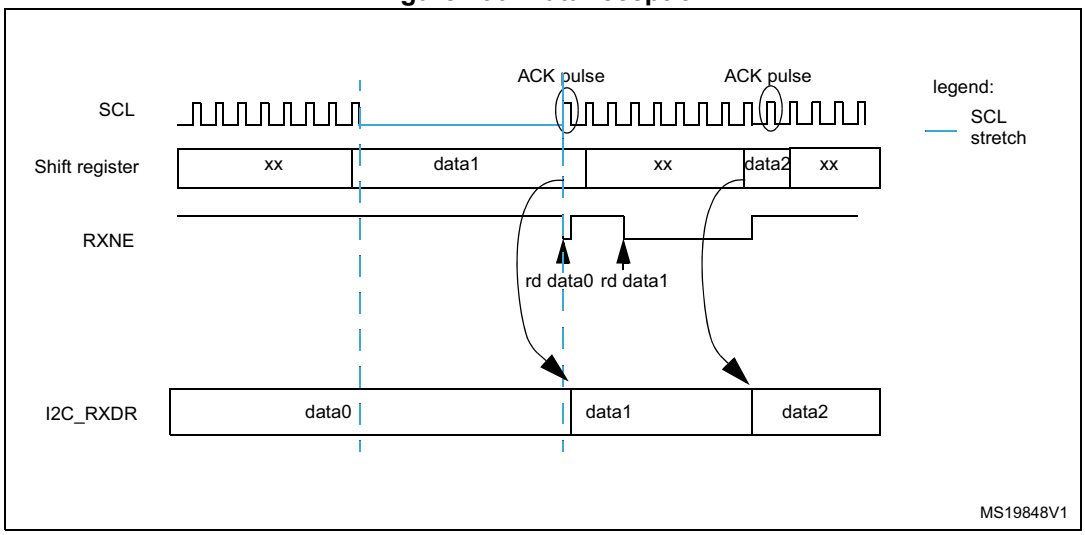
63.4.7 I2C data transfer

The data transfer is managed through transmit and receive data registers and a shift register.

Reception

The SDA input fills the shift register. After the eighth SCL pulse (when the complete data byte is received), the shift register is copied into the I2C_RXDR register if it is empty (RXNE = 0). If RXNE = 1, which means that the previous received data byte has not yet been read, the SCL line is stretched low until I2C_RXDR is read. The stretch occurs between the eighth and the ninth SCL pulse (before the acknowledge pulse).

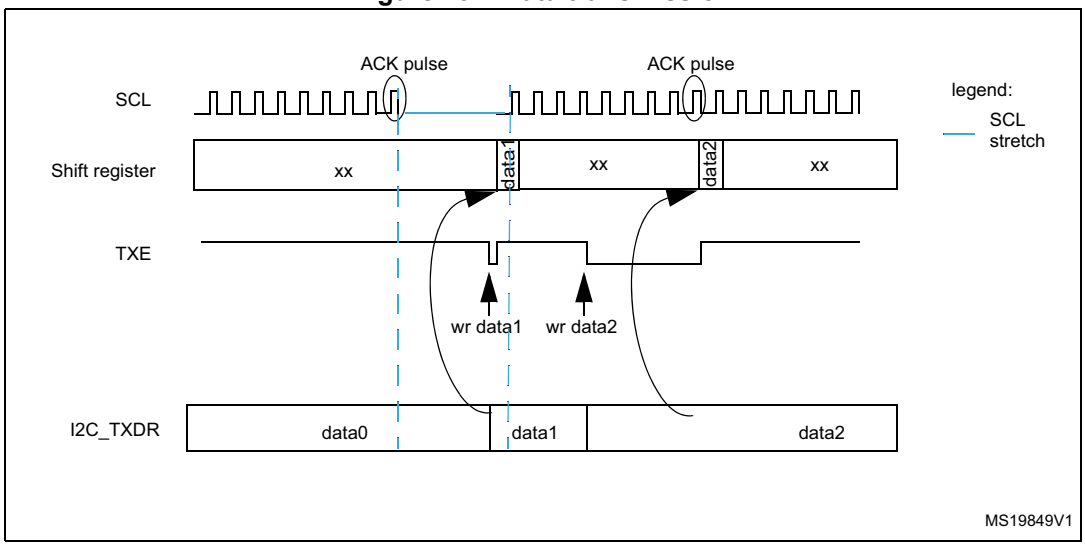
Figure 790. Data reception



Transmission

If the I2C_TXDR register is not empty (TXE = 0), its content is copied into the shift register after the ninth SCL pulse (the acknowledge pulse). Then the shift register content is shifted out on the SDA line. If TXE = 1, which means that no data is written yet in I2C_TXDR, the SCL line is stretched low until I2C_TXDR is written. The stretch starts after the ninth SCL pulse.

Figure 791. Data transmission



Hardware transfer management

The I2C features an embedded byte counter to manage byte transfer and to close the communication in various modes, such as:

- NACK, STOP and ReSTART generation in controller mode
- ACK control in target receiver mode
- PEC generation/checking

In controller mode, the byte counter is always used. By default, it is disabled in target mode. It can be enabled by software, by setting the SBC (target byte control) bit of the I2C_CR1 register.

The number of bytes to transfer is programmed in the NBYTES[7:0] bitfield of the I2C_CR2 register. If this number is greater than 255, or if a receiver wants to control the acknowledge value of a received data byte, the reload mode must be selected, by setting the RELOAD bit of the I2C_CR2 register. In this mode, the TCR flag is set when the number of bytes programmed in NBYTES[7:0] is transferred (when the associated counter reaches zero), and an interrupt is generated if TCIE is set. SCL is stretched as long as the TCR flag is set. TCR is cleared by software when NBYTES[7:0] is written to a non-zero value.

When NBYTES[7:0] is reloaded with the last number of bytes to transfer, the RELOAD bit must be cleared.

When RELOAD = 0 in controller mode, the counter can be used in two modes:

- **Automatic end** (AUTOEND = 1 in the I2C_CR2 register). In this mode, the controller automatically sends a STOP condition once the number of bytes programmed in the NBYTES[7:0] bitfield is transferred.
- **Software end** (AUTOEND = 0 in the I2C_CR2 register). In this mode, a software action is expected once the number of bytes programmed in the NBYTES[7:0] bitfield is transferred; the TC flag is set and an interrupt is generated if the TCIE bit is set. The SCL signal is stretched as long as the TC flag is set. The TC flag is cleared by software when the START or STOP bit of the I2C_CR2 register is set. This mode must be used when the controller wants to send a RESTART condition.

Caution: The AUTOEND bit has no effect when the RELOAD bit is set.

Table 608. I2C configuration

Function	SBC bit	RELOAD bit	AUTOEND bit
Controller Tx/Rx NBYTES + STOP	X	0	1
Controller Tx/Rx + NBYTES + RESTART	X	0	0
Target Tx/Rx, all received bytes ACKed	0	X	X
Target Rx with ACK control	1	1	X

63.4.8 I2C target mode

I2C target initialization

To work in target mode, the user must enable at least one target address. The I2C_OAR1 and I2C_OAR2 registers are available to program the target own addresses OA1 and OA2, respectively.

OA1 can be configured either in 7-bit (default) or in 10-bit addressing mode, by setting the OA1MODE bit of the I2C_OAR1 register.

OA1 is enabled by setting the OA1EN bit of the I2C_OAR1 register.

If an additional target addresses are required, the second target address OA2 can be configured. Up to seven OA2 LSBs can be masked, by configuring the OA2MSK[2:0] bitfield of the I2C_OAR2 register. Therefore, for OA2MSK[2:0] configured from 1 to 6, only OA2[7:2], OA2[7:3], OA2[7:4], OA2[7:5], OA2[7:6], or OA2[7] are compared with the received address. When OA2MSK[2:0] is other than 0, the address comparator for OA2 excludes the I2C reserved addresses (0000 XXX and 1111 XXX) and they are not acknowledged. If OA2MSK[2:0] = 7, all received 7-bit addresses are acknowledged (except reserved addresses). OA2 is always a 7-bit address.

When enabled through the specific bit, the reserved addresses can be acknowledged if they are programmed in the I2C_OAR1 or I2C_OAR2 register with OA2MSK[2:0] = 0.

OA2 is enabled by setting the OA2EN bit of the I2C_OAR2 register.

The general call address is enabled by setting the GCEN bit of the I2C_CR1 register.

When the I2C peripheral is selected by one of its enabled addresses, the ADDR interrupt status flag is set, and an interrupt is generated if the ADDRIE bit is set.

By default, the target uses its clock stretching capability, which means that it stretches the SCL signal at low level when required, to perform software actions. If the controller does not

support clock stretching, I2C must be configured with NOSTRETCH = 1 in the I2C_CR1 register.

After receiving an ADDR interrupt, if several addresses are enabled, the user must read the ADDCODE[6:0] bitfield of the I2C_ISR register to check which address matched. The DIR flag must also be checked to know the transfer direction.

Target with clock stretching

As long as the NOSTRETCH bit of the I2C_CR1 register is zero (default), the I2C peripheral operating as an I²C-bus target stretches the SCL signal in the following situations:

- The ADDR flag is set and the received address matches with one of the enabled target addresses.
  The stretch is released when the software clears the ADDR flag by setting the ADDRCF bit.
- In transmission, the previous data transmission is completed and no new data is written in I2C_TXDR register, or the first data byte is not written when the ADDR flag is cleared (TXE = 1).
  The stretch is released when the data is written to the I2C_TXDR register.
- In reception, the I2C_RXDR register is not read yet and a new data reception is completed.
  The stretch is released when I2C_RXDR is read.
- In target byte control mode (SBC bit set) with reload (RELOAD bit set), the last data byte transfer is finished (TCR bit set).
  The stretch is released when then TCR is cleared by writing a non-zero value in the NBYTES[7:0] bitfield.
- After SCL falling edge detection.
  The stretch is released after [(SDADEL + SCLDEL + 1) x (PRESC+ 1) + 1] x t_{I2CCLK} period.

Target without clock stretching

As long as the NOSTRETCH bit of the I2C_CR1 register is set, the I2C peripheral operating as an I²C-bus target does not stretch the SCL signal.

The SCL clock is not stretched while the ADDR flag is set.

In transmission, the data must be written in the I2C_TXDR register before the first SCL pulse corresponding to its transfer occurs. If not, an underrun occurs, the OVR flag is set in the I2C_ISR register and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set. The OVR flag is also set when the first data transmission starts and the STOPF bit is still set (has not been cleared). Therefore, if the user clears the STOPF flag of the previous transfer only after writing the first data to be transmitted in the next transfer, it ensures that the OVR status is provided, even for the first data to be transmitted.

In reception, the data must be read from the I2C_RXDR register before the ninth SCL pulse (ACK pulse) of the next data byte occurs. If not, an overrun occurs, the OVR flag is set in the I2C_ISR register, and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Target byte control mode

To allow byte ACK control in target reception mode, the target byte control mode must be enabled, by setting the SBC bit of the I2C_CR1 register. This is required to comply with SMBus standards.

The reload mode must be selected to allow byte ACK control in target reception mode (RELOAD = 1). To get control of each byte, NBYTES[7:0] must be initialized to 0x1 in the ADDR interrupt subroutine, and reloaded to 0x1 after each received byte. When the byte is received, the TCR bit is set, stretching the SCL signal low between the eighth and the ninth SCL pulse. The user can read the data from the I2C_RXDR register, and then decide to acknowledge it or not by configuring the ACK bit of the I2C_CR2 register. The SCL stretch is released by programming NBYTES to a non-zero value: the acknowledge or not-acknowledge is sent and the next byte can be received.

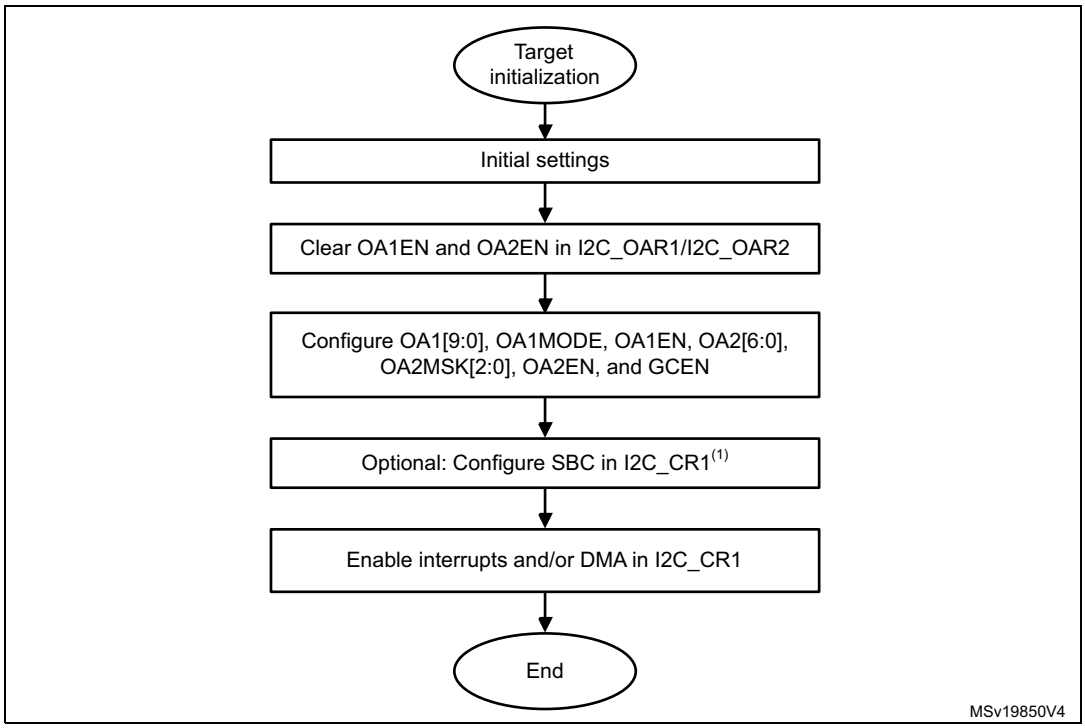
NBYTES[7:0] can be loaded with a value greater than 0x1. Receiving then continues until the corresponding number of bytes are received.

Note: The SBC bit must be configured when the I2C peripheral is disabled, when the target is not addressed, or when ADDR = 1.

The RELOAD bit value can be changed when ADDR = 1, or when TCR = 1.

Caution: The target byte control mode is not compatible with NOSTRETCH mode. Setting SBC when NOSTRETCH = 1 is not allowed.

Figure 792. Target initialization flow



1. SBC must be set to support SMBus features.

Target transmitter

A transmit interrupt status (TXIS) flag is generated when the I2C_TXDR register becomes empty. An interrupt is generated if the TXIE bit of the I2C_CR1 register is set.

The TXIS flag is cleared when the I2C_TXDR register is written with the next data byte to transmit.

When NACK is received, the NACKF flag is set in the I2C_ISR register and an interrupt is generated if the NACKIE bit of the I2C_CR1 register is set. The target automatically releases the SCL and SDA lines to let the controller perform a STOP or a RESTART condition. The TXIS bit is not set when a NACK is received.

When STOP is received and the STOPIE bit of the I2C_CR1 register is set, the STOPF flag of the I2C_ISR register is set and an interrupt is generated. In most applications, the SBC bit is usually programmed to 0. In this case, if TXE = 0 when the target address is received (ADDR = 1), the user can choose either to send the content of the I2C_TXDR register as the first data byte, or to flush the I2C_TXDR register, by setting the TXE bit in order to program a new data byte.

In target byte control mode (SBC = 1), the number of bytes to transmit must be programmed in NBYTES[7:0] in the address match interrupt subroutine (ADDR = 1). In this case, the number of TXIS events during the transfer corresponds to the value programmed in NBYTES[7:0].

Caution: When NOSTRETCH = 1, the SCL clock is not stretched while the ADDR flag is set, so the user cannot flush the I2C_TXDR register content in the ADDR subroutine to program the first data byte. The first data byte to send must be previously programmed in the I2C_TXDR register:

- This data can be the one written in the last TXIS event of the previous transmission message.
- If this data byte is not the one to send, the I2C_TXDR register can be flushed, by setting the TXE bit, to program a new data byte. The STOPF bit must be cleared only after these actions. This guarantees that they are executed before the first data transmission starts, following the address acknowledge.

  If STOPF is still set when the first data transmission starts, an underrun error is generated (the OVR flag is set).

  If a TXIS event (transmit interrupt or transmit DMA request) is required, the user must set the TXIS bit in addition to the TXE bit, to generate the event.

Figure 793. Transfer sequence flow for I2C target transmitter, NOSTRETCH = 0

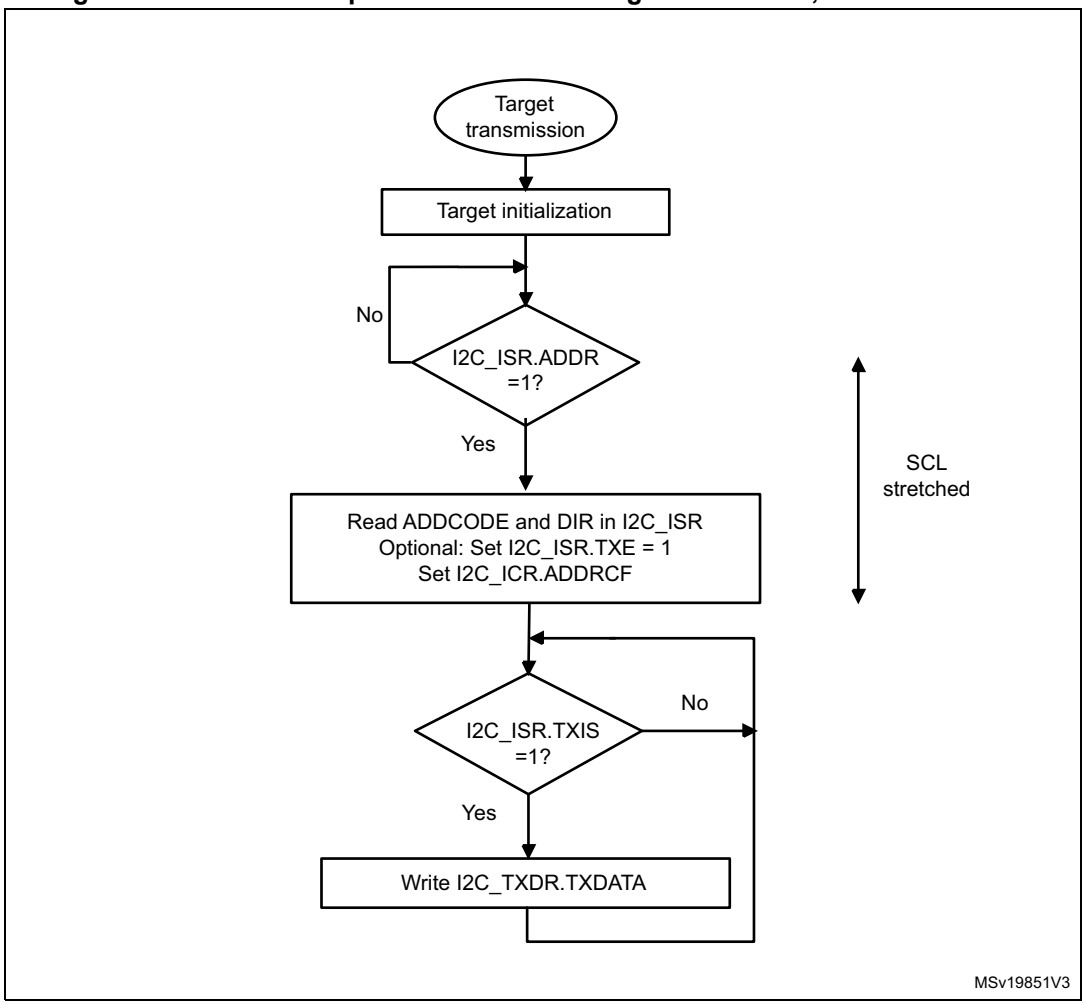


Figure 794. Transfer sequence flow for I2C target transmitter, NOSTRETCH = 1

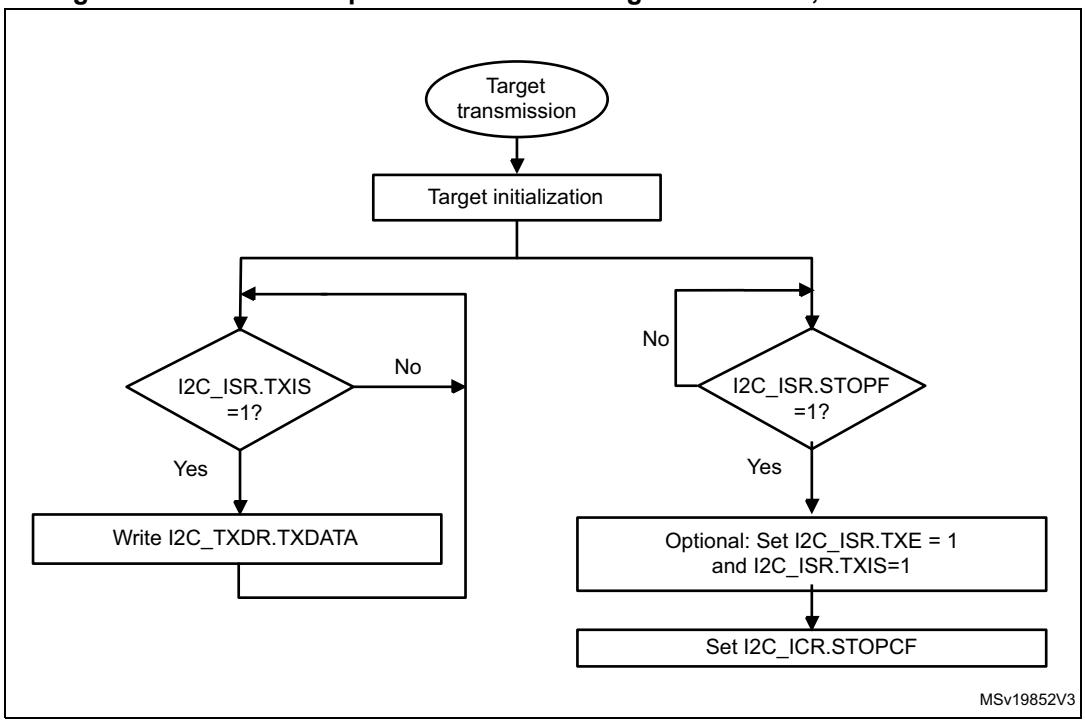
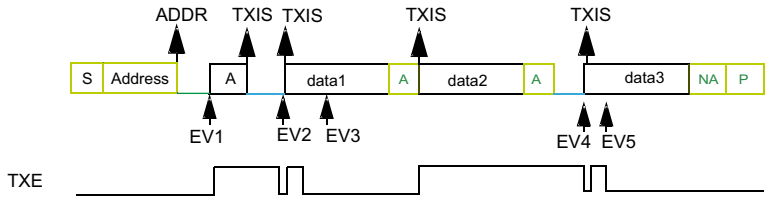


Figure 795. Transfer bus diagrams for I2C target transmitter (mandatory events only)

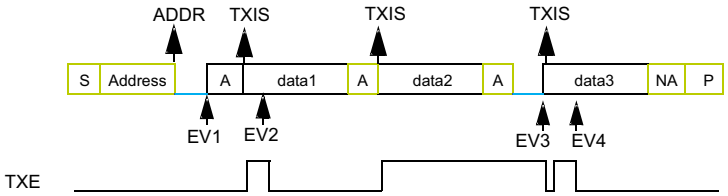
Example I2C target transmitter 3 bytes with 1st data flushed, NOSTRETCH=0:



legend:
transmission
reception
SCL stretch

EV1: ADDR ISR: check ADDCODE and DIR, set TXE, set ADDRCF
EV2: TXIS ISR: wr data1
EV3: TXIS ISR: wr data2
EV4: TXIS ISR: wr data3
EV5: TXIS ISR: wr data4 (not sent)

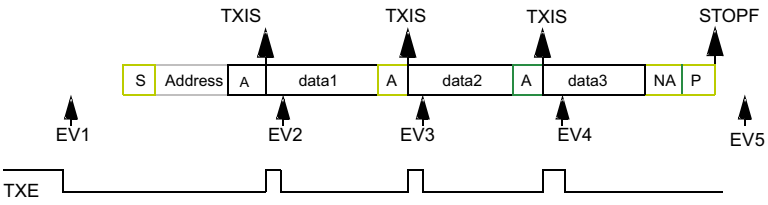
Example I2C target transmitter 3 bytes without 1st data flush, NOSTRETCH=0:



legend :
transmission
reception
SCL stretch

EV1: ADDR ISR: check ADDCODE and DIR, set ADDRCF
EV2: TXIS ISR: wr data2
EV3: TXIS ISR: wr data3
EV4: TXIS ISR: wr data4 (not sent)

Example I2C target transmitter 3 bytes, NOSTRETCH=1:



legend:
transmission
reception
SCL stretch

EV1: wr data1
EV2: TXIS ISR: wr data2
EV3: TXIS ISR: wr data3
EV4: TXIS ISR: wr data4 (not sent)
EV5: STOPF ISR: (optional: set TXE and TXIS), set STOPCF

MSv19853V3

Target receiver

The RXNE bit of the I2C_ISR register is set when the I2C_RXDR is full, which generates an interrupt if the RXIE bit of the I2C_CR1 register is set. RXNE is cleared when I2C_RXDR is read.

When STOP condition is received and the STOPIE bit of the I2C_CR1 register is set, the STOPF flag in the I2C_ISR register is set and an interrupt is generated.

Figure 796. Transfer sequence flow for I2C target receiver, NOSTRETCH = 0

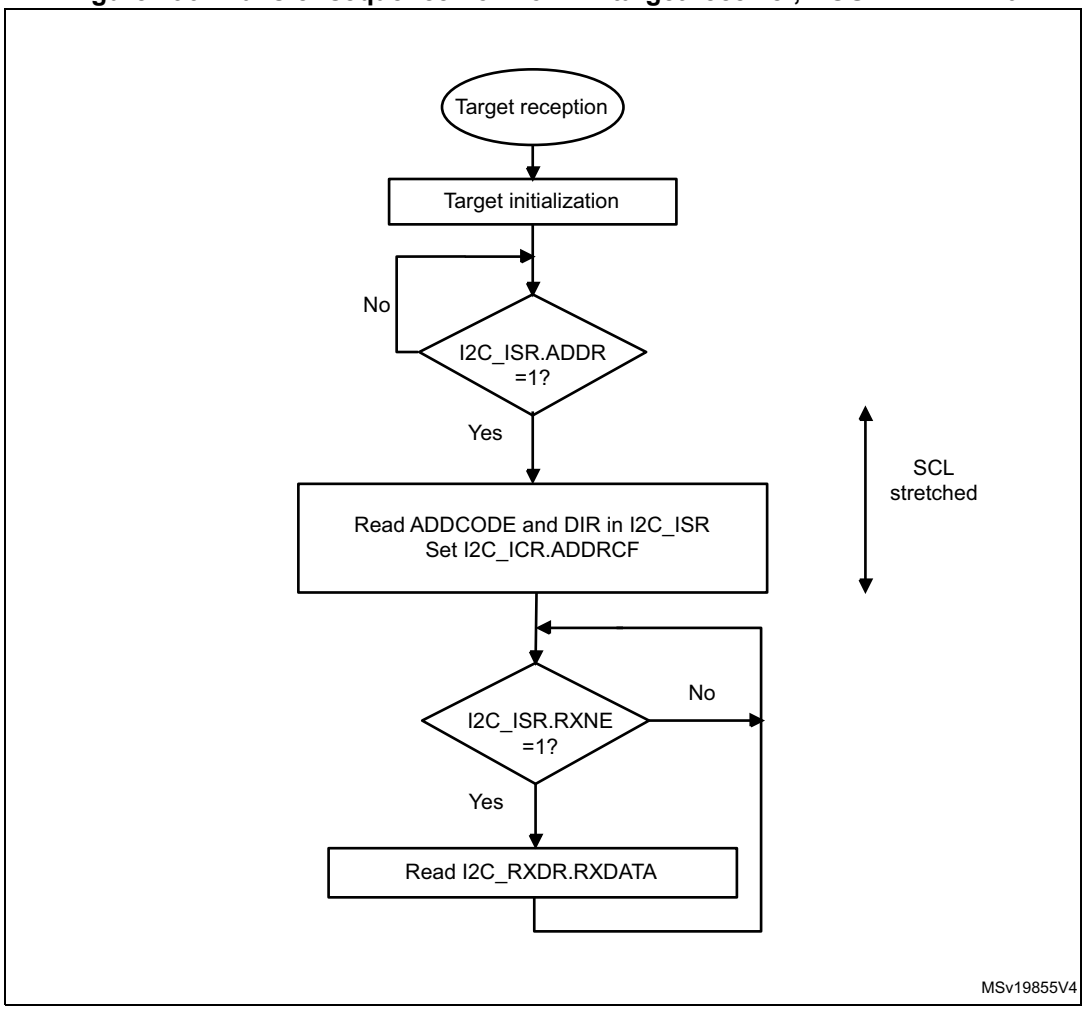


Figure 797. Transfer sequence flow for I2C target receiver, NOSTRETCH = 1

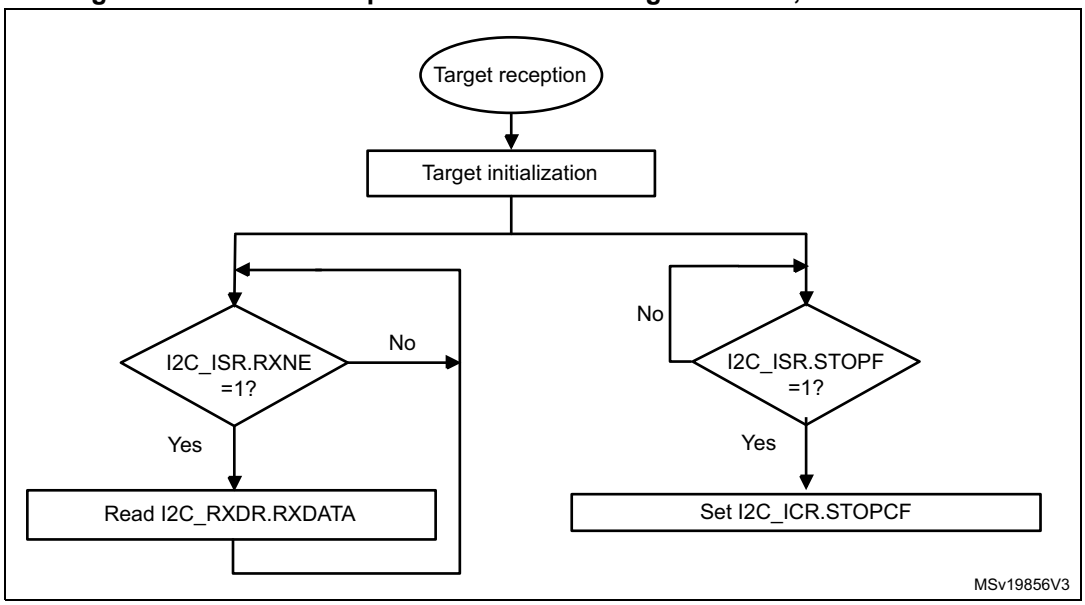
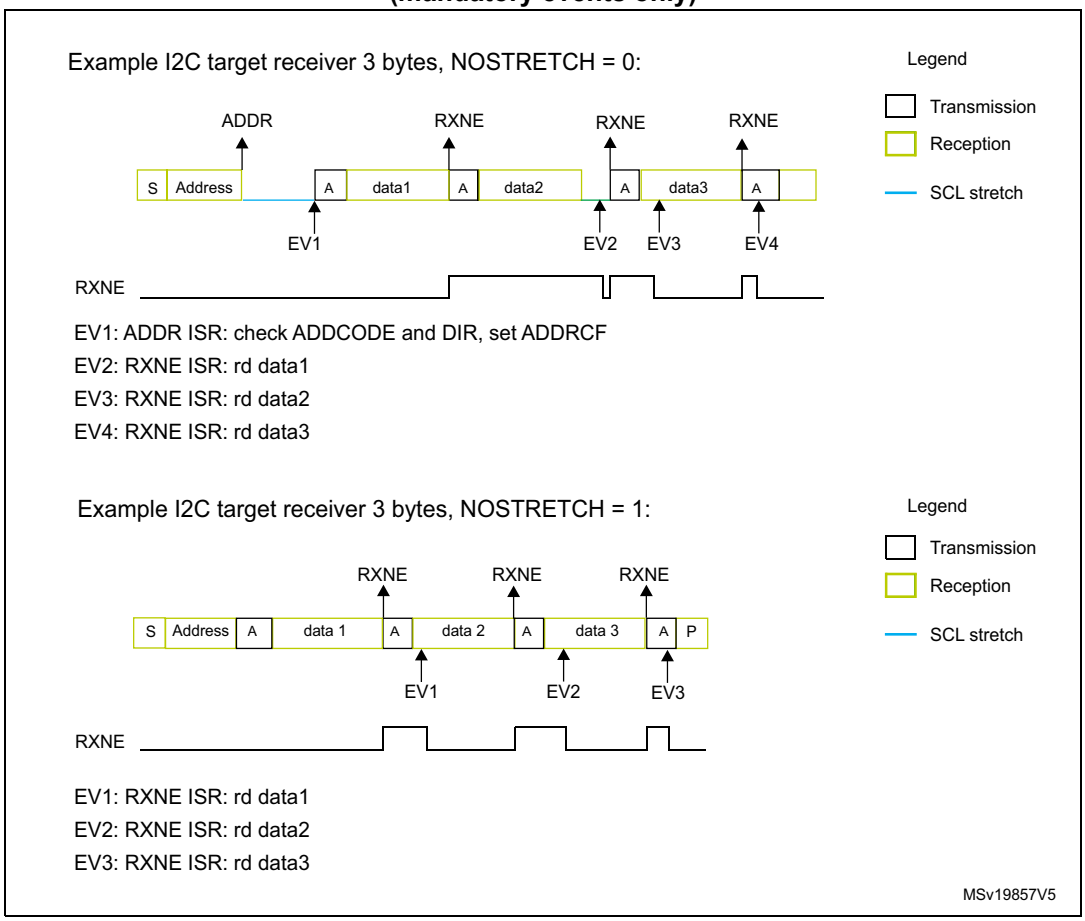


Figure 798. Transfer bus diagrams for I2C target receiver (mandatory events only)



63.4.9 I2C controller mode

I2C controller initialization

Before enabling the peripheral, the I2C controller clock must be configured, by setting the SCLH and SCLL bits in the I2C_TIMINGR register.

The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the *I2C Configuration* window.

A clock synchronization mechanism is implemented in order to support multicontroller environment and target clock stretching.

In order to allow clock synchronization:

- The low level of the clock is counted using the SCLL counter, starting from the SCL low level internal detection.
- The high level of the clock is counted using the SCLH counter, starting from the SCL high level internal detection.

I2C detects its own SCL low level after a t_{SYNC1} delay depending on the SCL falling edge, SCL input noise filters (analog and digital), and SCL synchronization to the I2CxCLK clock. I2C releases SCL to high level once the SCLL counter reaches the value programmed in the SCLL[7:0] bitfield of the I2C_TIMINGR register.

I2C detects its own SCL high level after a t_{SYNC2} delay depending on the SCL rising edge, SCL input noise filters (analog and digital), and SCL synchronization to the I2CxCLK clock. I2C ties SCL to low level once the SCLH counter reaches the value programmed in the SCLH[7:0] bitfield of the I2C_TIMINGR register.

Consequently the controller clock period is:

$t_{SCL} = t_{SYNC1} + t_{SYNC2} + \{[(SCLH+ 1) + (SCLL+ 1)] \times (PRESC+ 1) \times t_{I2CCLK}\}$

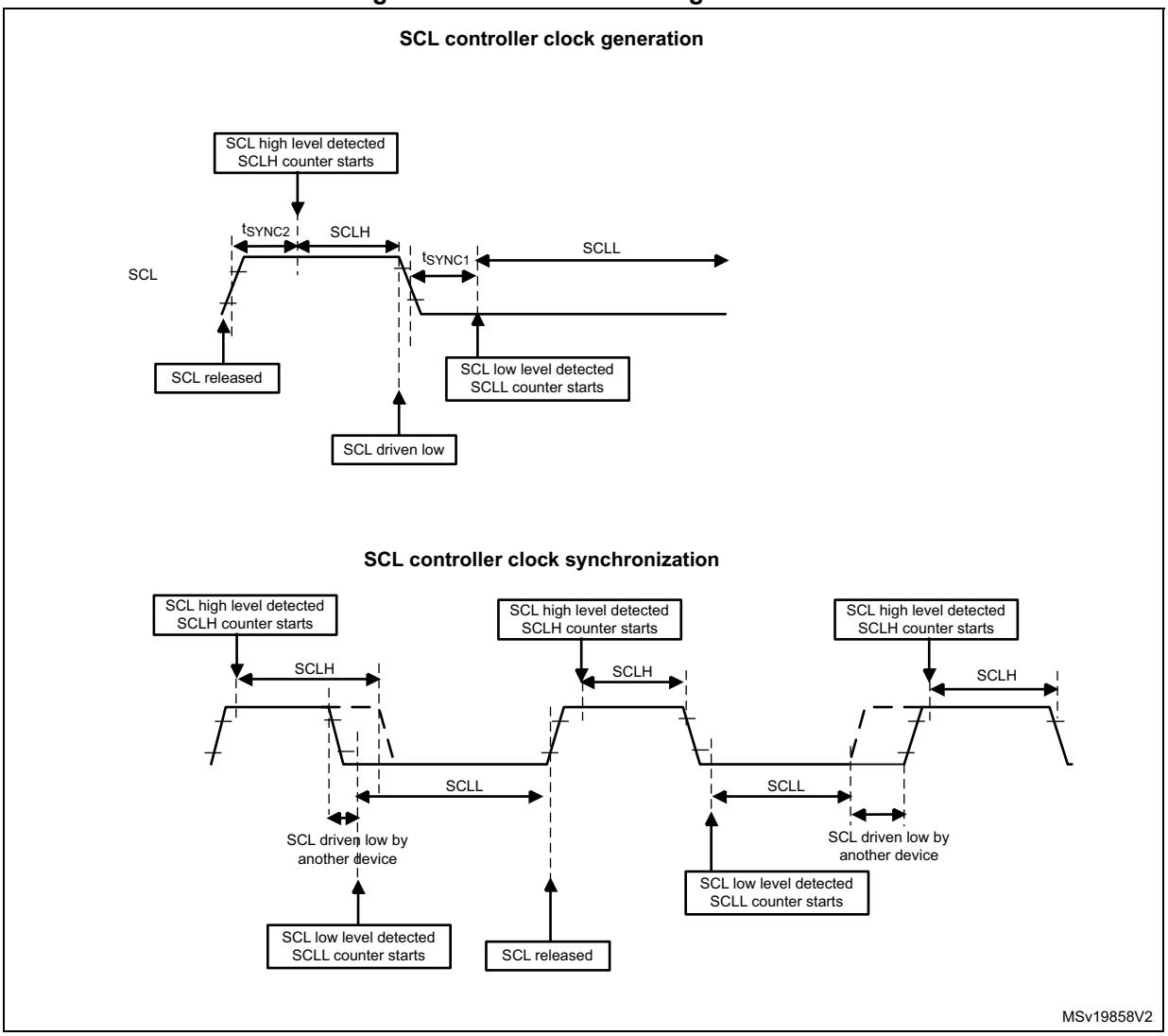
The duration of t_{SYNC1} depends upon:

- SCL falling slope
- input delay induced by the analog filter (when enabled)
- input delay induced by the digital filter (when enabled): DNF[3:0] x t_{I2CCLK}
- delay due to SCL synchronization with the i2c_ker_ck clock (two to three i2c_ker_ck periods)

The duration of t_{SYNC2} depends upon:

- SCL rising slope
- input delay induced by the analog filter (when enabled)
- input delay induced by the digital filter (when enabled): DNF[3:0] x t_{I2CCLK}
- delay due to SCL synchronization with the i2c_ker_ck clock (two to three i2c_ker_ck periods)

Figure 799. Controller clock generation



Caution: For compliance with the I²C-bus or SMBus specification, the controller clock must respect the timings in the following table.

Table 609. I²C-bus and SMBus specification clock timings

Symbol	Parameter	Standard-mode (Sm)		Fast-mode (Fm)		Fast-mode Plus (Fm+)		SMBus		Unit
		Min	Max	Min	Max	Min	Max	Min	Max	
f_{SCL}	SCL clock frequency	-	100	-	400	-	1000	-	100	kHz
$t_{HD:STA}$	Hold time (repeated) START condition	4.0	-	0.6	-	0.26	-	4.0	-	µs
$t_{SU:STA}$	Set-up time for a repeated START condition	4.7	-	0.6	-	0.26	-	4.7	-	
$t_{SU:STO}$	Set-up time for STOP condition	4.0	-	0.6	-	0.26	-	4.0	-	
t_{BUF}	Bus free time between a STOP and START condition	4.7	-	1.3	-	0.5	-	4.7	-	
t_{LOW}	Low period of the SCL clock	4.7	-	1.3	-	0.5	-	4.7	-	
t_{HIGH}	High period of the SCL clock	4.0	-	0.6	-	0.26	-	4.0	50	
t_r	Rise time of both SDA and SCL signals	-	1000	-	300	-	120	-	1000	ns
t_f	Fall time of both SDA and SCL signals	-	300	-	300	-	120	-	300	

Note: The SCLL[7:0] bitfield also determines the t_{BUF} and $t_{SU:STA}$ timings and SCLH[7:0] the $t_{HD:STA}$ and $t_{SU:STO}$ timings.

Refer to *Section 63.4.10* for examples of I2C_TIMINGR settings versus the i2c_ker_ck frequency.

Controller communication initialization (address phase)

To initiate the communication with a target to address, set the following bitfields of the I2C_CR2 register:

- ADD10: addressing mode (7-bit or 10-bit)
- SADD[9:0]: target address to send
- RD_WRN: transfer direction
- HEAD10R: in case of 10-bit address read, this bit determines whether the header only (for direction change) or the complete address sequence is sent.
- NBYTES[7:0]: the number of bytes to transfer; if equal to or greater than 255 bytes, the bitfield must initially be set to 0xFF.

Note: Changing these bitfields is not allowed as long as the START bit is set.

Before launching the communication, make sure that the I²C-bus is idle. This can be checked using the bus idle detection function or by verifying that the IDR bits of the GPIOs selected as SDA and SCL are set. Any low-level incident on the I²C-bus lines that coincides with the START condition asserted by the I2C peripheral may cause its deadlock if not filtered out by the input filters. If such incidents cannot be prevented, design the software so that it restores the normal operation of the I2C peripheral in case of a deadlock, by toggling the PE bit of the I2C_CR1 register.

To launch the communication, set the START bit of the I2C_CR2 register. The controller then automatically sends a START condition followed by the target address, either immediately if the BUSY flag is low, or t_{BUF} time after the BUSY flag transits from high to low state. The BUSY flag is set upon sending the START condition.

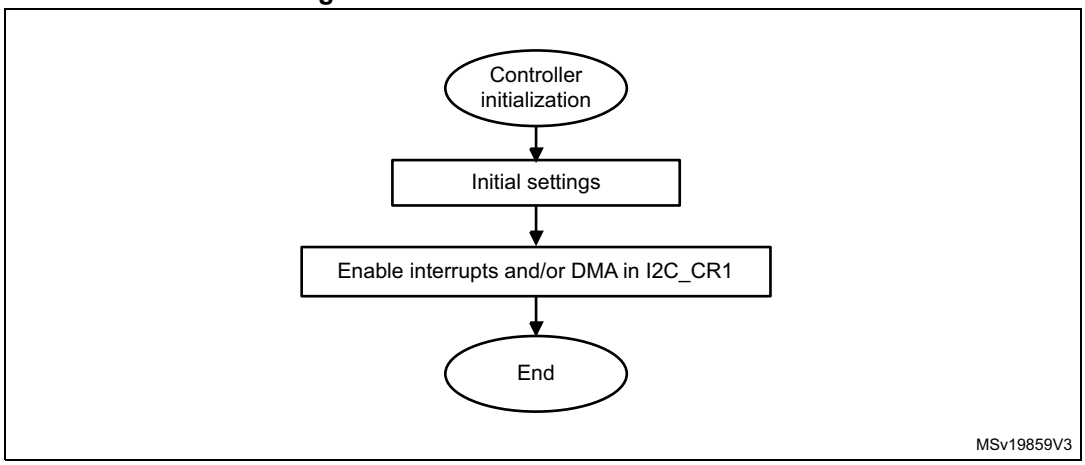
In case of an arbitration loss, the controller automatically switches back to target mode and can acknowledge its own address if it is addressed as a target.

Note: The START bit is reset by hardware when the target address is sent on the bus, whatever the received acknowledge value. The START bit is also reset by hardware upon arbitration loss.

In 10-bit addressing mode, the controller automatically keeps resending the target address in a loop until the first address byte (first seven address bits) is acknowledged by the target. Setting the ADDRCF bit makes I2C quit that loop.
If the I2C peripheral is addressed as a target (ADDR = 1) while the START bit is set, the I2C peripheral switches to target mode and the START bit is cleared.

Note: The same procedure is applied for a repeated START condition. In this case, BUSY = 1.

Figure 800. Controller initialization flow

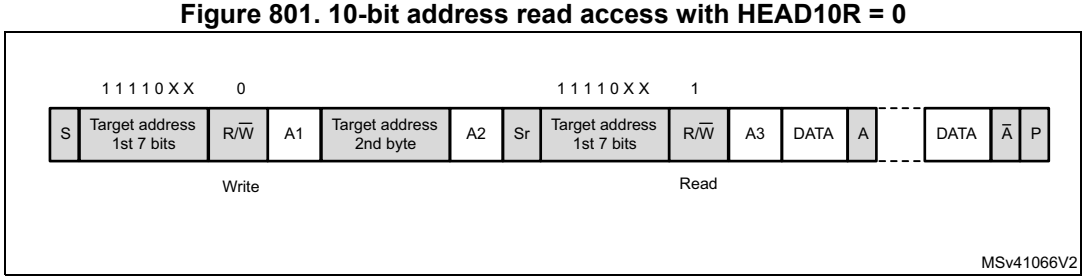


Initialization of a controller receiver addressing a 10-bit address target

If the target address is in 10-bit format, the user can choose to send the complete read sequence, by clearing the HEAD10R bit of the I2C_CR2 register. In this case, the controller automatically sends the following complete sequence after the START bit is set:

(RE)START + Target address 10-bit header Write + Target address second byte + (RE)START + Target address 10-bit header Read.

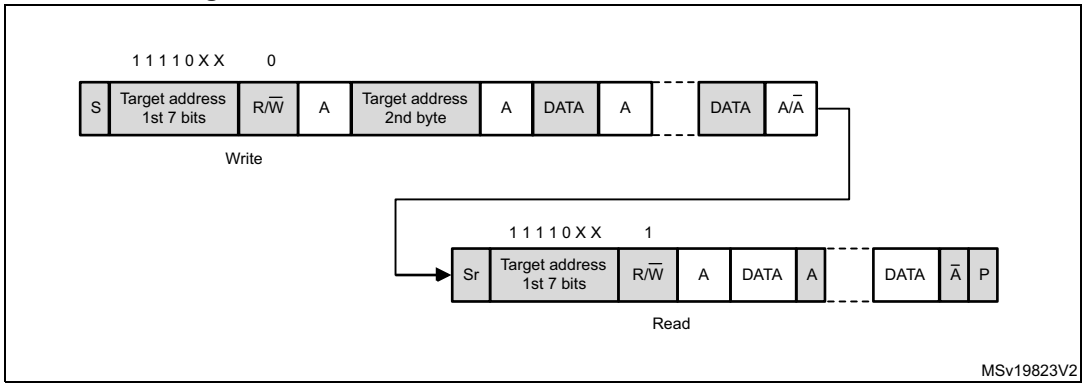
Figure 801. 10-bit address read access with HEAD10R = 0



If the controller addresses a 10-bit address target, transmits data to this target and then reads data from the same target, a controller transmission flow must be done first. Then a repeated START is set with the 10-bit target address configured with HEAD10R = 1. In this case, the controller sends this sequence:

RESTART + Target address 10-bit header Read.

Figure 802. 10-bit address read access with HEAD10R = 1



Controller transmitter

In the case of a write transfer, the TXIS flag is set after each byte transmission, after the ninth SCL pulse when an ACK is received.

A TXIS event generates an interrupt if the TXIE bit of the I2C_CR1 register is set. The flag is cleared when the I2C_TXDR register is written with the next data byte to transmit.

The number of TXIS events during the transfer corresponds to the value programmed in NBYTES[7:0]. If the total number of data bytes to transmit is greater than 255, the reload mode must be selected by setting the RELOAD bit in the I2C_CR2 register. In this case, when the NBYTES[7:0] number of data bytes is transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written with a non-zero value.

When RELOAD = 0 and the number of data bytes defined in NBYTES[7:0] is transferred:

- In automatic end mode (AUTOEND = 1), a STOP condition is automatically sent.
- In software end mode (AUTOEND = 0), the TC flag is set and the SCL line is stretched low, to perform software actions:
  - A RESTART condition can be requested by setting the START bit of the I2C_CR2 register with the proper target address configuration and the number of bytes to transfer. Setting the START bit clears the TC flag and sends the START condition on the bus.
  - A STOP condition can be requested by setting the STOP bit of the I2C_CR2 register. This clears the TC flag and sends a STOP condition on the bus.

When a NACK is received, the TXIS flag is not set and a STOP condition is automatically sent. The NACKF flag of the I2C_ISR register is set. An interrupt is generated if the NACKIE bit is set.

Figure 803. Transfer sequence flow for I2C controller transmitter, N ≤ 255 bytes

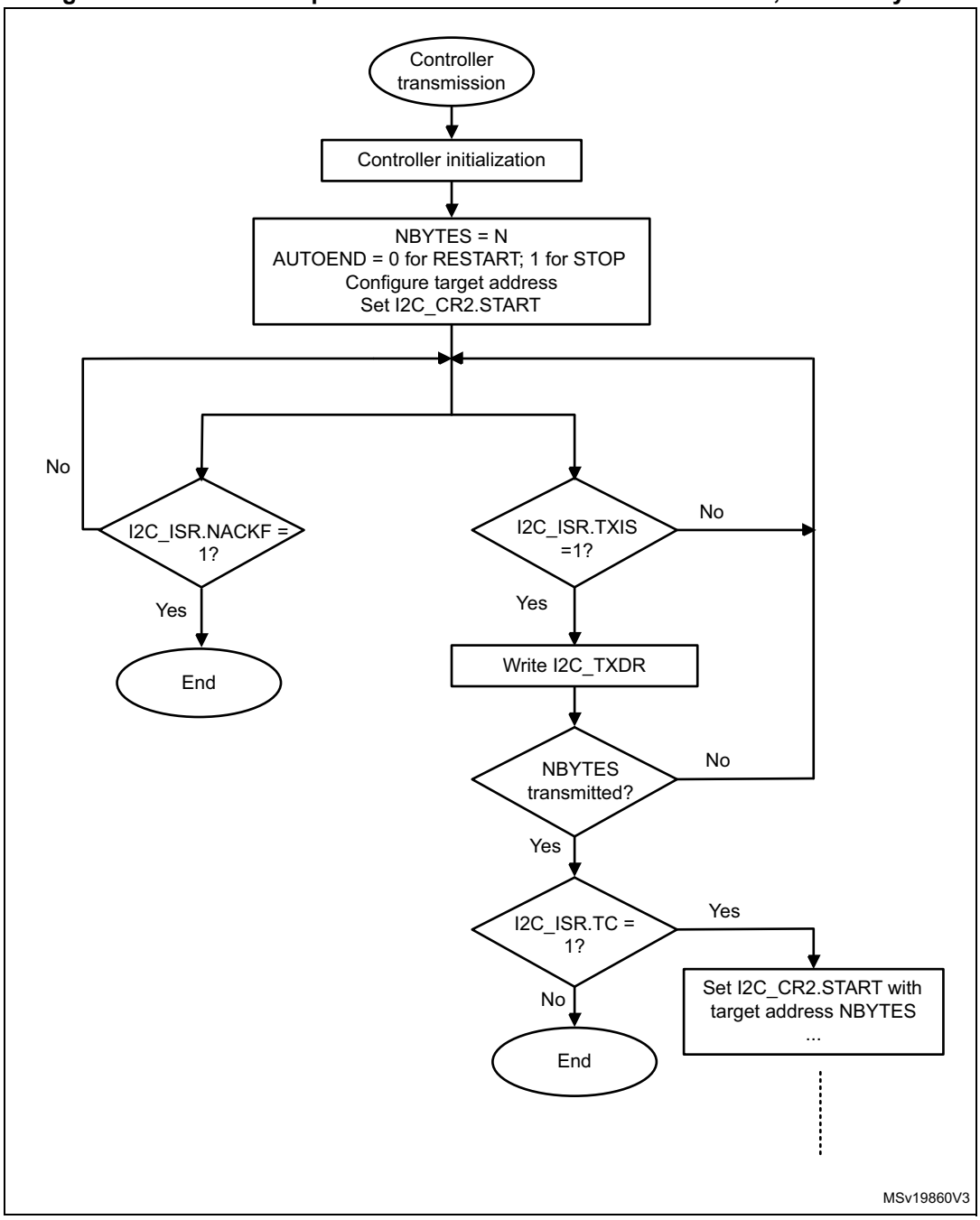


Figure 804. Transfer sequence flow for I2C controller transmitter, N > 255 bytes

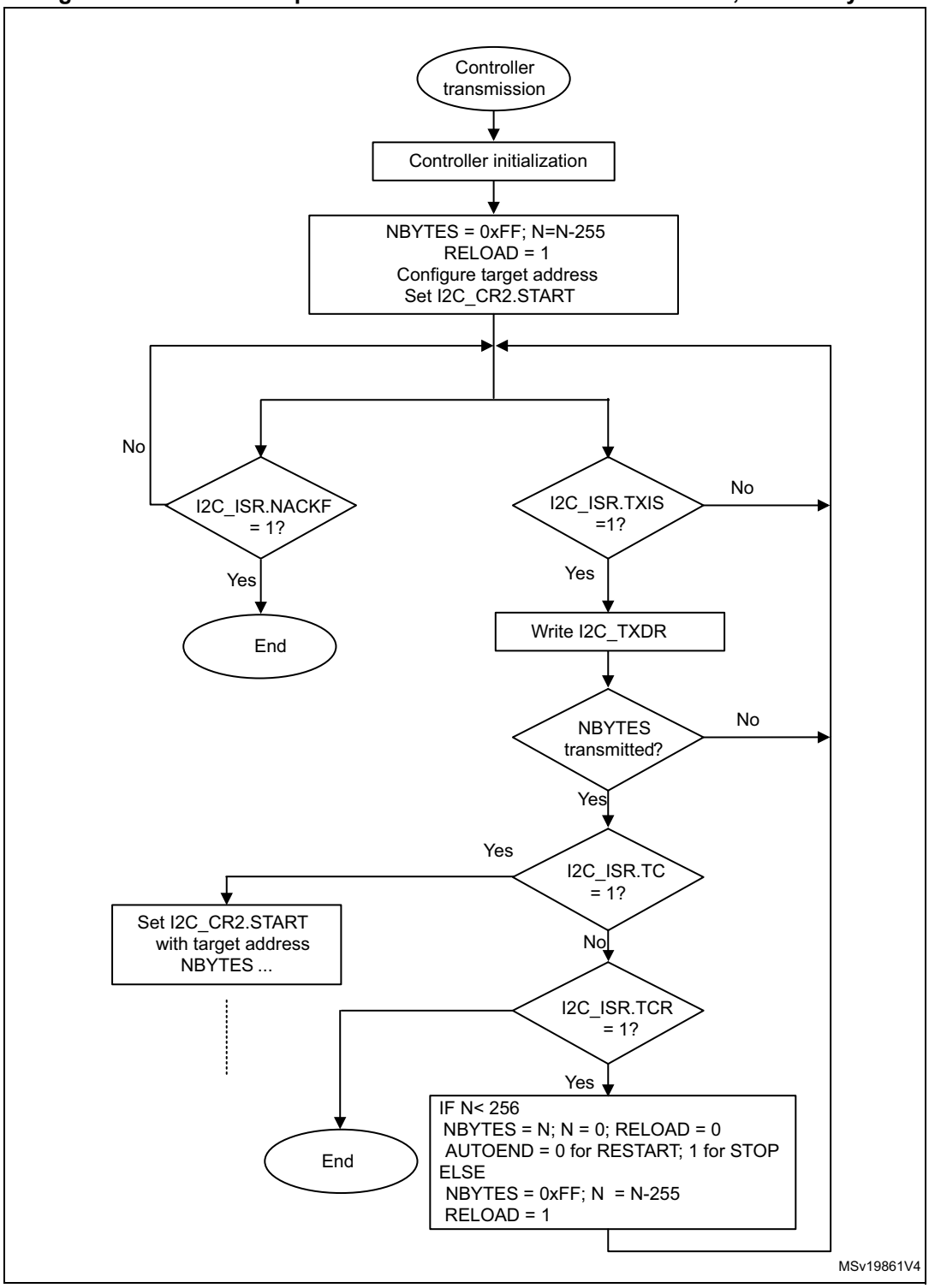
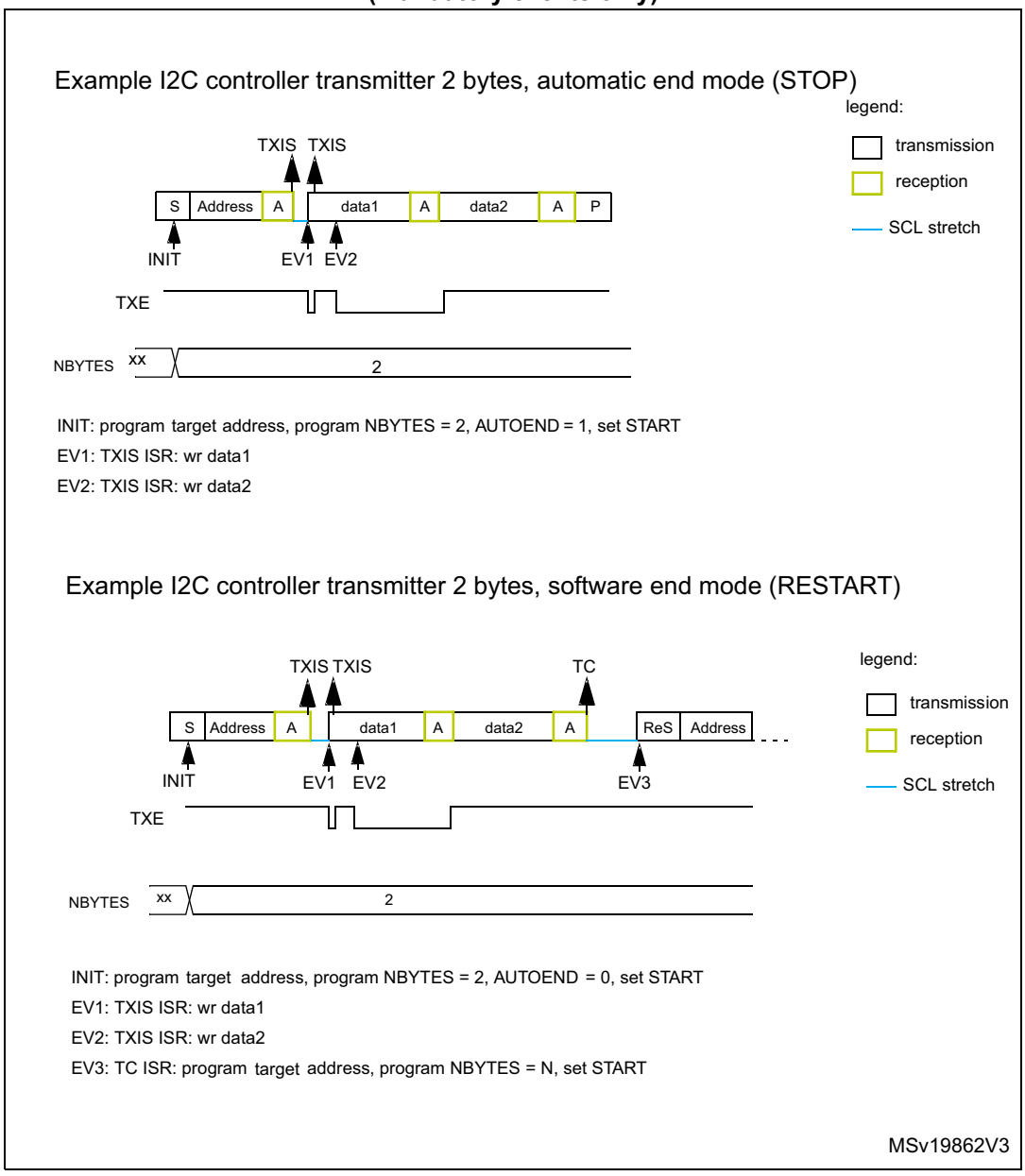


Figure 805. Transfer bus diagrams for I2C controller transmitter (mandatory events only)



Controller receiver

In the case of a read transfer, the RXNE flag is set after each byte reception, after the eighth SCL pulse. An RXNE event generates an interrupt if the RXIE bit of the I2C_CR1 register is set. The flag is cleared when I2C_RXDR is read.

If the total number of data bytes to receive is greater than 255, select the reload mode, by setting the RELOAD bit of the I2C_CR2 register. In this case, when the NBYTES[7:0] number of data bytes is transferred, the TCR flag is set and the SCL line is stretched low until NBYTES[7:0] is written with a non-zero value.

When RELOAD = 0 and he number of data bytes defined in NBYTES[7:0] is transferred:

- In automatic end mode (AUTOEND = 1), a NACK and a STOP are automatically sent after the last received byte.
- In software end mode (AUTOEND = 0), a NACK is automatically sent after the last received byte. The TC flag is set and the SCL line is stretched low in order to allow software actions:
  - A RESTART condition can be requested by setting the START bit of the I2C_CR2 register, with the proper target address configuration and the number of bytes to transfer. Setting the START bit clears the TC flag and sends the START condition and the target address on the bus.
  - A STOP condition can be requested by setting the STOP bit of the I2C_CR2 register. This clears the TC flag and sends a STOP condition on the bus.

Figure 806. Transfer sequence flow for I2C controller receiver, N ≤ 255 bytes

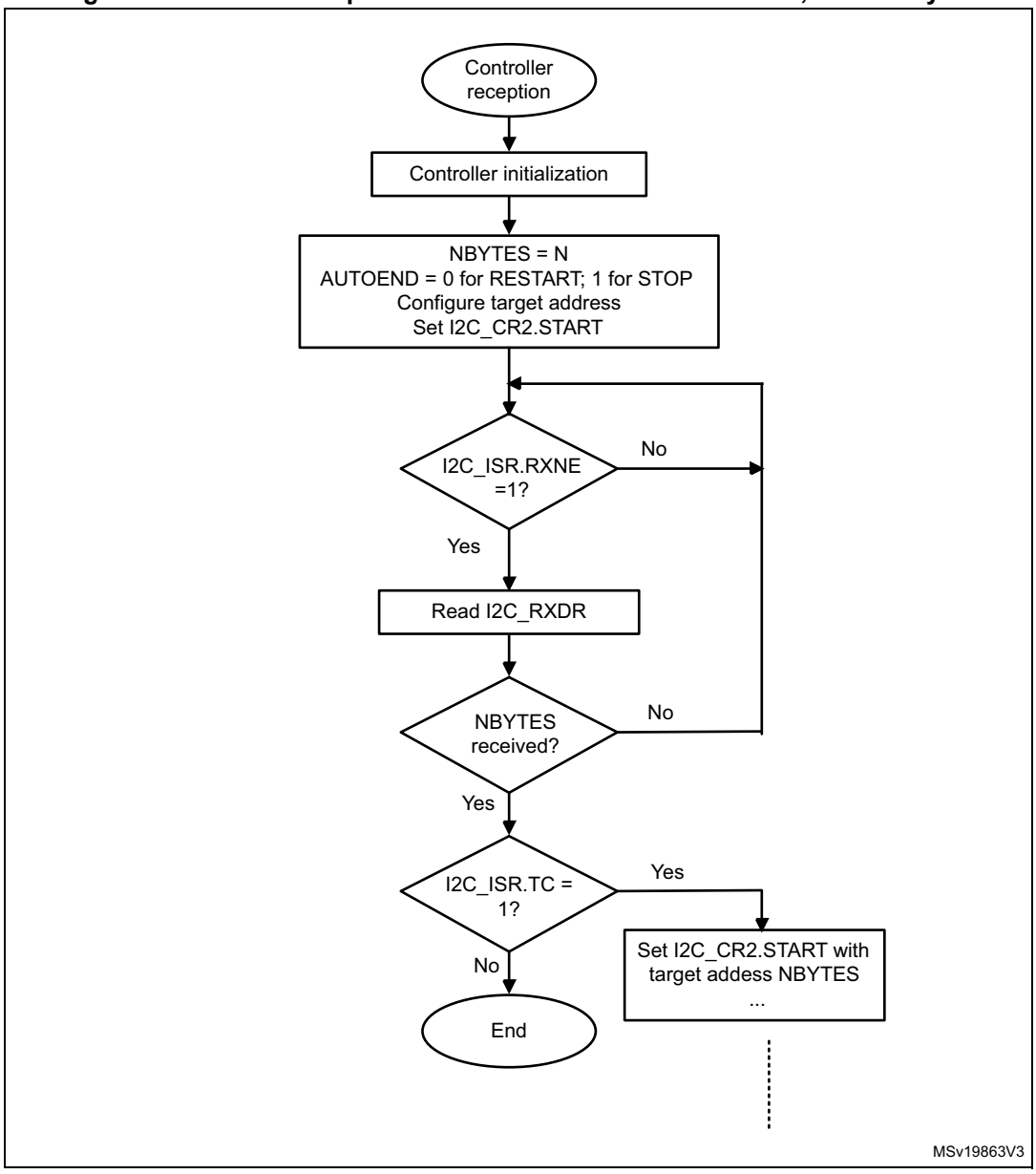


Figure 807. Transfer sequence flow for I2C controller receiver, N > 255 bytes

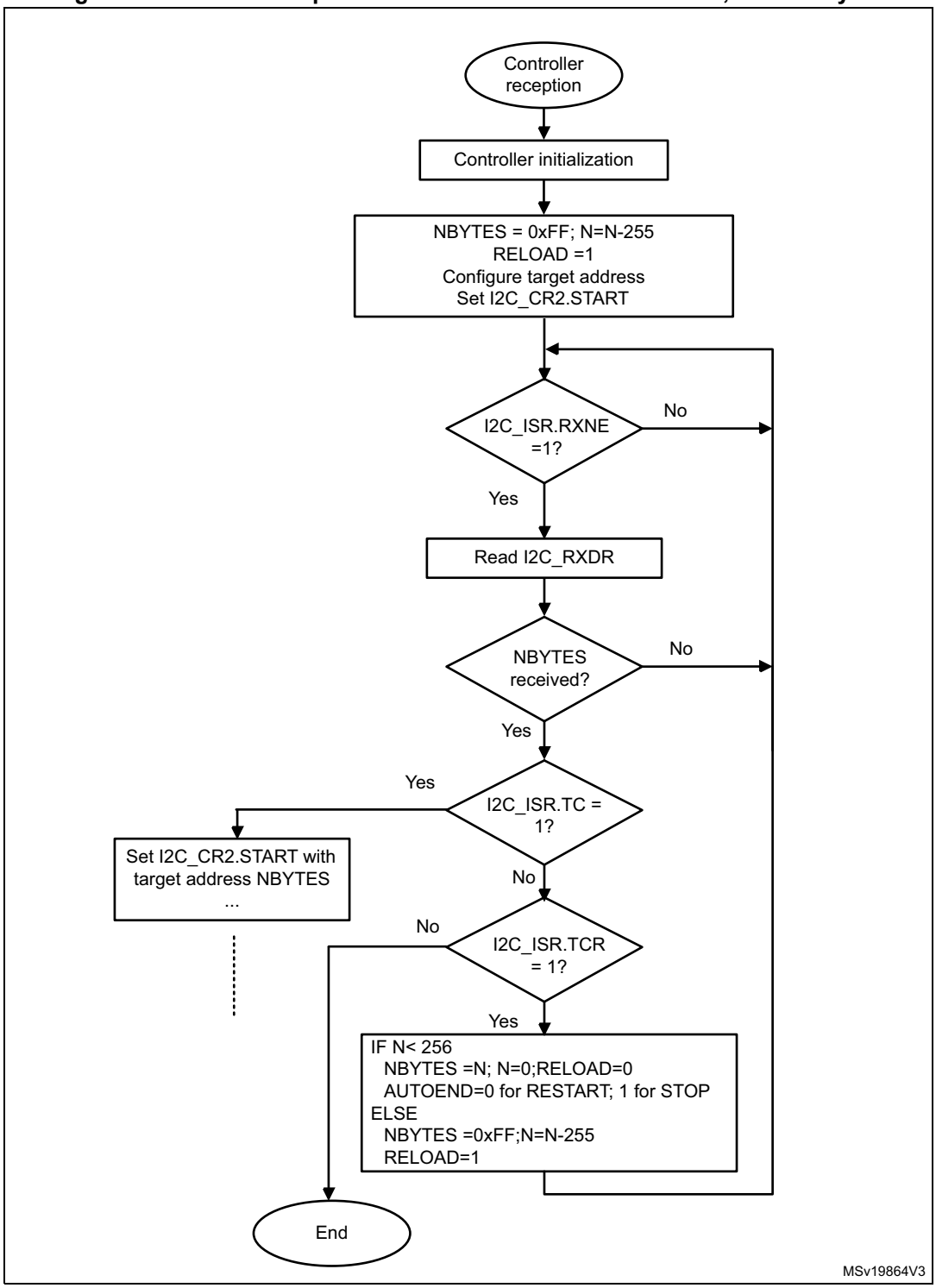
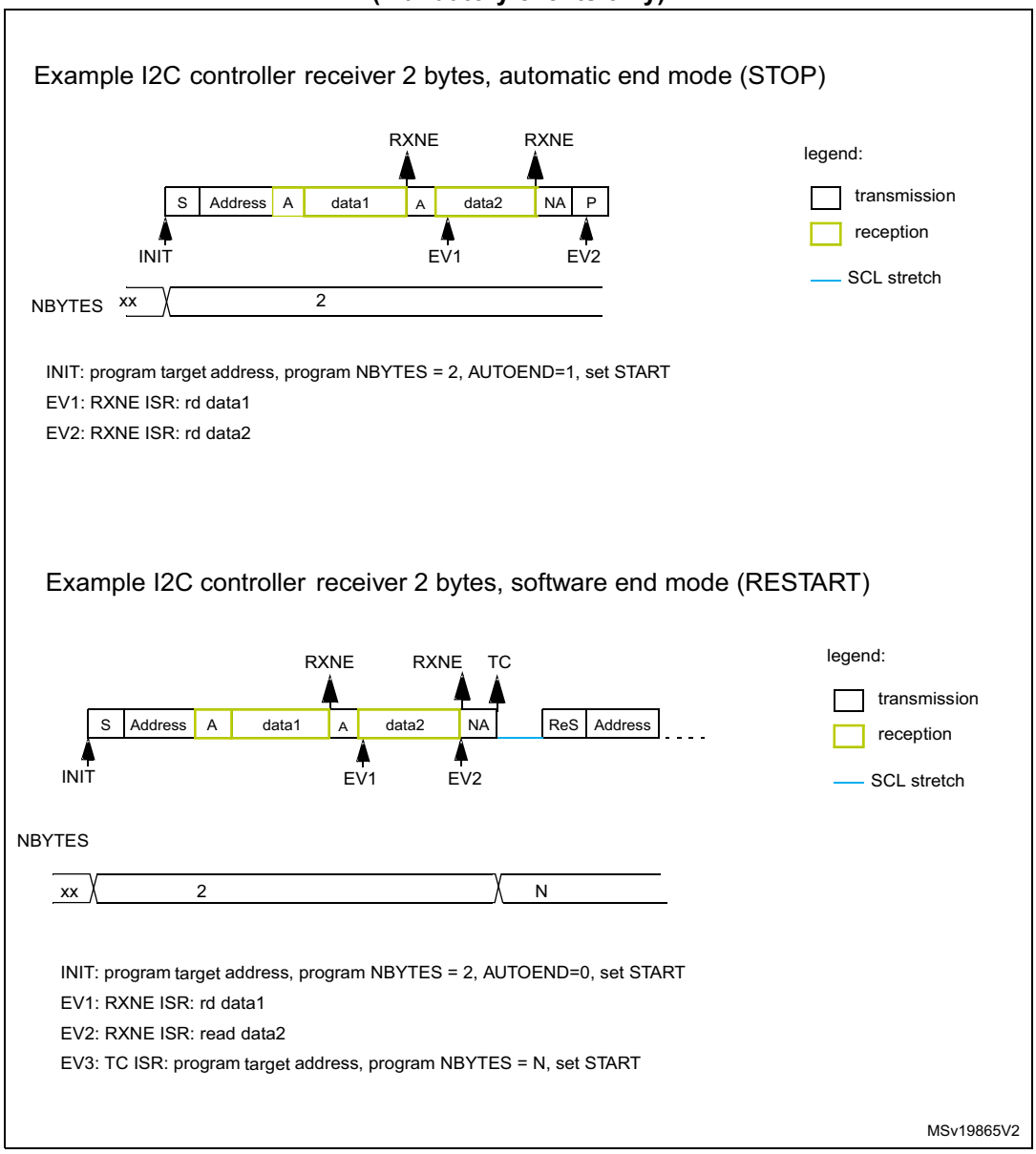


Figure 808. Transfer bus diagrams for I2C controller receiver (mandatory events only)



63.4.10 I2C_TIMINGR register configuration examples

The following tables provide examples of how to program the I2C_TIMINGR register to obtain timings compliant with the I²C-bus specification. To get more accurate configuration values, use the STM32CubeMX tool (*I2C Configuration* window).

Table 610. Timing settings for f_{I2CCLK} of 8 MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	500 kHz
PRESC[3:0]	0x1	0x1	0x0	0x0
SCLL[7:0]	0xC7	0x13	0x9	0x6
t_{SCLL}	200 x 250 ns = 50 µs	20 x 250 ns = 5.0 µs	10 x 125 ns = 1250 ns	7 x 125 ns = 875 ns
SCLH[7:0]	0xC3	0xF	0x3	0x3
t_{SCLH}	196 x 250 ns = 49 µs	16 x 250 ns = 4.0 µs	4 x 125 ns = 500 ns	4 x 125 ns = 500 ns
t_{SCL}(1)	~100 µs(2)	~10 µs(2)	~2.5 µs(3)	~2.0 µs(4)
SDADEL[3:0]	0x2	0x2	0x1	0x0
t_{SDADEL}	2 x 250 ns = 500 ns	2 x 250 ns = 500 ns	1 x 125 ns = 125 ns	0 ns
SCLDEL[3:0]	0x4	0x4	0x3	0x1
t_{SCLDEL}	5 x 250 ns = 1250 ns	5 x 250 ns = 1250 ns	4 x 125 ns = 500 ns	2 x 125 ns = 250 ns

1. t_{SCL} is greater than t_{SCLL} + t_{SCLH} due to SCL internal detection delay. Values provided for t_{SCL} are examples only.
2. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 500 ns. Example with t_{SYNC1} + t_{SYNC2} = 1000 ns.
3. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 500 ns. Example with t_{SYNC1} + t_{SYNC2} = 750 ns.
4. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 500 ns. Example with t_{SYNC1} + t_{SYNC2} = 655 ns.

Table 611. Timing settings for f_{I2CCLK} of 16 MHz

Parameter	Standard-mode (Sm)		Fast-mode (Fm)	Fast-mode Plus (Fm+)
	10 kHz	100 kHz	400 kHz	1000 kHz
PRESC[3:0]	0x3	0x3	0x1	0x0
SCLL[7:0]	0xC7	0x13	0x9	0x4
t_{SCLL}	200 x 250 ns = 50 µs	20 x 250 ns = 5.0 µs	10 x 125 ns = 1250 ns	5 x 62.5 ns = 312.5 ns
SCLH[7:0]	0xC3	0xF	0x3	0x2
t_{SCLH}	196 x 250 ns = 49 µs	16 x 250 ns = 4.0 µs	4 x 125 ns = 500 ns	3 x 62.5 ns = 187.5 ns
t_{SCL}(1)	~100 µs(2)	~10 µs(2)	~2.5 µs(3)	~1.0 µs(4)
SDADEL[3:0]	0x2	0x2	0x2	0x0
t_{SDADEL}	2 x 250 ns = 500 ns	2 x 250 ns = 500 ns	2 x 125 ns = 250 ns	0 ns
SCLDEL[3:0]	0x4	0x4	0x3	0x2
t_{SCLDEL}	5 x 250 ns = 1250 ns	5 x 250 ns = 1250 ns	4 x 125 ns = 500 ns	3 x 62.5 ns = 187.5 ns

1. t_{SCL} is greater than t_{SCLL} + t_{SCLH} due to SCL internal detection delay. Values provided for t_{SCL} are examples only.
2. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 250 ns. Example with t_{SYNC1} + t_{SYNC2} = 1000 ns.
3. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 250 ns. Example with t_{SYNC1} + t_{SYNC2} = 750 ns.
4. t_{SYNC1} + t_{SYNC2} minimum value is 4 x t_{I2CCLK} = 250 ns. Example with t_{SYNC1} + t_{SYNC2} = 500 ns.

63.4.11 SMBus specific features

Introduction

The system management bus (SMBus) is a two-wire interface through which various devices can communicate with each other and with the rest of the system. It is based on operation principles of the I²C-bus. The SMBus provides a control bus for system and power management related tasks.

The I2C peripheral is compatible with the SMBus specification (http://smbus.org).

The system management bus specification refers to three types of devices:

- **Target** is a device that receives or responds to a command.
- **Controller** is a device that issues commands, generates clocks, and terminates the transfer.
- **Host** is a specialized controller that provides the main interface to the system CPU. A host must be a controller-target and must support the SMBus *host notify* protocol. Only one host is allowed in a system.

The I2C peripheral can be configured as a controller or a target device, and also as a host.

Bus protocols

There are eleven possible command protocols for any given device. The device can use any or all of them to communicate. These are: *Quick Command*, *Send Byte*, *Receive Byte*, *Write Byte*, *Write Word*, *Read Byte*, *Read Word*, *Process Call*, *Block Read*, *Block Write*, and *Block Write-Block Read Process Call*. The protocols must be implemented by the user software.

For more details on these protocols, refer to the SMBus specification (http://smbus.org).

STM32CubeMX implements an SMBus stack thanks to X-CUBE-SMBUS, a downloadable software pack that allows basic SMBus configuration per I2C instance.

Address resolution protocol (ARP)

SMBus target address conflicts can be resolved by dynamically assigning a new unique address to each target device. To provide a mechanism to isolate each device for the purpose of address assignment, each device must implement a unique 128-bit device identifier (UDID). In the I2C peripheral, it is implemented by software.

The I2C peripheral supports the Address resolution protocol (ARP). The SMBus device default address (0b1100 001) is enabled by setting the SMBDEN bit of the I2C_CR1 register. The ARP commands must be implemented by the user software.

Arbitration is also performed in target mode for ARP support.

For more details on the SMBus address resolution protocol, refer to the SMBus specification (http://smbus.org).

Received command and data acknowledge control

An SMBus receiver must be able to NACK each received command or data. In order to allow the ACK control in target mode, the target byte control mode must be enabled, by setting the SBC bit of the I2C_CR1 register. Refer to *Target byte control mode* for more details.

Host notify protocol

To enable the host notify protocol, set the SMBHEN bit of the I2C_CR1 register. The I2C peripheral then acknowledges the SMBus host address (0b0001 000).

When this protocol is used, the device acts as a controller and the host as a target.

SMBus alert

The I2C peripheral supports the SMBALERT# optional signal through the SMBA pin. With the SMBALERT# signal, an SMBus target device can signal to the SMBus host that it wants to talk. The host processes the interrupt and simultaneously accesses all SMBALERT# devices through the alert response address (0b0001 100). Only the device/devices which pulled SMBALERT# low acknowledges/acknowledge the alert response address.

When the I2C peripheral is configured as an SMBus target device (SMBHEN = 0), the SMBA pin is pulled low by setting the ALERTEN bit of the I2C_CR1 register. The alert response address is enabled at the same time.

When the I2C peripheral is configured as an SMBus host (SMBHEN = 1), the ALERT flag of the I2C_ISR register is set when a falling edge is detected on the SMBA pin and ALERTEN = 1. An interrupt is generated if the ERRIE bit of the I2C_CR1 register is set. When ALERTEN = 0, the alert line is considered high even if the external SMBA pin is low.

Note: If the SMBus alert pin is not required, keep the ALERTEN bit cleared. The SMBA pin can then be used as a standard GPIO.

Packet error checking

A packet error checking mechanism introduced in the SMBus specification improves reliability and communication robustness. The packet error checking is implemented by appending a packet error code (PEC) at the end of each message transfer. The PEC is calculated by using the $C(x) = x^8 + x^2 + x + 1$ CRC-8 polynomial on all the message bytes (including addresses and read/write bits).

The I2C peripheral embeds a hardware PEC calculator and allows a not acknowledge to be sent automatically when the received byte does not match the hardware calculated PEC.

Timeouts

To comply with the SMBus timeout specifications, the I2C peripheral embeds hardware timers.

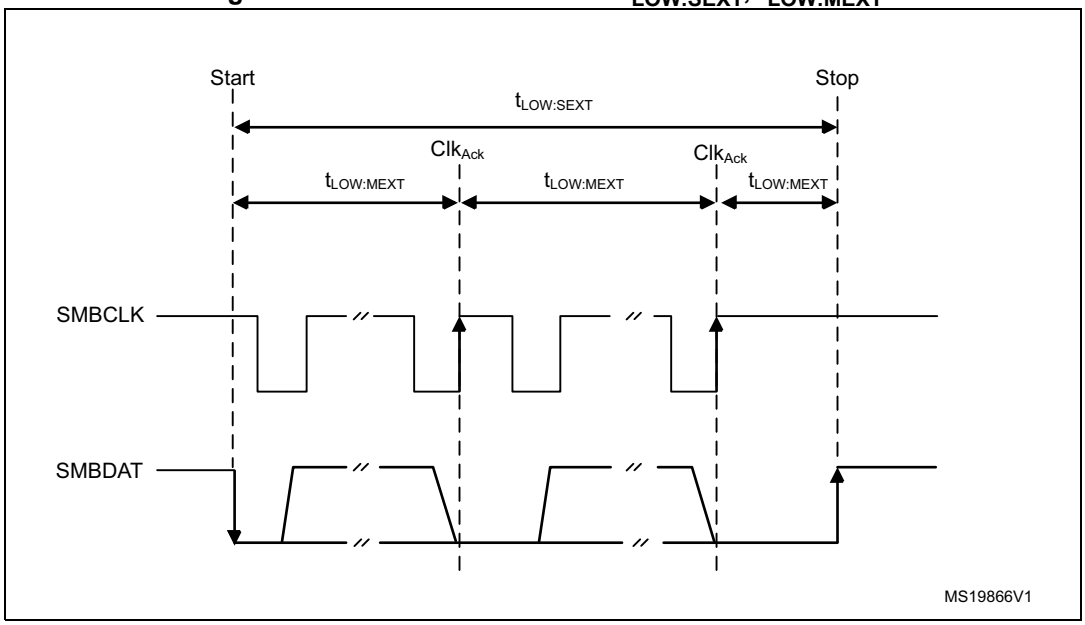
Table 612. SMBus timeout specifications

Symbol	Parameter	Limits		Unit
		Min	Max	
$t_{TIMEOUT}$	Detect clock low timeout	25	35	ms
$t_{LOW:SEXT}$(1)	Cumulative clock low extend time (target device)	-	25	
$t_{LOW:MEXT}$(2)	Cumulative clock low extend time (controller device)	-	10	

1. $t_{LOW:SEXT}$ is the cumulative time a given target device is allowed to extend the clock cycles in one message from the initial START to the STOP. It is possible that another target device or the controller also extends the clock causing the combined clock low extend time to be greater than $t_{LOW:SEXT}$. The value provided applies to a single target device connected to a full-target controller.

2. $t_{LOW:MEXT}$ is the cumulative time a controller device is allowed to extend its clock cycles within each byte of a message as defined from START-to-ACK, ACK-to-ACK, or ACK-to-STOP. It is possible that a target device or another controller also extends the clock, causing the combined clock low time to be greater than $t_{LOW:MEXT}$ on a given byte. The value provided applies to a single target device connected to a full-target controller.

Figure 809. Timeout intervals for $t_{LOW:SEXT}$, $t_{LOW:MEXT}$



Bus idle detection

A controller can assume that the bus is free if it detects that the clock and data signals have been high for $t_{IDLE} > t_{HIGH}$(max) (refer to the table in *Section 63.4.9*).

This timing parameter covers the condition where a controller is dynamically added to the bus, and may not have detected a state transition on the SMBCLK or SMBDAT lines. In this case, the controller must wait long enough to ensure that a transfer is not currently in progress. The I2C peripheral supports a hardware bus idle detection.

63.4.12 SMBus initialization

In addition to the I2C initialization for the I²C-bus, the use of the peripheral for the SMBus communication requires some extra initialization steps.

Received command and data acknowledge control (target mode)

An SMBus receiver must be able to NACK each received command or data. To allow ACK control in target mode, the target byte control mode must be enabled, by setting the SBC bit of the I2C_CR1 register. Refer to *Target byte control mode* for more details.

Specific addresses (target mode)

The specific SMBus addresses must be enabled if required. Refer to *Bus idle detection* for more details.

The SMBus device default address (0b1100 001) is enabled by setting the SMBDEN bit of the I2C_CR1 register.

The SMBus host address (0b0001 000) is enabled by setting the SMBHEN bit of the I2C_CR1 register.

The alert response address (0b0001100) is enabled by setting the ALERTEN bit of the I2C_CR1 register.

Packet error checking

PEC calculation is enabled by setting the PECEN bit of the I2C_CR1 register. Then the PEC transfer is managed with the help of the hardware byte counter associated with the NBYTES[7:0] bitfield of the I2C_CR2 register. The PECEN bit must be configured before enabling the I2C.

The PEC transfer is managed with the hardware byte counter, so the SBC bit must be set when interfacing the SMBus in target mode. The PEC is transferred after transferring NBYTES[7:0] - 1 data bytes, if the PECBYTE bit is set and the RELOAD bit is cleared. If RELOAD is set, PECBYTE has no effect.

Caution: Changing the PECEN configuration is not allowed when the I2C peripheral is enabled.

Table 613. SMBus with PEC configuration

Mode	SBC bit	RELOAD bit	AUTOEND bit	PECBYTE bit
Controller Tx/Rx NBYTES + PEC+ STOP	X	0	1	1
Controller Tx/Rx NBYTES + PEC + ReSTART	X	0	0	1
Target Tx/Rx with PEC	1	0	X	1

Timeout detection

The timeout detection is enabled by setting the TIMOUTEN and TEXTEN bits of the I2C_TIMEOUTR register. The timers must be programmed in such a way that they detect a timeout before the maximum time given in the SMBus specification.

$t_{TIMEOUT}$ check

To check the $t_{TIMEOUT}$ parameter, load the 12-bit TIMEOUTA[11:0] bitfield with the timer reload value. Keep the TIDLE bit at 0 to detect the SCL low level timeout.

Then set the TIMOUTEN bit of the I2C_TIMEOUTR register, to enable the timer.

If SCL is tied low for longer than the (TIMEOUTA + 1) x 2048 x t_{I2CCLK} period, the TIMEOUT flag of the I2C_ISR register is set.

Refer to *Table 614*.

Caution: Changing the TIMEOUTA[11:0] bitfield and the TIDLE bit values is not allowed when the TIMEOUTEN bit is set.

$t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ check

A 12-bit timer associated with the TIMEOUTB[11:0] bitfield allows checking $t_{LOW:SEXT}$ for the I2C peripheral operating as a target, or $t_{LOW:MEXT}$ when it operates as a controller. As the standard only specifies a maximum, the user can choose the same value for both. The timer is then enabled by setting the TEXTEN bit in the I2C_TIMEOUTR register.

If the SMBus peripheral performs a cumulative SCL stretch for longer than the (TIMEOUTB + 1) x 2048 x t_{I2CCLK} period, and within the timeout interval described in *Bus idle detection* section, the TIMEOUT flag of the I2C_ISR register is set.

Refer to *Table 615*.

Caution: Changing the TIMEOUTB[11:0] bitfield value is not allowed when the TEXTEN bit is set.

Bus idle detection

To check the t_{IDLE} period, the TIMEOUTA[11:0] bitfield associated with 12-bit timer must be loaded with the timer reload value. Keep the TIDLE bit at 1 to detect both SCL and SDA high level timeout. Then set the TIMOUTEN bit of the I2C_TIMEOUTR register to enable the timer.

If both the SCL and SDA lines remain high for longer than the (TIMEOUTA + 1) x 4 x t_{I2CCLK} period, the TIMEOUT flag of the I2C_ISR register is set.

Refer to *Table 616*.

Caution: Changing the TIMEOUTA[11:0] bitfield and the TIDLE bit values is not allowed when the TIMEOUTEN bit is set.

63.4.13 SMBus I2C_TIMEOUTR register configuration examples

The following tables provide examples of settings to reach desired $t_{TIMEOUT}$, $t_{LOW:SEXT}$, $t_{LOW:MEXT}$, and t_{IDLE} timings at different f_{I2CCLK} frequencies.

Table 614. TIMEOUTA[11:0] for maximum $t_{TIMEOUT}$ of 25 ms

f_{I2CCLK}	TIMEOUTA[11:0]	TIDLE	TIMEOUTEN	$t_{TIMEOUT}$
8 MHz	0x61	0	1	98 x 2048 x 125 ns = 25 ms
16 MHz	0xC3	0	1	196 x 2048 x 62.5 ns = 25 ms

Table 615. TIMEOUTB[11:0] for maximum $t_{LOW:SEXT}$ and $t_{LOW:MEXT}$ of 8 ms

f_{I2CCLK}	TIMEOUTB[11:0]	TEXTEN	$t_{LOW:SEXT}$ $t_{LOW:MEXT}$
8 MHz	0x1F	1	32 x 2048 x 125 ns = 8 ms
16 MHz	0x3F	1	64 x 2048 x 62.5 ns = 8 ms

Table 616. TIMEOUTA[11:0] for maximum t_{IDLE} of 50 µs

f_{I2CCLK}	TIMEOUTA[11:0]	TIDLE	TIMEOUTEN	t_{IDLE}
8 MHz	0x63	1	1	100 x 4 x 125 ns = 50 µs
16 MHz	0xC7	1	1	200 x 4 x 62.5 ns = 50 µs

63.4.14 SMBus target mode

In addition to I2C target transfer management (refer to *Section 63.4.8: I2C target mode*), this section provides extra software flowcharts to support SMBus.

SMBus target transmitter

When using the I2C peripheral in SMBus mode, set the SBC bit to enable the PEC transmission at the end of the programmed number of data bytes. When the PECBYTE bit is set, the number of bytes programmed in NBYTES[7:0] includes the PEC transmission. In that case, the total number of TXIS interrupts is NBYTES[7:0] - 1, and the content of the I2C_PECR register is automatically transmitted if the controller requests an extra byte after the transfer of the NBYTES[7:0] - 1 data bytes.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 810. Transfer sequence flow for SMBus target transmitter N bytes + PEC

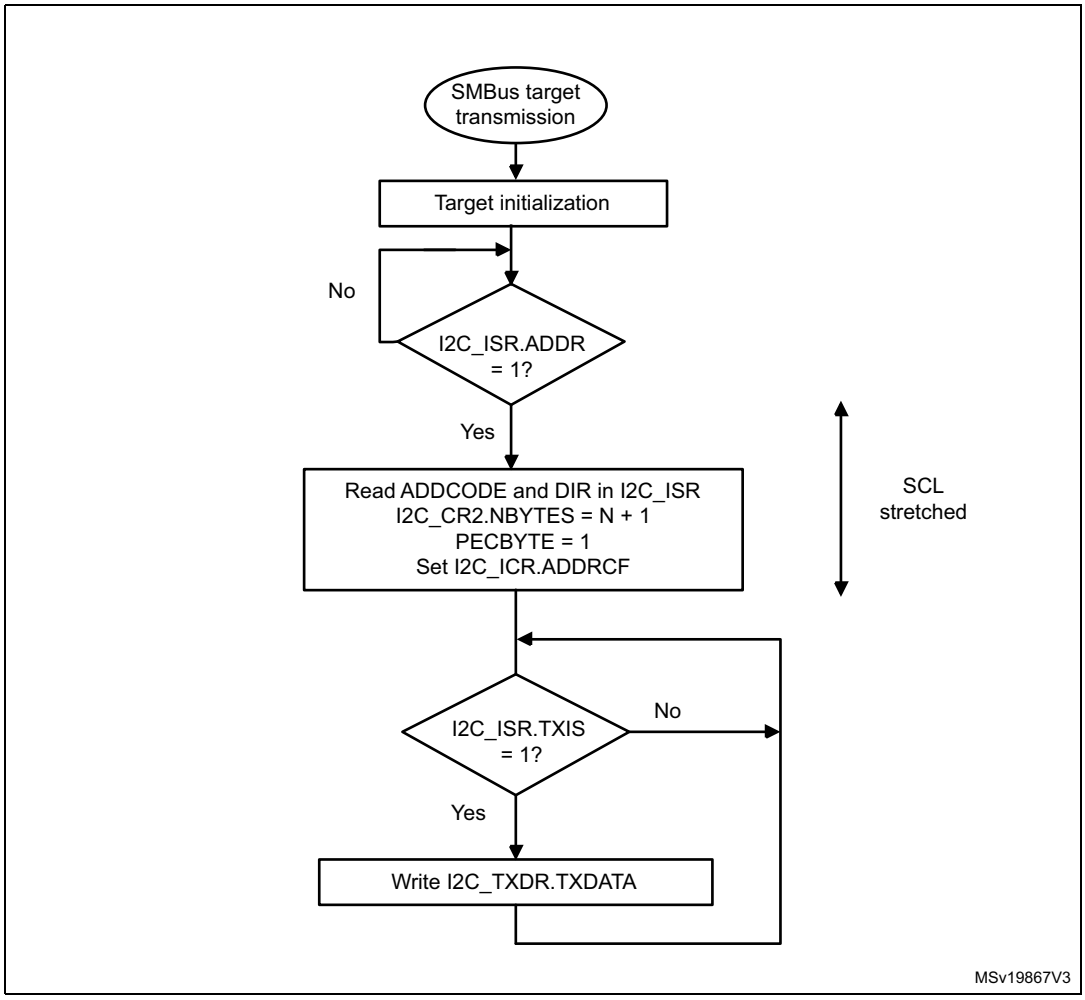
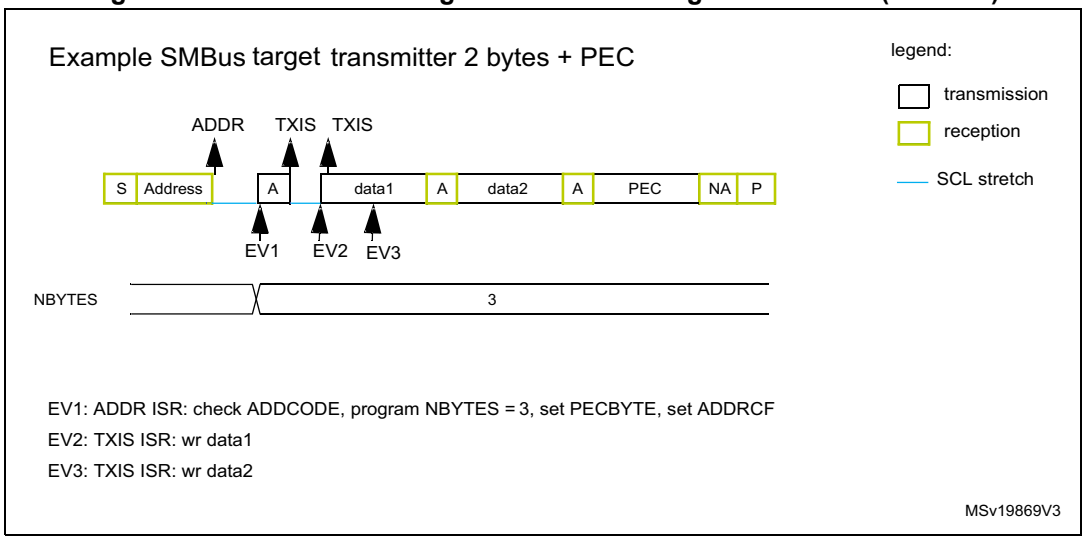


Figure 811. Transfer bus diagram for SMBus target transmitter (SBC = 1)



SMBus target receiver

When using the I2C peripheral in SMBus mode, set the SBC bit to enable the PEC checking at the end of the programmed number of data bytes. To allow the ACK control of each byte, the reload mode must be selected (RELOAD = 1). Refer to *Target byte control mode* for more details.

To check the PEC byte, the RELOAD bit must be cleared and the PECBYTE bit must be set. In this case, after the receipt of NBYTES[7:0] - 1 data bytes, the next received byte is compared with the internal I2C_PECR register content. A NACK is automatically generated if the comparison does not match, and an ACK is automatically generated if the comparison matches, whatever the ACK bit value. Once the PEC byte is received, it is copied into the I2C_RXDR register like any other data, and the RXNE flag is set.

Upon a PEC mismatch, the PECERR flag is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

If no ACK software control is required, the user can set the PECBYTE bit and, in the same write operation, load NBYTES[7:0] with the number of bytes to receive in a continuous flow. After the receipt of NBYTES[7:0] - 1 bytes, the next received byte is checked as being the PEC.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 812. Transfer sequence flow for SMBus target receiver N bytes + PEC

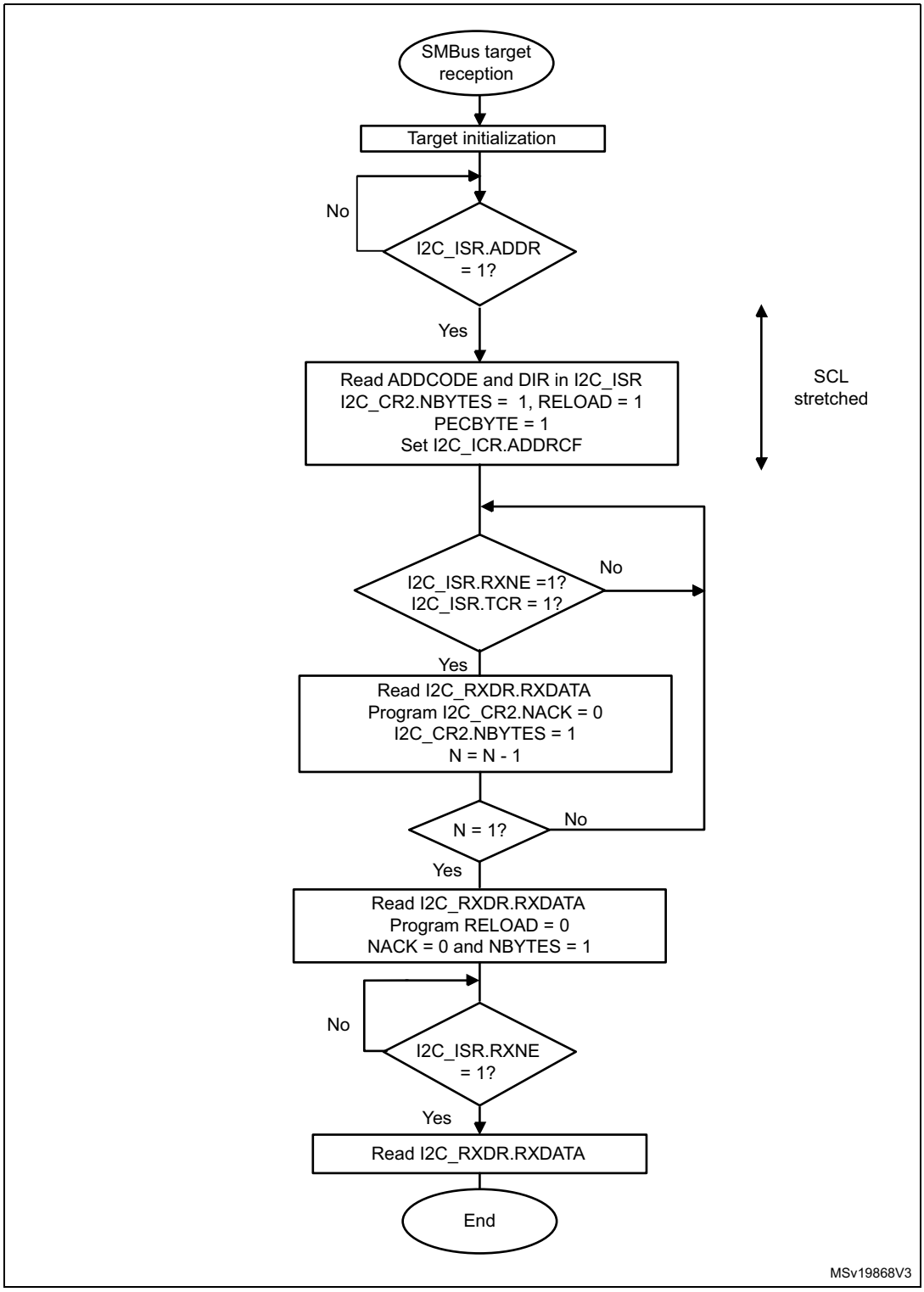
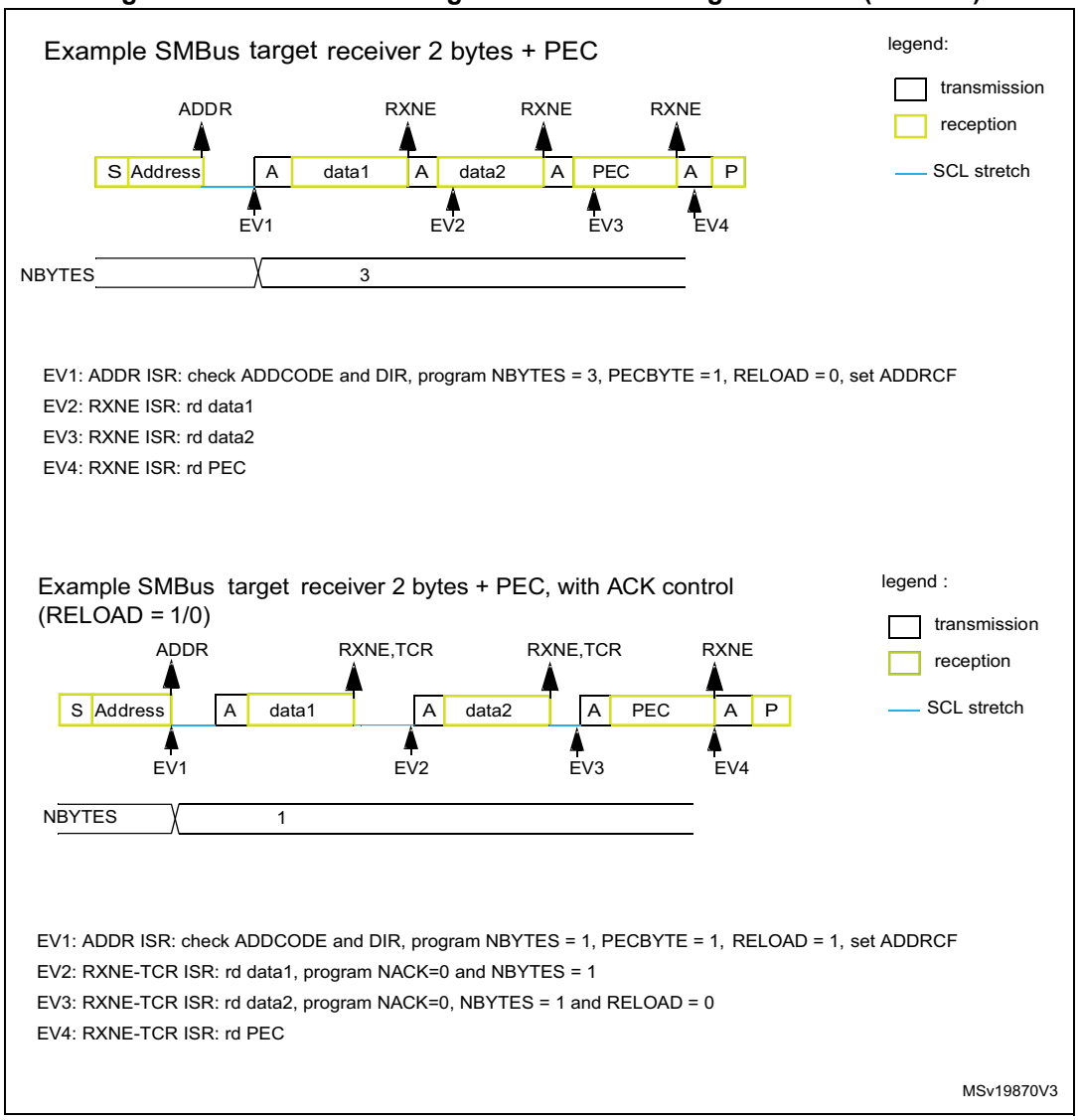


Figure 813. Bus transfer diagrams for SMBus target receiver (SBC = 1)



63.4.15 SMBus controller mode

In addition to I2C controller transfer management (refer to *Section 63.4.9: I2C controller mode*), this section provides extra software flowcharts to support SMBus.

SMBus controller transmitter

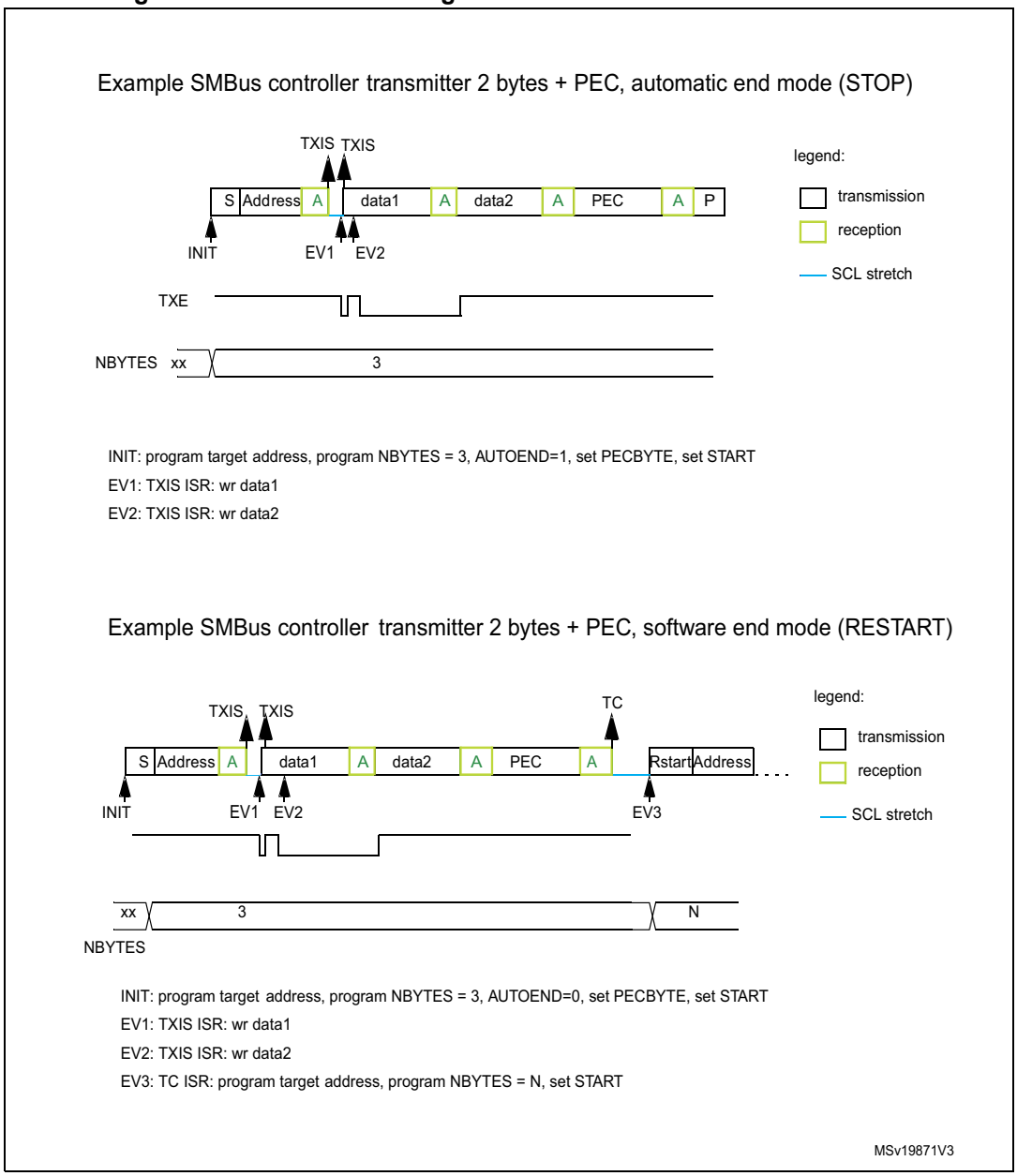
When the SMBus controller wants to transmit the PEC, the PECBYTE bit must be set and the number of bytes must be loaded in the NBYTES[7:0] bitfield, before setting the START bit. In this case, the total number of TXIS interrupts is NBYTES[7:0] - 1. So if the PECBYTE bit is set when NBYTES[7:0] = 0x1, the content of the I2C_PECR register is automatically transmitted.

If the SMBus controller wants to send a STOP condition after the PEC, the automatic end mode must be selected (AUTOEND = 1). In this case, the STOP condition automatically follows the PEC transmission.

When the SMBus controller wants to send a RESTART condition after the PEC, the software mode must be selected (AUTOEND = 0). In this case, once NBYTES[7:0] - 1 are transmitted, the I2C_PECR register content is transmitted. The TC flag is set after the PEC transmission, stretching the SCL line low. The RESTART condition must be programmed in the TC interrupt subroutine.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 814. Bus transfer diagrams for SMBus controller transmitter



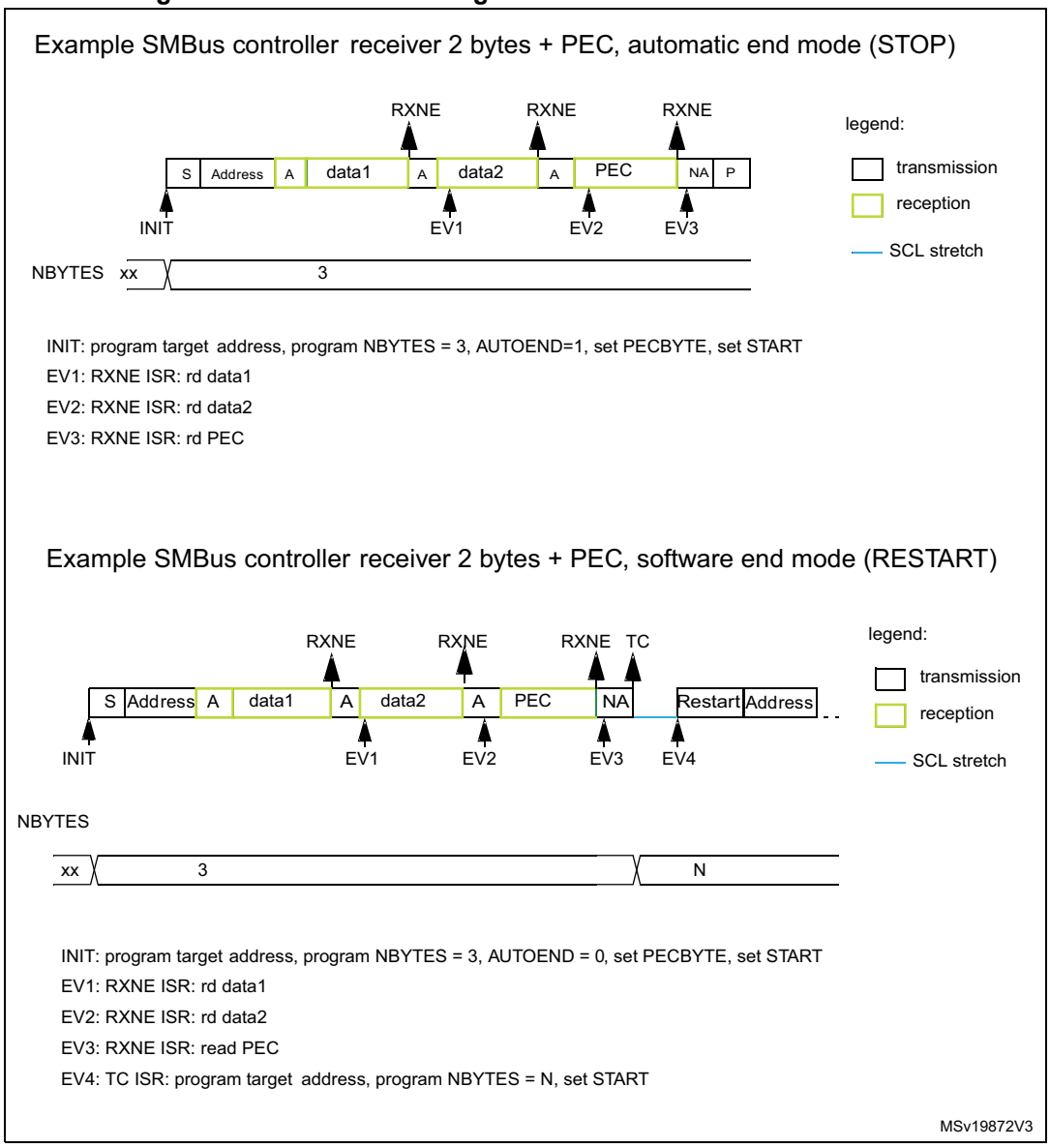
SMBus controller receiver

When the SMBus controller wants to receive, at the end of the transfer, the PEC followed by a STOP condition, the automatic end mode can be selected (AUTOEND = 1). The PECBYTE bit must be set and the target address programmed before setting the START bit. In this case, after the receipt of NBYTES[7:0] - 1 data bytes, the next received byte is automatically checked versus the I2C_PECR register content. A NACK response is given to the PEC byte, followed by a STOP condition.

When the SMBus controller receiver wants to receive, at the end of the transfer, the PEC byte followed by a RESTART condition, the software mode must be selected (AUTOEND = 0). The PECBYTE bit must be set and the target address programmed before setting the START bit. In this case, after the receipt of NBYTES[7:0] - 1 data bytes, the next received byte is automatically checked versus the I2C_PECR register content. The TC flag is set after the PEC byte reception, stretching the SCL line low. The RESTART condition can be programmed in the TC interrupt subroutine.

Caution: The PECBYTE bit has no effect when the RELOAD bit is set.

Figure 815. Bus transfer diagrams for SMBus controller receiver



63.4.16 Wake-up from Stop mode on address match

The I2C peripheral is able to wake up the device from Stop mode (APB clock is off), when the device is addressed. All addressing modes are supported.

The wake-up from Stop mode is enabled by setting the WUPEN bit of the I2C_CR1 register. The HSI or MSI oscillator must be selected as the clock source for i2c_ker_ck to allow the wake-up from Stop mode.

In Stop mode, the HSI or MSI oscillator is stopped. Upon detecting START condition, the I2C interface starts the HSI or MSI oscillator and stretches SCL low until the oscillator wakes up.

HSI or MSI is then used for the address reception.

If the received address matches the device own address, I2C stretches SCL low until the device wakes up. The stretch is released when the ADDR flag is cleared by software. Then the transfer goes on normally.

If the address does not match, the HSI or MSI oscillator is stopped again and the device does not wake up.

Note: When the system clock is used as I2C clock, or when WUPEN = 0, the HSI or MSI oscillator does not start upon receiving START condition.

Only an ADDR interrupt can wake the device up. Therefore, do not enter Stop mode when I2C is performing a transfer, either as a controller or as an addressed target after the ADDR flag is set. This can be managed by clearing the SLEEPDEEP bit in the ADDR interrupt routine and setting it again only after the STOPF flag is set.

Caution: The digital filter is not compatible with the wake-up from Stop mode feature. Before entering Stop mode with the WUPEN bit set, deactivate the digital filter, by writing zero to the DNF[3:0] bitfield.

Caution: The feature is only available when the HSI or MSI oscillator is selected as the I2C clock.

Caution: Clock stretching must be enabled (NOSTRETCH = 0) to ensure proper operation of the wake-up from Stop mode feature.

Caution: If the wake-up from Stop mode is disabled (WUPEN = 0), the I2C peripheral must be disabled before entering Stop mode (PE = 0).

63.4.17 Error conditions

The following errors are the conditions that can cause the communication to fail.

Bus error (BERR)

A bus error is detected when a START or a STOP condition is detected and is not located after a multiple of nine SCL clock pulses. START or STOP condition is detected when an SDA edge occurs while SCL is high.

The bus error flag is set only if the I2C peripheral is involved in the transfer as controller or addressed target (that is, not during the address phase in target mode).

In case of a misplaced START or RESTART detection in target mode, the I2C peripheral enters address recognition state like for a correct START condition.

When a bus error is detected, the BERR flag of the I2C_ISR register is set, and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Arbitration loss (ARLO)

An arbitration loss is detected when a high level is sent on the SDA line, but a low level is sampled on the SCL rising edge.

In controller mode, arbitration loss is detected during the address phase, data phase and data acknowledge phase. In this case, the SDA and SCL lines are released, the START control bit is cleared by hardware and the controller switches automatically to target mode.

In target mode, arbitration loss is detected during data phase and data acknowledge phase. In this case, the transfer is stopped and the SCL and SDA lines are released.

When an arbitration loss is detected, the ARLO flag of the I2C_ISR register is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Overrun/underrun error (OVR)

An overrun or underrun error is detected in target mode when NOSTRETCH = 1 and:

- In reception when a new byte is received and the RXDR register has not been read yet. The new received byte is lost, and a NACK is automatically sent as a response to the new byte.
- In transmission:
  - When STOPF = 1 and the first data byte must be sent. The content of the I2C_TXDR register is sent if TXE = 0, 0xFF if not.
  - When a new byte must be sent and the I2C_TXDR register has not been written yet, 0xFF is sent.

When an overrun or underrun error is detected, the OVR flag of the I2C_ISR register is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Packet error checking error (PECERR)

A PEC error is detected when the received PEC byte does not match the I2C_PECR register content. A NACK is automatically sent after the wrong PEC reception.

When a PEC error is detected, the PECERR flag of the I2C_ISR register is set and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Timeout error (TIMEOUT)

A timeout error occurs for any of these conditions:

- TIDLE = 0 and SCL remains low for the time defined in the TIMEOUTA[11:0] bitfield: this is used to detect an SMBus timeout.
- TIDLE = 1 and both SDA and SCL remains high for the time defined in the TIMEOUTA [11:0] bitfield: this is used to detect a bus idle condition.
- Controller cumulative clock low extend time reaches the time defined in the TIMEOUTB[11:0] bitfield (SMBus $t_{LOW:MEXT}$ parameter).
- Target cumulative clock low extend time reaches the time defined in the TIMEOUTB[11:0] bitfield (SMBus $t_{LOW:SEXT}$ parameter).

When a timeout violation is detected in controller mode, a STOP condition is automatically sent.

When a timeout violation is detected in target mode, the SDA and SCL lines are automatically released.

When a timeout error is detected, the TIMEOUT flag is set in the I2C_ISR register and an interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

Alert (ALERT)

The ALERT flag is set when the I2C peripheral is configured as a host (SMBHEN = 1), the SMBALERT# signal detection is enabled (ALERTEN = 1), and a falling edge is detected on the SMBA pin. An interrupt is generated if the ERRIE bit of the I2C_CR1 register is set.

63.5 I2C in low-power modes

Table 617. Effect of low-power modes to I2C

Mode	Description
Sleep	No effect. I2C interrupts cause the device to exit the Sleep mode.
Stop(1)	The contents of I2C registers are kept. – WUPEN = 1 and I2C is clocked by an internal oscillator (HSI or MSI). The address recognition is functional. The I2C address match condition causes the device to exit the Stop mode. – WUPEN = 0: I2C must be disabled before entering Stop mode.
Standby	The I2C peripheral is powered down. It must be reinitialized after exiting Standby mode.

1. Refer to *Section 63.3: I2C implementation* for information about the Stop modes supported by each instance. If the wake-up from a specific stop mode is not supported, the instance must be disabled before entering that specific Stop mode.

63.6 I2C interrupts

The following table gives the list of I2C interrupt requests.

Table 618. I2C interrupt requests

Interrupt acronym	Interrupt event	Event flag	Enable control bit	Interrupt clear method	Exit Sleep mode	Exit Stop SVOS high modes	Exit Stop SVOS low, Standby modes
I2C_EV	Receive buffer not empty	RXNE	RXIE	Read I2C_RXDR register	Yes	No	No
	Transmit buffer interrupt status	TXIS	TXIE	Write I2C_TXDR register			
	STOP detection interrupt flag	STOPF	STOPIE	Write STOPCF = 1			
	Transfer complete reload	TCR	TCIE	Write I2C_CR2 with NBYTES[7:0] ≠ 0			
	Transfer complete	TC		Write START = 1 or STOP = 1			
	Address matched	ADDR	ADDRIE	Write ADDRCF = 1		Yes(1)	
	NACK reception	NACKF	NACKIE	Write NACKCF = 1		No	
I2C_ERR	Bus error	BERR	ERRIE	Write BERRCF = 1	Yes	No	No
	Arbitration loss	ARLO		Write ARLOCF = 1			
	Overrun/underrun	OVR		Write OVRCF = 1			
I2C_ERR	PEC error	PECERR	ERRIE	Write PECERRCF = 1	Yes	No	No
	Timeout/ t_{LOW} error	TIMEOUT		Write TIMEOUTCF = 1			
	SMBus alert	ALERT		Write ALERTCF = 1			

1. The ADDR match event can wake up the device from Stop mode only if the I2C instance supports the wake-up from Stop mode feature. Refer to *Section 63.3: I2C implementation*.

63.7 I2C DMA requests

63.7.1 Transmission using DMA

DMA (direct memory access) can be enabled for transmission by setting the TXDMAEN bit of the I2C_CR1 register. Data is loaded from an SRAM area configured through the DMA peripheral (see *Section 14: Direct memory access controller (DMA)*) to the I2C_TXDR register whenever the TXIS bit is set.

Only the data are transferred with DMA.

In controller mode, the initialization, the target address, direction, number of bytes and START bit are programmed by software (the transmitted target address cannot be transferred with DMA). When all data are transferred using DMA, DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter. Refer to *Controller transmitter*.

In target mode:

- With NOSTRETCH = 0, when all data are transferred using DMA, DMA must be initialized before the address match event, or in ADDR interrupt subroutine, before clearing ADDR.
- With NOSTRETCH = 1, the DMA must be initialized before the address match event.

The PEC transfer is managed with the counter associated to the NBYTES[7:0] bitfield. Refer to *SMBus target transmitter* and *SMBus controller transmitter*.

Note: If DMA is used for transmission, it is not required to set the TXIE bit.

63.7.2 Reception using DMA

DMA (direct memory access) can be enabled for reception by setting the RXDMAEN bit of the I2C_CR1 register. Data is loaded from the I2C_RXDR register to an SRAM area configured through the DMA peripheral (refer to *Section 14: Direct memory access controller (DMA)*) whenever the RXNE bit is set. Only the data (including PEC) are transferred with DMA.

In controller mode, the initialization, the target address, direction, number of bytes and START bit are programmed by software. When all data are transferred using DMA, DMA must be initialized before setting the START bit. The end of transfer is managed with the NBYTES counter.

In target mode with NOSTRETCH = 0, when all data are transferred using DMA, DMA must be initialized before the address match event, or in the ADDR interrupt subroutine, before clearing the ADDR flag.

The PEC transfer is managed with the counter associated to the NBYTES[7:0] bitfield. Refer to *SMBus target receiver* and *SMBus controller receiver*.

Note: If DMA is used for reception, it is not required to set the RXIE bit.

63.8 I2C debug modes

When the device enters debug mode (core halted), the SMBus timeout either continues working normally or stops, depending on the DBG_I2Cx_STOP bits in the DBG block.

63.9 I2C registers

Refer to *Section 1.2* for the list of abbreviations used in register descriptions.

The registers are accessed by words (32-bit).

63.9.1 I2C control register 1 (I2C_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access, until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STOPFACLR	ADDRACLR	Res.	Res.	Res.	Res.	Res.	FMP	PECEN	ALERTEN	SMBDEN	SMBHEN	GCEN	WUPEN	NOSTRETCH	SBC
rw	rw						rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDMAEN	TXDMAEN	Res.	ANFOFF	DNF[3:0]				ERRIE	TCIE	STOPIE	NACKIE	ADDRIE	RXIE	TXIE	PE
rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **STOPFACLR**: STOP detection flag (STOPF) automatic clear
0: STOPF flag is set by hardware, cleared by software by setting STOPCF bit.
1: STOPF flag remains cleared by hardware. This mode can be used in NOSTRETCH target mode, to avoid the overrun error if the STOPF flag is not cleared before next data transmission. This allows a target data management by DMA only, without any interrupt from peripheral.

Bit 30 **ADDRACLR**: Address match flag (ADDR) automatic clear
0: ADDR flag is set by hardware, cleared by software by setting ADDRCF bit.
1: ADDR flag remains cleared by hardware. This mode can be used in target mode, to avoid the ADDR clock stretching if the I2C enables only one target address. This allows a target data management by DMA only, without any interrupt from peripheral.

Bits 29:25 Reserved, must be kept at reset value.

Bit 24 **FMP**: Fast-mode Plus drive enable
0: FMP I/O drive disabled
1: FMP I/O drive enabled

Bit 23 **PECEN:** PEC enable
0: PEC calculation disabled
1: PEC calculation enabled

Bit 22 **ALERTEN**: SMBus alert enable

0: The SMBALERT# signal on SMBA pin is not supported in host mode (SMBHEN = 1). In device mode (SMBHEN = 0), the SMBA pin is released and the alert response address header is disabled (0001100x followed by NACK).
1: The SMBALERT# signal on SMBA pin is supported in host mode (SMBHEN = 1). In device mode (SMBHEN = 0), the SMBA pin is driven low and the alert response address header is enabled (0001100x followed by ACK).

Note: When ALERTEN = 0, the SMBA pin can be used as a standard GPIO.

Bit 21 **SMBDEN**: SMBus device default address enable

0: Device default address disabled. Address 0b1100001x is NACKed.
1: Device default address enabled. Address 0b1100001x is ACKed.

Bit 20 **SMBHEN**: SMBus host address enable

0: Host address disabled. Address 0b0001000x is NACKed.
1: Host address enabled. Address 0b0001000x is ACKed.

Bit 19 **GCEN**: General call enable

0: General call disabled. Address 0b00000000 is NACKed.
1: General call enabled. Address 0b00000000 is ACKed.

Bit 18 **WUPEN**: Wake-up from Stop mode enable

0: Wake-up from Stop mode disabled.
1: Wake-up from Stop mode enabled.

Note: WUPEN can be set only when DNF[3:0] = 0000.

Bit 17 **NOSTRETCH**: Clock stretching disable

This bit is used to disable clock stretching in target mode. It must be kept cleared in controller mode.
0: Clock stretching enabled
1: Clock stretching disabled

Note: This bit can be programmed only when the I2C peripheral is disabled (PE = 0).

Bit 16 **SBC**: Target byte control

This bit is used to enable hardware byte control in target mode.
0: Target byte control disabled
1: Target byte control enabled

Bit 15 **RXDMAEN**: DMA reception requests enable

0: DMA mode disabled for reception
1: DMA mode enabled for reception

Bit 14 **TXDMAEN**: DMA transmission requests enable

0: DMA mode disabled for transmission
1: DMA mode enabled for transmission

Bit 13 Reserved, must be kept at reset value.

Bit 12 **ANFOFF:** Analog noise filter OFF

0: Analog noise filter enabled
1: Analog noise filter disabled

Note: This bit can be programmed only when the I2C peripheral is disabled (PE = 0).

Bits 11:8 **DNF[3:0]**: Digital noise filter

These bits are used to configure the digital noise filter on SDA and SCL input. The digital filter, filters spikes with a length of up to DNF[3:0] * t_{I2CCLK}

0000: Digital filter disabled

0001: Digital filter enabled and filtering capability up to one t_{I2CCLK}

...

1111: digital filter enabled and filtering capability up to fifteen t_{I2CCLK}

Note: If the analog filter is enabled, the digital filter is added to it. This filter can be programmed only when the I2C peripheral is disabled (PE = 0).

Bit 7 **ERRIE**: Error interrupts enable

0: Error detection interrupts disabled

1: Error detection interrupts enabled

Note: Any of these errors generates an interrupt:

- arbitration loss (ARLO)

- bus error detection (BERR)

- overrun/underrun (OVR)

- timeout detection (TIMEOUT)

- PEC error detection (PECERR)

- alert pin event detection (ALERT)

Bit 6 **TCIE**: Transfer complete interrupt enable

0: Transfer complete interrupt disabled

1: Transfer complete interrupt enabled

Note: Any of these events generates an interrupt:

Transfer complete (TC)

Transfer complete reload (TCR)

Bit 5 **STOPIE**: STOP detection interrupt enable

0: STOP detection (STOPF) interrupt disabled

1: STOP detection (STOPF) interrupt enabled

Bit 4 **NACKIE**: Not acknowledge received interrupt enable

0: Not acknowledge (NACKF) received interrupts disabled

1: Not acknowledge (NACKF) received interrupts enabled

Bit 3 **ADDRIE**: Address match interrupt enable (target only)

0: Address match (ADDR) interrupts disabled

1: Address match (ADDR) interrupts enabled

Bit 2 **RXIE**: RX interrupt enable

0: Receive (RXNE) interrupt disabled

1: Receive (RXNE) interrupt enabled

Bit 1 **TXIE**: TX interrupt enable

0: Transmit (TXIS) interrupt disabled

1: Transmit (TXIS) interrupt enabled

Bit 0 **PE**: Peripheral enable

0: Peripheral disabled

1: Peripheral enabled

Note: When PE = 0, the I2C SCL and SDA lines are released. Internal state machines and status bits are put back to their reset value. When cleared, PE must be kept low for at least three APB clock cycles.

63.9.2 I2C control register 2 (I2C_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	PECBYTE	AUTOEND	RELOAD	NBYTES[7:0]							
					rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NACK	STOP	START	HEAD10R	ADD10	RD_WRN	SADD[9:0]									
rs	rs	rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **PECBYTE**: Packet error checking byte

This bit is set by software, and cleared by hardware when the PEC is transferred, or when a STOP condition or an Address matched is received, also when PE = 0.

0: No PEC transfer

1: PEC transmission/reception is requested

Note: Writing 0 to this bit has no effect.

This bit has no effect when RELOAD is set, and in target mode when SBC = 0.

Bit 25 **AUTOEND**: Automatic end mode (controller mode)

This bit is set and cleared by software.

0: software end mode: TC flag is set when NBYTES data are transferred, stretching SCL low.

1: Automatic end mode: a STOP condition is automatically sent when NBYTES data are transferred.

Note: This bit has no effect in target mode or when the RELOAD bit is set.

Bit 24 **RELOAD**: NBYTES reload mode

This bit is set and cleared by software.

0: The transfer is completed after the NBYTES data transfer (STOP or RESTART follows).

1: The transfer is not completed after the NBYTES data transfer (NBYTES is reloaded). TCR flag is set when NBYTES data are transferred, stretching SCL low.

Bits 23:16 **NBYTES[7:0]**: Number of bytes

The number of bytes to be transmitted/received is programmed there. This field is don't care in target mode with SBC = 0.

Note: Changing these bits when the START bit is set is not allowed.

Bit 15 **NACK**: NACK generation (target mode)

The bit is set by software, cleared by hardware when the NACK is sent, or when a STOP condition or an Address matched is received, or when PE = 0.

0: an ACK is sent after current received byte.

1: a NACK is sent after current received byte.

Note: Writing 0 to this bit has no effect.

This bit is used only in target mode: in controller receiver mode, NACK is automatically generated after last byte preceding STOP or RESTART condition, whatever the NACK bit value.

When an overrun occurs in target receiver NOSTRETCH mode, a NACK is automatically generated, whatever the NACK bit value.

When hardware PEC checking is enabled (PECBYTE = 1), the PEC acknowledge value does not depend on the NACK value.

Bit 14 **STOP**: STOP condition generation

This bit only pertains to controller mode. It is set by software and cleared by hardware when a STOP condition is detected or when PE = 0.

0: No STOP generation

1: STOP generation after current byte transfer

Note: Writing 0 to this bit has no effect.

Bit 13 **START**: START condition generation

This bit is set by software. It is cleared by hardware after the START condition followed by the address sequence is sent, by an arbitration loss, by an address matched in target mode, by a timeout error detection, or when PE = 0.

0: No START generation

1: RESTART/START generation:

If the I2C is already in controller mode with AUTOEND = 0, setting this bit generates a repeated START condition when RELOAD = 0, after the end of the NBYTES transfer.

Otherwise, setting this bit generates a START condition once the bus is free.

Note: Writing 0 to this bit has no effect.

The START bit can be set even if the bus is BUSY or I2C is in target mode.

This bit has no effect when RELOAD is set.

Bit 12 **HEAD10R**: 10-bit address header only read direction (controller receiver mode)

0: The controller sends the complete 10-bit target address read sequence: START + 2 bytes 10-bit address in write direction + RESTART + first seven bits of the 10-bit address in read direction.

1: The controller sends only the first seven bits of the 10-bit address, followed by read direction.

Note: Changing this bit when the START bit is set is not allowed.

Bit 11 **ADD10**: 10-bit addressing mode (controller mode)

0: The controller operates in 7-bit addressing mode

1: The controller operates in 10-bit addressing mode

Note: Changing this bit when the START bit is set is not allowed.

Bit 10 **RD_WRN**: Transfer direction (controller mode)

0: Controller requests a write transfer

1: Controller requests a read transfer

Note: Changing this bit when the START bit is set is not allowed.

Bits 9:0 **SADD[9:0]**: Target address (controller mode)

Condition: In 7-bit addressing mode (ADD10 = 0):
SADD[7:1] must be written with the 7-bit target address to be sent. Bits SADD[9], SADD[8] and SADD[0] are don't care.

Condition: In 10-bit addressing mode (ADD10 = 1):
SADD[9:0] must be written with the 10-bit target address to be sent.

Note: Changing these bits when the START bit is set is not allowed.

63.9.3 I2C own address 1 register (I2C_OAR1)

Address offset: 0x08

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OA1EN	Res.	Res.	Res.	Res.	OA1MODE	OA1[9:0]									
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **OA1EN**: Own address 1 enable

0: Own address 1 disabled. The received target address OA1 is NACKed.
1: Own address 1 enabled. The received target address OA1 is ACKed.

Bits 14:11 Reserved, must be kept at reset value.

Bit 10 **OA1MODE**: Own address 1 10-bit mode

0: Own address 1 is a 7-bit address.
1: Own address 1 is a 10-bit address.

Note: This bit can be written only when OA1EN = 0.

Bits 9:0 **OA1[9:0]**: Interface own target address

7-bit addressing mode: OA1[7:1] contains the 7-bit own target address. Bits OA1[9], OA1[8] and OA1[0] are don't care.

10-bit addressing mode: OA1[9:0] contains the 10-bit own target address.

Note: These bits can be written only when OA1EN = 0.

63.9.4 I2C own address 2 register (I2C_OAR2)

Address offset: 0x0C

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access, until the previous one is completed. The latency of the second write access can be up to 2x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OA2EN	Res.	Res.	Res.	Res.	OA2MSK[2:0]			OA2[7:1]							Res.
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **OA2EN**: Own address 2 enable
0: Own address 2 disabled. The received target address OA2 is NACKed.
1: Own address 2 enabled. The received target address OA2 is ACKed.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:8 **OA2MSK[2:0]**: Own address 2 masks
000: No mask
001: OA2[1] is masked and don't care. Only OA2[7:2] are compared.
010: OA2[2:1] are masked and don't care. Only OA2[7:3] are compared.
011: OA2[3:1] are masked and don't care. Only OA2[7:4] are compared.
100: OA2[4:1] are masked and don't care. Only OA2[7:5] are compared.
101: OA2[5:1] are masked and don't care. Only OA2[7:6] are compared.
110: OA2[6:1] are masked and don't care. Only OA2[7] is compared.
111: OA2[7:1] are masked and don't care. No comparison is done, and all (except reserved) 7-bit received addresses are acknowledged.
Note: These bits can be written only when OA2EN = 0.
As soon as OA2MSK ≠ 0, the reserved I2C addresses (0b0000xxx and 0b1111xxx) are not acknowledged, even if the comparison matches.

Bits 7:1 **OA2[7:1]**: Interface address
7-bit addressing mode: 7-bit address
Note: These bits can be written only when OA2EN = 0.

Bit 0 Reserved, must be kept at reset value.

63.9.5 I2C timing register (I2C_TIMINGR)

Address offset: 0x10

Reset value: 0x0000 0000

Access: no wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRESC[3:0]				Res.	Res.	Res.	Res.	SCLDEL[3:0]				SDADEL[3:0]			
rw	rw	rw	rw					rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCLH[7:0]								SCLL[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **PRESC[3:0]**: Timing prescaler

This field is used to prescale i2c_ker_ck to generate the clock period t_{PRESC} used for data setup and hold counters (refer to section *I2C timings*), and for SCL high and low level counters (refer to section *I2C controller initialization*).

t_{PRESC} = (PRESC + 1) x t_{I2CCLK}

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:20 **SCLDEL[3:0]**: Data setup time

This field is used to generate a delay t_{SCLDEL} = (SCLDEL + 1) x t_{PRESC} between SDA edge and SCL rising edge. In controller and in target modes with NOSTRETCH = 0, the SCL line is stretched low during t_{SCLDEL}.

Note: t_{SCLDEL} is used to generate $t_{SU:DAT}$ timing.

Bits 19:16 **SDADEL[3:0]**: Data hold time

This field is used to generate the delay t_{SDADEL} between SCL falling edge and SDA edge. In controller and in target modes with NOSTRETCH = 0, the SCL line is stretched low during t_{SDADEL}.

t_{SDADEL}= SDADEL x t_{PRESC}

Note: SDADEL is used to generate $t_{HD:DAT}$ timing.

Bits 15:8 **SCLH[7:0]**: SCL high period (controller mode)

This field is used to generate the SCL high period in controller mode.

t_{SCLH} = (SCLH + 1) x t_{PRESC}

Note: SCLH is also used to generate $t_{SU:STO}$ and $t_{HD:STA}$ timing.

Bits 7:0 **SCLL[7:0]**: SCL low period (controller mode)

This field is used to generate the SCL low period in controller mode.

t_{SCLL} = (SCLL + 1) x t_{PRESC}

Note: SCLL is also used to generate t_{BUF} and $t_{SU:STA}$ timings.

Note: This register must be configured when the I2C peripheral is disabled (PE = 0).

Note: The STM32CubeMX tool calculates and provides the I2C_TIMINGR content in the I2C Configuration window.

63.9.6 I2C timeout register (I2C_TIMEOUTR)

Address offset: 0x14

Reset value: 0x0000 0000

Access: no wait states, except if a write access occurs while a write access is ongoing. In this case, wait states are inserted in the second write access until the previous one is completed. The latency of the second write access can be up to 2 x i2c_pclk + 6 x i2c_ker_ck.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TEXTEN	Res.	Res.	Res.	TIMEOUTB[11:0]											
rw				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIMOUTEN	Res.	Res.	TIDLE	TIMEOUTA[11:0]											
rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TEXTEN**: Extended clock timeout enable

0: Extended clock timeout detection is disabled

1: Extended clock timeout detection is enabled. When a cumulative SCL stretch for more than $t_{LOW:EXT}$ is done by the I2C interface, a timeout error is detected (TIMEOUT = 1).

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:16 **TIMEOUTB[11:0]**: Bus timeout B

This field is used to configure the cumulative clock extension timeout:

– Controller mode: the controller cumulative clock low extend time ($t_{LOW:MEXT}$) is detected

– Target mode: the target cumulative clock low extend time ($t_{LOW:SEXT}$) is detected

$t_{LOW:EXT}$ = (TIMEOUTB + TIDLE = 01) x 2048 x t_{I2CCLK}

Note: These bits can be written only when TEXTEN = 0.

Bit 15 **TIMOUTEN**: Clock timeout enable

0: SCL timeout detection is disabled

1: SCL timeout detection is enabled. When SCL is low for more than $t_{TIMEOUT}$ (TIDLE = 0) or high for more than t_{IDLE} (TIDLE = 1), a timeout error is detected (TIMEOUT = 1).

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **TIDLE**: Idle clock timeout detection

0: TIMEOUTA is used to detect SCL low timeout

1: TIMEOUTA is used to detect both SCL and SDA high timeout (bus idle condition)

Note: This bit can be written only when TIMOUTEN = 0.

Bits 11:0 **TIMEOUTA[11:0]**: Bus timeout A

This field is used to configure:

The SCL low timeout condition $t_{TIMEOUT}$ when TIDLE = 0

$t_{TIMEOUT}$= (TIMEOUTA + 1) x 2048 x t_{I2CCLK}

The bus idle condition (both SCL and SDA high) when TIDLE = 1

t_{IDLE}= (TIMEOUTA + 1) x 4 x t_{I2CCLK}

Note: These bits can be written only when TIMOUTEN = 0.

63.9.7 I2C interrupt and status register (I2C_ISR)

Address offset: 0x18

Reset value: 0x0000 0001

Access: no wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDCODE[6:0]							DIR
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BUSY	Res.	ALERT	TIMEOUT	PECERR	OVR	ARLO	BERR	TCR	TC	STOPF	NACKF	ADDR	RXNE	TXIS	TXE
r		r	r	r	r	r	r	r	r	r	r	r	r	rs	rs

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:17 **ADDCODE[6:0]**: Address match code (target mode)

These bits are updated with the received address when an address match event occurs (ADDR = 1). In the case of a 10-bit address, ADDCODE provides the 10-bit header followed by the two MSBs of the address.

Bit 16 **DIR**: Transfer direction (target mode)

This flag is updated when an address match event occurs (ADDR = 1).

0: Write transfer, target enters receiver mode.

1: Read transfer, target enters transmitter mode.

Bit 15 **BUSY**: Bus busy

This flag indicates that a communication is in progress on the bus. It is set by hardware when a START condition is detected, and cleared by hardware when a STOP condition is detected, or when PE = 0.

Bit 14 Reserved, must be kept at reset value.

Bit 13 **ALERT**: SMBus alert

This flag is set by hardware when SMBHEN = 1 (SMBus host configuration), ALERTEN = 1 and an SMBALERT# event (falling edge) is detected on SMBA pin. It is cleared by software by setting the ALERTCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 12 **TIMEOUT**: Timeout or t_{LOW} detection flag

This flag is set by hardware when a timeout or extended clock timeout occurred. It is cleared by software by setting the TIMEOUTCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 11 **PECERR**: PEC error in reception

This flag is set by hardware when the received PEC does not match with the PEC register content. A NACK is automatically sent after the wrong PEC reception. It is cleared by software by setting the PECCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 10 **OVR**: Overrun/underrun (target mode)

This flag is set by hardware in target mode with NOSTRETCH = 1, when an overrun/underrun error occurs. It is cleared by software by setting the OVRCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 9 **ARLO**: Arbitration lost

This flag is set by hardware in case of arbitration loss. It is cleared by software by setting the ARLOCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 8 **BERR**: Bus error

This flag is set by hardware when a misplaced START or STOP condition is detected whereas the peripheral is involved in the transfer. The flag is not set during the address phase in target mode. It is cleared by software by setting the BERRCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 7 **TCR**: Transfer complete reload

This flag is set by hardware when RELOAD = 1 and NBYTES data have been transferred. It is cleared by software when NBYTES is written to a non-zero value.

Note: This bit is cleared by hardware when PE = 0.

This flag is only for controller mode, or for target mode when the SBC bit is set.

Bit 6 **TC**: Transfer complete (controller mode)

This flag is set by hardware when RELOAD = 0, AUTOEND = 0 and NBYTES data have been transferred. It is cleared by software when START bit or STOP bit is set.

Note: This bit is cleared by hardware when PE = 0.

Bit 5 **STOPF**: STOP detection flag

This flag is set by hardware when a STOP condition is detected on the bus and the peripheral is involved in this transfer:

- as a controller, provided that the STOP condition is generated by the peripheral.
- as a target, provided that the peripheral has been addressed previously during this transfer.

It is cleared by software by setting the STOPCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 4 **NACKF**: Not acknowledge received flag

This flag is set by hardware when a NACK is received after a byte transmission. It is cleared by software by setting the NACKCF bit.

Note: This bit is cleared by hardware when PE = 0.

Bit 3 **ADDR**: Address matched (target mode)

This bit is set by hardware as soon as the received target address matched with one of the enabled target addresses. It is cleared by software by setting *ADDRCF bit.*

Note: This bit is cleared by hardware when PE = 0.

Bit 2 **RXNE**: Receive data register not empty (receivers)

This bit is set by hardware when the received data is copied into the I2C_RXDR register, and is ready to be read. It is cleared when I2C_RXDR is read.

Note: This bit is cleared by hardware when PE = 0.

Bit 1 **TXIS**: Transmit interrupt status (transmitters)

This bit is set by hardware when the I2C_TXDR register is empty and the data to be transmitted must be written in the I2C_TXDR register. It is cleared when the next data to be sent is written in the I2C_TXDR register.

This bit can be written to 1 by software only when NOSTRETCH = 1, to generate a TXIS event (interrupt if TXIE = 1 or DMA request if TXDMAEN = 1).

Note: This bit is cleared by hardware when PE = 0.

Bit 0 **TXE**: Transmit data register empty (transmitters)

This bit is set by hardware when the I2C_TXDR register is empty. It is cleared when the next data to be sent is written in the I2C_TXDR register.

This bit can be written to 1 by software in order to flush the transmit data register I2C_TXDR.

Note: This bit is set by hardware when PE = 0.

63.9.8 I2C interrupt clear register (I2C_ICR)

Address offset: 0x1C

Reset value: 0x0000 0000

Access: no wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ALERTCF	TIMOUTCF	PECCF	OVRCF	ARLOCF	BERRCF	Res.	Res.	STOPCF	NACKCF	ADDRCF	Res.	Res.	Res.
		w	w	w	w	w	w			w	w	w			

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **ALERTCF**: Alert flag clear
Writing 1 to this bit clears the ALERT flag in the I2C_ISR register.

Bit 12 **TIMOUTCF**: Timeout detection flag clear
Writing 1 to this bit clears the TIMEOUT flag in the I2C_ISR register.

Bit 11 **PECCF**: PEC error flag clear
Writing 1 to this bit clears the PECERR flag in the I2C_ISR register.

Bit 10 **OVRCF**: Overrun/underrun flag clear
Writing 1 to this bit clears the OVR flag in the I2C_ISR register.

Bit 9 **ARLOCF**: Arbitration lost flag clear
Writing 1 to this bit clears the ARLO flag in the I2C_ISR register.

Bit 8 **BERRCF**: Bus error flag clear
Writing 1 to this bit clears the BERRF flag in the I2C_ISR register.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **STOPCF**: STOP detection flag clear
Writing 1 to this bit clears the STOPF flag in the I2C_ISR register.

Bit 4 **NACKCF**: Not acknowledge flag clear
Writing 1 to this bit clears the NACKF flag in I2C_ISR register.

Bit 3 **ADDRCF**: Address matched flag clear
Writing 1 to this bit clears the ADDR flag in the I2C_ISR register. Writing 1 to this bit also clears the START bit in the I2C_CR2 register.

Bits 2:0 Reserved, must be kept at reset value.

63.9.9 I2C PEC register (I2C_PECR)

Address offset: 0x20

Reset value: 0x0000 0000

Access: no wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PEC[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PEC[7:0]:** Packet error checking register
This field contains the internal PEC when PECEN=1.
The PEC is cleared by hardware when PE = 0.

63.9.10 I2C receive data register (I2C_RXDR)

Address offset: 0x24

Reset value: 0x0000 0000

Access: no wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXDATA[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **RXDATA[7:0]:** 8-bit receive data
Data byte received from the I²C-bus.

63.9.11 I2C transmit data register (I2C_TXDR)

Address offset: 0x28

Reset value: 0x0000 0000

Access: no wait states

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXDATA[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TXDATA[7:0]:** 8-bit transmit data

Data byte to be transmitted to the I²C-bus

Note: These bits can be written only when TXE = 1.

63.9.12 I2C register map

The table below provides the I2C register map and the reset values.

Table 619. I2C register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	I2C_CR1	STOPFACLR	ADDRACLR	Res.	Res.	Res.	Res.	Res.	FMP	PECEN	ALERTEN	SMBDEN	SMBHEN	GCEN	WUPEN	NOSTRETCH	SBC	RXDMAEN	TXDMAEN	Res.	ANFOFF	DNF[3:0]				ERRIE	TCIE	STOPIE	NACKIE	ADDRIE	RXIE	TXIE	PE
	Reset value	0	0						0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	I2C_CR2	Res.	Res.	Res.	Res.	Res.	PECBYTE	AUTOEND	RELOAD	NBYTES[7:0]								NACK	STOP	START	HEAD10R	ADD10	RD_WRN	SADD[9:0]									
	Reset value						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	I2C_OAR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OA1EN	Res.	Res.	Res.	Res.	OA1MODE	OA1[9:0]									
	Reset value																	0					0	0	0	0	0	0	0	0	0	0	0
0x0C	I2C_OAR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OA2EN	Res.	Res.	Res.	Res.	OA2MSK [2:0]			OA2[7:1]							Res.
	Reset value																	0					0	0	0	0	0	0	0	0	0	0	
0x10	I2C_TIMINGR	PRESC[3:0]				Res.	Res.	Res.	Res.	SCLDEL [3:0]				SDADEL [3:0]				SCLH[7:0]								SCLL[7:0]							
	Reset value	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	I2C_TIMEOUTR	TEXTEN	Res.	Res.	Res.	TIMEOUTB[11:0]												TIMOUTEN	Res.	Res.	TIDLE	TIMEOUTA[11:0]											
	Reset value	0				0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	I2C_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDCODE[6:0]							DIR	BUSY	Res.	ALERT	TIMEOUT	PECERR	OVR	ARLO	BERR	TCR	TC	STOPF	NACKF	ADDR	RXNE	TXIS	TXE
	Reset value									0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x1C	I2C_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ALERTCF	TIMOUTCF	PECCF	OVRCF	ARLOCF	BERRCF	Res.	Res.	STOPCF	NACKCF	ADDRCF	Res.	Res.	Res.
	Reset value																			0	0	0	0	0	0			0	0	0			
0x20	I2C_PECR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PEC[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x24	I2C_RXDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXDATA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x28	I2C_TXDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXDATA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

64 Improved inter-integrated circuit (I3C)

64.1 I3C introduction

The I3C interface handles communication with other devices such as sensor(s) and host processor.

An I3C bus is a two-wire, serial single-ended, multi-drop bus, intended to improve the legacy I^2C bus. The I3C interface improves the features of the I^2C interface (adding especially in-band interrupts, dynamic address assignment, energy-efficient data rates, power management) meanwhile preserving some backward compatibility: it allows an I^2C target to operate on an I3C bus in legacy I^2C standard-mode (Sm), fast-mode (Fm), or legacy I^2C fast-mode plus (Fm+), provided that the latter does not perform clock stretching.

The I3C SDR-only peripheral implements all the required features and many optional features of the MIPI® I3C specification v1.1 (see *Table 624: I3C peripheral controller/target features versus MIPI v1.1*). It can control all I3C bus-specific sequencing, protocol, arbitration, and timing.

After reset, the software can initialize the I3C peripheral as primary controller or as target:

- As I3C controller, the software can enable the I3C peripheral and initialize the I3C bus.
- As I3C target, the software can enable the I3C peripheral to operate on an I3C bus.

The I3C peripheral can be used with DMA, to off-load the CPU.

64.2 I3C main features

The I3C peripheral supports:

- MIPI® I3C specification v1.1 (see details in *Table 624*), as:
  - I3C SDR-only primary controller
  - I3C SDR-only secondary controller
  - I3C SDR-only target
- I3C SCL bus clock frequency up to 12.5 MHz
- Registers configuration from the host application via the APB slave port
- Queued data transfers:
  - Transmit FIFO (TX-FIFO) for data bytes/words to be transmitted on the I3C bus
  - Receive FIFO (RX-FIFO) for received data bytes/words on the I3C bus
  - For each FIFO, optional DMA mode with a dedicated DMA channel
- Queued control/status transfers, when controller:
  - Control FIFO (C-FIFO) for control words to be sent on the I3C bus
  - Optional status FIFO (S-FIFO) for status words as received on the I3C bus
  - For each FIFO, optional DMA mode with a dedicated DMA channel
- Messages:
  - Legacy I^2C read/write messages to legacy I^2C targets in Sm/Fm/Fm+
  - I3C SDR read/write private messages

  - I3C SDR broadcast CCC messages (see details in *Table 630*)
  - I3C SDR read/write direct CCC messages (see details in *Table 630*)
- Frame-level management, when controller:
  - Optional C-FIFO and TX-FIFO preload
  - Multiple messages encapsulation
  - Optional arbitrable header generation on the I3C bus
  - HDR exit pattern generation on the I3C bus for error recovery
- Programmable bus timing, when controller:
  - SCL high and low period
  - SDA hold time
  - Bus free (minimum) time
  - Bus available/idle condition time
  - Clock stall time
- Target-initiated requests management:
  - Simultaneous support up to four I3C targets, when controller
  - In-band interrupts, with programmable IBI payload (up to 4 bytes), with pending read notification support
  - Bus control request, with recovery flow support and hand-off delay
  - Hot-join mechanism
- HDR exit pattern detection, when target
- Bus error management:
  - CEx with x = 0, 1, 2, 3 when controller
  - TEx with x = 0, 1, ... , 6 when target
  - Bus control switch error and recovery
  - Target reset
- Individual programmable event-based management:
  - Per-event identification with flag reporting and clear control
  - Host application notification via flag polling, and/or via interrupt with a per-event programmable enable
  - Error type identification
- Wake-up from Stop mode(s), as controller (see *Section 64.3.2*):
  - On an in-band interrupt without payload
  - On a hot-join request
  - On a controller-role request
- Wake-up from Stop mode(s), as target (see *Section 64.3.2*):
  - On a reset pattern
  - On a missed start on the serial bus
- Multi-clock domain management:
  - Separate APB clock and kernel clock, driven from independently programmed clock sources via the RCC, in addition to SCL clock
  - Minimum operating frequency for the kernel clock and the APB clock vs. the application-driven SCL clock (see clocks constraints in *Section 64.6.2*)

64.3 I3C implementation

64.3.1 I3C instantiation

There are several I3C instances in the device, all with the same feature set:

Table 620. I3C instantiation

Instance(s)
I3C1, I3C2

64.3.2 I3C wake-up from low-power mode(s)

The I3C peripheral can wake up the device from a low-power mode, as detailed in *Table 621*. For more details about the wake-up capabilities, refer to *Section 64.13*.

Table 621. I3C wake-up

Wake-up
From Stop mode with SVOS high

64.3.3 I3C FIFOs

The FIFOs are implemented as defined in *Table 622*.

Table 622. I3C FIFOs implementation

FIFO	Content	Unit	Size (in unit)	Used as controller/target (rationale)
C-FIFO	(32-bit) Control words	Word	2	Controller (a frame can be based on multiple control words; this is not the case as target)
S-FIFO	(32-bit) Status words			Controller (target: status only in register mode)
TX-FIFO	Transmitted data	Byte	8	Controller and target
RX-FIFO	Received data			

64.3.4 I3C triggers

This feature is not available in this product: no hardware trigger signal is connected as an input to the I3C peripheral.

64.3.5 I3C interrupt(s)

The interrupt mapping is implemented as detailed in *Table 623*.

Table 623. I3C interrupt(s)

Signal name	Signal type	Description
i3c_err_it	O	Error interrupt line
i3c_evt_it	O	Event interrupt line

64.3.6 I3C MIPI® support

The I3C peripheral supports the MIPI specification v1.1, as defined in *Table 624*.

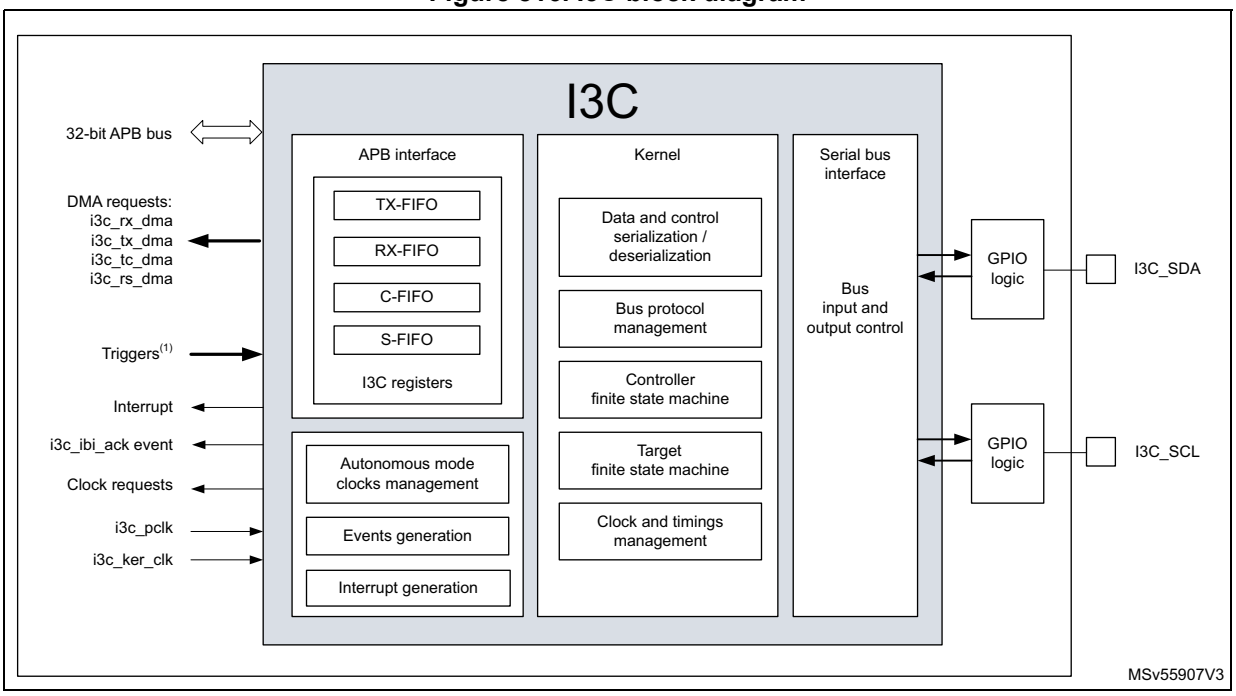
Table 624. I3C peripheral controller/target features versus MIPI v1.1

Feature	MIPI I3C v1.1	I3C peripheral		Comments
		When controller	When target	
I3C SDR message	X	X	X	-
Legacy I^2C message (Sm/Fm/Fm+)	X	X	-	Mandatory when controller if the I3C bus is mixed with (external) legacy I^2C target(s). Optional in MIPI v1.1 when target.
HDR DDR message	X	-	-	Optional in MIPI v1.1
HDR-TSL/TSP, HDR-BT	X	-	-	Optional in MIPI v1.1
Dynamic address assignment	X	X	X	Supported broadcast ENTDAA when controller and target. Supported broadcast SETAASA and direct SETDASA when controller.
7-bit static I^2C address	X	X	-	No support of the I3C peripheral as target on an I^2C bus.
Grouped addressing	X	X	-	Optional in MIPI v1.1
CCCs	X	X	X	Mandatory and some optional CCCs supported (refer to *Table 630* when controller/target).
Error detection and recovery	X	X	X	-
In-band interrupt (with MDB)	X	X	X	-
Secondary controller	X	X	X	-
Hot-join mechanism	X	X	X	-
Target reset	X	X	X	-
Synchronous timing control	X	X	-	Optional in MIPI v1.1
Asynchronous timing control 0	X	X	-	Mandatory in MIPI v1.1 when controller (refer to *Timing control and sensor acquisition timestamping when I3C controller*). Optional in MIPI v1.1 when target.
Asynchronous timing control 1, 2, 3	X	-	-	Optional in MIPI v1.1
Device to device tunneling	X	X	-	Optional in MIPI v1.1
Multi-lane data transfer	X	-	-	Optional in MIPI v1.1
Monitoring device early termination	X	-	-	Optional in MIPI v1.1

64.4 I3C block diagram

The I3C block diagram is illustrated in *Figure 816*.

Figure 816. I3C block diagram



1. Refer to *Section 64.3.4: I3C triggers*.

64.5 I3C pins and internal signals

Table 625. I3C input/output pins

Pin name	Pin type	Description
I3C_SDA	Input/output	Serial bus data line
I3C_SCL	Input/output	Serial bus clock line

Note: During initialization, the selected GPIOs for I3C_SCL and I3C_SDA should be configured as alternate function and no pull.

Table 626. I3C internal input/output signals

Signal name	Signal type	Description
i3c_pclk	I	APB clock
i3c_ker_clk	I	Kernel clock (also named as I3CCLK)
i3c_pclk_req	O	APB clock request
i3c_ker_clk_req	O	Kernel clock request
i3c_it(1)	O	Global interrupt line

Table 626. I3C internal input/output signals (continued)

Signal name	Signal type	Description
i3c_err_it(1)	O	Error interrupt line
i3c_evt_it(1)	O	Event interrupt line
i3c_rx_dma	O	DMA request for reading received bytes/words from RX-FIFO
i3c_tx_dma	O	DMA request for writing to be transmitted bytes/words to TX-FIFO
i3c_tc_dma	O	DMA request for writing to be transmitted control words to C-FIFO, when the I3C peripheral acts as controller
i3c_rs_dma	O	DMA request for reading status words from S-FIFO, when the I3C peripheral acts as controller
i3c_ibi_ack	O	IBI acknowledge event when the I3C peripheral acts as controller, for timing control with asynchronous mode 0

1. Refer to *Section 64.3.5: I3C interrupt(s)*.

64.6 I3C reset and clocks

64.6.1 I3C reset

On a system reset, the I3C peripheral is reset.

Alternatively, the software can reset specifically the I3C peripheral by writing the corresponding reset control bit (I3CxRST) of the reset and clock controller (RCC). Refer to the RCC section of this document for more details.

Additionally, when acting as target, the enabled I3C peripheral (EN = 1 in the I3C_CFGR register) can receive an in-band reset pattern on the I3C bus from the controller. The software is then notified (when RSTF = 1 in the I3C_EVR register and/or the corresponding interrupt is enabled) to perform the requested action, as registered in RSTACT[1:0] of the I3C_DEVR0 register, on the former reception of the broadcast or direct RSTACT CCC. Refer to *Table 630* and *Section 64.16.16* for more details.

This reset interrupt notification can be used to wake up from a low power mode.

For more details about the corresponding low-power mode(s), refer to the power management in the PWR section.

64.6.2 I3C clocks and requirements

As indicated in *Figure 816*, the I3C peripheral is implemented with several clock domains:

- SCL bus clock: for the I3C bus interface
  - When controller: the user must set and can adjust SCL/SDA timings by programming *I3C timing register 0 (I3C_TIMINGR0)*, *I3C timing register 1 (I3C_TIMINGR1)* and *I3C timing register 2 (I3C_TIMINGR2)*, as summarized in *Controller initialization* and *Updating the configuration for a transfer, as controller*.
  - When target: the user must set and comply with the bus available condition (t_{AVAL} for an in-band interrupt or controller-role request), and the bus idle condition (t_{IDLE} for a hot-join request), by programming *I3C timing register 1 (I3C_TIMINGR1)*, as summarized in *Target initialization*.

- I3CCLK kernel clock: for the I3C protocol management, data and control serialization/deserialization, controller and target finite state machines, bus clock and timings management
- APB clock: for the APB interface, DMA interface, events, and interrupt generation

APB clock and kernel clocks are driven from independently programmed clock sources via the RCC (refer to section Reset and clock control (RCC) for more details).

I3C kernel clock requirement, as controller

According to the intended value of the SCL clock on the bus, the application must guarantee that the frequency of the I3CCLK kernel clock be at least 2x the frequency of the SCL clock

Note: Sustaining $F_{SCL\ max}$ = 12.9 MHz means a frequency of the I3CCLK kernel clock > 25.8 MHz.

I3C kernel clock requirements, as target

According to the intended value of the SCL clock on the bus, the application must guarantee a minimum operating frequency for the I3CCLK kernel clock, meeting the following constraints:

1. Period of the I3CCLK kernel clock < t_{HIGH} (SCL clock high period)
   - $t_{HIGH\ min}$ = 24 ns. A frequency higher than 41.7 MHz guarantees this constraint, which can be relaxed, depending on the I3C bus/controller
2. Period of the I3CCLK kernel clock < t_{CASr} (clock after repeated start condition)
   - $t_{CASr\ min}$ = $t_{CAS\ min}$ / 2 = 19.2 ns. A frequency higher than 52 MHz guarantees this constraint, which can be relaxed, depending on the I3C bus/controller
3. Two periods of the I3CCLK kernel clock < t_{LOW_OD} (SCL clock low period in open drain)
   - $t_{LOW_OD\ min}$ = 200 ns. A frequency higher than 10 MHz guarantees this constraint, which can be relaxed, depending on the I3C bus/controller
4. Frequency of the I3CCLK kernel clock > 2.5x frequency of the SCL clock
   - $F_{SCL\ max}$ = 12.9 MHz. A frequency higher than 32.3 MHz guarantees this constraint, which can be relaxed, depending on the I3C controller

APB clock requirement

According to the intended value of the SCL clock on the bus, the application must guarantee a minimum operating frequency for the APB clock:

APB clock period < 3x (SCL clock period) - I3CCLK kernel clock period

This means that $F_{APB} > [F_{SCL} \times F_{I3CCLK} / (3x\ F_{I3CCLK} - F_{SCL})]$

Note: This equation can be simplified to a minimum value of 5 MHz for the APB frequency.

64.7 I3C peripheral state and programming

64.7.1 I3C peripheral state

The I3C peripheral plays the role of I3C bus controller, or the role of an I3C target. In any case (see *Figure 817* and *Figure 818*), the peripheral is in one of the following states:

- Disabled state:
  - After an I3C reset (system reset or I3C reset from RCC), the I3C peripheral is in disabled state.
  - When the software sets to 1 bit EN in the I3C_CFGR register, the I3C peripheral takes into account the value of the different initialized configuration registers, and switches to the (enabled and) idle state. The peripheral is initialized as I3C target or I3C controller.
  - As I3C controller, then software can enable the I3C peripheral and initialize the I3C bus.
  - As I3C target, then the software can enable the I3C peripheral to operate on an I3C bus.
- Idle state:
  - After being enabled (EN = 1), the I3C peripheral activates its I3CCLK and SCL clock domains, and is able to communicate on the I3C bus.
  - The software can partly update the I3C peripheral configuration, see *Updating the configuration for a transfer, as controller* and *Updating the configuration of the I3C peripheral, as target* for more details.
  - The I3C peripheral switches to the (enabled and) active state in either one of the following conditions:

    Once the software initiates a transfer (as controller: when the software initiates a frame transfer; as target: when the software initiates an IBI/CR/HJ request).

    Once the hardware receives a request from another I3C device on the bus (as controller: after a start request from a target and a maximum T_{CAS} time; as target: when receiving a broadcast/direct CCC or a private read/write).
- Active state:
  - The I3C peripheral executes the transfer(s) on the bus.
  - When the requested transfer(s) is(are) completed, the software is notified by an event from the *I3C event register (I3C_EVR)*, and the corresponding interrupt is enabled by *I3C interrupt enable register (I3C_IER)*. The I3C peripheral switches to idle state, is still able to communicate on the bus, and can be (partly) reconfigured:

    As controller: the raised event/flag can be frame completed (FCF), IBI/controller-role/hot-join request completed (IBIF/CRF/HJF), or transfer error (ERRF).

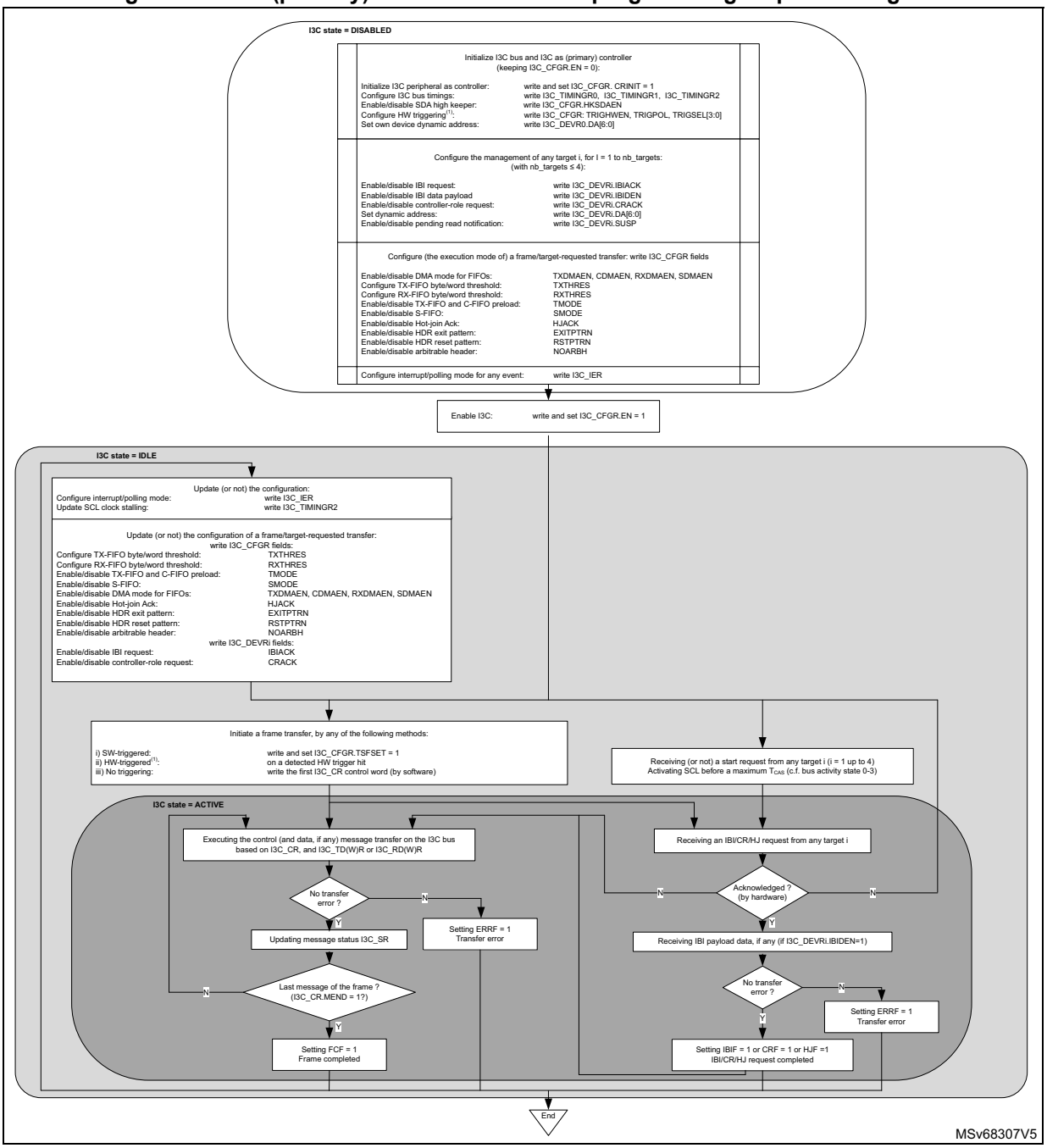
    As target: the raised event/flag can be dynamic address assignment completed (DAUPDF), IBI completed (IBIENDF), controller-role gaining completed (CRUPDF), broadcast/direct CCC completed (xxUPDF/RSTF/GETF/STAF), private read/write completed (FCF), or transfer error (ERRF).

Note: The software can disable the I3C peripheral (write EN = 0), partially resetting it (subparts within the SCL clock domain and the I3CCLK kernel clock domain). Event, interrupt, and clock request generation are also impacted. The previously written configuration of the APB registers is kept and not modified.

64.7.2 I3C controller state and programming sequence

Figure 817 illustrates the overall programming sequence of the I3C peripheral acting as (primary) controller, including state transitions, main subtasks, and conditions, as explained in this section.

Figure 817. I3C (primary) controller state and programming sequence diagram



1. This feature is implementation-dependent and can be unavailable. Refer to *Section 64.3.4: I3C triggers*.

Controller initialization

When the controller is in disabled state (EN = 0 in the I3C_CFGR register), the software must initialize as follows:

- Configure *I3C configuration register (I3C_CFGR)* with the following fields:
  - CRINIT = 1: as I3C bus controller
  - HKSDAEN: high keeper on SDA enable/disable
- Configure I3C bus timings:
  a) *I3C timing register 0 (I3C_TIMINGR0)*:
  - SCL clock high time period (t_{DIG_H}, $t_{DIG_H_MIXED}$) in legacy I^2C and I3C open-drain/push-pull
  - SCL clock low time period (t_{DIG_L}, $t_{DIG_OD_L}$) in legacy I^2C and I3C open-drain/push-pull

  b) *I3C timing register 1 (I3C_TIMINGR1)*:
  - SDA hold time in push-pull (t_{HD_PP})
  - bus free condition time (I3C $t_{CAS,}$ legacy I^2C t_{BUF})
  - I3C repeated start timing (t_{CASr}, t_{CBSr})
  - I3C stop timing (t_{CBP})
  - SCL clock low maximum stalling on the ENTDAA CCC ($t_{STALLDAA}$), or on the ACK/NACK of a legacy I^2C or address phase of an I3C transfer, or the parity bit of a write data transfer, or on the ACK/NACK data phase of a legacy I^2C write, or on the transition bit of an I3C read transfer, or on the ACK/NACK phase of a legacy I^2C write (t_{STALL}), to adjust SCL clock low stalling, if needed by the I3C peripheral itself, when used as controller
  - $t_{NEWCRLock}$ for controller-role hand-off procedure (after GETACCCR CCC)

  c) *I3C timing register 2 (I3C_TIMINGR2)*:
  - SCL clock low stalling time, with separated enable/disable for each phase, to adjust SCL clock low stalling, if needed on the SDA hand-off with the addressed I3C target or legacy I^2C target
- Configure its own dynamic address: DA [6:0] field of the *I3C own device characteristics register (I3C_DEVR0)*
- Configure the management of any device target x: *I3C device x characteristics register (I3C_DEVRx)*, for x = 1 to x ≤ 4
- Configure the execution mode of a frame transfer or a target-requested transfer: *I3C configuration register (I3C_CFGR)*, with the following fields:
  - TXDMAEN, CDMAEN, RXDMAEN, SDMAEN: DMA mode enable/disable for respectively, TX-FIFO, C-FIFO, RX-FIFO, S-FIFO
  - TXTHRES, RXTHRES: respectively TX-FIFO and RX-FIFO byte/world threshold
  - TMODE: transmit mode (enable/disable for both TX-FIFO and C-FIFO preload)
  - SMODE: S-FIFO enable/disable
  - EXITPTRN, RSTPTRN: exit, reset pattern enable/disable
  - HJACK: hot-join acknowledge enable/disable
  - NOARBH: arbitrable header disable/enable
- Configure interrupt generation or polling mode from any event: *I3C interrupt enable register (I3C_IER)*

Then, the software can enable the I3C peripheral (set EN = 1).

Note: The software can write once all the fields of the I3C_CFGR while enabling it.

Start a controller-initiated frame transfer

When the controller is in enabled state (EN = 1 in the I3C_CFGR register), the software can initiate a frame transfer by any of the following configuration methods:

- Software-triggering: on a write and set TSFSET = 1 in the I3C_CFGR register
  - This causes the hardware to raise the flag CFNFF = 1 in the I3C_EVR register, to request a first control word I3C_CR to be written.
- No triggering: on a write of the first control word I3C_CR by software.

Then, regardless of the frame starting method, the I3C peripheral switches to active state. While the control word is not the last message of the I3C frame (while MEND = 0 in the I3C_CR register), and while there is no transfer error (while ERRF = 1 in the I3C_EVR register), the hardware keeps requesting a next control word, and continuing the frame transfer:

- If the C-FIFO in not configured in DMA mode (CDMAEN = 0 in the I3C_CFGR register), the software writes a next control word following the flag CFNFF = 1 in the I3C_EVR register, or the corresponding interrupt if enabled (if CFNFIE = 1 in I3C_IER register)
- If the C-FIFO in configured in DMA mode (CDMAEN = 1), a next control word is automatically pushed and written by the allocated DMA channel consequently to the asserted I3C DMA request (i3c_tc_dma).

Start and receiving a target-initiated transfer

When the controller is in enabled state (EN = 1 in the I3C_CFGR register), concurrently to a possible controller-initiated transfer, a target can initiate a transfer by issuing a start request (drive SDA low), provided the controller has allowed a hot-join request, an IBI request, or a controller-role request via the I3C_DEVR0 register.

In this case, even though the controller software has no intent to start a frame transfer, the hardware switches to active state (activates the SCL clock before a maximum t_{CAS} time defined as 1 µs, 100 µs, 2 ms, or 50 ms, depending on, respectively, the bus activity state 0, 1, 2, or 3) to receive the hot-join/in-band interrupt/controller-role request from the target.

For more information about the execution of target-initiated I3C bus transfer and its related programming as a controller, refer to the relevant figures in *Section 64.9*:

- *Figure 829: IBI transfer, as controller/target*
- *Figure 830: Hot-join request transfer, as controller/target*
- *Figure 831: Controller-role request transfer, as controller/target*

Executing a (controller-initiated) frame transfer

The controller executes on the bus the frame transfer until the completion of the last message (FCF = 1 in the I3C_EVR register), or a transfer error (ERRF = 1 in the I3C_EVR register), and the corresponding interrupt, if enabled. This is based on I3C_CR and I3C_TD(W)R registers, written explicitly by software or pushed by the allocated DMA channel, and based on the I3C_RD(W)R, read explicitly by the software or by the allocated DMA channel. Then the I3C controller switches back to idle state.

For more information about the execution of controller-initiated I3C bus transfer and its related programming as a controller, refer to figures in *Section 64.9*:

- *Figure 819: I3C CCC messages, as controller*
- *Figure 820: I3C broadcast ENTDAA CCC, as controller*
- *Figure 821: I3C broadcast, direct read and direct write RSTACT CCC, as controller*
- *Figure 826: I3C private read/write messages, as controller*
- *Figure 828: Legacy I2C read/write messages, as controller*

Figure 817 does not include the management of the FIFOs (TX-FIFO, RX-FIFO, C-FIFO, and S-FIFO). This is detailed in *Section 64.10*.

For each completed message without transfer error, the hardware reports the exchanged transfer on the I3C bus by updating *I3C status register (I3C_SR)*, which can be read or not by the software when the S-FIFO is disabled (SMODE = 0 in the I3C_CFGR register).

- In the case of a direct CCC read or a private read transfer, in addition to the completion of the last message (FCF = 1 in the I3C_EVR register) or a transfer error (ERRF = 1 in the I3C_EVR register) and the corresponding interrupt if enabled, and provided that the S-FIFO is disabled for the status register I3C_SR (SMODE = 0 in the I3C_CFGR register), the software is notified if the read transfer is ended prematurely by the target by RXTGTENDF = 1 in the I3C_EVR register, and the corresponding interrupt, if enabled. The software can then read I3C_SR, to get more information about the executed transfer.

Alternatively, if the S-FIFO is enabled (SMODE = 1 in the I3C_CFGR register), the status register I3C_SR must be read for each executed message, either directly by the software (notified by SFNEF = 1 in the I3C_EVR register and the corresponding interrupt, if enabled), or via the DMA (if SDMAEN = 1 in the I3C_CFGR register), no matter if a read is prematurely ended by the target or not. Frame completion (FCF = 1 in the I3C_EVR register) occurs only after reading the status of the last message (S-FIFO is empty). For more information, refer to *Section 64.10.4*.

Updating the configuration for a transfer, as controller

Back in idle state, the software can update the configuration of the I3C peripheral before the next transfer:

- Modify SCL clock stalling via *I3C timing register 2 (I3C_TIMINGR2)*
- Modify the interrupt/polling mode policy via *I3C interrupt enable register (I3C_IER)*
- Modify the following fields of the *I3C configuration register (I3C_CFGR)*:
  - TXTHRES, RXTHRES
  - TMODE, SMODE
  - TXDMAEN, CDMAEN, RXDMAEN, SDMAEN
  - EXITPTRN, RSTPTRN
  - NOARBH

- Modify/prepare the control words, status words, read/write data of the next frame transfer to be executed, by software and/or DMA
  - *I3C message control register (I3C_CR)*, *I3C message control register [alternate] (I3C_CR)*
  - *I3C status register (I3C_SR)*
  - *I3C transmit data byte register (I3C_TDR)*, *I3C transmit data word register (I3C_TDWR)*
  - *I3C receive data byte register (I3C_RDR)*, *I3C receive data word register (I3C_RDWR)*
- Typically after having issued and completed a broadcast/direct DISEC/ENEC CCC:
  - Modify the hot-join acknowledge policy via bit HJACK in the I3C_CFGR register
  - Modify IBI/CR acknowledge policy to any target x, via *I3C device x characteristics register (I3C_DEVRx)*

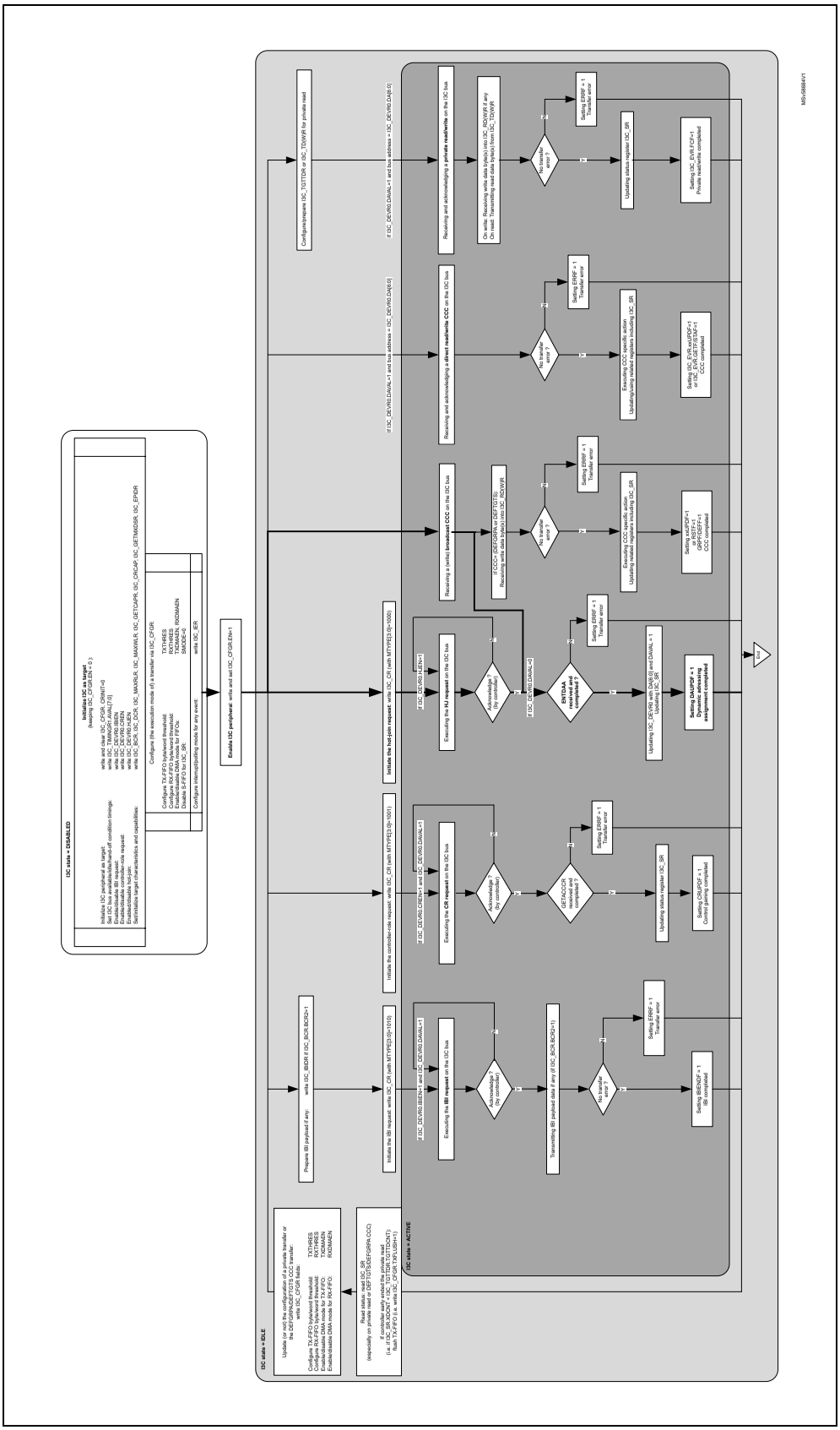
The registers usage versus the I3C peripheral role as controller is summarized in *Section 64.8.1*.

The static/dynamic registers fields usage when acting as controller is summarized in *Table 628*.

64.7.3 I3C target state and programming sequence

Figure 818 illustrates the overall programming sequence of the I3C peripheral acting as target, including state transitions, main subtasks, and conditions, as explained in this section.

Figure 818. I3C target state and programing sequence diagram



Target initialization

When the target is in disabled state (EN = 0 in the I3C_CFGR register), the software must initialize as follows:

- Set the I3C peripheral as target: write and clear CRINIT = 0 in the I3C_CFGR register
- Set the I3C bus timings via *I3C timing register 1 (I3C_TIMINGR1)*: write AVAL[7:0] for:
  - Bus available condition time (t_{AVAL}) for IBI or controller-role request
  - Bus idle condition time (t_{IDLE}) for hot-join request
  - $t_{NEWCRLock}$ for controller-role hand-off procedure (after GETACCCR CCC)
- Configure target-initiated requests: write *I3C own device characteristics register (I3C_DEVR0)* with
  - IBIEN: in-band interrupt (also known as IBI) request enable/disable
  - CREN: controller-role request enable/disable
  - HJEN: hot-join request enable/disable
- Initialize target characteristics and capabilities: write
  - *I3C bus characteristics register (I3C_BCR)*
  - *I3C device characteristics register (I3C_DCR)*
  - *I3C maximum read length register (I3C_MAXRLR)*
  - *I3C maximum write length register (I3C_MAXWLR)*
  - *I3C get capability register (I3C_GETCAPR)*
  - *I3C controller-role capability register (I3C_CRCAPR)*
  - *I3C get max data speed register (I3C_GETMXDSR)*
  - *I3C extended provisioned ID register (I3C_EPIDR)*
- Configure the execution mode of a transfer: *I3C configuration register (I3C_CFGR)*, with the following fields:
  - TXDMAEN, RXDMAEN: DMA mode enable/disable for, respectively, TX-FIFO and RX-FIFO
  - TXTHRES, RXTHRES: respectively TX-FIFO and RX-FIFO byte/world threshold
  - Disable S-FIFO: SMODE = 0 (default/reset value)
- Configure interrupt generation or polling mode from any event: *I3C interrupt enable register (I3C_IER)*

Then, the software can enable the I3C peripheral (write and set EN = 1).

Note: The software can write once all the fields of the I3C_CFGR register while enabling it.

Receiving a (broadcast CCC, direct read/write CCC or private read/write) message on the I3C bus

When the target is in idle state (EN = 1 in the I3C_CFGR register), the target is ready to receive a communication message on the I3C bus from the controller, and is ready to switch to the active state.

Typically, first, the active target is receiving a broadcast ENTDAA CCC, possibly after optional received broadcast ENEC/DISEC CCC(s), and is then assigned a dynamic address. The event DAUPF in the I3C_EVR register is raised to 1, the related interrupt is generated if enabled, and the target goes back to idle state.

After that, the idle target is ready to receive any other broadcast CCC message, or direct read/write CCC, or private read/write message from the controller.

For more information about the execution of controller-initiated I3C bus transfers and its related programming as a target, including the updated I3C registers and fields, refer to figures in *Section 64.9*:

- *Figure 822: I3C CCC messages, as target*
- *Figure 823: I3C broadcast ENTDAA CCC, as target*
- *Figure 824: I3C broadcast DEFTGTS CCC, as target*
- *Figure 825: I3C broadcast DEFGRPA CCC, as target*
- *Figure 827: I3C private read/write messages, as target*

Figure 818 does not include the FIFOs management (TX-FIFO, RX-FIFO). This is detailed in *Section 64.10*.

Read the message status register

For each received and completed message without transfer error, the hardware reports the exchanged transfer on the bus by updating the *I3C status register (I3C_SR)*, which can be read by the software after being notified by the corresponding flag in the *I3C event register (I3C_EVR)*, or by the corresponding interrupt if enabled in the *I3C interrupt enable register (I3C_IER)*.

I3C status register (I3C_SR) must be read by the software after the following messages:

- A private read: to get the number of exchanged data bytes, as the controller can have ended the transfer earlier than expected by the target (if XDCNT[15:0] in the I3C_SR register is lower than TGTTDCNT[15:0] in the I3C_TGTTDR register). If so, software must flush the TX-FIFO (write TXFLUSH = 1 in the I3C_CFGR register).
- A DEFTGTS CCC or a DEFGRPA CCC: to get the number of received data bytes in the RX-FIFO

Start a (target-initiated) transfer on the I3C bus

When the target goes first from disabled to idle state (software writes EN = 1 in the I3C_CFGR register), concurrently to be able to receive a broadcast CCC from the controller, the software can initiate a hot-join request (the software writes MTYPE[3:0] = 1000 in the I3C_CR register) to be eligible to participate to a next ENTDAA CCC, provided it is allowed to do so (HJEN = 1 in the I3C_DEVR0 register).

Once a dynamic address is assigned (DAUPF = 1 in the I3C_EVR register), more generally and possibly concurrently to a frame transfer emitted by the controller, the software can initiate an IBI (in-band interrupt request) to the controller, or a controller-role request by writing the related control word into the I3C_CR register.

For more information about the execution of target-initiated I3C bus transfer, and its related programming as a target, refer to figures in *Section 64.9*:

- *Figure 829: IBI transfer, as controller/target*
- *Figure 830: Hot-join request transfer, as controller/target*
- *Figure 831: Controller-role request transfer, as controller/target*

Updating the configuration of the I3C peripheral, as target

Back in idle state, the software can update the configuration of the I3C target before a next transfer:

- Modify the interrupt/polling mode policy via *I3C interrupt enable register (I3C_IER)*
- Modify following fields of the *I3C configuration register (I3C_CFGR)*:
  - TXTHRES, RXTHRES
  - TXDMAEN, RXDMAEN
- Modify/prepare the *I3C IBI payload data register (I3C_IBIDR)*, if any payload (if BCR2 = 1 in the I3C_BCR register), before initiating an IBI transfer (write *I3C message control register [alternate] (I3C_CR)* with MTYPE[3:0] = 1010)
- Modify/prepare *I3C target transmit configuration register (I3C_TGTTDR)*, to disable or enable the TX-FIFO to be preloaded with a defined number of data bytes to be transmitted, before receiving a private read or a direct CCC read (out of the GETSTATUS CCC)

The registers usage vs. the I3C peripheral role as target is summarized in *Section 64.8.1*.

The static/dynamic registers fields usage, when acting as target, is summarized in *Table 629*.

64.8 I3C registers and programming

64.8.1 I3C register set, as controller/target

Table 627 lists the registers and their usage versus the I3C peripheral role.

Table 627. I3C register usage

Register	Used as controller	Used as target
I3C_CR	X	X
I3C_CFGR	X	X
I3C_RDR	X	X
I3C_RDWR	X	X
I3C_TDR	X	X
I3C_TDWR	X	X
I3C_IBIDR	X	X
I3C_TGTTDR	-	X
I3C_SR	X	X
I3C_SER	X	X
I3C_RMR	X	X
I3C_EVR	X	X
I3C_IER	X	X
I3C_CEVR	X	X
I3C_DEVR0	X	X

Table 627. I3C register usage (continued)

Register	Used as controller	Used as target
I3C_DEVRx (x = 1 to 4)	X	-
I3C_MAXRLR	-	X
I3C_MAXWLR	-	X
I3C_TIMINGR0	X	-
I3C_TIMINGR1	X	X
I3C_TIMINGR2	X	-
I3C_BCR	-	X
I3C_DCR	-	X
I3C_GETCAPR	-	X
I3C_CRCAPR	-	X
I3C_GETMXDSR	-	X
I3C_EPIDR	-	X

64.8.2 I3C registers and fields use versus peripheral state, as controller

When the I3C peripheral acts as controller, *Table 628* lists the registers and their usage versus the controller state (disabled, idle, and active).

Table 628. I3C registers/fields usage versus controller state

Register	Used as controller	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_CR	X	-	-	X

Table 628. I3C registers/fields usage versus controller state (continued)

Register	Used as controller	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_CFGR	X	CRINIT HKSDAEN EN(1)	Write: any used field except {CRINIT, HKSDAEN}, namely {CDMAEN SDMAEN TXDMAEN RXDMAEN TMODE SMODE TXTHRES RXTHRES HJACK EXITPTRN RSTPTRN NOARBH}(2) CFLUSH SFLUSH TXFLUSH RXFLUSH TSFSET	-
I3C_RDR	X	-	-	Read
I3C_RDWR	X	-	-	Read
I3C_TDR	X	-	-	Write
I3C_TDWR	X	-	-	Write
I3C_IBIDR	X	-	-	Read
I3C_TGTTDR	-			
I3C_SR	X	-	-	Read
I3C_SER	X	-	Read	-
I3C_RMR	X	-	-	Read RADD[6:0] IBIRDCNT[2:0]

Table 628. I3C registers/fields usage versus controller state (continued)

Register	Used as controller	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_EVR	X	-	-	Read (controller-role fields): HJF CRF IBIF FCF ERRF RXTGTENDF RXFNEF TXFNFF SFNEF CFNFF RXLASTF TXLASTF TXFEF CFEF
I3C_IER	X	Write xIE with x = HJ, CR, IBI, FC, ERR, RXTGTEND, RXFNE, TXFNF, SFNE, CFNF(2)		
I3C_CEVR	X	-	-	Write (controller-role fields, refer to I3C_EVR)
I3C_DEVR0	X	Write DA[6:0](2)		-
I3C_DEVRx x = 1..4	X	Write SUSP, IBIDEN, IBIACK, CRACK, DA[6:0](2)		-
I3C_MAXRLR	-			
I3C_MAXWLR	-			
I3C_TIMINGR0	X	X	-	-
I3C_TIMINGR1	X	X	-	-
I3C_TIMINGR2	X	Write(2)		
I3C_BCR	-			
I3C_DCR	-			
I3C_GETCAPR	-			
I3C_CRCAPR	-			
I3C_GETMXDSR	-			
I3C_EPIDR	-			

1. Bit EN in the I3C_CFGR register is written and set in disabled state (when the same bit is 0). This field can be also written and de-asserted in idle state.
2. These fields are typically written and initialized in disabled state during bus configuration. They are not write-protected when EN = 0, and can be also written and updated in other state(s).

64.8.3 I3C registers and fields usage versus peripheral state, as target

When the I3C peripheral acts as target, *Table 629* lists the registers and their usage versus the I3C target state (disabled, idle, and active).

Table 629. I3C registers/fields usage versus target state

Register	Used as target	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_CR	X	-	MTYPE[3:0] DCNT[2:0]	-
I3C_CFGR	X	CRINIT EN(1)	Write: any used field except CRINIT, namely: {TXDMAEN RXDMAEN TXTHRES RXTHRES}(2) TXFLUSH RXFLUSH	-
I3C_RDR	X	-	-	Read
I3C_RDWR	X	-	-	Read
I3C_TDR	X	-	-	Write
I3C_TDWR	X	-	-	Write
I3C_IBIDR	X	-	Write	-
I3C_TGTTDR	X	-	Write	-
I3C_SR	X	-	-	Read DIR, XDCNT[15:0]
I3C_SER	X	-	Read DOVR, STALL, PERR, CODERR[3:0]	-
I3C_RMR	X	-	-	Read RCODE[7:0]

Table 629. I3C registers/fields usage versus target state (continued)

Register	Used as target	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_EVR	X	-	-	Read (target-role fields): GRPF DEFF INTUPDF ASUPDF RSTF MRLUPDF MWLUPDF DAUPDF STAF GETF WKPF CRUPDF IBIENDF ERRF FCF RXFNEF TXFNFF TXLASTF TXFEF
I3C_IER	X	Write xIE with x = GRP, DEF, INTUPD, ASUPD, RST, MRLUPD, MWLUPD, DAUPD, STA, GET, WKP, CRUPD, IBIEND, ERR, FC, RXFNE, TXFNF(2)		
I3C_CEVR	X	-	-	Write (target-role fields, refer to I3C_EVR)
I3C_DEVR0	X	HJEN CREN IBIEN	Read RSTVAL, RSTACT[1:0], and AS[1:0]	-
I3C_DEVRx x = 1..4	-			
I3C_MAXRLR	X	X	-	-
I3C_MAXWLR	X	X	-	-
I3C_TIMINGR0	-			
I3C_TIMINGR1	X	AVAL[7:0]	-	-
I3C_TIMINGR2	-			
I3C_BCR	X	X	-	-
I3C_DCR	X	X	-	-
I3C_GETCAPR	X	X	-	-
I3C_CRCAPR	X	X	-	-

Table 629. I3C registers/fields usage versus target state (continued)

Register	Used as target	Writable only in disabled state	Typically written/read in idle state	Typically written/read in idle or active state
I3C_GETMXDSR	X	X	-	-
I3C_EPIDR	X	X	-	-

1. Bit EN in the I3C_CFGR register is written and set in disabled state (when the same bit is 0). This field can be also written and de-asserted in idle state.
2. These fields are typically written and initialized in disabled state during I3C bus configuration. They are not write-protected when EN = 0.

64.9 I3C bus transfers and programming

64.9.1 I3C command set (CCCs), as controller/target

The list of the supported I3C command set (for example, list of CCCs, common command codes) and the overview of how they are handled by the I3C peripheral acting as controller or target, is specified in *Table 630*.

Table 630. List of supported I3C CCCs, as controller/target

CCC name	CCC value	Read /write	With/without defining byte With/without sub-command byte	With/without optional data byte(s)	Use as controller	Use as target, raised I3C_EVR event	When target: specific action
Broadcast CCCs							
ENEC	0x00	Write	No defining/sub-command byte	With one data byte (enable target events byte)	X	X, INTUPDF	Update and enable I3C_DEVR0: HJEN, CREN, IBIEN if any
DISEC	0x01			With one data byte (disable target events byte)	X	X, INTUPDF	Update and disable I3C_DEVR0: HJEN, CREN, IBIEN if any
ENTASx x = 0...3	0x02 ... 0x05			No data byte	X	X, ASUPDF	Update I3C_DEVR0.AS[1:0]
RSTDAA	0x06			-	X	X, DAUPDF	Clear I3C_DEVR0.DAVAL = 0
ENTDAA	0x07			-	X	X, DAUPDF if DAVAL = 0	Update I3C_DEVR0: DA[6:0] and set DAVAL = 1
DEFTGTS	0x08			With [1+ 4x (1+ number_of_targets)] x data bytes	X	X, DEFF	Update I3C_RDR/ I3C_RDWR. Refer to *Figure 824*.
SETMWL	0x09			With two data byte	X	X, MWLUPDF	Update I3C_MAXWLR
SETMRL	0x0A			With 2 or 3 data bytes	X	X, MRLUPDF	Update I3C_MAXRLR
ENTTM	0x0B			With one data byte	X	-	-
SETXTIME	0x28		With sub-command byte	Without or with one or more data bytes	X	-	-
SETAASA	0x29		No defining/sub-command byte	No data byte	X	-	-
RSTACT	0x2A		With defining byte (0x00, 0x01 or 0x02)		X	X, RSTF after detected reset pattern	Update I3C_DEVR0: RSTACT[1:0] and set RSTVAL = 1
DEFGRPA	0x2B		No defining/sub-command byte	With several data bytes	X	X, GRPF	Update I3C_RDR/ RDWR. Refer to *Figure 825*.
RSTGRPA	0x2C			No data byte	X	-	-

Table 630. List of supported I3C CCCs, as controller/target (continued)

CCC name	CCC value	Read /write	With/without defining byte With/without sub-command byte	With/without optional data byte(s)	Use as controller	Use as target, raised I3C_EVR event	When target: specific action
Direct CCCs							Action if ACK (if I3C target Address = I3C_DEVR0.DA[6:0] and I3C_DEVR0.DAVAL = 1) (else NACK)
ENEC	0x80	Write	No defining/sub-command byte	With one data byte (enable target events byte)	X	X, INTUPDF	Update and enable I3C_DEVR0: HJEN, CREN, IBIEN if any
DISEC	0x81			With one data byte (disable target events byte)	X	X, INTUPDF	Update and disable I3C_DEVR0: HJEN, CREN, IBIEN if any
ENTASx x = 0...3	0x82...0x85			No data byte	X	X, ASUPDF	Update I3C_DEVR0.AS[1:0]
SETDASA	0x87			No data byte	X	-	-
SETNEWDA	0x88			With one data byte	X	X, DAUPDF if DAVAL = 1	Update I3C_DEVR0: DA[6:0] (and keep DAVAL = 1)
SETMWL	0x89			With two data bytes	X	X, MWLUPDF	Update I3C_MAXWLR
SETMRL	0x8A			With two or three data bytes	X	X, MRLUPDF	Update I3C_MAXRLR
GETMWL	0x8B	Read	No defining/sub-command byte	With two data bytes	X	X, GETF	Return data bytes from I3C_MAXWLR[15:0]. Refer to *Section 64.16.19*.
GETMRL	0x8C			With two or three data bytes	X	X, GETF	Return data bytes from I3C_MAXRLR[15:0] and if I3C_BCR.BCR2 = 1 return third byte from I3C_MAXRLR.IBIP[2:0]. Refer to *Section 64.16.18*.

Table 630. List of supported I3C CCCs, as controller/target (continued)

CCC name	CCC value	Read /write	With/without defining byte With/without sub-command byte	With/without optional data byte(s)	Use as controller	Use as target, raised I3C_EVR event	When target: specific action
GETPID	0x8D	Read	No defining/sub-command byte	With six data bytes	X	X, GETF	Return data bytes from I3C_EPIDR. Refer to *Section 64.16.28*.
GETBCR	0x8E			With one data byte	X	X, GETF	Return data byte from I3C_BCR[7:0]. Refer to *Section 64.16.23*.
GETDCR	0x8F				X	X, GETF	Return I3C_DCR[7:0]. Refer to *Section 64.16.24*.
GETSTATUS	0x90		With or without defining byte (TGTSTAT, PRECR)	With two data bytes (format 1 or format 2 with PRECR)	X	X, STAF if format 1 X, GETF if format 2	Return 2 data bytes, as detailed in *Section 64.9.9*.
GETACCCR	0x91		No defining/sub-command byte	With one data byte	X	X, CRUPDF	Return data byte from I3C_DEVR0.DA[6:0] with parity bit
GETMXDS	0x94		With or without defining byte (WRRDTURN, CRHDLY)	With two data bytes (format 1) or 5 data bytes (format 2 or format 3 with WRRDTURN) or 1 data byte (format 3 with CRHDLY)	X	X, GETF	Return data byte(s) from I3C_GETMXDSR. Refer to *Section 64.16.27*.
GETCAPS	0x95		With or without defining byte (TGTSTAT, CRCAPS)	With 3 data bytes (format 1 or format 2 with TGTSTAT) or two data bytes (format 2 with CRCAPS)	X	X, GETF	Return 3 GETCAPx data bytes from I3C_GETCAPR (refer to *Section 64.16.25*) or Return 2 CRCAPx data bytes from I3C_CRCAPR (refer to *Section 64.16.26*)
D2DXFER	0x97	Write	With defining byte	With defining byte	X	-	-
SETXTIME	0x98		With sub-command byte	With sub-command byte	X		
GETXTIME	0x99	Read	No defining/sub-command byte	No defining/sub-command byte	X		

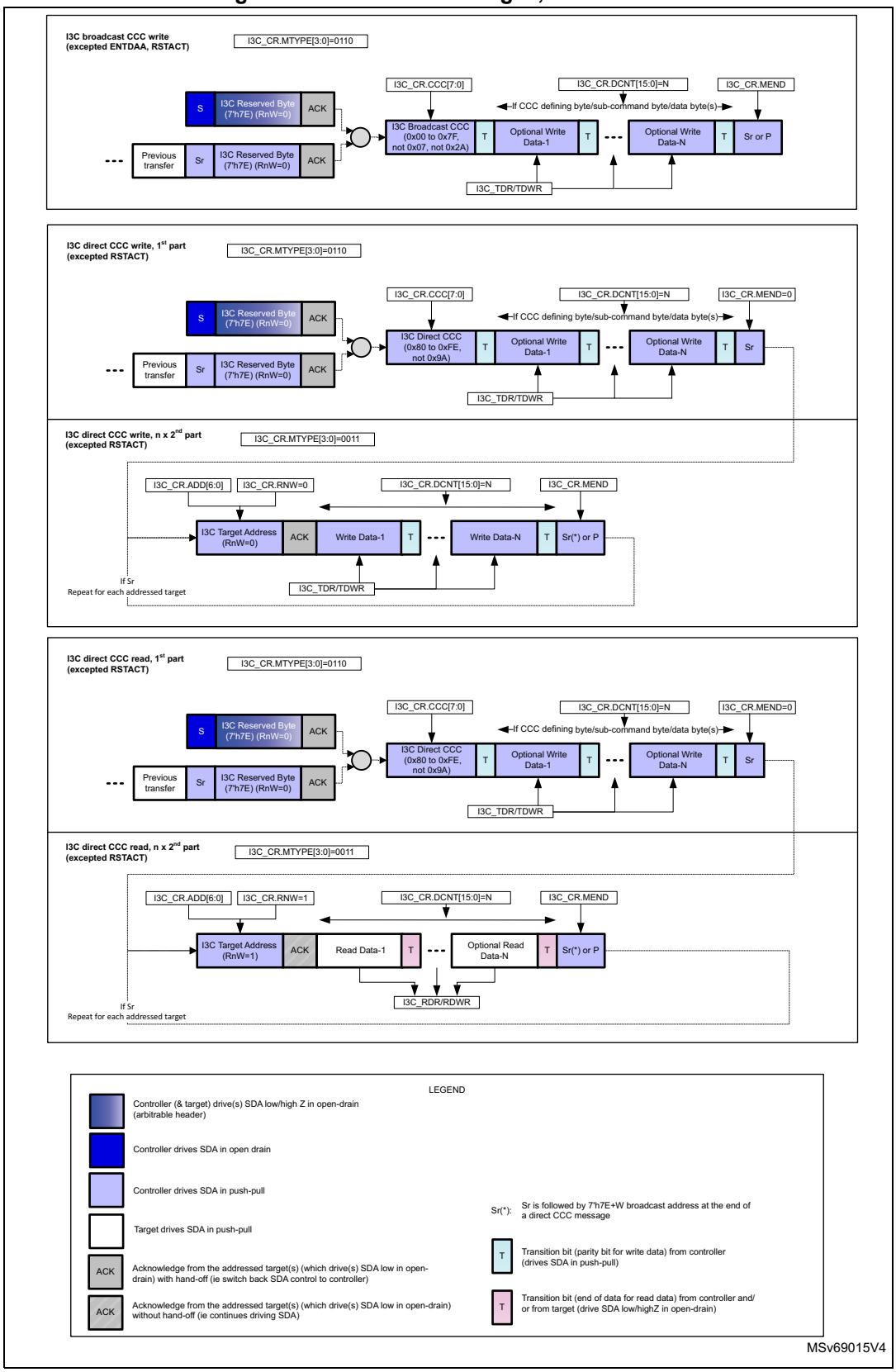
Table 630. List of supported I3C CCCs, as controller/target (continued)

CCC name	CCC value	Read /write	With/without defining byte With/without sub-command byte	With/without optional data byte(s)	Use as controller	Use as target, raised I3C_EVR event	When target: specific action
RSTACT	0x9A	Read/ Write	With defining byte (0x00, 0x01, or 0x02)	With defining byte (0x00, 0x01, or 0x02)	X	X, RSTF if detected reset pattern	Read: return data byte from RSTACT[1:0] in the I3C_DEVR0 register. Write: update I3C_DEVR0: RSTACT[1:0] and set RSTVAL = 1
SETGRPA	0x9B	Write	No defining/sub-command byte	No defining/sub-command byte	X	-	-
RSTGRPA	0x9C				X		

64.9.2 I3C broadcast/direct CCC transfer (except ENTDAA, RSTACT), as controller

Figure 819 illustrates I3C broadcast CCC write transfer (except ENTDAA, RSTACT), and direct CCC read/write transfer, as communicated on the I3C bus, and as programmed when acting as controller.

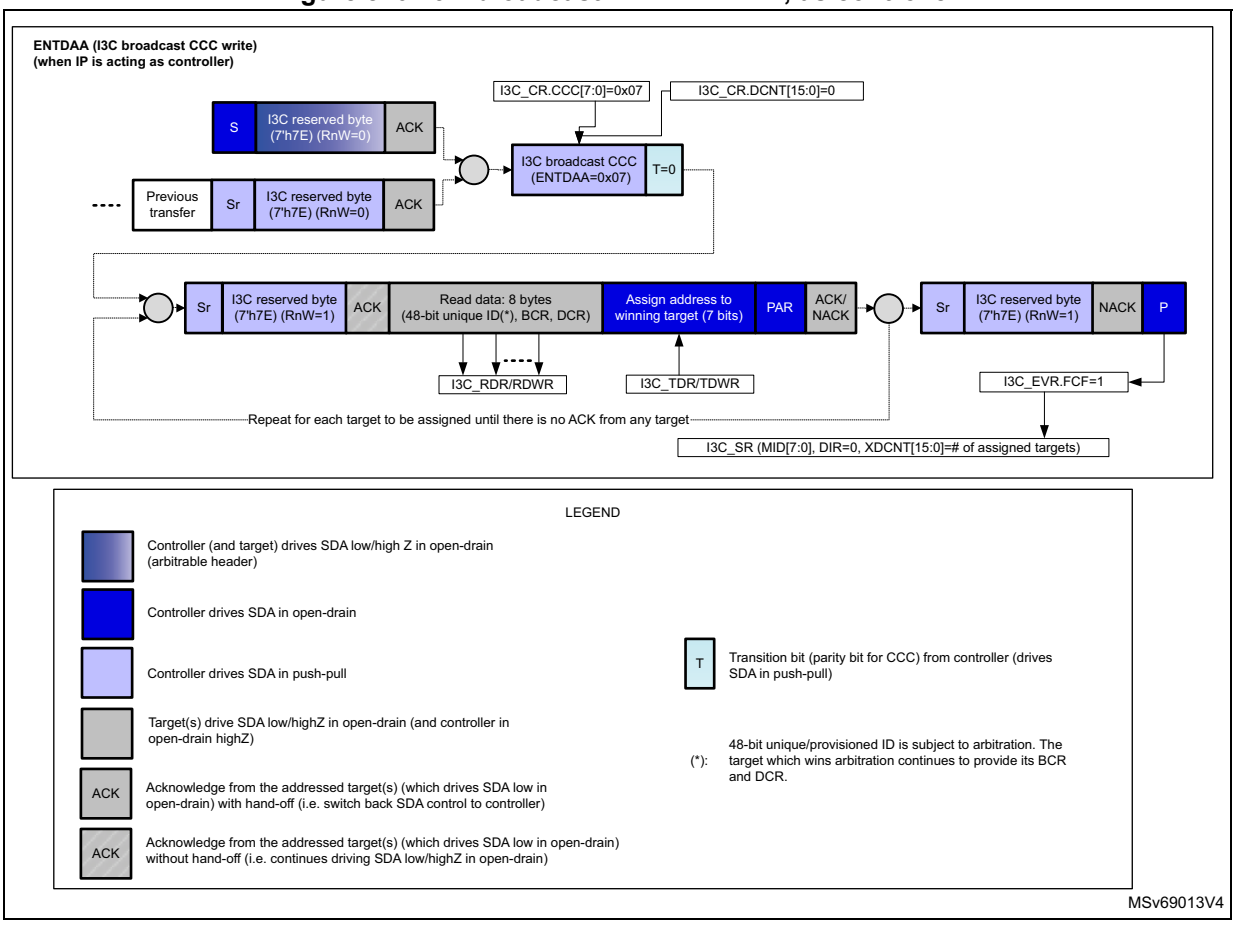
Figure 819. I3C CCC messages, as controller



64.9.3 I3C broadcast ENTDAA CCC transfer, as controller

Figure 820 illustrates I3C broadcast ENTDAA CCC, as communicated on the I3C bus, and as programmed when acting as controller.

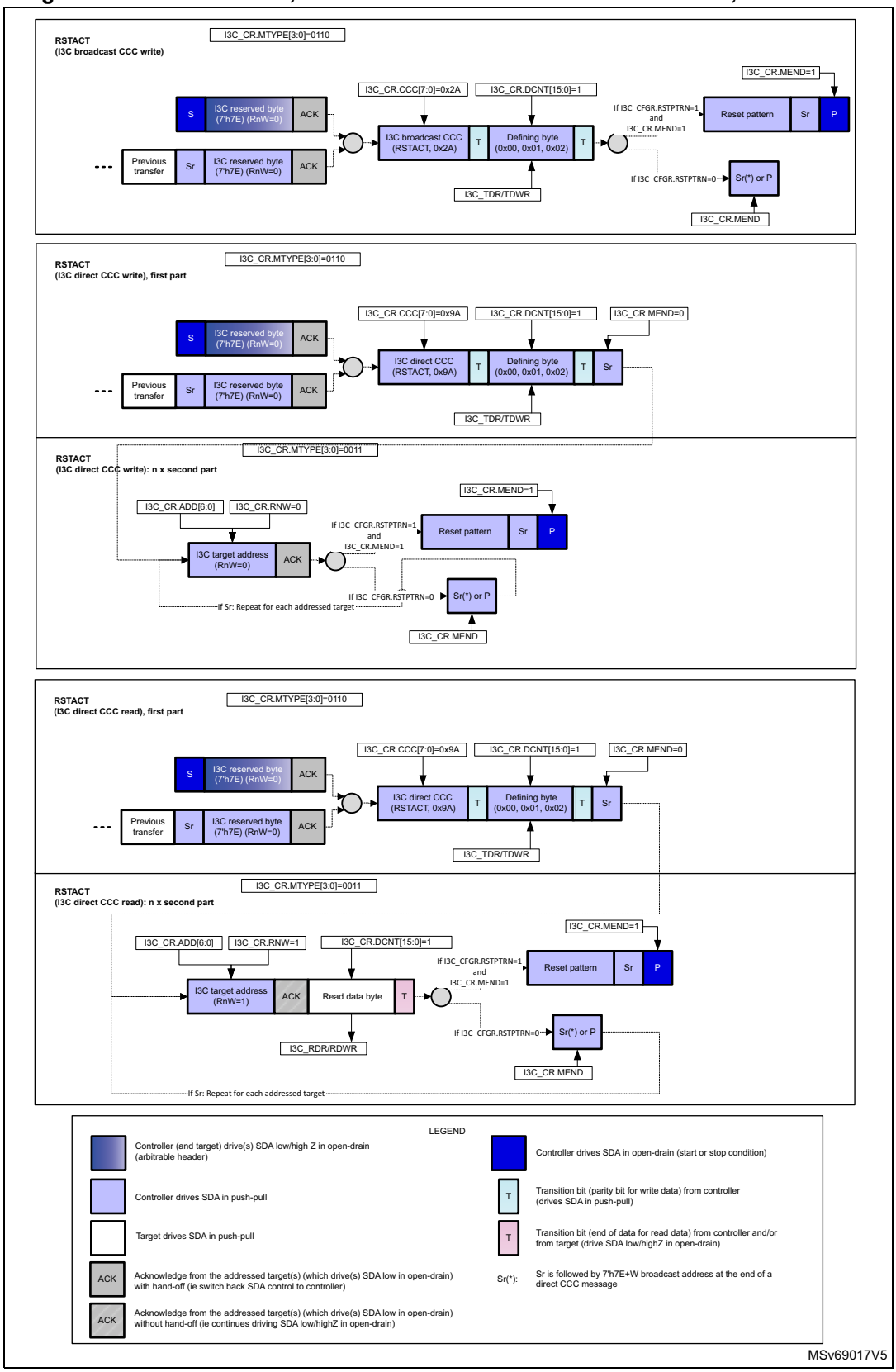
Figure 820. I3C broadcast ENTDAA CCC, as controller



64.9.4 I3C broadcast/direct RSTACT CCC transfer, as controller

Figure 821 illustrates I3C broadcast (write), direct write and read RSTACT CCC, as communicated on the I3C bus, and as programmed when acting as controller.

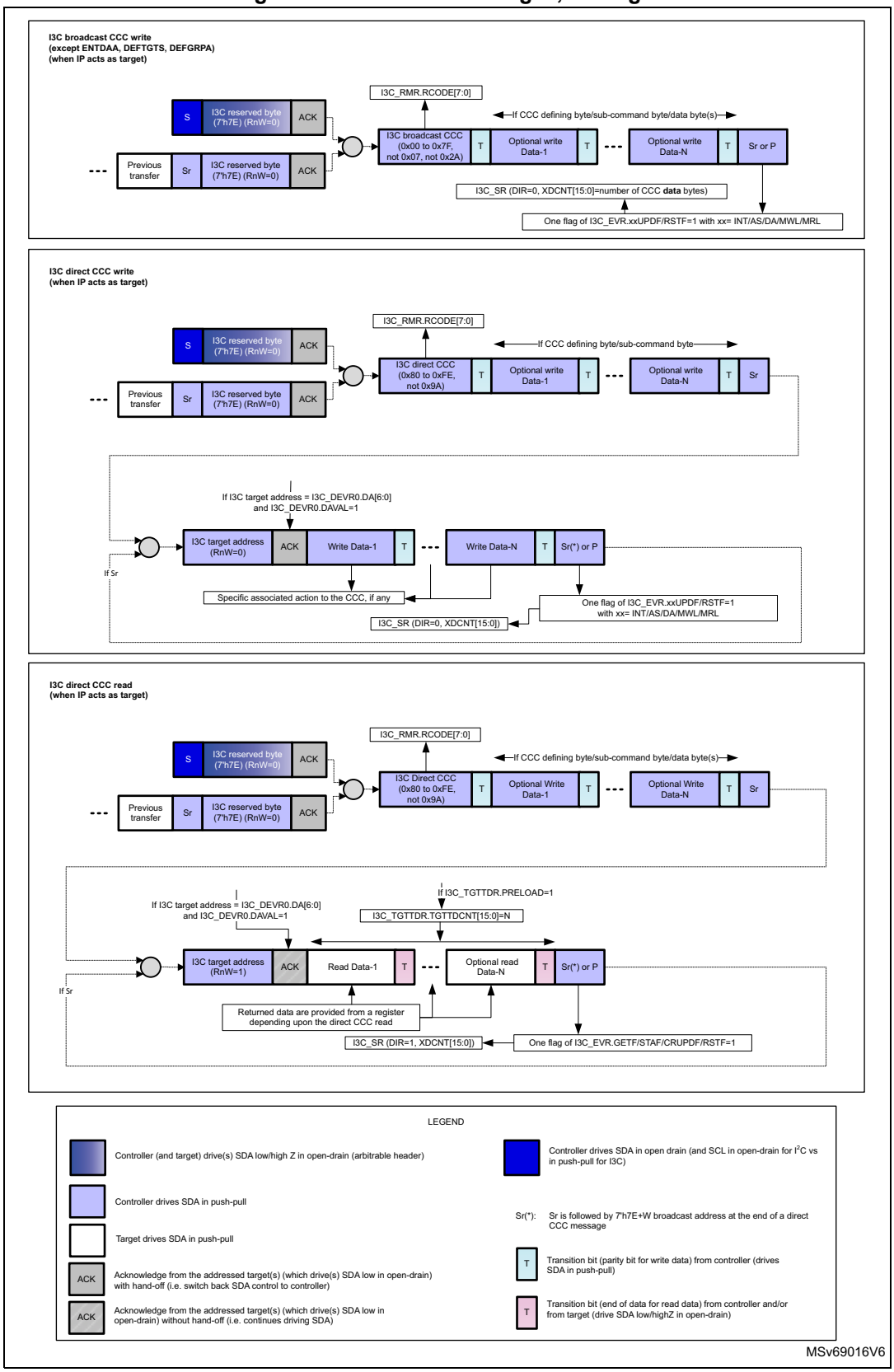
Figure 821. I3C broadcast, direct read and direct write RSTACT CCC, as controller



64.9.5 I3C broadcast/direct CCC transfer (except ENTDAA, DEFTGTS, DEFGRPA), as target

Figure 822 illustrates I3C broadcast CCC write transfer (except ENTDAA, DEFTGTS, DEFGRPA), direct CCC read/write transfer, as communicated on the I3C bus, and as programmed when acting as target.

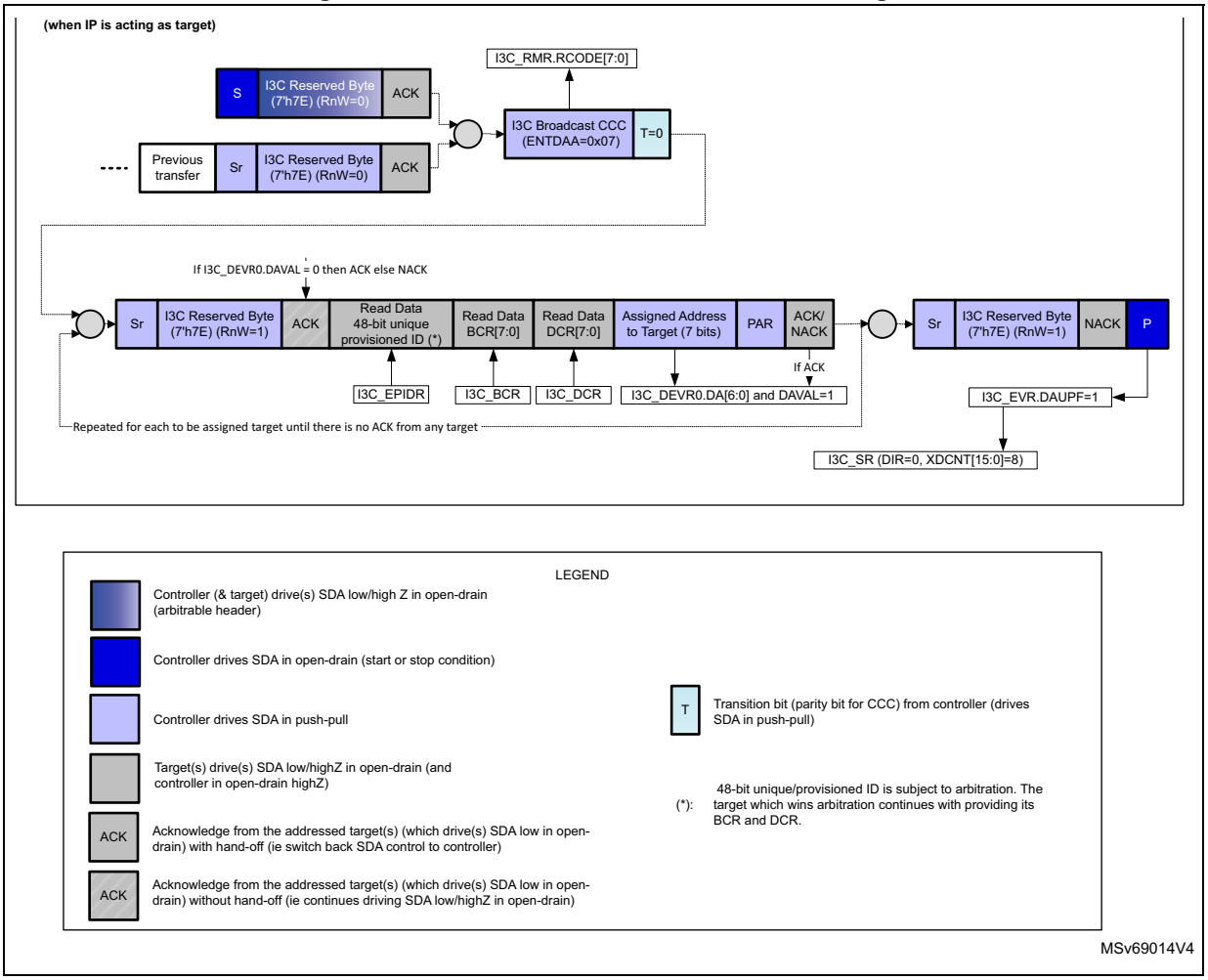
Figure 822. I3C CCC messages, as target



64.9.6 I3C broadcast ENTDAA CCC transfer, as target

Figure 823 illustrates I3C broadcast ENTDAA CCC, as communicated on the I3C bus, and as programmed when acting as target.

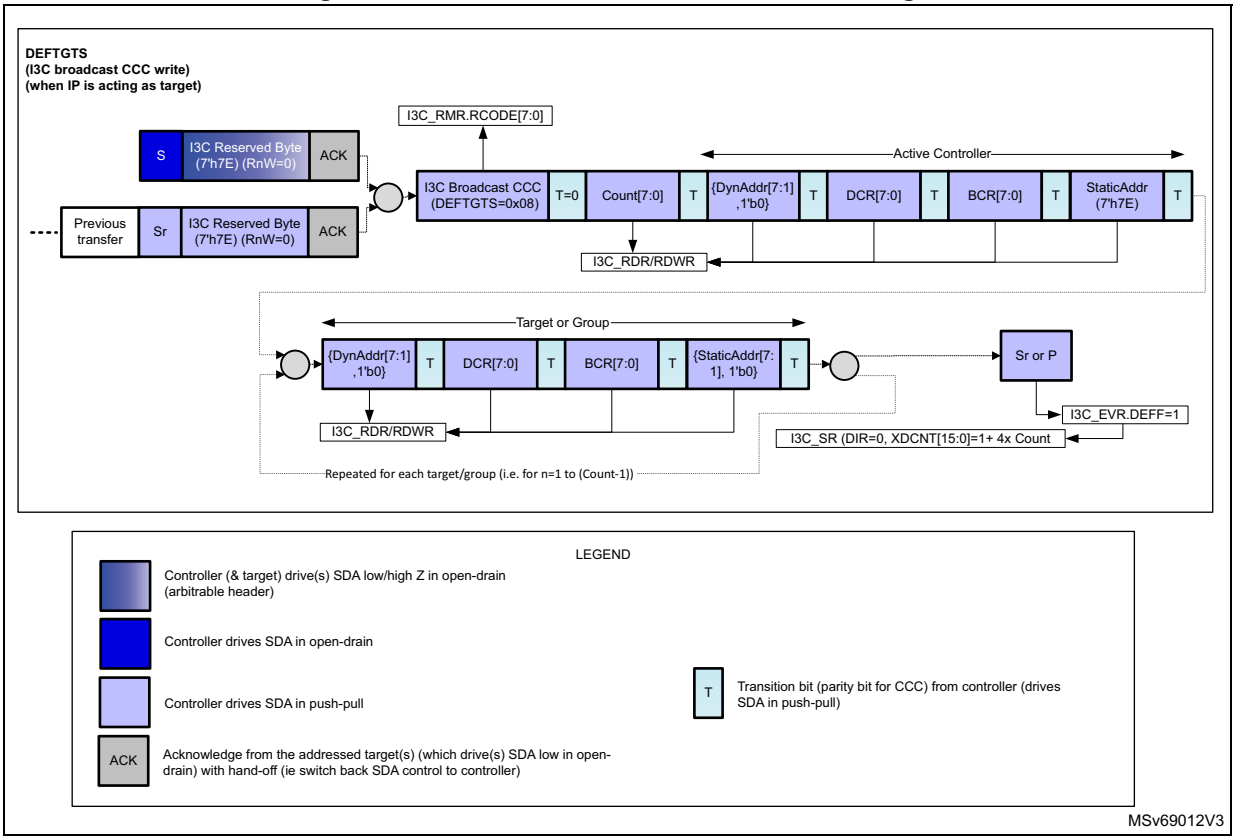
Figure 823. I3C broadcast ENTDAA CCC, as target



64.9.7 I3C broadcast DEFTGTS CCC transfer, as target

Figure 824 illustrates I3C broadcast DEFTGTS CCC, as communicated on the I3C bus, and as programmed when acting as target.

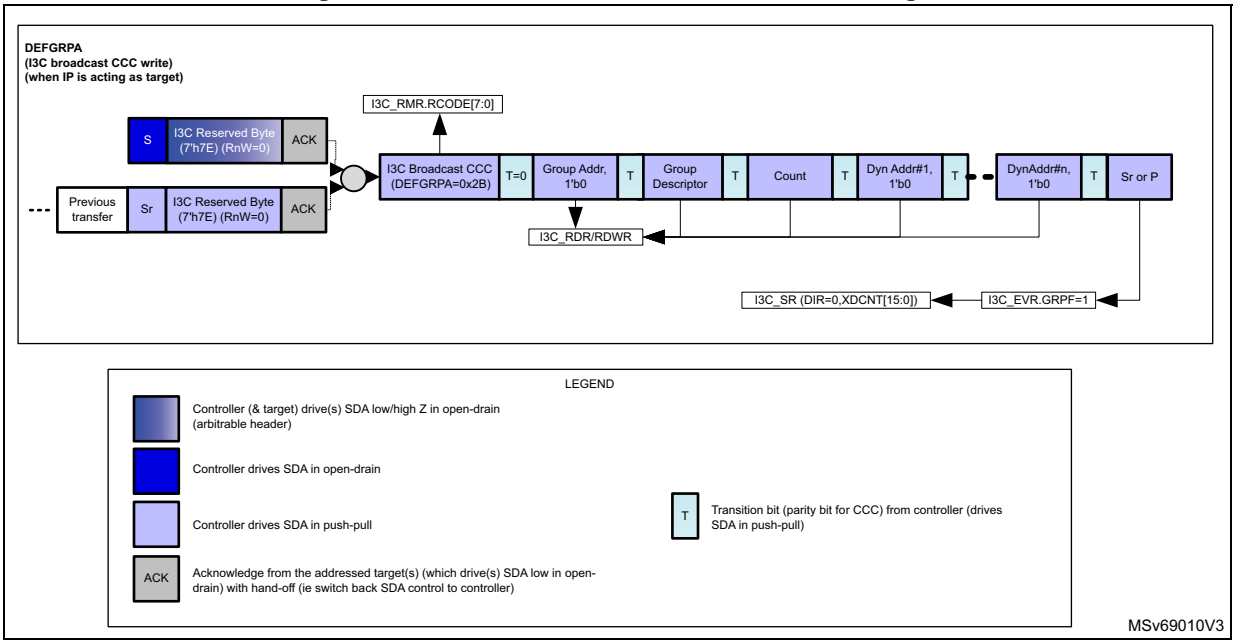
Figure 824. I3C broadcast DEFTGTS CCC, as target



64.9.8 I3C broadcast DEFGRPA CCC transfer, as target

Figure 825 illustrates I3C broadcast DEFGRPA CCC, as communicated on the I3C bus, and as programmed when acting as target.

Figure 825. I3C broadcast DEFGRPA CCC, as target



64.9.9 I3C direct GETSTATUS CCC response, as target

When the I3C acts as target, the hardware returns two data bytes on reception of GETSTATUS CCC, with format 1 (without defining byte or with defining byte TGTSTAT = 0x00), or format 2 (with defining byte PRECR = 0x91).

The returned 2-byte STATUS[15:0] with format 1 on the I3C bus is then as follows:

- STATUS[15:14] = 00 (unused)
- STATUS[13] = 1 if a missed start was detected since the former GETSTATUS CCC, else 0
- STATUS[12] = 1 if an overrun/underrun error was detected since the former GETSTATUS CCC, else 0
- STATUS[11] = 1 if an SCL stable for more than 125 µs was detected during an SDR read since the former GETSTATUS CCC, else 0
- STATUS[10:8] = 000 to 110: encoded value x = 0 to 6, corresponding to a target error TEx if a protocol error was detected since the former GETSTATUS CCC (if STATUS[5] = 1), else 000
- STATUS[7:6] = 00 (ready to prepare for hand-off procedure)
- STATUS[5] = 1 if a protocol error was detected since the former GETSTATUS CCC, else 0
- STATUS[4] = 0 (reserved)
- STATUS[3:1] = 000 (unused)
- STATUS[0] = 1 if there is a pending interrupt (if an IBI is configured in the I3C_CR register, and IBIEN = 1 and DAVAL = 1 in the I3C_DEVR0.register, and the IBI is not yet acknowledged by the controller neither disabled via DISEC), else 0

The returned 2-byte STATUS[15:0] with format 2 on the I3C bus is then as follows:

- STATUS[15:8] = 0000 0000 (unused)
- STATUS[7:2] = 00000 (unused)
- STATUS[1] = 1 if a received DEFTGTS or a received DEFGRPA CCC is still under software processing, and the related event is not yet cleared by software (DEFF = 1 or GRPF = 1 in the I3C_EVR register); the controller must wait before issuing a GETACCR CCC (else it is not acknowledged)
- STATUS[0] = 1 if a DEFTGTS or DEFGRPA CCC may have been missed. This bit is asserted if a missed start is detected (WKPF = 1 in the I3C_EVR register), de-asserted if DEFF = 1 or GRPF = 1 in the I3C_EVR register.

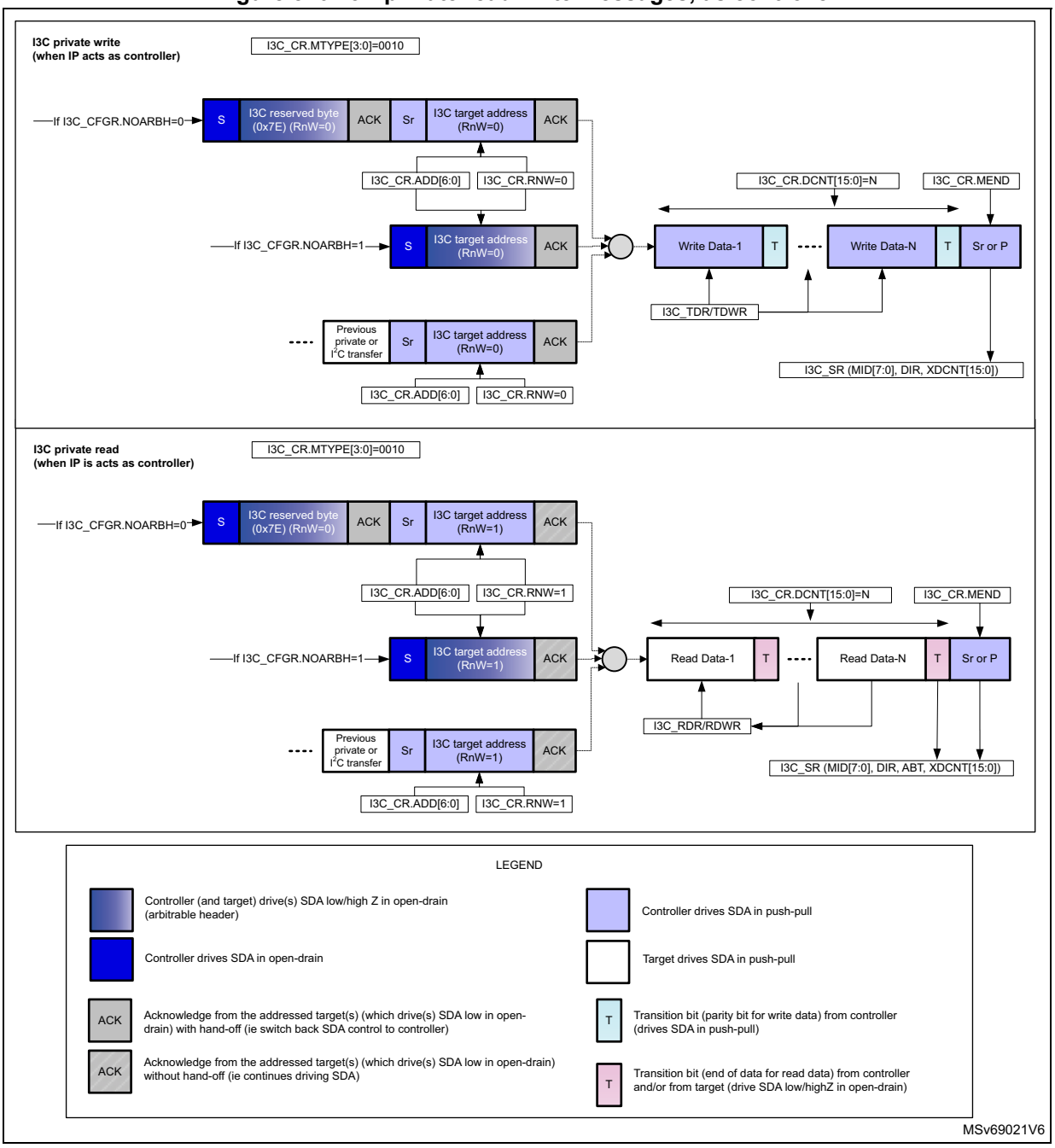
Completion of a GETSTATUS CCC of format 1 is reported by STAF = 1 in the I3C_EVR register, and the corresponding interrupt if enabled (if STAIE = 1 in the I3C_IER register).

Completion of a GETSTATUS CCC of format 2 is reported by GETF = 1 in the I3C_EVR register, and the corresponding interrupt if enabled (if GETIE = 1 in the I3C_IER register).

64.9.10 I3C private read/write transfer, as controller

Figure 826 illustrates private read/write transfer, as communicated on the I3C bus, and as programmed when acting as controller.

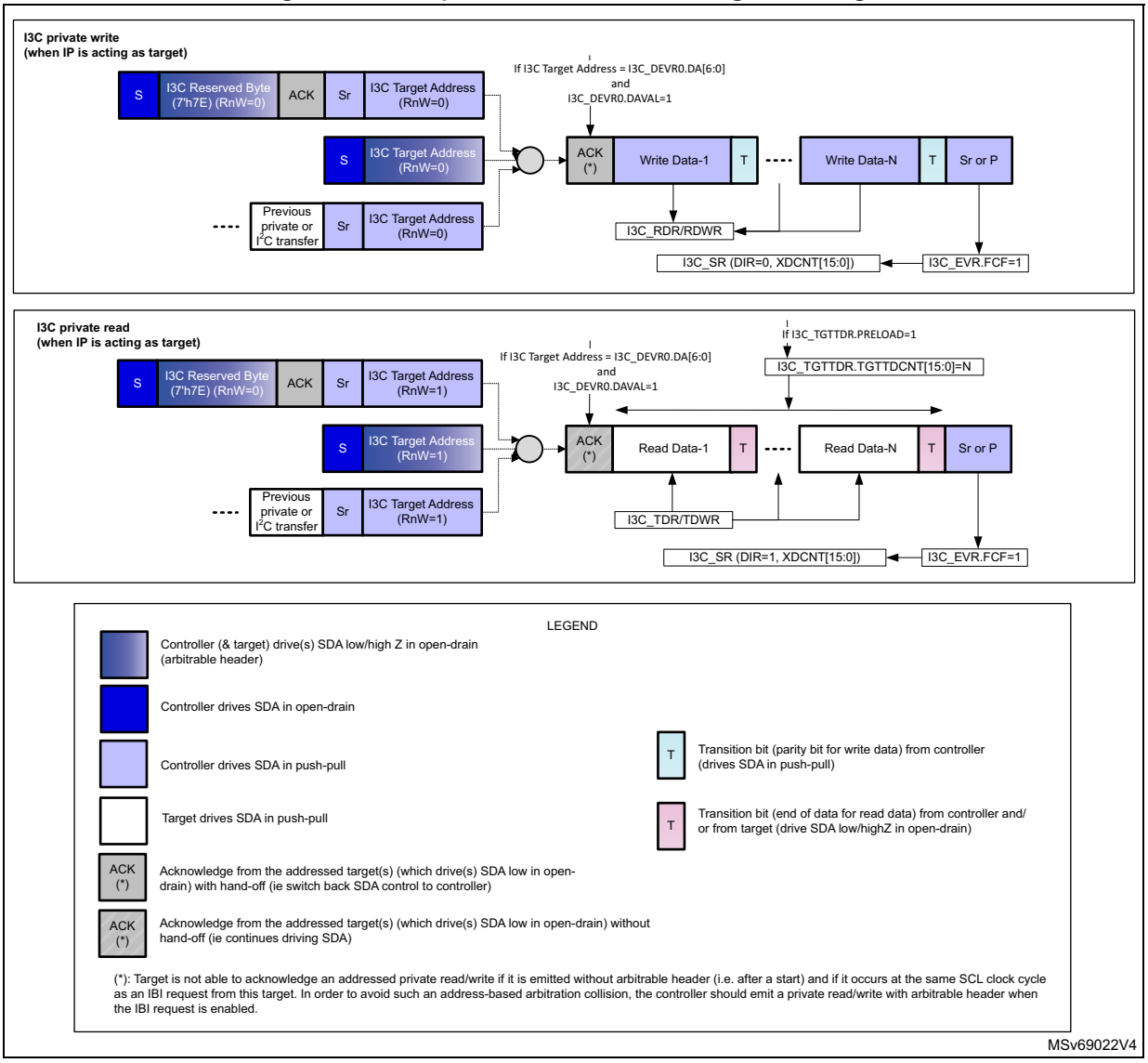
Figure 826. I3C private read/write messages, as controller



64.9.11 I3C private read/write transfer, as target

Figure 827 illustrates I3C private read/write transfer, as communicated on the I3C bus, and as programmed when acting as target.

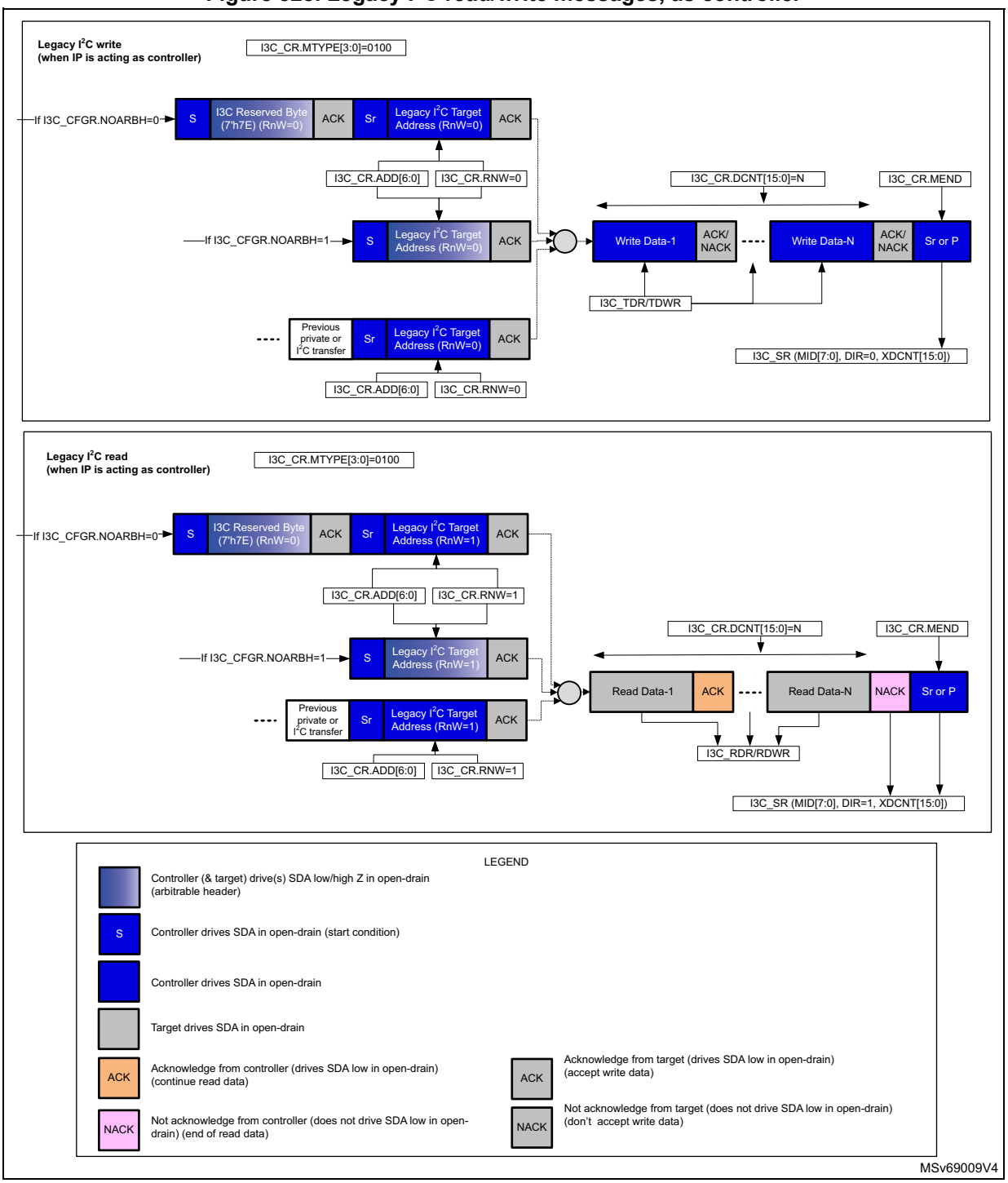
Figure 827. I3C private read/write messages, as target



64.9.12 Legacy I²C read/write transfer, as controller

Figure 828 illustrates legacy I²C read/write transfer, as communicated on the I3C bus, and as programmed when acting as controller.

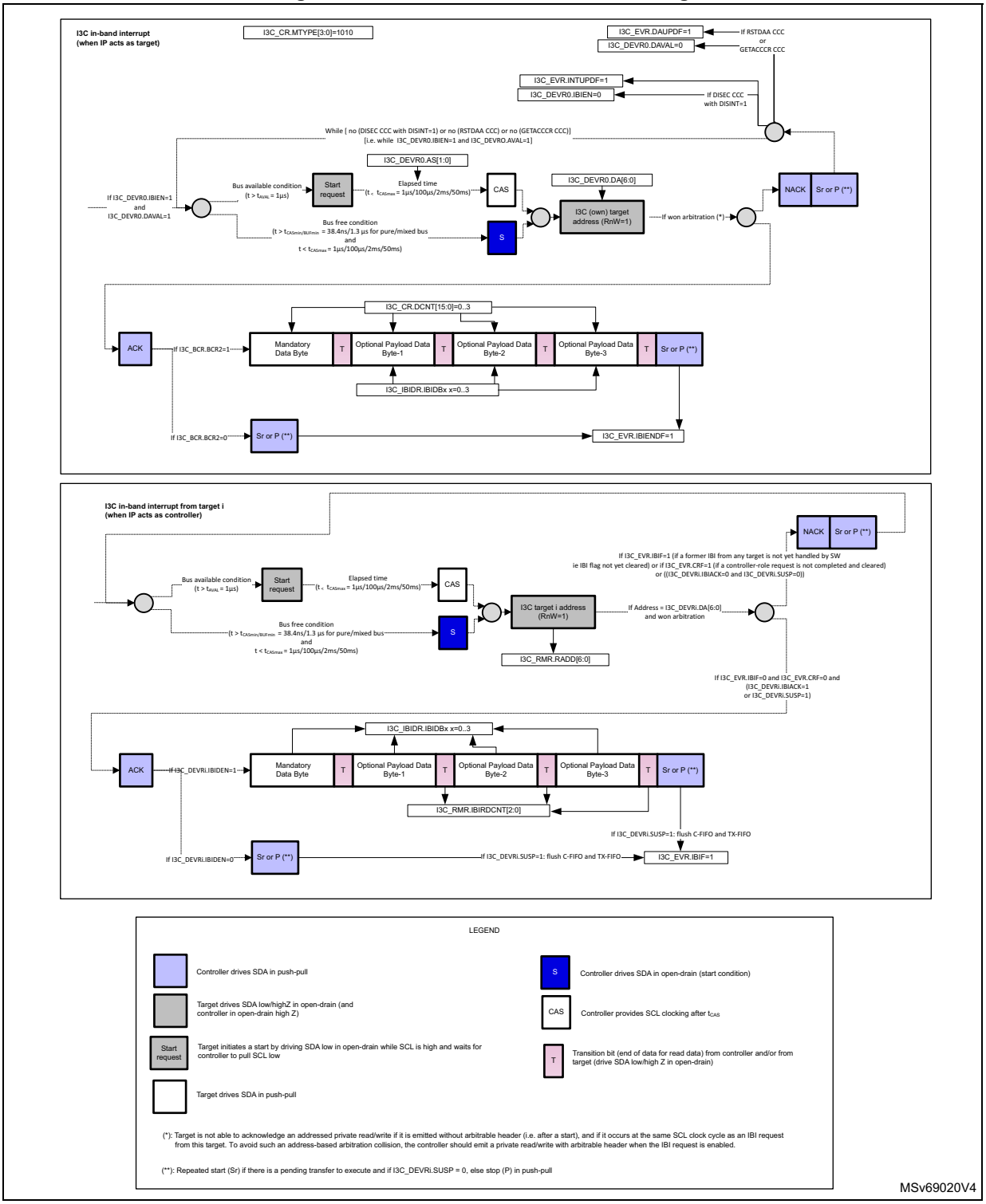
Figure 828. Legacy I²C read/write messages, as controller



64.9.13 I3C IBI transfer, as controller/target

Figure 829 illustrates IBI (in-band interrupt) transfer, as communicated on the I3C bus, and as programmed when acting as target or received as controller.

Figure 829. IBI transfer, as controller/target



When the I3C peripheral acts as controller, the I3C_IBIDR register is used to receive the IBI data payload. Consequently, the IBI request from the target must not exceed a 4-byte data payload. If there is more information to be exchanged in the context of this in-band interrupt, the controller software must issue a private read.

Timing control and sensor acquisition timestamping when I3C controller

When the I3C peripheral acts a controller, it can use timing control and asynchronous mode 0 to timestamp the moment at which the target(s) acquired sampled data, and to time-correlate them.

To do this, the software must go through the following sequence:

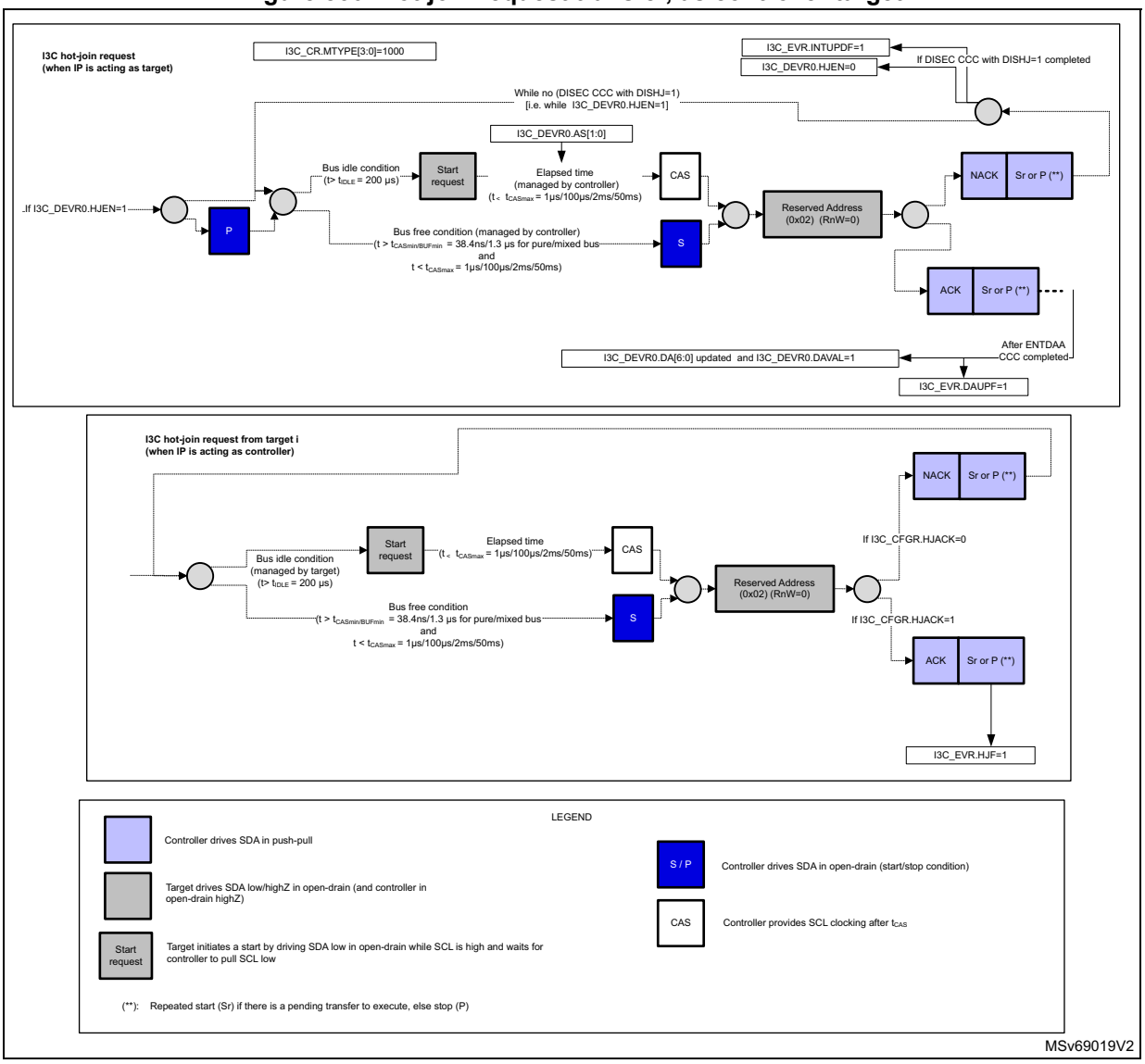
1. Allocate and configure a timer in capture mode, to be connected to the I3C generated event i3c_ibi_ack so that for each IBI acknowledge the counter of the timer can be captured.
2. Program a broadcast/direct SETXTIME CCC with sub-command byte 0xDT to enter asynchronous mode 0 on the target(s) side.
3. After being notified of the command completion by the flag and/or the related interrupt (FCF flag is set in I3C_EVR register), reset and enable the timer to start the counter on the controller side.
4. After being notified that an IBI is complete by the flag and/or the related interrupt (IBIF = 1 in the I3C_EVR register), and if the IBI is with timing info (if IBIDB0[7:5] = 100 in the I3C_IBIDR register), read the captured value of the timer, as C_TIM. Given that:
   a) C_C2 = 9 × (SCLL_PP[7:0] + 1 + SCLH_I3C[7:0] + 1) × T_{I3CCLK}
   b) T_C1 (16-bit value) = {IBIDB2, IBIDB1} in the I3C_IBIDR register
   c) T_C2 (8-bit value) = IBID3 in the I3C_IBIDR register

   Compute the timestamp of the sampled data received from the target, as follows (see MIPI spec formula for more details): C_TS = C_TIM - C_C2 x T_C1 / T_C2.
5. Repeat step *4*, for each received IBI.
6. Generate a broadcast/direct SETXTIME CCC with subcommand byte 0xFF to exit asynchronous mode 0 and disable/deallocate the timer resource.

64.9.14 I3C hot-join request transfer, as controller/target

Figure 830 illustrates hot-join request transfer, as communicated on the I3C bus, and as programmed when acting as target or received as controller.

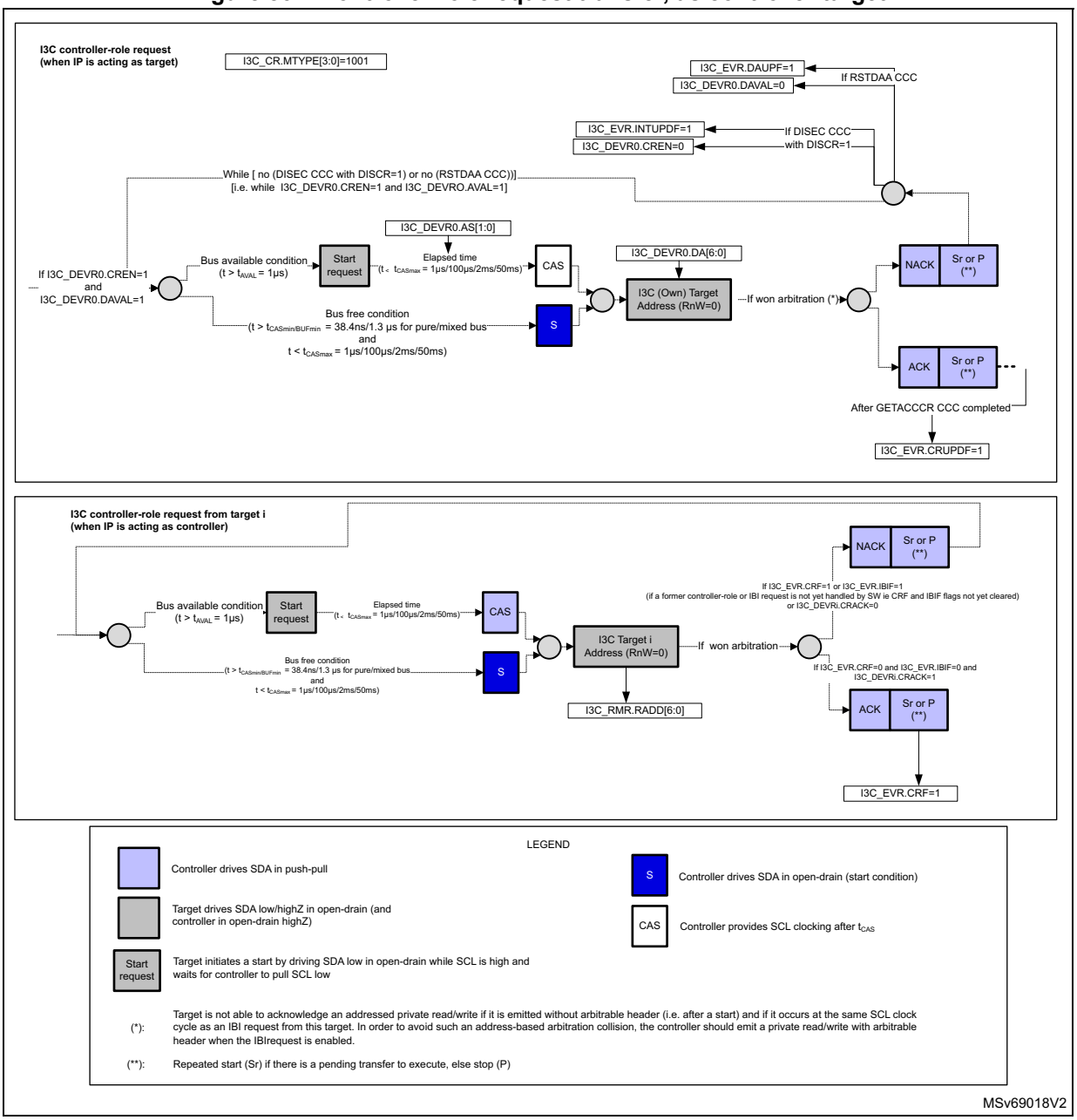
Figure 830. Hot-join request transfer, as controller/target



64.9.15 I3C controller-role request transfer, as controller/target

Figure 831 illustrates controller-role request transfer, as communicated on the I3C bus, and as programmed when acting as target or received as controller.

Figure 831. Controller-role request transfer, as controller/target



64.10 I3C FIFOs management, as controller

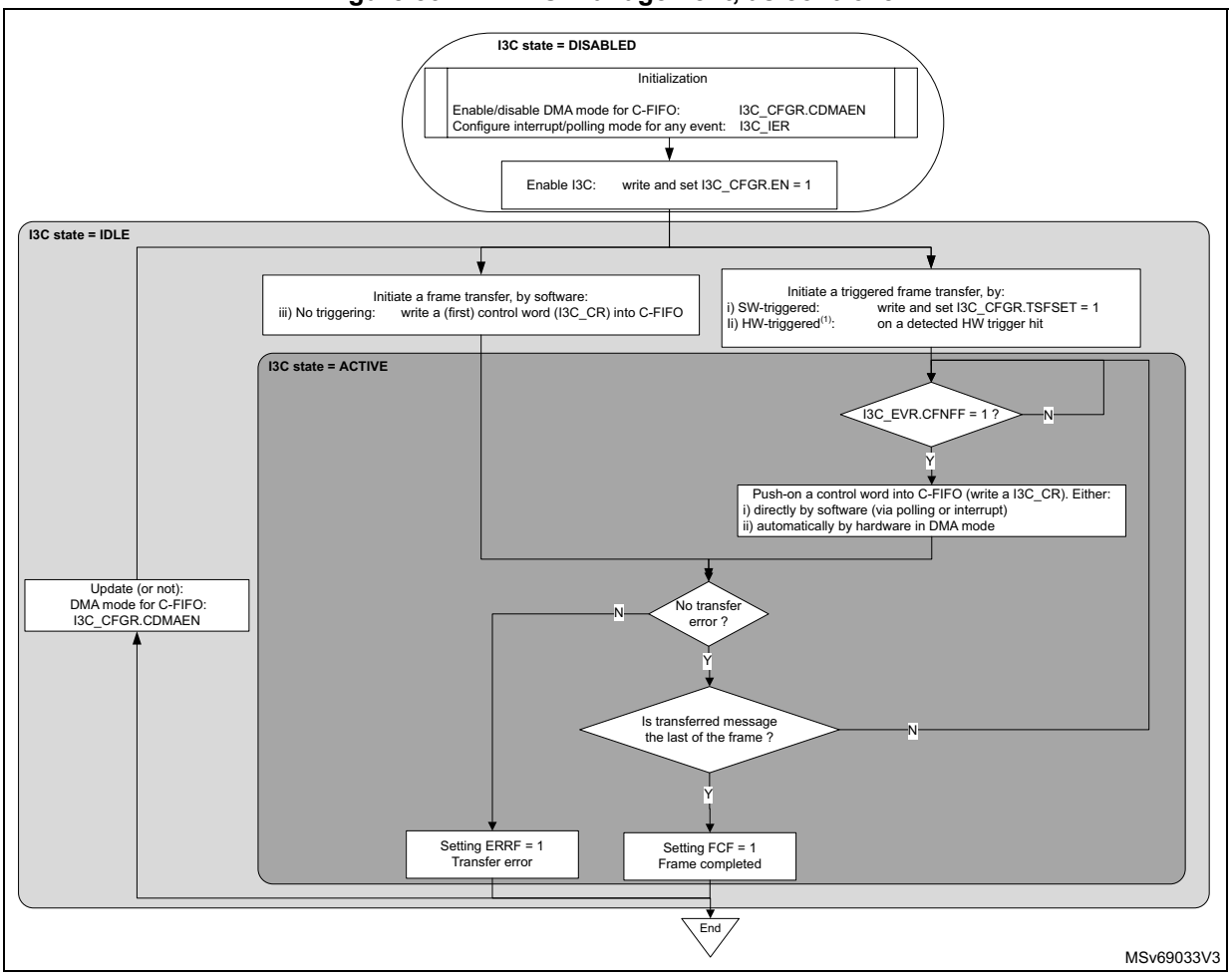
64.10.1 C-FIFO management, as controller

When controller, as illustrated in figures of *Section 64.9*, C-FIFO can be used during any of the following transfers:

- broadcast CCC (*Figure 819*, *Figure 820*, *Figure 821*)
- direct read/write CCC (*Figure 821*)
  - command part, first message
  - data part, next message(s)
- private read/write (*Figure 826*)
- legacy I2C read/write (*Figure 828*)
- software-initiated error recovery (SCL forced to be stopped until next header message followed by HDR exit pattern)

Figure 832 illustrates the management of the C-FIFO for queuing control word(s) on the I3C bus, when the I3C peripheral acts as controller.

Figure 832. C-FIFO management, as controller



1. This feature is implementation-dependent and can be unavailable. Refer to *Section 64.3.4*.

First, the software must initialize the C-FIFO management via CDMAEN in the I3C_CFGR register, to be written either:

- directly by the software (if CDMAEN = 0) at the control word level:
  - via polling mode (CFNFIE = 0 in the I3C_IER register): waiting for a next control word is requested by the hardware (CFNFF = 1 in the I3C_IER register) before an explicit write to the I3C_CR register
  - via enabled interrupt notification if CFNFIE = 1
- by the allocated DMA channel (if CDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_tc_dma):
  - as configured at block level, the DMA is automatically pushing-on/writing control word(s) into the I3C_CR register from its memory source buffer, until the frame completion (a stop is emitted on the I3C bus after the last message of the frame), unless a transfer error occurs.

In any case, if C-FIFO is empty and a restart must be emitted with a new control word, a C-FIFO underrun is reported (ERRF = 1 in the I3C_EVR register and COVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_IER register, an interrupt is generated.

The DMA mode for the C-FIFO management can be modified when the I3C peripheral is not in active state.

When controller, if a transfer error occurs (ERRF = 1 in the I3C_EVR register), the C-FIFO is flushed automatically by the hardware.

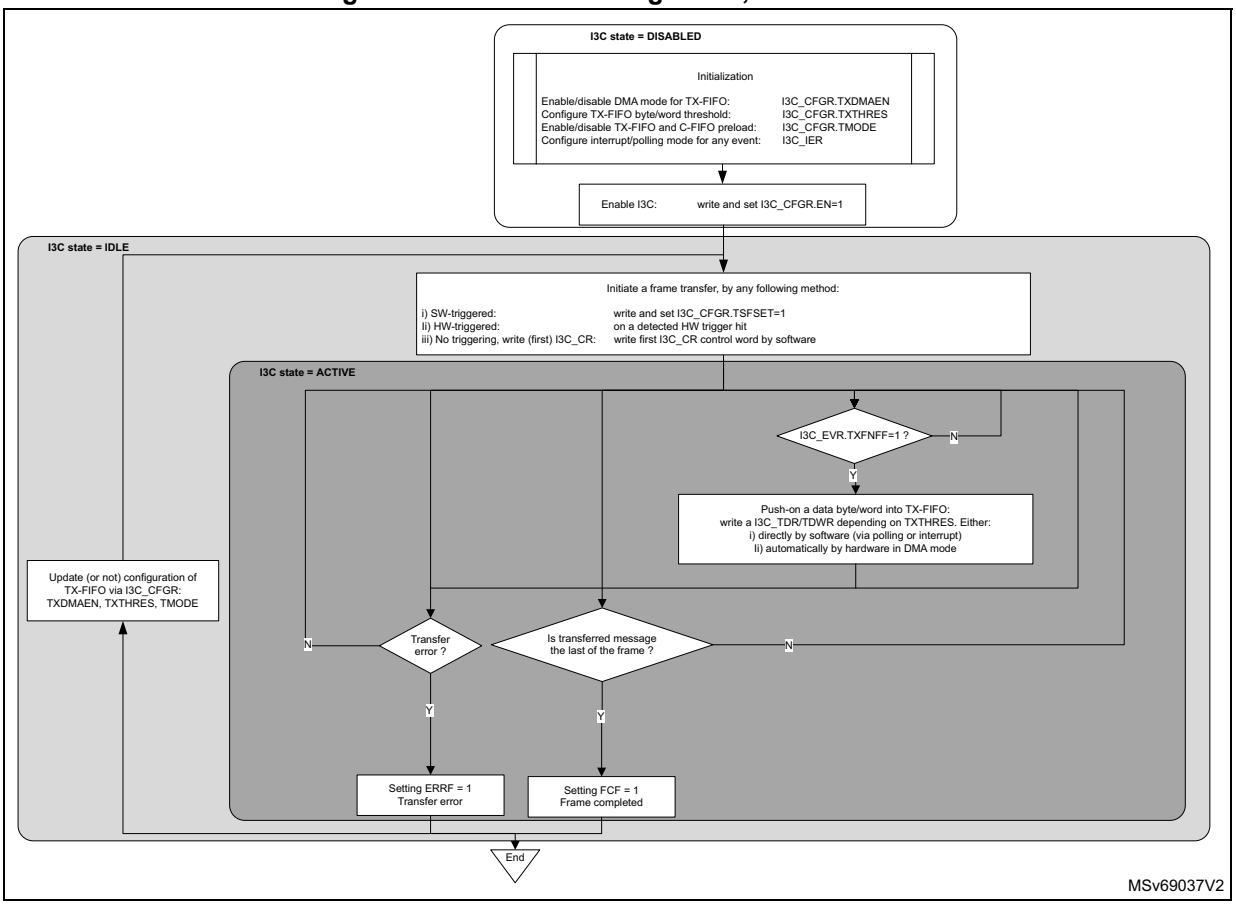
64.10.2 TX-FIFO management, as controller

When controller, as shown in figures of *Section 64.9*, TX-FIFO can be used during any of the following transfers:

- broadcast or direct CCC (including ENTDAA and RSTACT), if a defining/sub-command byte or if data byte(s) are present (*Figure 819*, *Figure 820*, *Figure 821*)
- private write (*Figure 826*)
- legacy I2C write (*Figure 828*)

Figure 833 illustrates the management of the TX-FIFO for queuing data bytes or word(s) to be transmitted on the I3C bus, when the I3C peripheral acts as controller.

Figure 833. TX-FIFO management, as controller



First, the software must initialize the TX-FIFO management via the following fields in the I3C_CFGR register:

- TXDMAEN: enable/disable DMA mode for TX-FIFO
- TXTHRES: push-on data byte(s) or word(s) into TX-FIFO
- TMODE: enable/disable TX-FIFO and C-FIFO preload

Then, depending upon bit TXDMAEN in the I3C_CFGR register, the TX-FIFO is written either:

- directly by the software (if TXDMAEN = 0) at the byte/word level:
  - via polling mode (TXFNFIE = 0 in the I3C_IER register): waiting for a next data byte/word is requested by the hardware (TXFNFF = 1 in the I3C_IER register) before an explicit write to the I3C_TDR or I3C_TDWR register, depending upon bit TXTHRES in the I3C_CGFR register
  - via enabled interrupt notification if TXFNFIE = 1
- by the allocated DMA channel (if TXDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_tx_dma):
  - as configured at DMA block level, the DMA is automatically pushing-on/writing data bytes/words to the I3C_TDR or I3C_TDWR register (depending upon bit TXTHRES in the I3C_CGFR. register) from its memory source buffer, until the frame completion (a stop is emitted on the I3C bus after the last message of the frame), unless a transfer error occurs.

An I3C message begins from a start or a repeated start, and ends with a stop or a repeated start. At message level, the last data byte/word to be transmitted is flagged by TXLASTF = 1 in the I3C_EVR register. When an I3C frame is described with multiple messages (separated by a repeated start), this event can be used by the software to update the pointer to the buffer where the byte(s)/word(s) of the next message is/are stored.

When frame completion is reported (FCF = 1 in the I3C_EVR register, and the corresponding interrupt is enabled), the TX-FIFO is empty.

If the TX-FIFO is empty and a data byte must be transmitted on the I3C bus, a TX-FIFO underrun is reported (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_IER register, an interrupt is generated.

The configuration for the TX-FIFO management can be modified when the I3C peripheral is not in active state.

When controller, if a transfer error occurs (ERRF = 1), the TX-FIFO is automatically flushed by the hardware.

No C-FIFO/TX-FIFO preload

As defined in *Table 622*, C-FIFO size is two words, TX-FIFO size is 8 bytes.

When no C-FIFO/TX-FIFO preload is configured (TMODE = 0 in the I3C_CFGR register), the I3C peripheral emits a start on the I3C bus as soon as the first control word is written into the C-FIFO. Then, it decodes the I3C_CR register, and requires a next data byte/word to be written, if needed within this message. As soon as a next control word is detected as required by the hardware to be transmitted on the I3C bus (if a repeated start must be emitted on the I3C bus or if the C-FIFO gets available room), this control word is requested to be written into the C-FIFO until the last message (MEND = 1 in the I3C_CR register).

Similarly, as soon as another data byte/word is detected as required by the hardware to be transmitted on the I3C bus (if a repeated start must be emitted on the I3C bus, and if TX-FIFO is not full, and if data byte(s)/word(s) must be transmitted during this I3C message), this data byte/word must be written in the TX-FIFO.

C-FIFO and TX-FIFO preload

When C-FIFO/TX-FIFO preload is configured (TMODE = 1 in the I3C_CFGR register), before emitting a start on the bus, the I3C peripheral waits for loading as much as possible both the C-FIFO and the TX-FIFO, as follows:

1. Wait for a first control word to be written into the C-FIFO.
2. Wait for data byte(s)/word(s) to be written in the TX-FIFO, if any, as defined by the first control word (if RNW = 0 and DCNT[15:0] = 0 in the I3C_CR register), and up to the TX-FIFO size.
3. If TX-FIFO is not full and if the first control word is not the last of the frame (MEND = 0 in the I3C_CR register):
   a) Wait for a second control word to be written into the C-FIFO, then C-FIFO is full.
   b) If TX-FIFO is not full, wait for data byte(s)/word(s) to be written in the TX-FIFO, if any, as defined by the second control word (RNW = 0 and DCNT[15:0] = 0 in the I3C_CR register), and up to the TX-FIFO size.

Then, as soon as a next control word is detected as required by the hardware to be transmitted on the I3C bus (if a repeated start must be emitted on the I3C bus), this control

word is requested to be written into the C-FIFO until the last message (MEND = 1 in the I3C_CR register).

Similarly, as soon as a next data byte/word is detected as required by the hardware to be transmitted on the I3C bus (if a repeated start must be emitted on the I3C bus and if TX-FIFO is not full and if data byte(s)/word(s) are to be transmitted during this I3C message), this data byte/word must be written in the TX-FIFO.

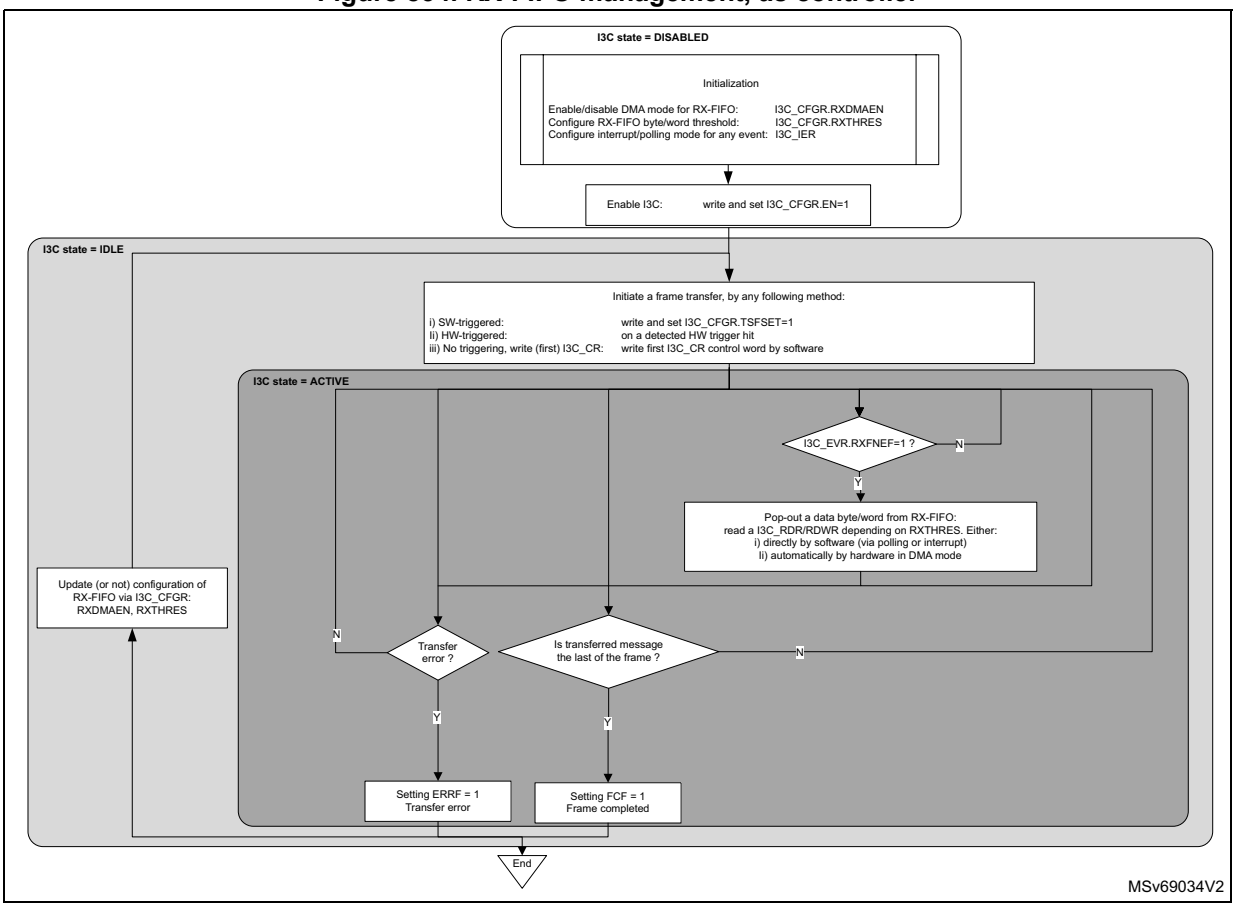
64.10.3 RX-FIFO management, as controller

When controller, as shown in figures of *Section 64.9*, RX-FIFO is used during any of the following transfers:

- broadcast ENTDAA CCC (*Figure 820*)
- direct CCC read (*Figure 819*), including direct RSTACT CCC read (*Figure 821*)
- private read (*Figure 826*)
- legacy I2C read (*Figure 828*)

Figure 834 illustrates the management of the RX-FIFO for queuing and popping-out data bytes or word(s) as received from the I3C bus, when the I3C peripheral acts as controller.

Figure 834. RX-FIFO management, as controller



First, the software must initialize the RX-FIFO management via the following fields of the I3C_CFGR register:

- RXDMAEN: enable/disable DMA mode for RX-FIFO
- RXTHRES: pop-out data byte(s) or word(s) from RX-FIFO

Then, depending on RXDMAEN, the RX-FIFO is read either:

- directly by the software (RXDMAEN = 0) at the byte/word level:
  - via polling mode (RXFNEIE = 0 in the I3C_IER register): waiting for a next data byte/word is requested by the hardware (RXFNEF = 1 in the I3C_IER register) before an explicit read to the I3C_RDR or I3C_RDWR register, depending upon bit RXTHRES in the I3C_CGFR register
  - via enabled interrupt notification if RXFNEIE = 1 in the I3C_IER register
- by the allocated DMA channel (if RXDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_rx_dma):
  - as configured at DMA block level, the DMA automatically pops-out/reads data bytes/words from the I3C_RDR or I3C_RDWR register (depending upon bit RXTHRES), and writes them into its memory destination buffer, until the frame completion (a stop is emitted on the I3C bus after the last message of the frame), unless a transfer error occurs.

An I3C message begins from a start or a repeated start, and ends with a stop or a repeated start. At message level, the last received data byte/word from the I3C bus is flagged by IRXLASTF = 1 in the I3C_EVR register. When an I3C frame is described with multiple messages (separated by a repeated start), this event can be used by the software for updating the pointer to the buffer where is/are stored the data byte(s)/word(s) of the next message.

If RX-FIFO is full and a data byte is received on the I3C bus, an RX-FIFO overrun is reported (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_IER register, an interrupt is generated.

The configuration for the RX-FIFO management can be modified when the I3C peripheral is not in active state.

Early read termination from the target

A private read message can be early completed (also known as prematurely ended) by the addressed target.

- If RXDMAEN = 1:
  - the software must allocate, for the DMA request i3c_rx_dma, a DMA channel x, capable of peripheral early termination (refer to DMA implementation section).
  - The software must configure the DMA channel x to enable the DMA peripheral-flow control mode via PFREQ = 1 in the DMA_CxTR2 register, to be able to perform a(n) (early or not) DMA block completion.
  - Then, on block completion, the software can read BR1.BNDT[15:0] in DMA_Cx and/or DMA_CxSAR register(s), to get the effective number of DMA transferred bytes.
- If RXDMAEN = 0:
  - at message level, the last received data byte/word from the I3C bus is flagged by RXFNEF = 1 and RXLASTF = 1 in the I3C_EVR register.

In any case, if the S-FIFO is disabled (if SMODE = 0 in the I3C_CFGR register), the software is notified that an early read termination occurs by RXTGTENDF = 1 in the I3C_EVR register, and the corresponding interrupt if enabled. Then, software can read the status register I3C_SR to check information related to the last message ,and get the number of received data bytes on the prematurely ended read transfer (XDCNT[15:0] in the I3C_SR register).

In any case, if the S-FIFO is enabled (SMODE = 1 in the I3C_CFGR register), the software or the DMA (depending upon SDMAEN in the I3C_CFGR register) must read for each message the status register I3C_SR. The number of effective received data bytes on the prematurely ended read message is reported by XDCNT[15:0] (and then ABT = 1) in the I3C_SR register.

For more information, refer to *I3C status register (I3C_SR)* and *Section 64.10.4*.

64.10.4 S-FIFO management, as controller

When controller, S-FIFO can be used by the software to be able to read the *I3C status register (I3C_SR)* for each transferred message.

Reading status register with disabled S-FIFO

If SMODE = 0 in the I3C_CFGR register, the S-FIFO is disabled and the status register can be read as a usual register:

- the register content is overwritten by hardware when is transferee a new message
- I3C_SR contains the status of the last transferred message
- SCL clock is not stalled if status register is not read.

If SMODE = 0, for the specific case of a private read prematurely ended by the target:

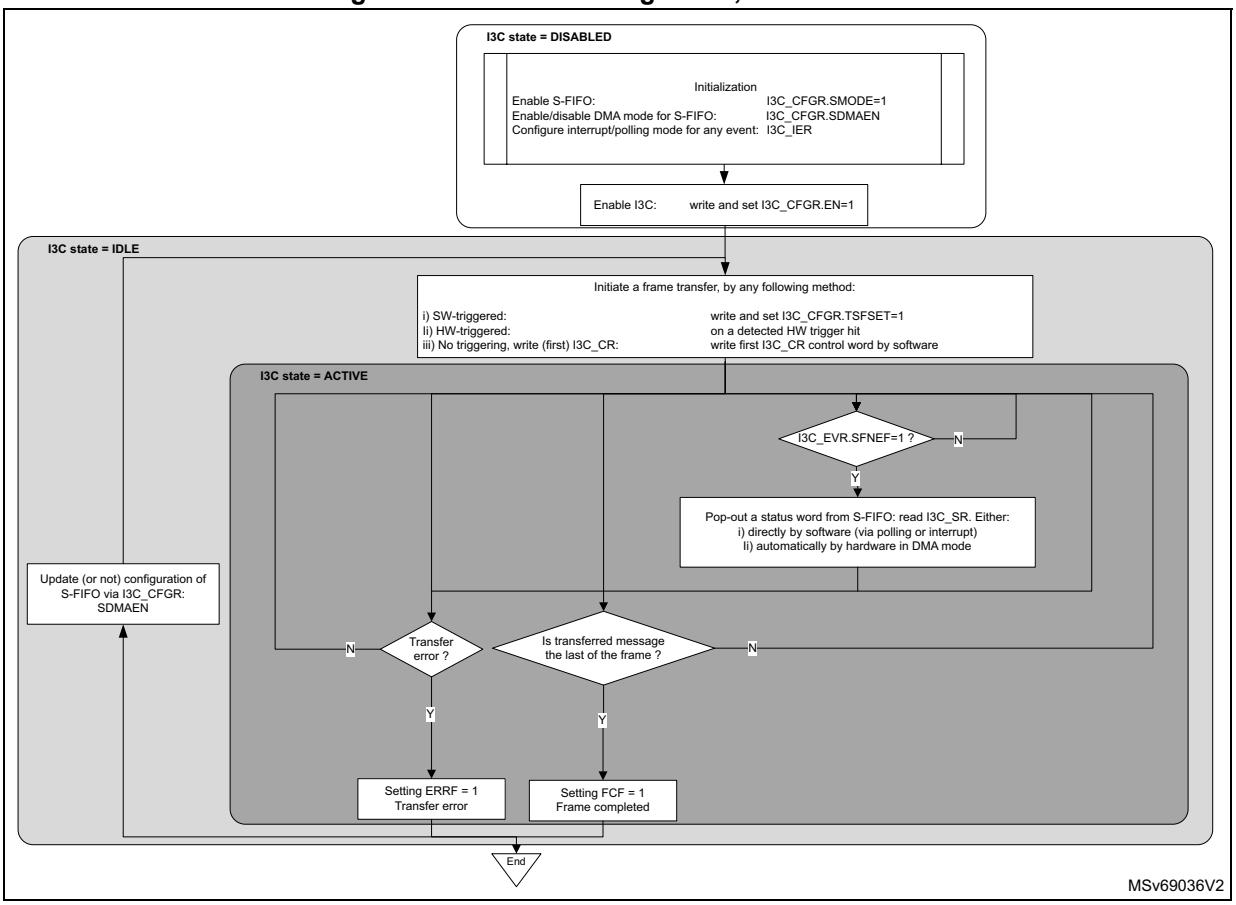
- the software is notified by the flag RXTGTENDF = 1 in the I3C_EVR register and the corresponding interrupt if enabled.
- Then, until that the software has cleared the event flag write and set CRXTGTENDF = 1 in the I3C_CEVR register:
  - no more data byte can be received on the I3C bus and written by the hardware into I3C_RDR/I3C_RDWR registers
  - I3C_SR cannot be updated
  - SCL clock is stalled if needed

Typically, I3C_SR can be read when FCF = 1, or ERRF = 1, or RXTGTENDF = 1 in the I3C_EVR register. XDCNT[15:0] in the I3C_SR register can be read to get the effective number of received data bytes on a private read, after an early termination from the target (refer to *Section 64.10.3*).

Reading status register with enabled S-FIFO

If SMODE = 1 in the I3C_CFGR register, the S-FIFO is enabled. *Figure 835* illustrates the management of the S-FIFO for queuing and popping-out a status word for each executed message on the I3C bus, when the I3C peripheral acts as controller.

Figure 835. S-FIFO management, as controller



First, the software must initialize the S-FIFO management via the SDMAEN field (enable/disable DMA mode for S-FIFO) in the I3C_CFGR register. Then, depending on SDMAEN, the S-FIFO is read either:

- directly by the software (if SDMAEN = 0):
  - via polling mode (SFNEIE = 0 in the I3C_IER register): waiting for a next status word is requested by the hardware (SFNEF = 1 in the I3C_IER register) before an explicit read to the I3C_SR register
  - via enabled interrupt notification (SFNEIE = 1)
- by the allocated DMA channel (if SDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_rs_dma):
  - as configured at DMA block level, the DMA is automatically popping-out/reading status words from the I3C_SR register and writing them into its memory destination buffer, until the frame completion (a stop is emitted on the I3C bus after the last message of the frame), unless a transfer error occurs.

Each message status must be read, else an overrun error is asserted by the hardware (ERRF = 1 in the I3C_EVR register and COVR = 1 in the I3C_SER register) when the S-FIFO is full and a next message status must be written. The corresponding interrupt is raised if enabled by ERRIE = 1 in the I3C_IER register.

The frame completion (FCF = 1 in the I3C_EVR register) is reported only when the S-FIFO is empty.

The configuration for the S-FIFO management can be modified when the I3C peripheral is not in active state.

64.11 I3C FIFOs management, as target

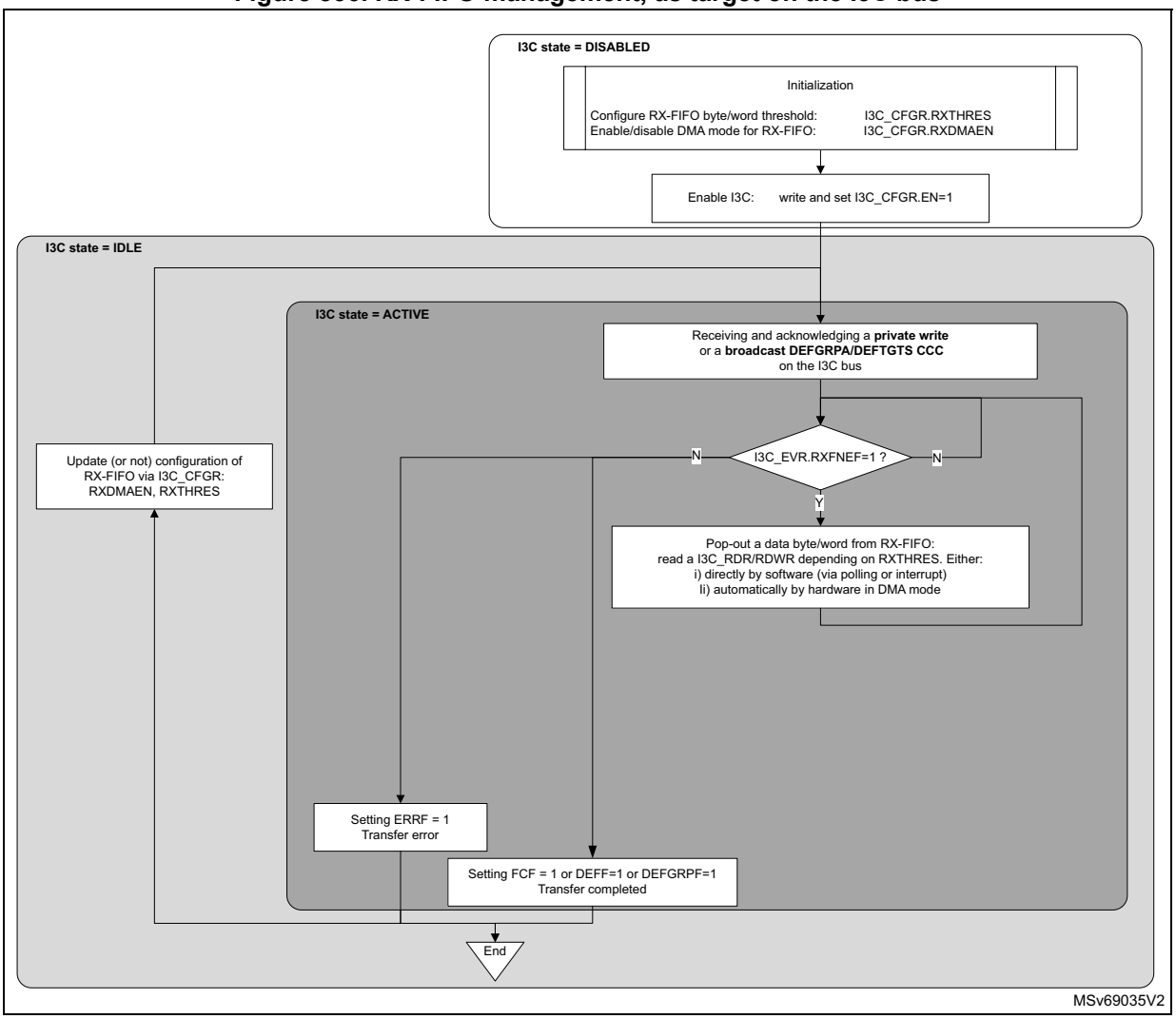
64.11.1 RX-FIFO management, as target

When target, as shown in figures of *Section 64.9*, the RX-FIFO is used during any of the following received and acknowledged transfers:

- broadcast DEFTGTS CCC (*Figure 824*)
- broadcast DEFGRPA CCC (*Figure 825*)
- private write (*Figure 827*)

Figure 836 illustrates the management of the RX-FIFO for queuing and popping-out data bytes or word(s) as received on the I3C bus, when the I3C peripheral acts as target.

Figure 836. RX-FIFO management, as target on the I3C bus



First, the software must initialize the RX-FIFO management via the following I3C_CFGR register fields:

- RXDMAEN: enable/disable DMA mode for RX-FIFO
- RXTHRES: pop-out data byte(s) or word(s) from RX-FIFO

Then, depending on RXDMAEN, the RX-FIFO is read either:

- directly by the software (if RXDMAEN = 0) at the byte/word level:
  - via polling mode (RXFNEIE = 0 in the I3C_IER register): waiting for a next data byte/word is requested by the hardware (RXFNEF = 1 in the I3C_IER register) before an explicit read to the I3C_RDR or I3C_RDWR registers, depending upon RXTHRES in the I3C_CGFR register
  - via enabled interrupt notification if RXFNEIE = 1
- by the allocated DMA channel (if RXDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_rx_dma):
  - as configured at DMA block level, the DMA is automatically popping-out/reading data bytes/words from the I3C_RDR or I3C_RDWR register (depending upon RXTHRES) and writing them into its memory destination buffer, until the transfer completion (in the I3C_EVR register, FCF = 1 in case of private write, or GRPF = 1 in case of a DEFGRPA CCC, or DEFF = 1 in case of DEFTGTS CCC), unless a transfer error occurs.

If RX-FIFO is full and a new data byte is received on the I3C bus, a RX-FIFO overrun is reported (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register) and the corresponding interrupt if enabled.

When the transfer is completed (in the I3C_EVR register, FCF = 1, or GRPF = 1, or DEFF = 1):

- RX-FIFO is empty
- Until software has not processed the RX data buffer corresponding to the completed or private write / DEFTGTS / DEFGRPA transfer (meaning until software has not cleared the corresponding flag write and set CFCF / CDEFF / CGRPF to 1 in the I3C_CEVR register), if a next private write / DEFTGTS CCC / DEFGRPA CCC is received and a data byte must be written by the hardware into the RX-FIFO, a transfer error is reported with a RX-FIFO overrun (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register) and the corresponding interrupt if enabled.

The configuration for the RX-FIFO management can be modified when the I3C peripheral is not in active state.

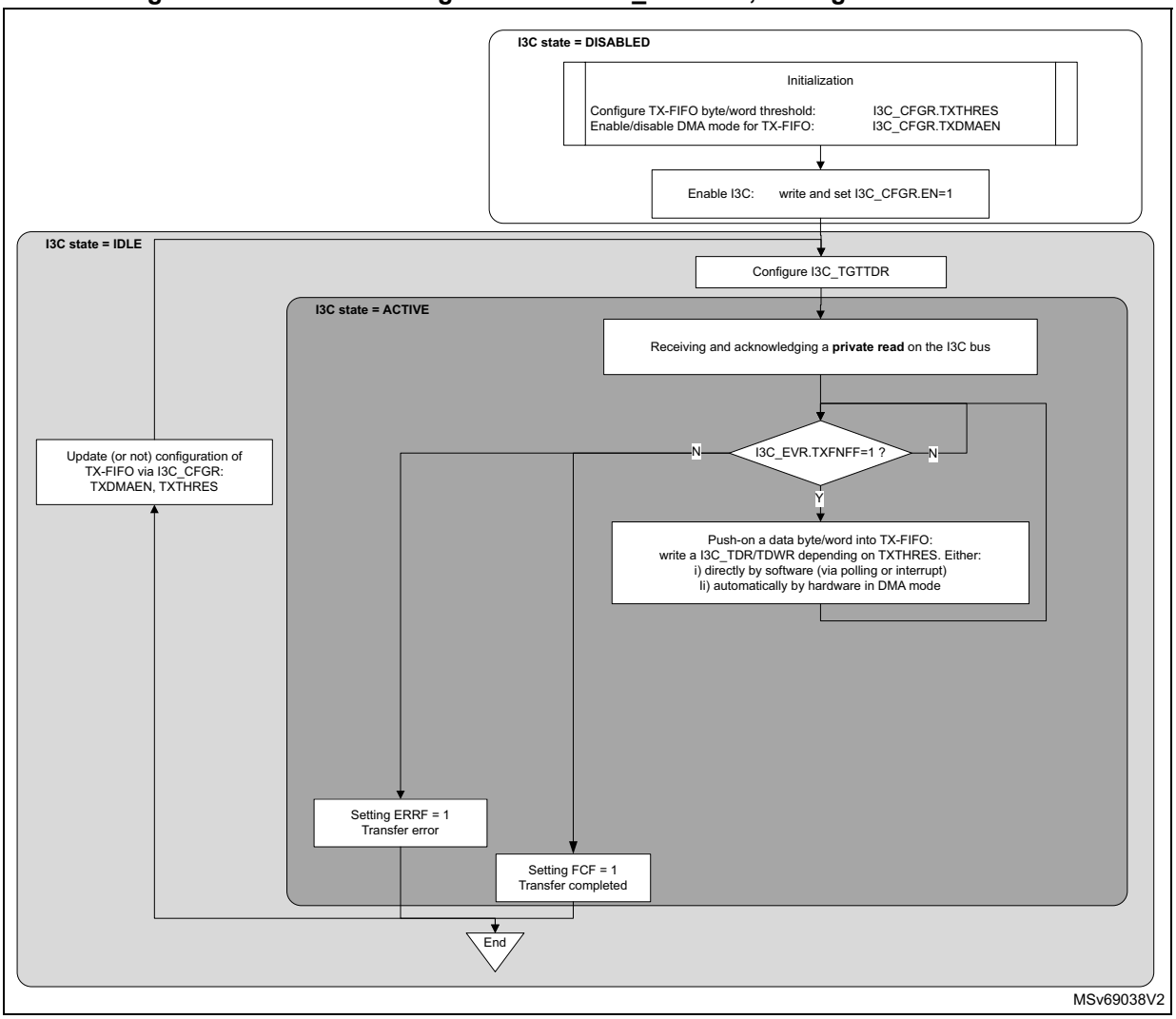
64.11.2 TX-FIFO management, as target

Main scheme

When target, as shown in figures of *Section 64.9*, the TX-FIFO is used only during a private read (*Figure 827*).

Figure 837 illustrates the management of the TX-FIFO for queuing and pushing-on data bytes or word(s) to be transmitted on the I3C bus, when the I3C peripheral acts as target.

Figure 837. TX-FIFO management with I3C_TGTTDR, as target on the I3C bus



First, the software must initialize the TX-FIFO management and write I3C_CFGR with

- TXDMAEN: enable/disable DMA mode for TX-FIFO
- TXTHRES: push-on data byte(s) or word(s) to TX-FIFO

Then, before receiving a private read on the I3C bus , the software must configure the *I3C target transmit configuration register (I3C_TGTTDR)* to preload the TX-FIFO with a number of data bytes (write TGTTDCNT[15:0] and PRELOAD = 1 in a single access), so that data bytes from the target are ready to be transmitted on the serial bus:

- If PRELOAD = 1 and TGTTDCNT[15:0] > TX-FIFO size, TX-FIFO is first preloaded up to the FIFO size.
- If PRELOAD = 1 and TGTTDCNT[15:0] ≤ TX-FIFO size, TX-FIFO is preloaded up to TGTTDCNT[15:0].

Note: TX-FIFO size is 8 bytes (refer to Table 622).

Depending upon TXDMAEN, the TX-FIFO is preloaded either:

- directly by the software (TXDMAEN = 0) at the byte/word level:
  - via polling mode (TXFNFIE = 0 in the I3C_IER register): waiting for a next data byte/word is requested by the hardware (TXFNFF = 1 in the I3C_IER register) before an explicit write to the I3C_TDR or I3C_TDWR register, depending upon TXTHRES in the I3C_CGFR register
  - via enabled interrupt notification if TXFNFIE = 1
- by the allocated DMA channel (TXDMAEN = 1) to the corresponding DMA request from the I3C peripheral (i3c_tx_dma):
  - as configured at DMA block level, the DMA is automatically pushing-on/writing data bytes/words to the I3C_TDR or I3C_TDWR register (depending upon TXTHRES) from its memory source buffer, until the transfer completion (FCF = 1 in the IEC_EVR register), unless a transfer error occurs.

Then, in the same way, either directly by the software or by the allocated DMA channel, if there are remaining TGTTDCNT[15:0] in the I3C_TGTTDR register to be loaded into TX-FIFO for continuing the private read (set PRELOAD = 1 and TGTTDCNT[15:0] > TX-FIFO size), and provided that the private read is not yet completed by the controller:

- If TXTHRES = 0: when a byte is transmitted on the serial bus, a next byte is preloaded into TX-FIFO
- If TXTHRES = 1: when four bytes are transmitted on the serial bus, the next word is preloaded into TX-FIFO.

The private read transfer is completed (FCF = 1 in the I3C_EVR register) when either the target or the controller first terminates the data byte transfer.

On transfer completion, the software can:

- read XDCNT[15:0] in the I3C_SR register: the effective number of transmitted data bytes
- read TGTTDCNT[15:0] in the I3C_TGTTDR register: the remaining number of bytes to be loaded and transmitted on the serial bus
- flush TX-FIFO: write and set (or not) TXFLUSH = 1 in the I3C_CFGR register, continue (or not) for another read, depending upon the user application

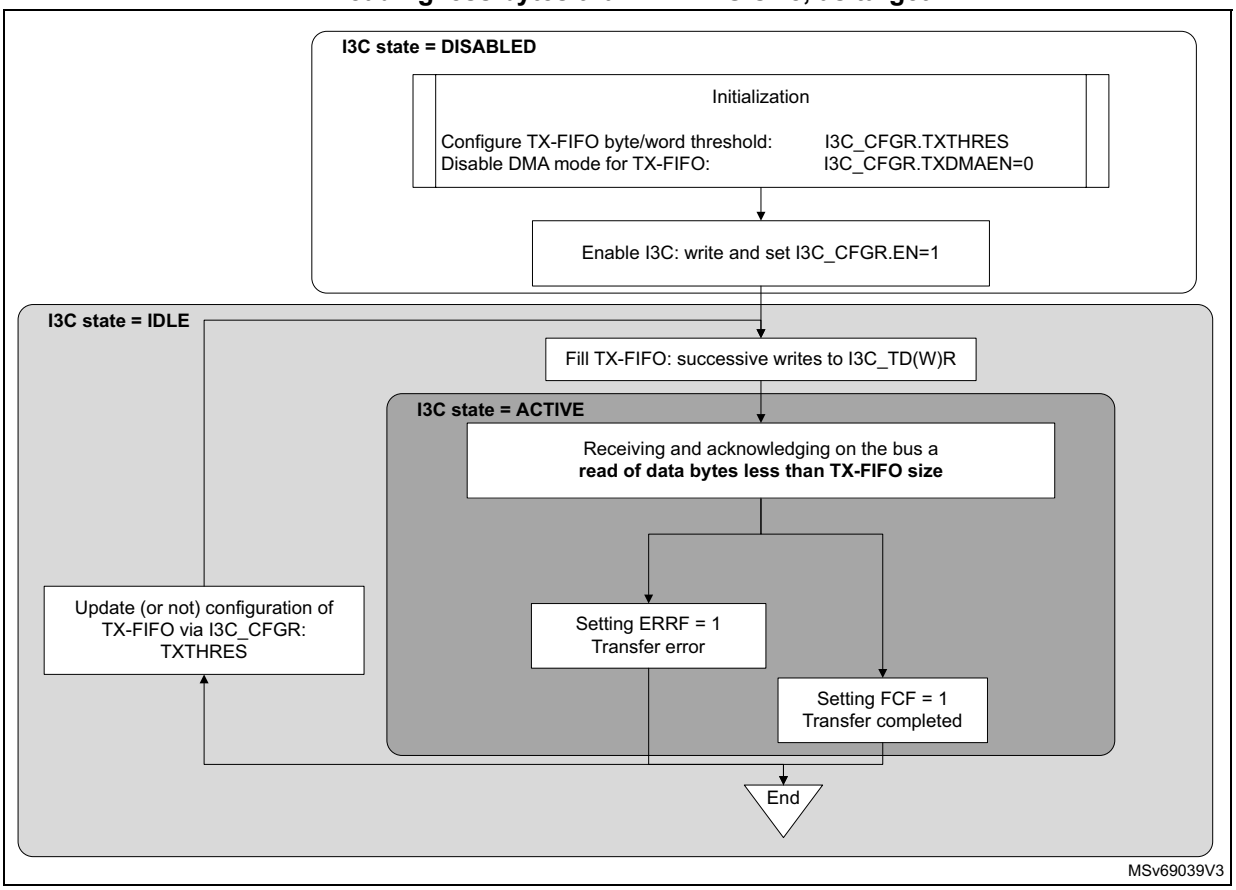
If TX-FIFO is empty and a data byte must be transmitted on the serial bus, a TX-FIFO underrun is reported (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_EVR register, an interrupt is generated.

The configuration for the TX-FIFO management can be modified when the I3C peripheral is not in active state.

Alternative without I3C_TGTTDR, if less bytes than the TX-FIFO size

Alternatively to the use of the I3C_TGTTDR register, as shown in the *Figure 838*, when the DMA is not used (TDMAEN = 0), provided that the number of data bytes to be read on the serial bus is less that the TX-FIFO size, the software can directly prepare and fill the TX-FIFO with the number of bytes to be read directly by the software, by successive writes to the I3C_TDR or I3C_I3C_TDWR, depending upon TXTHRES.

Figure 838. TX-FIFO management by software without I3C_TGTTDR if reading less bytes than TX-FIFO size, as target



64.12 I3C error management

64.12.1 Controller error management

Table 631 enumerates the I3C bus error conditions, and, for each of them, the corresponding detection, action and reporting when the I3C peripheral acts as controller.

Table 631. I3C controller error management

Error type	Description	Error detection	Controller action	Reported error(1)
CE0	Illegally formatted CCC (e.g. less returned data byte(s))	Prematurely ended read data by target on a direct CCC read(2)	Hardware emits a stop.	ERRF = 1 and PERR = 1 and CODERR[3:0] = 0000

Table 631. I3C controller error management (continued)

Error type	Description	Error detection	Controller action	Reported error(1)
CE1	Monitoring error	An incorrect ACK is detected at the end of a legacy I^2C read	Hardware keeps SCL running for nine clock cycles and another byte to be possibly exchanged, then emits again a NACK, followed by a stop.	ERRF = 1 and PERR = 1 and CODERR[3:0] = 0001
		Restart or stop cannot be generated at the end of an I3C SDR read	SCL is kept running. Software can stop SCL by writing a control word with MTYPE[3:0] = 0000 in the I3C_CR register (header message), then must typically wait at least 150 µs before emitting another message.	
		After a CE1 error, a start cannot be generated		
CE2	No response to broadcast address (0b111_1110)	Header (0b111_1110 + RnW = 0) is detected as NACK-ed during a message different from escalation fault or reset pattern message	Hardware emits an HDR exit pattern followed by a stop.	ERRF = 1 and PERR = 1 and CODERR[3:0] = 0010
CE3	Failed controller hand-off	New controller has not driven SCL low after having SDA low (after a start via a test header or a start request from a target) and after the delay time defined by its activity state has elapsed	Hardware emits a start + 0b111_1110 + RnW = 0, followed by ACK/NACK from target(s), then stops.	ERRF = 1 and PERR = 1 and CODERR[3:0] = 0011
-	Incorrect returned 7-bit target address with parity bit on a GETACCCR CCC	Incorrect dynamic address and/or parity bit is detected	Hardware cancels the GETACCCR CCC by emitting a Restart + 0b111_1110 + RnW = 0, followed by ACK/NACK from target(s), then stops.	ERRF = 1 and DERR = 1
-	Addressed target is NACK-ed on a direct CCC read for the first time	NACK is detected	Hardware performs a single-retry by emitting a Restart + 7-bit same target address + RnW = 1.	-
-	Addressed target is NACK-ed on either a direct CCC write, an I3C private read/write, a legacy I^2C, or a direct CCC read for the second time	NACK is detected	Hardware emits a stop.	ERRF = 1 and ANACK= 1

Table 631. I3C controller error management (continued)

Error type	Description	Error detection	Controller action	Reported error(1)
-	Assigned/transmitted dynamic address with parity bit is NACK-ed on a ENTDAA CCC for the first time	NACK is detected	Hardware performs a single-retry assignation loop by emitting a Restart + 0b111_1110 + RnW = 1, followed by an ACK and 8-byte read data from the (most-priority) target, followed by the assigned address + parity bit.	-
-	Assigned/transmitted dynamic address with parity bit is NACK-ed on a ENTDAA CCC for the second time	NACK is detected	Hardware emits a stop.	ERRF = 1 and DNACK= 1
-	Write data is NACK-ed on a legacy I2C write			
-	Control word, status word, transmitted data or read data is not written/read in time vs. I3C bus timings	SCL stall timeout	Hardware emits a stop.	ERRF = 1 and (COVR = 1 or DOVR = 1)

1. ERRF in the I3C_EVR register, PERR, CODERR[3:0], DERR, ANACK, DNACK, COVR, and DOVR in the I3C_SER register.
2. MIPI v1.1: on a GETCAPS CCC, the number of received data bytes can be 2, 3 or 4. However a target compliant with MIPI v1.0 can return only the first byte as per previously named GETHDRCAP CCC. As a result, CE0 is not generated if the number is lower than 4.
On a GETMXDS CCC, the number of received data bytes can be 2 or 5. CE0 is not generated if the number is lower than 5.

64.12.2 Target error management

Table 632 enumerates the I3C bus error conditions, the corresponding detection, action, and reporting when the I3C peripheral acts as target.

Table 632. I3C target error management

Error type	Description	Error detection	Next (when emitted by the controller)	Reported error(1)
TE0	Invalid broadcast address (0b111_1110+RnW = 0) or Invalid 7-bit dynamic address + RnW = 1 after DAA assignment	A forbidden address is detected after a start or a repeated start	Hardware waits for an HDR exit pattern	ERRF = 1 and PERR = 1 and ODERR[3:0] = 1000
TE1	CCC code	A CCC code is detected with a parity error		ERRF = 1 and PERR = 1 and ODERR[3:0] = 1001

Table 632. I3C target error management (continued)

Error type	Description	Error detection	Next (when emitted by the controller)	Reported error(1)
TE2	Write data	A write data byte is detected with a parity error on an I3C private write message	Hardware waits for a stop or a repeated start	ERRF = 1 and PERR = 1 and ODERR[3:0] = 1010
TE3	Assigned address during dynamic address arbitration	Assigned dynamic address is detected with a parity error on a ENTDAA CCC		ERRF = 1 and PERR = 1 and ODERR[3:0] = 1011
TE4	0b111_1110 + RnW = 1 missing after Sr during dynamic address arbitration	Missing {0b111_1110 + RnW = 1} after Sr during dynamic address arbitration is detected		ERRF = 1 and PERR = 1 and ODERR[3:0] = 1100
TE5	Transaction after detecting CCC	Invalid CCC direction is detected vs. direction provided during address phase, on a CCC direct read/write		ERRF = 1 and PERR = 1 and ODERR[3:0] = 1101
TE6	Monitoring error of I3C bus	SDA is detected as driven at unexpected value on an data read (during a CCC direct read or a private read or an IBI)	Hardware releases SDA, then waits for a stop or a repeated start	ERRF = 1 and PERR = 1 and ODERR[3:0] = 1110
-	Monitoring SCL on an SDR data read	SCL is detected as stable for more than 125 µs on an SDR data read (during a CCC direct read or a private read or an IBI)		ERRF = 1 and STALL = 1
-	Data to be transmitted not written/available in time vs. I3C bus timings	Data to be transmitted not written by software or DMA in due time on a private read	Hardware detects a stop or a repeated start	ERRF = 1 and DOVR = 1
-	Received data cannot be registered in time vs. I3C bus timings	Received data is not read by software or DMA in due time on a private write or a DEFTGTS or a DEFGRPA CCC		

1. ERRF in the I3C_EVR register, PERR, ODERR[3:0], STALL, and DOVR in the I3C_SER register.

64.13 I3C wake-up from low-power mode(s)

64.13.1 Wake-up from Stop

The user must first configure the reset and clock controller (RCC) to set up the clock data path down to the I3C peripheral: to select the source oscillator for the I3C kernel clock, the source oscillator for the I3C APB clock, and to set the clocks frequency. At initialization via the RCC, the user must enable the I3C clocks for a given I3C peripheral to be functional, separately for Run/Sleep mode and for Stop mode. For more details about RCC programming, refer to Section Reset and clock control (RCC).

The I3C hardware automatically manages its own clocks gating and generates a separated clock request output signal to the RCC for its kernel clock and its APB clock, whenever the device is in Run, Sleep or Stop mode.

When entering a Stop mode, the V_{CORE} domain is supplied, by default any clock oscillator is disabled, and in any case, neither the system clock nor a peripheral clock is running in the domain.

As controller: wake-up on an IBI without MDB, a hot-join request or a controller-role request

When the peripheral acts as controller, before the product enters a low-power mode, the software must issue an ENTASx CCC, generally with x = 0, 1, 2 or 3, to inform targets that the I3C controller is not expected to exit from idle state neither to communicate on the I3C bus before an interval of, respectively, 1 µs, 100 µs, 2 ms or 50 ms, has elapsed. This delay defines the T_{CAS} delay for the controller to set the SCL bus clock low and running, after a start condition. More specifically for a Stop mode, the CCC must be restricted to an ENTASx with the value x = 1, 2, or 3.

The general scheme for the I3C wake-up from Stop is the following:

1. First the I3C peripheral requests its kernel clock to the system:
   - On a detected start condition on the I3C bus (SDA line is detected to be driven low while SCL is high).
   - As controller, as initiated by an external I3C target device for a hot-join/in-band interrupt/controller-role request. Once the kernel clock is provided and running, the I3C hardware uses an internal timer to wait for the corresponding T_{CAS} time to elapse, then drives low SCL and continues toggling SCL to let the target perform its hot-join/in-band interrupt/controller-role request on the I3C bus.
   - On a detected hardware trigger to initiate an I3C transfer on the I3C bus (refer to *Section 64.3.4: I3C triggers* to determine whether the trigger feature is supported).
2. The system enables the source oscillator of the I3C kernel clock, and the clock gets ready (after few microseconds from HSI). The user can keep the source oscillator ON in Stop mode to reduce this startup latency, at the expense of power consumption.
   - The I3C peripheral maintains the kernel clock request until the generation of the Stop condition on the I3C bus.
3. After that the kernel clock is running, as controller the I3C peripheral requests its APB clock to the system:
   - On the ACKed address of a received IBI request (*Figure 829*), and it maintains the APB clock request until that IBIF is cleared.

     If the IBI is without MDB: a Stop is normally generated on the I3C bus, even if the APB clock is not yet provided.
   - On the ACKed address of a received controller-role request (*Figure 830*), and it maintains the APB clock request until that CRF is cleared.
   - On the ACKed address of a hot-join request (*Figure 831*), and it maintains the APB clock request until that HJF is cleared.
4. The system is notified of the I3C APB clock request, and the power management unit of the PWR module is awaken.
5. An additional delay may be needed for the regulator, if it must increase the voltage for Run mode.
6. The system enables the system clock that drives the APB clock. There is an additional delay if the selected oscillator source for the system and APB clocks is not the same as

the one driving the I3C kernel clock. If so, the system enables the source oscillator of the system clock, which gets ready after a delay.

7. With the APB clock running, the I3C peripheral can log the I3C transfer in its status and data registers. When the bus transfer is completed, the I3C peripheral generates the corresponding flag (IBIF/CRF/HJF), and the enabled interrupt can wake up the CPU.

As target: wake-up on a I3C reset pattern

The target reset pattern is a specific scheme for a controller to wake up and possibly reset a target from a low power mode. It consists both in the RSTACT CCC and in the in-band reset pattern generation.

As target, the sequence for the I3C wake-up from Stop on a reset pattern is the following:

1. When the I3C peripheral detects a reset pattern on the bus (14 SDA transitions while SCL is kept low), it requests its APB clock to the system to set the RSTF flag of the I3C_EVR register
2. The system is notified of the I3C APB clock request, and the power management unit of the PWR module is awaken (after few microseconds).
3. An additional delay may be needed for the regulator, if it must increase the voltage for Run mode.
4. The system enables the source oscillator of the system clock, which gets ready after few microseconds.
5. With the APB clock running, the I3C peripheral can raise the RSTF flag of the I3C_EVR register, and the enabled interrupt can wake up the CPU.

As target: wake-up on a missed start on the serialbus

1. The I3C peripheral requests its kernel clock to the system:
   - On a detected start condition on the serial bus (SDA line is detected to be driven low while SCL is high)
   - If the kernel clock is not provided before that SCL is driven low by the external controller, then the I3C target detects that a bus start is missed and waits for the kernel clock to be provided. A controller message that is specifically addressed to this low-power device may have been missed (and NAcked). However the I3C peripheral can also detect a missed start if the controller started a CCC or private or legacy I^2C message to another addressed target, or if another target initiated a start for an IBI/CR/HJ request.
   - On a detected hardware trigger to initiate an I3C IBI/CR/HJ transfer on the I3C bus (refer to *Section 64.3.4: I3C triggers* to determine whether the trigger feature is supported).
2. The system enables the source oscillator of the I3C kernel clock, and the clock gets ready (after few microseconds if HSI). The user may keep the source oscillator ON in

Stop mode to reduce this startup latency, at the expense of power consumption. Especially if the controller is in an Activity State of x = 0 (with a 1 µs T_{CAS} latency).

3. After that the kernel clock is running, as target, the I3C peripheral requests its APB clock to the system to raise the WKPF flag:
4. The system is notified of the I3C APB clock request, and the power management unit of the PWR module is awaken (after few microseconds).
5. An additional delay may be needed for the regulator, if it must increase the voltage for Run mode.
6. The system enables the system clock, which drives the APB clock. There is an additional delay if the selected oscillator source for the system and APB clocks is not the same as the one driving the I3C kernel clock. If so, the system enables the source oscillator of the system clock, which gets ready after few microseconds.
7. With the APB clock running:
   – The missed start flag (WKPF of the I3C_EVR register) is raised if it occurred, and the enabled interrupt can wake up the CPU. It is then known that a serial bus transaction was missed, but not whether the target was addressed or not. The software uses a timeout to return to Stop mode, if it is not addressed by the controller again within this interval.

64.14 I3C in low-power modes

Table 633. Effect of low-power modes

Mode	Description
Sleep	No effect. I3C interrupts cause the device to exit Sleep mode.
Stop(1)	The content of the I3C registers is kept when entering Stop mode. I3C hardware manages automatically its own clocks gating and generates, for its kernel clock and APB clock, a clock request output signal to the RCC. I3C interrupts can cause the device to wake up and exit Stop mode.
Standby	The I3C peripheral is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 64.3.2: I3C wake-up from low-power mode(s)* to see which, if any, Stop modes are supported.

64.15 I3C interrupts

Table 634. I3C interrupt requests

Interrupt acronym		Interrupt event	Used as		Interrupt enable: field in I3C_IER	Event flag: field in I3C_EVR	Event clear method: write 1 to field in I3C_CEVR
			Controller	Target			
I3C	EVT	A control word is requested	X	-	CFNFIE	CFNFF	-
		A status word is available	X	-	SFNEIE	SFNEF	-
		Data to be transmitted are requested	X	X	TXFNFIE	TXFNFF	-
		Received data are available	X	X	RXFNEIE	RXFNEF	-
		Controller: frame transfer is completed Target: private transfer is completed	X	X	FCIE	FCF	CFCF
		A private read transfer is prematurely ended by the target (and SMODE = 0 in the I3C_CFGR register)	X	-	RXTGTENDIE	RXTGTENDF	CRXTGTENDF
		An IBI request is received	X	-	IBIIE	IBIF	CIBIF
		A controller-role request is received	X	-	CRIE	CRF	CCRF
		A hot-join request is received	X	-	HJIE	HJF	CHJF
		IBI request is completed	-	X	IBIENDIE	IBIENDF	CIBIENDF
		Serial bus start is missed	-	X	WKPIE	WKPF	CWKPF
		Direct GETACCR CCC is received	-	X	CRUPDIE	CRUPDF	CCRUPDF
		Direct GETSTATUS CCC is received	-	X	STAIE	STAF	CSTAF
		Any direct GETxxx CCC (except GETSTATUS) is received	-	X	GETIE	GETF	CGETF
		Dynamic address is updated (broadcast ENTDAA/RSTDAA or direct SETNEWDA is received)	-	X	DAUPDIE	DAUPDF	CDAUPDF
		Direct SETMWL CCC is received	-	X	MWLUPDIE	MWLUPDF	CMWLUPDF
		Direct SETMRL CCC is received	-	X	MRLUPDIE	MRLUPDF	CMRLUPDF
		A reset pattern is detected	-	X	RSTIE	RSTF	CRSTF
		Bus activity state is updated (direct/broadcast ENTASx CCC is received)	-	X	ASUPDIE	ASUPDF	CASUPDF
		Broadcast/direct ENEC/DISEC CCC is received	-	X	INTUPDIE	INTUPDF	CINTUPDF
		Broadcast DEFTGTS CCC is received	-	X	DEFIE	DEFF	CDEFF
		Broadcast DEFGRPA CCC is received	-	X	GRPIE	GRPF	CGRPF
	ERR	An error occurred	X	X	ERRIE	ERRF	CERRF

64.16 I3C registers

The I3C registers must be accessed with a 32-bit word aligned address.

Note: I3C_RDR and I3C_TDR registers must be accessed with a single significant LSB data byte for, respectively, reading RX-FIFO and writing TX-FIFO.

64.16.1 I3C message control register (I3C_CR)

Address offset: 0x000

Reset value: 0x0000 0000

This register must be used to control the message to emit on the I3C bus:

- when I3C acts as controller (bit[30] = MTYPE[3] = 0): if there is no CCC code to be emitted bits[29:27] = MTYPE[2:0] differ from 110; else the alternate register description *Section 64.16.2* must be considered.
- when I3C acts as target (bit[30] = MTYPE[3] = 1).

When I3C acts as controller:

- If the control FIFO (C-FIFO) is not full (CFNFF = 1 in the I3C_EVR register), writing into this register means pushing a new control word into the C-FIFO; either by software, or automatically by DMA, as defined by CDMAEN in the I3C_CFGR register.
- If C-FIFO is empty and a restart must be emitted with a new control word, the I3C hardware asserts the control FIFO error underrun flag (COVR = 1 in the I3C_SER register). If enabled by ERRIE = 1 in the I3C_IER register, an interrupt is generated.
- After the last message of the frame is completed (a message with MEND = 1 in the I3C_CR register), the I3C hardware asserts the frame completed flag (FCF = 1 in the I3C_EVR register) and the corresponding interrupt, if enabled.

When I3C acts as target, this register is used in register mode:

- Software writes into this register to initiate a command (IBI, controller-role or hot-join request) on the I3C bus.
- C-FIFO is disabled, and there is no DMA mode neither for control words.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MEND	MTYPE[3:0]				Res.	Res.	Res.	ADD[6:0]							RNW
w	w	w	w	w				w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DCNT[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit 31 **MEND**: Message end type/last message of a frame (when the I3C acts as controller)

0: this message from controller is followed by a repeated start (Sr), before another message must be emitted

1: this message from controller ends with a stop (P), being the last message of a frame

Bits 30:27 **MTYPE[3:0]**: Message type (whatever I3C acts as controller/target)

Condition: when I3C acts as I3C controller

0000: **SCL clock is forced to stop** until a next control word is executed

Bits[26:0] are ignored. On a CE1 error detection (ERRF = 1 in the I3C_EVR register and CODERR[3:0] = 0001 in the I3C_SER register) where a start/restart/stop is prevented from being generated, the software must use this message type for SCL “stuck at” recovery. Refer to *Table 631*.

0001: **header message**

Bits[26:0] are ignored. If the addressed target is not responding with an ACK to a private/direct message, as an escalation stage after a failed GETSTATUS tentative, the software must program this with EXITPTRN = 1 in the I3C_CFGR register, so that an HDR exit pattern is emitted on the bus, whatever the header is ACK-ed or NACK-ed (to avoid the target to consider that the I3C bus is in HDR mode). Refer to *Table 631* and MIPI specification about escalation handling.

0010: **private message** (refer to *Figure 826*)

Bits[23:17] (ADD[6:0]) are the emitted 7-bit dynamic address.
Bit[16] (RNW) is the emitted RnW bit.
Bits[15:0] (DCNT[15:0]) are the number of programmed data bytes.
The transferred private message is:

– {S / S + 0b111_1110 + RnW = 0 + Sr/Sr+*} + 7-bit DynAddr + RnW + (8-bit Data + T)* + Sr/P.
– After an S (start), depending upon bit NOARBH in the I3C_CFGR register, the arbitrable header (0b111_1110 + RnW = 0) is inserted or not.
– Sr+*: after an Sr (repeated start), the hardware automatically inserts (0b111_1110 + RnW = 0) if needed, if it follows a previous message without ending by a P (stop).

0011: **direct message (second part of an I3C SDR direct CCC command)** (refer to *Figure 819*)

Bits[23:17] (ADD[6:0]) are the emitted 7-bit dynamic address.
Bit[16] (RNW) is the emitted RnW bit.
Bits[15:0] (DCNT[15:0]) are the number of programmed data bytes.
The transferred direct message is: Sr + 7-bit DynAddr + RnW + (8-bit Data + T)* + Sr/P

0100: **legacy I^2C message** (refer to *Figure 828*)

Bits[23:17] (ADD[6:0]) are the emitted 7-bit static address.
Bit[16] (RNW) is the emitted RnW bit.
Bits[15:0] (DCNT[15:0]) are the number of programmed data bytes.
The transferred legacy I^2C message is:

– {S / S + 0b111_1110 + RnW = 0 + Sr/Sr+*} + 7-bit StaAddr + RnW + (8-bit data + T)* + Sr/P.
– After an S, depending on NOARBH, the arbitrable header (0b111_1110 + RnW = 0) is inserted or not.
– Sr+*: after an Sr (repeated start), the hardware automatically inserts (0b111_1110 + RnW = 0) if needed (if it follows a previous message without ending by a P (stop)).

Others: reserved

Condition: when I3C acts as I3C target

1000: **hot-join request** (W) (refer to *Figure 830*)

The transferred hot-join request is {S +} 0b000_0010 addr + RnW = 0.
Writing the control word initiates the hot-join request if target is allowed to do so (HJEN = 1 in the I3C_DEVR0 register), either actively after a bus idle condition via the hardware issuing a start request (SDA low) and waiting for the controller to activate SCL clock, or passively if the controller initiates a concurrent message.

1001: **controller-role request** (W) (refer to *Figure 831*)

The transferred controller-role request is {S +} DA[6:0] + RnW = 0 (DA in the I3C_DEVR0 register)
Writing the control word initiates the controller-role request if target is allowed to do so (CREN = 1 and DAVAL = 1 in the I3C_DEVR0 register), either actively after a bus idle condition via the hardware issuing a start request (SDA low) and waiting for the controller to activate SCL clock, or passively if the controller initiates a concurrent message.

1010: **IBI (in-band interrupt) request** (R) (refer to *Figure 829*)

Bits[15:0] (DCNT[15:0]) are the number of the IBI data payload (including the first MDB), if any.
The transferred IBI request is {S +} DA[6:0] + RnW = 1 + optional IBI data payload. Writing the control word initiates the IBI request if target is allowed to do so (IBIEN = 1 and DAVAL = 1 in the I3C_DEVR0.register), either actively after a bus idle condition via the hardware issuing a start request (SDA low) and waiting for the controller to activate SCL clock, or passively if the controller initiates a concurrent message.
When acknowledged from controller, the transmitted IBI payload data (optional, depending upon BCR2 in the I3C_BCR register) is defined by DCNT[15:0] in the I3C_CR register and I3C_IBIDR, and must be consistently programmed vs. the IBI payload data size defined by IBIP[2:0] in the I3C_IBIDR register.

Others: reserved

Bits 26:24 Reserved, must be kept at reset value.

Bits 23:17 **ADD[6:0]**: 7-bit I3C dynamic / I^2C static target address (when I3C acts as controller)
When I3C acts as controller, this field is used if MTYPE[3:0] = 0010 (private message), or MTYPE[3:0] = 0011 (direct message), or MTYPE[3:0] = 0100 (legacy I^2C message)

Bit 16 **RNW**: Read / non-write message (when I3C acts as controller)
When I3C acts as controller, this field is used if MTYPE[3:0] = 0010 (private message), or MTYPE[3:0] = 0011 (direct message), or MTYPE[3:0] = 0100 (legacy I^2C message), to emit the RnW bit on the I3C bus.
0: write message
1: read message

Bits 15:0 **DCNT[15:0]**: Count of data to transfer during a read or write message, in bytes (whatever I3C acts as controller/target)
When I3C acts as controller, this field is used if MTYPE[3:0] = 0010 (private message), or MTYPE[3:0] = 0011 (direct message), or MTYPE[3:0] = 0100 (legacy I^2C message), to set the number of exchanged data bytes on the bus. In case of a private or legacy I^2C read/write message, this field must be non-null.
When I3C acts as target, this field is used if MTYPE[3:0] = 1010 (IBI request) and if any IBI data payload (data to be transmitted if BCR2 = 1 in the I3C_BCR register), to set the number of bytes of the IBI data payload (1, 2, 3, or 4).
Linear encoding up to 64 Kbytes - 1
0x0000: no data to transfer
0x0001: 1 byte
0x0002: 2 bytes
...
0xFFFF: 64 Kbytes - 1 byte

64.16.2 I3C message control register [alternate] (I3C_CR)

Address offset: 0x000

Reset value: 0x0000 0000

This write register description must be used to control the message for when the controller has to emit a CCC (whatever is the type of the CCC: for a CCC broadcast, a CCC direct, or a CCC Enter HDR).

This is the alternate description of register I3C_CR, for when MTYPE[3:0] = 0110. Else refer to *I3C message control register (I3C_CR)*.

If the control FIFO (also known as C-FIFO) is not full (CFNFF = 1 in the I3C_EVR register), writing into this register means pushing a new control word into the C-FIFO; either by the software or automatically by DMA, as defined by the CDMAEN bit in the I3C_CFGR register.

When the last message of the frame is completed (a message with MEND = 1 in the I3C_CR register), the I3C hardware asserts the frame completed flag (FCF = 1 in the I3C_EVR register) and the corresponding interrupt, if enabled.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MEND	MTYPE[3:0]				Res.	Res.	Res.	CCC[7:0]							
w	w	w	w	w				w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DCNT[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bit 31 **MEND**: Message end type/last message of a frame (when I3C acts as controller)

0: this message from controller is followed by a repeated start (Sr), before another message must be emitted

1: the message from the controller ends with a stop (P), being the last message of a frame

Bits 30:27 **MTYPE[3:0]**: Message type (when I3C acts as controller)

Condition: when I3C acts as I3C controller

0110: broadcast/direct CCC command (refer to *Table 630*, *Figure 819*, *Figure 820*, *Figure 821*)

Bits[23:16] (CCC[7:0]) are the emitted 8-bit CCC code

Bits[15:0] (DCNT[15:0]) are the number of the CCC defining bytes, or CCC sub-command bytes, or CCC data bytes.

If Bit[23] = CCC[7] = 1: this is the first part of an I3C SDR direct CCC command

The transferred direct CCC command (first part) message is:

– {S / S + 0b111_1110 +RnW = 0 / Sr+*} + (direct CCC + T) + (8-bit Data + T)* + Sr

– After an S (start), depending upon NOARBH in the I3C_CFGR register, the arbitrable header (0b111_1110 + RnW = 0) is inserted or not.

– Sr+*: after an Sr (repeated start), the hardware automatically inserts (0b111_1110 + R/W).

If Bit[23] = CCC[7] = 0: this is an I3C SDR broadcast CCC command (including specific ENTDAA, refer to *Figure 820*)

The transferred broadcast CCC command message is:

– {S / S + 0b111_1110 + RnW = 0 / Sr+*} + (broadcast CCC + T) + (8-bit Data + T)* + Sr/P

– After an S (start), depending on NOARBH, the arbitrable header (0b111_1110 + RnW = 0) is inserted or not.

– Sr+*: after an Sr (repeated start), the hardware automatically inserts (0b111_1110 + R/W).

Others: reserved

Bits 26:24 Reserved, must be kept at reset value.

Bits 23:16 **CCC[7:0]**: 8-bit CCC code (when I3C acts as controller)

If bit[23] = CCC[7] = 1, this is the first part of an I3C SDR direct CCC command.

If bit[23] = CCC[7] = 0, this is an I3C SDR broadcast CCC command (including ENTDAA).

Bits 15:0 **DCNT[15:0]**: Count of related data to the CCC command to transfer as CCC defining bytes, or CCC sub-command bytes, or CCC data bytes, in bytes

Linear encoding up to 64 Kbytes - 1.

0x0000: no data to transfer.

Note: Value mandatory when emitting ENTDAA broadcast CCC (refer to Figure 820).

0x0001: 1 byte

Note: Value mandatory when emitting RSTACT direct/broadcast CCC (refer to Figure 821).

0x0002: 2 bytes

...

0xFFFF: 64 Kbytes - 1 byte

64.16.3 I3C configuration register (I3C_CFGR)

Address offset: 0x004

Reset value: 0x0000 0000

This register is used to configure:

- features that apply when the I3C acts as controller or target: RX-FIFO and TX-FIFO management (RXDMAEN, RXTHRES, RXFLUSH, TXDMAEN, TXTHRES, TXFLUSH), I3C peripheral role (CRINIT)
- dedicated features when the I3C acts as a controller: frame-based control-word triggering (TSFSET), FIFOs management (TMODE, SMODE, SFLUSH, SDMAEN, CDMAEN), and miscellaneous ones (HJACK, HKSDAEN, EXITPTRN, RSTPATRN, NOARBH)

The configuration fields CRINIT, HKSDAEN can be modified only when EN = 0. This condition is respected if they are modified at the same time when EN is set to 1 (it is not necessary to set EN later on, with another write operation).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TSFSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFLUSH	CDMA EN	TMODE	SMODE	SFLUSH	SDMA EN
	w									w	rw	rw	rw	w	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TX THRES	TX FLUSH	TX DMAEN	Res.	RX THRES	RX FLUSH	RX DMAEN	HJACK	Res.	HK SDAEN	EXIT PTRN	RST PTRN	NO ARBH	CRINIT	EN
	rw	w	rw		rw	w	rw	rw		rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **TSFSET**: Frame transfer set (software trigger) (when I3C acts as controller)

This bit can only be written. When I3C acts as I3C controller:

0: no action

1: setting this bit initiates a frame transfer by causing the hardware to assert the flag CFNFF in the I3C_EVR register (C-FIFO not full and a control word is needed)

Note: If this bit is not set, the other alternative for the software to initiate a frame transfer is to directly write the first control word register (I3C_CR) while C-FIFO is empty (CFEF = 1 in the I3C_EVR register). Then, if the first written control word is not tagged as a message end (MEND = 0 in the I3C_CR register), it causes the hardware to assert CFNFF.

Bits 29:24 Reserved, must be kept at reset value.

Bit 23 Reserved, must be kept at reset value.

Bit 22 Reserved, must be kept at reset value.

Bit 21 **CFLUSH**: C-FIFO flush (when I3C acts as controller)

This bit can only be written.

0: no action

1: flush C-FIFO

Bit 20 **CDMAEN**: C-FIFO DMA request enable (when I3C acts as controller)

When I3C acts as controller:

0: DMA mode is disabled for C-FIFO

- Software writes and pushes control word(s) into C-FIFO (writes I3C_CR register), as needed for a given frame
- A next control word transfer can be written by software either via polling on the flag CFNFF = 1 in the I3C_EVR register, or via interrupt notification (enabled by CFNFIE = 1 in the I3C_IER register).

1: DMA mode is enabled for C-FIFO

- DMA writes and pushes control word(s) into C-FIFO (writes I3C_CR register), as needed for a given frame.
- A next control word transfer is automatically written by the programmed hardware (via the asserted C-FIFO DMA request from the I3C and the programmed DMA channel).

Bit 19 **TMODE**: Transmit mode (when I3C acts as controller)

When I3C acts as controller, this bit is used for the C-FIFO and TX-FIFO management vs. the emitted frame on the I3C bus.

0: C-FIFO and TX-FIFO are not preloaded before starting to emit a frame transfer.

A frame transfer starts as soon as the first control word is present in C-FIFO.

1: C-FIFO and TX-FIFO are first preloaded (also TX-FIFO if needed, depending on the frame format) before starting to emit a frame transfer. Refer to *Section 64.10.2* for more details.

Bit 18 **SMODE**: S-FIFO enable / status receive mode (when I3C acts as controller)

When I3C acts as controller, this bit is used to enable the FIFO for the status (S-FIFO) of the exchanged message on the I3C bus.

When I3C acts as target, this bit must be cleared.

0: S-FIFO is disabled

- Status register (I3C_SR) is used without FIFO mechanism.
- There is no SCL stalling if a new status register content is not read.
- Status register must be read before being overwritten by the hardware.
- Must have SDMAEN = 0 in the I3C_CFGR register.

1: S-FIFO is enabled.

- Each message status must be read.
- There is SCL stalling when the S-FIFO is full and a next message status must be read.
- S-FIFO overrun error is reported after the maximum SCL clock stalling time.

Bit 17 **SFLUSH**: S-FIFO flush (when I3C acts as controller)

This bit can be written and used only when I3C acts as controller.

0: no action

1: flush S-FIFO

Bit 16 **SDMAEN**: S-FIFO DMA request enable (when I3C acts as controller)

This bit must be cleared if SMODE = 0 in the I3C_CFGR register (S-FIFO is disabled). In other words, DMA mode cannot be used if S-FIFO is disabled. Then the status register I3C_SR can be read or not.

This bit can be set or cleared if SMODE = 1 (S-FIFO is enabled). In other words, status register I3C_SR must be read for each message, either by software, or via an allocated DMA channel.

0: DMA mode is disabled for reading status register I3C_SR

- SMODE = 0: software can read the I3C_SR register after a completed frame (FCF = 1 in the I3C_EVR register) or an error (ERRF = 1 in the I3C_EVR register). Via polling on these register flags or via interrupt notification (enabled by FCIE = 1 and ERRIE = 1 in the I3C_IER register).
- SMODE = 1: software must read and pop a status word from S-FIFO (read I3C_SR register) after each asserted flag SFNEF = 1. Via polling on this register flag or via interrupt notification (enabled by SFNEIE = 1 in the I3C_IER register).

1: DMA mode is enabled for reading status register I3C_SR

- Must have SMODE = 1 in the I3C_CFGR register (S-FIFO enabled)
- DMA reads and pops status word(s) from S-FIFO (it reads I3C_SR register)
- Status word(s) are automatically read by the programmed hardware (via the asserted S-FIFO DMA request from the I3C and the programmed DMA channel).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TXTHRES**: TX-FIFO threshold (whatever I3C acts as controller/target)

This threshold defines, compared to the TX-FIFO level, when the TXFNFF flag is set in the I3C_EVR register (and consequently if TXDMAEN = 1 when is asserted a DMA TX request).

0: 1-byte threshold

TXFNFF is set when 1 byte must be written in TX-FIFO (in I3C_TDR).

1: 1-word / 4-byte threshold

TXFNFF is set when 1 word / 4 bytes must be written in TX-FIFO (in the I3C_TDWR register). If the a number of the last transmitted data is not a multiple of 4 bytes (XDCNT[1:0] = 00 in the I3C_SR register), only the relevant 1, 2, or 3 valid LSB bytes of the last word are taken into account by the hardware, and sent on the I3C bus.

Bit 13 **TXFLUSH**: TX-FIFO flush (whatever I3C acts as controller/target)

This bit can only be written.

When the I3C acts as target, this bit can be used to flush the TX-FIFO on a private read if the controller has aborted the data read (driven low the T bit), and there is/are remaining data in the TX-FIFO (ABT = 1, and XDCNT[15:0] in the I3C_SR register < TGTTDCNT[15:0] in the I3C_TGTTDR register).

0: no action

1: flush TX-FIFO

Bit 12 **TXDMAEN**: TX-FIFO DMA request enable (whatever I3C acts as controller/target)

0: DMA mode is disabled for TX-FIFO

- Software writes and pushes a data byte/word into TX-FIFO (writes I3C_TDR or I3C_TDWR register), to be transmitted over the I3C bus.
- A next data byte/word must be written by the software either via polling on the flag TXFNFF = 1 or via interrupt notification (enabled by TXFNFIE = 1).

1: DMA mode is enabled for TX-FIFO

- DMA writes and pushes data byte(s)/word(s) into TX-FIFO (writes I3C_TDR or I3C_TDWR register).
- A next data byte/word transfer is automatically pushed by the programmed hardware (via the asserted TX-FIFO DMA request from the I3C and the programmed DMA channel).

Bit 11 Reserved, must be kept at reset value.

Bit 10 **RXTHRES**: RX-FIFO threshold (whatever I3C acts as controller/target)

This threshold defines, compared to the RX-FIFO level, when the RXFNEF flag in the I3C_EVR register is set (and consequently if RXDMAEN = 1 when is asserted a DMA RX request).

0: 1-byte threshold

RXFNEF is set when 1 byte must be read in RX-FIFO (in the I3C_RDR register).

1: 1-word/4-bytes threshold

RXFNEF is set when 1 word / 4 bytes is/are to be read in RX-FIFO (in I3C_RDWR). In the case of a number of last received data being not a multiple of 4 bytes, only the relevant 1, 2 or 3 valid LSB bytes of the last word are to be considered by the software. The number of effective received data bytes is reported by XDCNT[15:0] in the I3C_SR register.

Bit 9 **RXFLUSH**: RX-FIFO flush (whatever I3C acts as controller/target)

This bit can only be written.

0: no action

1: flush RX-FIFO

Bit 8 **RXDMAEN**: RX-FIFO DMA request enable (whatever I3C acts as controller/target)

0: DMA mode is disabled for RX-FIFO

- Software reads and pops a data byte/word from RX-FIFO (it reads I3C_RDR or I3C_RDWR register).
- A next data byte/word must be read by the software either via polling flag RXFNEF = 1 in the I3C_EVR register, or via interrupt notification (enabled by RXFNEIE = 1 in the I3C_IER register).

1: DMA mode is enabled for RX-FIFO

- DMA reads and pops data byte(s)/word(s) from RX-FIFO (reads I3C_RDR or I3C_RDWR register).
- A next data byte/word is automatically read by the programmed hardware (via the asserted RX-FIFO DMA request from the I3C and the programmed DMA channel).

Bit 7 **HJACK**: Hot-join request acknowledge (when I3C acts as a controller)

0: hot-join request is not acknowledged

After the NACK, the controller continues as initially programmed (the hot-joining target is aware of the NACK and must emit another hot-join request later on).

1: hot-join request is acknowledged

After the ACK, the controller continues as initially programmed. The software is notified by the HJ interrupt (flag HJF is set in the I3C_EVR register), and must initiate the ENTDAA sequence later on, potentially preventing other hot-join requests with a disable target events command (DISEC, with DISHJ = 1).

Bit 6 Reserved, must be kept at reset value.

Bit 5 **HKSDAEN:** High-keeper enable on SDA line (when I3C acts as a controller)

0: High-keeper is disabled

1: High-keeper is enabled, and the weak pull-up is effective on the T bit, instead of the open-drain class pull-up.

Note: This bit can be modified only when EN = 0 in the I3C_CFGR register.

Bit 4 **EXITPTRN:** HDR exit pattern enable (when I3C acts as a controller)

This bit can be modified only when there is no on-going frame.

0: HDR exit pattern is not sent after the issued message header (MTYPE[3:0] = 0001 in the I3C_CR register). This is used to send the header, to test ownership of the bus when there is a suspicion of a problem after controller-role hand-off (new controller did not assert its controller-role by accessing the previous one in less than the delay defined by the activity state).

1: HDR exit pattern is sent after the issued message header (MTYPE[3:0] = 0001). This is used on a controller error detection and escalation handling, in case of a not responding target to a private message or a direct read CCC.

The HDR exit pattern is sent whatever the message header {S/Sr + 0x7E addr + W} is ACK-ed or NACK-ed..

Bit 3 **RSTPTRN:** HDR reset pattern enable (when I3C acts as a controller)

This bit can be modified only when there is no on-going frame.

0: standard stop emitted at the end of a frame

1: HDR reset pattern is inserted before the stop of any emitted frame that includes a RSTACT CCC command

Bit 2 **NOARBH**: No arbitrable header after a start (when I3C acts as a controller)

This bit can be modified only when there is no on-going frame.

0: An arbitrable header (0b111_1110 + RnW = 0) is emitted after a start and before a legacy I^2C message or an I3C SDR private read/write message (default).

1: No arbitrable header

- The target address is emitted directly after a start in case of a legacy I^2C message or an I3C SDR private read/write message.

- This is a more performing option (when the emission of the 0x7E arbitrable header is useless), but must be used only when the controller is sure that the addressed target device cannot emit concurrently an IBI or a controller-role request (to ensure no misinterpretation and no potential conflict between the address emitted by the controller in open-drain mode and the same address a target device can emit after a start, for IBI or MR).

Bit 1 **CRINIT**: Initial controller/target role

This bit can be modified only when EN = 0 in the I3C_CFGR register.

0: target role

Once enabled by setting EN = 1, the I3C peripheral initially acts as a target. I3C does not drive SCL line and does not enable SDA pull-up, until it eventually acquires the controller role.

1: controller role

Once enabled by setting EN = 1, the I3C peripheral initially acts as a controller. It has the I3C controller role, so drives SCL line and enables SDA pull-up, until it eventually offers the controller role to an I3C secondary controller.

Bit 0 **EN**: I3C enable (whatever I3C acts as controller/target)

0: I3C is disabled

- Except registers, the I3C peripheral is under reset (partial reset).

- Before clearing EN, when I3C acts as a controller, all the possible target requests must be disabled using DISEC CCC.

- When I3C acts as a target, software must not disable the I3C, unless a partial reset is needed.

1: I3C is enabled

In this state, some register fields cannot be modified (like CRINIT, HKSDAEN for the I3C_CFGR)

64.16.4 I3C receive data byte register (I3C_RDR)

Address offset: 0x010

Reset value: 0x0000 0000

This register is used to read received data bytes from the I3C bus if the RX-FIFO is configured with a byte-based read access (RXTHRES = 0 in the I3C_CFGR register). Then:

- On read, I3C_RDR returns a 32-bit data word, with the received data byte in LSB position, and other reserved bits read as 0.
- When RXDMAEN = 1 in the I3C_CFGR register: programmed I3C and DMA automatically manage by hardware the relevant and successive reads.
- When RXDMAEN = 0: before a read, the software must wait to be notified that a next received data byte must be read via the RX-FIFO non empty flag (RXFNEF = 1 in the I3C_EVR register) or via the corresponding interrupt if enabled. Last received byte of a given message to be read is also marked with RXLASTF = 1 in the I3C_EVR register when acting as controller.
- If the RX-FIFO is full, a new byte is received and cannot be pushed into the RX-FIFO without waiting anymore, the software is notified by error flag ERRF = 1 in the I3C_EVR register and by a data overrun flag DOVRF = 1 in the I3C_SER register (and corresponding interrupt if enabled).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDB0[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **RDB0[7:0]**: 8-bit received data on I3C bus.

64.16.5 I3C receive data word register (I3C_RDWR)

Address offset: 0x014

Reset value: 0x0000 0000

This register is used to read received data bytes from the I3C bus if the RX-FIFO is configured with a 32-bit word-based read access (RXTHRES = 1 in the I3C_CFGR register). Then:

- When RXDMAEN = 1 in the I3C_CFGR register: programmed I3C and DMA automatically manage by hardware the relevant and successive reads.
- When RXDMAEN = 0: before a read, the software must first wait to be notified that must be read a next received data word via the RX-FIFO non empty flag (RXFNEF = 1 in the I3C_EVR register) or via the corresponding interrupt if enabled. Last received

word/byte(s) of a given message to be read is also marked with RXLASTF = 1 in the I3C_EVR register when acting as controller.

- If the RX-FIFO holds less than four bytes, a read on I3C_RDWR returns a data word padded with null byte(s): the available byte(s) in LSB position(s) is (are) padded with zero byte(s) in MSB position(s).
- If the RX-FIFO is full and a new byte is received and cannot be pushed into the RX-FIFO without waiting anymore, the software is notified by an error flag ERRF = 1 in the I3C_EVR register and a data overrun DOVR = 1 in the I3C_SER register (and corresponding interrupt if enabled).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDB3[7:0]								RDB2[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDB1[7:0]								RDB0[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **RDB3[7:0]**: 8-bit received data (latest byte on I3C bus).

Bits 23:16 **RDB2[7:0]**: 8-bit received data (next byte after RDB1 on I3C bus).

Bits 15:8 **RDB1[7:0]**: 8-bit received data (next byte after RDB0 on I3C bus).

Bits 7:0 **RDB0[7:0]**: 8-bit received data (earliest byte on I3C bus).

64.16.6 I3C transmit data byte register (I3C_TDR)

Address offset: 0x018

Reset value: 0x0000 0000

This register is used to write data bytes to be transmitted over the I3C bus.

This register implements a byte-based write access to the transmit FIFO (TX-FIFO), and is used when TXTHRES = 0 in the I3C_CFGR register.

When the I3C acts as controller:

- When TXDMAEN = 1 and if TXTHRES = 0 in the I3C_CFGR register: programmed I3C and DMA automatically manage by hardware the relevant and successive writes.
- When TXDMAEN = 0 and if TXTHRES = 0: before a write, the software must wait to be notified that must be written a next data byte via the TX-FIFO non full flag (TXFNFF = 1 in the I3C_EVR register) or via the corresponding interrupt if enabled. Last transmitted byte of a given message to be written is also marked with TXLASTF = 1 in the I3C_EVR register.

When the I3C acts as target:

- When TXDMAEN = 1 and if TXTHRES = 0 and if PRELOAD = 1 in the I3C_TGTTDR register: programmed I3C and DMA automatically manage by hardware the relevant and successive number of writes to I3C_TDR register, according to TGTTDCNT[15:0] in the I3C_TGTDR register.
- When TXDMAEN = 0 and if TXTHRES = 0:
  - when PRELOAD = 1 in the I3C_TGTTDR register: before a write, the software must wait to be notified that must be written a next data byte via the TX-FIFO non

full flag (TXFNFF = 1) or via the corresponding interrupt, if enabled. The last transmitted byte of the initially programmed TGTTDCNT[15:0] is also marked with TXLASTF = 1 in the I3C_EVR register.

– when PRELOAD = 0: before a write, the software must wait to be notified that can be written up to 8 next data bytes (TX-FIFO size) via the TX-FIFO empty flag (TXFEF = 1 in the I3C_EVR register), or via the corresponding interrupt if enabled. If the software needs to write another data byte, it must wait for the TX-FIFO be empty (TXFEF = 1), and then write this data byte to be transmitted before nine SCL clock periods elapse, to avoid a data underrun error flag (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register).

If the TX-FIFO is empty and the controller cannot wait longer a data byte to be transmitted, the software is notified by an error flag ERRF = 1 in the I3C_EVR register and a data underrun flag DOF = 1 (and corresponding interrupt if enabled) in the I3C_SER register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDB0[7:0]							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TDB0[7:0]**: 8-bit data to transmit on I3C bus.

64.16.7 I3C transmit data word register (I3C_TDWR)

Address offset: 0x01C

Reset value: 0x0000 0000

This register is used to write 32-bit data words to be transmitted over the I3C bus.

This register implements a word-based write access to the transmit FIFO (TX-FIFO), and is used when TXTHRES = 1 in the I3C_CFGR register.

When the I3C acts as controller:

- When TXDMAEN = 1 and if TXTHRES = 1 in the I3C_CFGR register, programmed I3C and DMA automatically manage by hardware the relevant and successive writes.
- When TXDMAEN = 0 and if TXTHRES = 1: before a write, the software must wait to be notified that next data word/byte(s) must be written via the TX-FIFO non full flag (TXFNFF = 1 in the I3C_EVR register), or via the corresponding interrupt if enabled. Last transmitted word/byte(s) of a given message to be written in TX-FIFO is also marked with TXLASTF = 1 in the I3C_EVR register.

When the I3C acts as target:

- When TXDMAEN = 1 and if TXTHRES = 1 and if PRELOAD = 1 in the I3C_TGTTDR register: programmed I3C and DMA automatically manage by hardware the relevant and successive number of writes to the I3C_TDWR register, as per TGTTDCNT[15:0]

in the I3C_TGTDR register.

- When TXDMAEN = 0 and if TXTHRES = 1:
  - when PRELOAD = 1: before a write, the software must wait to be notified that must be written a next data word via the TX-FIFO non full flag (TXFNFF = 1 in the I3C_EVR register) or via the corresponding interrupt, if enabled. Last transmitted word of the initially programmed TGTTDCNT[15:0] in the I3C_TGTTDR register is also marked with TXLASTF = 1 in the I3C_EVR register.
  - when PRELOAD = 0: before a write, the software must wait to be notified that can be written up to two next data words (TX-FIFO size) via the TX-FIFO empty flag (TXFEF = 1 in the I3C_EVR register) or via the corresponding interrupt if enabled. If the software needs to write another data word, it must wait for TX-FIFO to be empty (TXFEF = 1), and then write the next data word to be transmitted before nine SCL clock periods elapse, to avoid a data underrun error flag (ERRF = 1 in the I3C_EVR register and DOVR = 1 in the I3C_SER register).

If the TX-FIFO is empty and the controller/target cannot wait longer a data byte to be transmitted, the software is notified by an error flag ERRF = 1 in the I3C_EVR register and a data underrun flag DOF = 1 in the I3C_SER register (and corresponding interrupt if enabled).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TDB3[7:0]								TDB2[7:0]							
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TDB1[7:0]								TDB0[7:0]							
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:24 **TDB3[7:0]**: 8-bit transmit data (latest byte on I3C bus).

Bits 23:16 **TDB2[7:0]**: 8-bit transmit data (next byte after TDB1[7:0] on I3C bus).

Bits 15:8 **TDB1[7:0]**: 8-bit transmit data (next byte after TDB0[7:0] on I3C bus).

Bits 7:0 **TDB0[7:0]**: 8-bit transmit data (earliest byte on I3C bus)

64.16.8 I3C IBI payload data register (I3C_IBIDR)

Address offset: 0x020

Reset value: 0x0000 0000

This register is used for the IBI payload data.

When I3C acts as target:

- if BCR2 = 0 in the I3C_BCR register, this register is useless.
- if BCR2 = 1, this register must be written by software
  - to be emitted on the I3C bus as the IBI payload data, after that the IBI request (MTYPE[3:0] = 1010 in the I3C_CR register) is acknowledged by the controller; with the IBI data payload size defined by DCNT[15:0] in the I3C_CR register.
  - Maximum (static) payload data size is given by IBIP[2:0] in the I3C_MAXRLR register. it can be 1, 2, 3 or 4 bytes.
  - DCNT[15:0] must be set between 1 (for the mandatory data byte MDB[7:0]) and the maximum IBIP[2:0].

When I3C acts as controller: if IBIACK= 1 in the I3C_DEVRx register, once it acknowledges on the I3C bus the IBI request from the target x:

- if IBIDEN = 0 (BCR[2] = 0 received from target x) in the I3C_DEVRx register, this register is useless.
- if IBIDEN = 1 (BCR[2] = 1 received from target x):
  - This register is internally written by hardware from the IBI payload data received on the I3C bus (including the first mandatory data byte MDB[7:0]).
  - When the last byte of the payload is received in I3C_IBIDR (when target drives T-bit = 0), IBIF flag is set (and corresponding interrupt if enabled) in the I3C_EVR register.
  - Then, the software can identify the target x via the received and logged 7-bit address in RADD[6:0] in the I3C_RMR register.
  - The software can read this register and interpret byte(s) according to the number of bytes effectively received and logged in IBIRDCNT[2:0] in the I3C_RMR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IBIDB3[7:0]								IBIDB2[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IBIDB1[7:0]								IBIDB0[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **IBIDB3[7:0]**: 8-bit IBI payload data (latest byte on I3C bus).

Bits 23:16 **IBIDB2[7:0]**: 8-bit IBI payload data (next byte on I3C bus after IBIDB1[7:0]).

Bits 15:8 **IBIDB1[7:0]**: 8-bit IBI payload data (next byte on I3C bus after IBIDB0[7:0]).

Bits 7:0 **IBIDB0[7:0]**: 8-bit IBI payload data (earliest byte on I3C bus, MDB[7:0] mandatory data byte).

64.16.9 I3C target transmit configuration register (I3C_TGTTDR)

Address offset: 0x024

Reset value: 0x0000 0000

When I3C acts as target, this register must be used to preload a number of data bytes in the TX-FIFO, so that they are ready to be transmitted on the I3C bus, when they are accepted/acknowledged by the controller, whatever the DMA mode is used or not (independently from TXDMAEN in the I3C_CFGR register).

When I3C acts as target, alternatively, if the number of data bytes to be transmitted is less than or equal to the TX-FIFO size (8 bytes), the software can directly use and write byte(s) into the I3C_TDR register (or I3C_TDWR register, depending upon TXTHRES in the I3C_CFGR register), via polling on the TX-FIFO empty flag (TXFEF = 1 in the I3C_EVR register), or via the corresponding enabled interrupt.

In any case, since an I3C target is unable to stretch/stall the SCL line, the software can identify a TX-FIFO underrun via DOR = 1 in the I3C_EVR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRE LOAD
															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TGTTDCNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **PRELOAD**: Preload of the TX-FIFO (when I3C is configured as target)

This bit must be written and asserted by software in the same access when is written and defined the number of bytes to preload into the TX-FIFO and to transmit.

This bit is cleared by hardware when all the data bytes to transmit are loaded into the TX-FIFO.

0: no TX-FIFO preload

1: TX-FIFO preload

Bits 15:0 **TGTTDCNT[15:0]**: Transmit data counter, in bytes (when I3C is configured as target)

This bitfield must be written by software in the same access when is asserted PRELOAD, in order to define the number of bytes to preload and to transmit.

This bitfield is updated by hardware and reports, when read, the remaining number of bytes to be loaded into the TX-FIFO.

64.16.10 I3C status register (I3C_SR)

Address offset: 0x030

Reset value: 0x0000 0000

This register is used to read the status about the exchanged message on the I3C bus:

- in FIFO mode: when the I3C acts as controller and if S-FIFO is enabled via SMODE = 1 in the I3C_CFGR register:
  - Software is notified via SFNEF = 1 in the I3C_EVR register if there is a status register to be read (and corresponding interrupt if enabled), when not in DMA mode (SDMAEN = 0)
  - In DMA mode (SDMAEN = 1), programmed I3C and DMA automatically manage by hardware the relevant and successive reads.
  - Software is notified via COVR = 1 in the I3C_SER register on an S-FIFO overflow (and ERRF = 1 in the I3C_EVR register and corresponding interrupt if enabled)
- in register mode: if SMODE = 0 in the I3C_CFGR register
  - Software can use the flags FCF in the I3C_SER register and ERRF = 1 in the I3C_EVR register (and corresponding interrupt if enabled) to read this register
  - This register can be overwritten by hardware on a new message completion, without any notification.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MID[7:0]								Res.	Res.	Res.	Res.	Res.	DIR	ABT	Res.
r	r	r	r	r	r	r	r						r	r	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XDCNT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **MID[7:0]**: Message identifier/counter of a given frame (when the I3C acts as controller)

When I3C acts as controller, this fbitield identifies the control word message (I3C_CR) to whom the I3C_SR status register refers.

First message of a frame is identified with MID[7:0] = 0.

This bitfield is incremented (by hardware) on the completion of a new message control word (I3C_CR) over I3C bus. This field is reset for every new frame start.

Bits 23:19 Reserved, must be kept at reset value.

Bit 18 **DIR**: Message direction

Whatever the I3C acts as controller or target, this bit indicates the direction of the related message on the I3C bus

0: write

1: read

Note: ENTDAA CCC is considered as a write command.

Bit 17 **ABT**: A private read message is ended prematurely by the target (when the I3C acts as controller)

When the I3C acts as controller, this bit indicates if the private read data transmitted by the target early terminates (the target drives T bit low earlier vs. what the controller expects in terms of programmed number of read data bytes DCNT[15:0] in the I3C_CR register).

0: no early completion from the target

1: early completion from the target

Bit 16 Reserved, must be kept at reset value.

Bits 15:0 **XDCNT[15:0]**: Data counter

Condition: during the dynamic address assignment process (ENTDAA CCC)

- When the I3C acts as controller: number of targets detected on the bus
- When the I3C acts as target: number of transmitted bytes

Condition: for other transfers, during the message

- Whatever the I3C acts as controller or target: number of data bytes read from or transmitted on the I3C bus during the message

64.16.11 I3C status error register (I3C_SER)

Address offset: 0x034

Reset value: 0x0000 0000

This read register is used to get more information about the error when an error is raised by hardware and notified to the software via the error flag ERRF = 1 in the I3C_EVR register (and corresponding interrupt if enabled).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	DERR	DNACK	ANACK	COVR	DOVR	STALL	PERR	CODERR[3:0]			
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **DERR**: Data error (when the I3C acts as controller)

0: no detected error

1: controller detected a data error during the controller-role hand-off procedure (GETACCCR CCC, formerly known as GETACCMST) when the received target address or/and the parity bit do no match. Active controller keeps controller-role.

Bit 9 **DNACK**: Data not acknowledged (when the I3C acts as controller)

0: no detected error

1: controller detected that a data byte is not acknowledged by a target, either during:

i) a legacy I2C write transfer

ii) the second trial when sending dynamic address during ENTDAA procedure

Bit 8 **ANACK**: Address not acknowledged (when the I3C is configured as controller)

0: no detected error

1: controller detected that the static/dynamic address was not acknowledged by a target, either during:

i) a legacy I2C read/write transfer

ii) a direct CCC write transfer

iii) the second trial of a direct CCC read transfer

iv) a private read/write transfer

Bit 7 **COVR**: C-FIFO underrun or S-FIFO overrun (when the I3C acts as controller)

0: no detected error

1: controller detected either:

i) a C-FIFO underrun: control FIFO is empty and a restart must be emitted

ii) an S-FIFO overrun: S-FIFO is full and a new message ends

Bit 6 **DOVR**: RX-FIFO overrun or TX-FIFO underrun

0: no detected error

1: whatever controller or target, hardware detected either:

i) a TX-FIFO underrun: TX-FIFO is empty and a write data byte must be transmitted

ii) a RX-FIFO overrun: RX-FIFO is full and a new data byte is received

Bit 5 **STALL**: SCL stall error (when the I3C acts as target)

0: no detected error

1: target detected that SCL was stable for more than 125 µs during an I3C SDR data read (during a direct CCC read, a private read, or an IB)

Bit 4 **PERR**: Protocol error

0: no detected error

1: whatever controller or target, hardware detected a protocol error, as detailed in CODERR[3:0]

Bits 3:0 **CODERR[3:0]**: Protocol error code/type
0000: CE0 error (transaction after sending CCC):
controller detected an illegally formatted CCC
0001: CE1 error (monitoring error):
controller detected that transmitted data on the bus is different from expected
0010: CE2 error (no response to broadcast address):
controller detected a not acknowledged broadcast address (0b111_1110)
0011: CE3 error (failed controller-role hand-off):
controller detected the new controller did not drive bus after controller-role hand-off
1000: TE0 error (invalid broadcast address 0b111_1110 + W):
target detected an invalid broadcast address 0b111_1110 + W
1001: TE1 error (CCC code):
target detected a parity error on a CCC code via a parity check (vs. T bit)
1010: TE2 error (write data):
target detected a parity error on a write data via a parity check (vs. T bit)
1011: TE3 error (assigned address during dynamic address arbitration):
target detected a parity error on the assigned address during dynamic address arbitration via a parity check (vs. PAR bit)
1100: TE4 error (0b111_1110 + R missing after Sr during dynamic address arbitration):
target detected a 0b111_1110 + R missing after Sr during dynamic address arbitration
1101: TE5 error (transaction after detecting CCC):
target detected an illegally formatted CCC
1110: TE6 error (monitoring error):
target detected that transmitted data on the bus is different from expected
others: reserved

64.16.12 I3C received message register (I3C_RMR)

Address offset: 0x040

Reset value: 0x0000 0000

When the I3C acts as controller, this read register is used to log the received target address, and the IBI received payload data size.

When the I3C acts as target, this read register is used to log the received CCC code

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RADD[6:0]							Res.
								r	r	r	r	r	r	r	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RCODE[7:0]								Res.	Res.	Res.	Res.	Res.	IBIRDCNT[2:0]		
r	r	r	r	r	r	r	r						r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:17 **RADD[6:0]**: Received target address (when the I3C is configured as controller)
When the I3C is configured as controller, this field logs the received dynamic address from the target during acknowledged IBI or controller-role request.

Bit 16 Reserved, must be kept at reset value.

Bits 15:8 **RCODE[7:0]**: Received CCC code (when the I3C is configured as target)
When the I3C is configured as target, this field logs the received CCC code.

Bits 7:3 Reserved, must be kept at reset value.

Bits 2:0 **IBIRDCNT[2:0]**: IBI received payload data count (when the I3C is configured as controller)
When the I3C is configured as controller, this field logs the number of data bytes effectively received in the I3C_IBIDR register.

64.16.13 I3C event register (I3C_EVR)

Address offset: 0x050

Reset value: 0x0000 0003

This is a read register, used for reporting event flags.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GRPF	DEFF	INT UPDF	AS UPDF	RSTF	MRL UPDF	MWL UPDF	DA UPDF	STAF	GETF	WKPF	Res.	HJF	CR UPDF	CRF	IBI ENDF
r	r	r	r	r	r	r	r	r	r	r		r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IBIF	Res.	Res.	Res.	ERRF	RXTGT ENDF	FCF	Res.	RX LASTF	TX LASTF	RX FNEF	TX FNFF	SFNEF	CFNFF	TXFEF	CFEF
r				r	r	r		r	r	r	r	r	r	r	r

Bit 31 **GRPF**: Group addressing flag (when the I3C acts as target)
When the I3C acts as target (and is typically controller-capable), this flag is asserted by hardware to indicate that the broadcast DEFGRPA CCC (define list of group addresses) has been received. Then, software can store the received data for when getting controller role.
The flag is cleared when software writes 1 into the corresponding CGRPF bit in the I3C_CR register.

Bit 30 **DEFF**: DEFTGTS flag (when the I3C acts as target)
When the I3C acts as target (and is typically controller capable), this flag is asserted by hardware to indicate that the broadcast DEFTGTS CCC (define list of targets) has been received. Then, software can store the received data for when getting the controller role.
The flag is cleared when software writes 1 into the corresponding CDEFF bit in the I3C_CEVR register.

Bit 29 **INTUPDF**: Interrupt/controller-role/hot-join update flag (when the I3C acts as target)
When the I3C acts as target, this flag is asserted by hardware to indicate that the direct or broadcast ENEC/DISEC CCC (enable/disable target events) has been received, where a target event is either an interrupt/IBI request, a controller-role request, or an hot-join request. Then, software must read respectively IBIEN, CREN, or HJEN in the I3C_DEVR0 register.
The flag is cleared when software writes 1 into the corresponding CINTUPDF bit in the I3C_CEVR register.

Bit 28 **ASUPDF**: Activity state update flag (when the I3C acts as target)
When the I3C acts as target, this flag is asserted by hardware to indicate that the direct or broadcast ENTASx CCC (with x = 0...3) has been received. Then, software must read AS[1:0] in the I3C_DEVR0 register.
The flag is cleared when software writes 1 into the corresponding CASUPDF bit in the I3C_CEVR register.

Bit 27 **RSTF**: Reset pattern flag (when the I3C acts as target)

When I3C acts as target, this flag is asserted by hardware to indicate that a reset pattern has been detected (14 SDA transitions while SCL is low, followed by repeated start, then stop).

Then, when not in Stop mode, software must read RSTACT[1:0] and RSTVAL in the I3C_DEVR0 register, to know the required reset level.

- If RSTVAL = 1: when the RSTF is asserted (and/or the corresponding interrupt if enabled), RSTACT[1:0] in the I3C_DEVR0 register dictates the reset action to be performed by the software, if any.
- If RSTVAL = 0: when the RSTF is asserted (and/or the corresponding interrupt if enabled), the software must issue an I3C reset after a first detected reset pattern, and a system reset on the second one.

When in Stop mode, the corresponding interrupt can be used to wake up the device.

The flag is cleared when software writes 1 into the corresponding CRSTF bit in the I3C_CEVR register.

Bit 26 **MRLUPDF**: Maximum read length update flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that a direct SETMRL CCC (set max read length) has been received. Then, software must read MRL[15:0] in the I3C_MAXRLR register to get the maximum read length value.

The flag is cleared when software writes 1 into the corresponding CMRLUPDF bit in the I3C_CEVR register.

Bit 25 **MWLUPDF**: Maximum write length update flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that a direct SETMWL CCC (set max write length) has been received. Then, software must read MWL[15:0] in the I3C_MAXRLR register to get the maximum write length value.

The flag is cleared when software writes 1 into the corresponding CMWLUPDF bit in the I3C_CEVR register.

Bit 24 **DAUPDF**: Dynamic address update flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that a dynamic address update has been received via any of the broadcast ENTDAA, RSTDAA and direct SETNEWDA CCC. Then, software must read DA[6:0] and DAVAL in the I3C_DEVR0 register to get the dynamic address update.

The flag is cleared when software writes 1 into the corresponding CDAUPDF bit in the I3C_CEVR register.

Bit 23 **STAF**: Get status flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that a direct GETSTATUS CCC of format 1 (without defining byte or with defining byte TGTSTAT) has been received.

The flag is cleared when software writes 1 into the corresponding CSTAF bit in the I3C_CEVR register.

Bit 22 **GETF**: Get flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that any direct CCC of get type (GET*** CCC) except the GETSTATUS of format 1 (but including GETSTATUS of format 2) has been received.

The flag is cleared when software writes 1 into the corresponding CGETF bit in the I3C_CEVR register.

Bit 21 **WKPF**: Wake-up/missed start flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that a start has been detected (an SDA falling edge followed by an SCL falling edge) but on the next SCL falling edge, the I3C kernel clock is (still) gated. Thus an I3C bus transaction may have been lost by the target.

The corresponding interrupt can be used to wake up the device from a low power (Sleep or Stop) mode.

The flag is cleared when software writes 1 into the corresponding CWKPF bit in the I3C_CEVR register.

Bit 20 Reserved, must be kept at reset value.

Bit 19 **HJF**: Hot-join flag (when the I3C acts as controller)

When the I3C acts as controller, this flag is asserted by hardware to indicate that an hot join request has been received.

The flag is cleared when software writes 1 into the corresponding CHJF bit in the I3C_CEVR register.

Bit 18 **CRUPDF**: Controller-role update flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that it has now gained the controller role after the completed controller-role hand-off procedure.

The flag is cleared when software writes 1 into the corresponding CCRUPDF bit in the I3C_CEVR register.

Bit 17 **CRF**: Controller-role request flag (when the I3C acts as controller)

When the I3C acts as controller, this flag is asserted by hardware to indicate that a controller-role request has been acknowledged and completed (by hardware). The software must then issue a GETACCCR CCC (get accept controller role) for the controller-role hand-off procedure.

The flag is cleared when software writes 1 into the corresponding CCRF bit in the I3C_CEVR register.

Bit 16 **IBIENDF**: IBI end flag (when the I3C acts as target)

When the I3C acts as target, this flag is asserted by hardware to indicate that an IBI transfer has been received and completed (IBI acknowledged and IBI data bytes read by controller if any).

The flag is cleared when software writes 1 into the corresponding CIBIENDF bit in the I3C_CEVR register.

Bit 15 **IBIF**: IBI flag (when the I3C acts as controller)

When the I3C acts as controller, this flag is asserted by hardware to indicate that an IBI request has been received.

The flag is cleared when software writes 1 into the corresponding CIBIF bit in the I3C_CEVR register.

Bits 14:12 Reserved, must be kept at reset value.

Bit 11 **ERRF**: Flag (whatever the I3C acts as controller/target)

This flag is asserted by hardware to indicate that an error occurred.Then, software must read I3C_SER to get the error type.

The flag is cleared when software writes 1 into the corresponding CERRF bit in the I3C_CEVR register.

Bit 10 **RXTGTENDF**: Target-initiated read end flag (when the I3C acts as controller)

When the I3C acts as controller, and only if the S-FIFO is disabled (SMODE = 0 in the I3C_CFGR register), this flag is asserted by hardware to indicate that the target has prematurely ended a read transfer. Then, software must read the status register I3C_SR to check information related to the last message and get the number of received data bytes on the prematurely read transfer (XDCNT in the I3C_SR register).

The flag is cleared when software writes 1 into the corresponding CRXTGTENDF bit in the I3C_CEVR register.

Bit 9 **FCF**: Frame complete flag (whatever the I3C acts as controller/target)

When the I3C acts as controller, this flag is asserted by hardware to indicate that a frame has been (normally) completed on the I3C bus, for example, when a stop is issued.

When the I3C acts as target, this flag is asserted by hardware to indicate that a message addressed to/by this target has been (normally) completed on the I3C bus, for example, when a next stop or repeated start is then issued by the controller.

The flag is cleared when software writes 1 into the corresponding CFCF bit in the I3C_CEVR register.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **RXLASTF**: Last read data byte/word flag (when the I3C acts as controller)

When the I3C acts as controller, this flag is asserted by hardware to indicate that the last data byte/word (depending upon RXTHRES in the I3C_CFGR register) of a message must be read from the RX-FIFO. The flag is de-asserted by hardware when the last data byte/word of a message is read.

Bit 6 **TXLASTF**: Last written data byte/word flag (whatever the I3C acts as controller/target)

This flag is asserted by hardware to indicate that the last data byte/word (depending upon TXTHRES in the I3C_CFGR register) of a message must be written to the TX-FIFO. The flag is de-asserted by hardware when the last data byte/word of a message is written.

Bit 5 **RXFNEF**: RX-FIFO not empty flag (whatever the I3C acts as controller/target)

This flag is asserted/de-asserted by hardware to indicate that a data byte must/must not be read from the RX-FIFO.

Note: The software must wait for RXFNEF = 1 (by polling or via the enabled interrupt) before reading from RX-FIFO (reading from I3C_RDR or I3C_RDWR, depending upon RXTHRES).

Bit 4 **TXFNFF**: TX-FIFO not full flag (whatever the I3C acts as controller/target)

This flag is asserted/de-asserted by hardware to indicate that a data byte/word must/must not be written to the TX-FIFO.

Note: The software must wait for TXFNFF = 1 (by polling or via the enabled interrupt) before writing to TX-FIFO (writing to I3C_TDR or I3C_TDWR, depending upon TXTHRES).

Note: When the I3C acts as target, if the software intends to use the TXFNFF flag for writing into I3C_TDR/I3C_TDWR, it must have configured and set the TX-FIFO preload (write PRELOAD in the I3C_TGTTDR register).

Bit 3 **SFNEF**: S-FIFO not empty flag (when the I3C acts as controller)

When the I3C acts as controller, if the S-FIFO is enabled (SMODE = 1 in the I3C_CFGR register), this flag is asserted by hardware to indicate that a status word must be read from the S-FIFO. The flag is de-asserted by hardware to indicate that a status word is not to be read from the S-FIFO.

Bit 2 **CFNFF**: C-FIFO not full flag (when the I3C acts as controller)

When the I3C acts as controller, this flag is asserted by hardware to indicate that a control word must be written to the C-FIFO. The flag is de-asserted by hardware to indicate that a control word is not to be written to the C-FIFO.

Note: The software must wait for CFNFF = 1 (by polling or via the enabled interrupt) before writing to C-FIFO (writing to I3C_CR).

Bit 1 **TXFEF**: TX-FIFO empty flag (whatever the I3C acts as controller/target)

This flag is asserted by hardware to indicate that the TX-FIFO is empty.

This flag is de-asserted by hardware to indicate that the TX-FIFO is not empty.

Bit 0 **CFEF**: C-FIFO empty flag (whatever the I3C acts as controller)

This flag is asserted by hardware to indicate that the C-FIFO is empty when controller, and that the I3C_CR register contains no control word (none IBI/CR/HJ request) when target.

This flag is de-asserted by hardware to indicate that the C-FIFO is not empty when controller, and that the I3C_CR register contains one control word (a pending IBI/CR/HJ request) when target.

Note: When the I3C acts as controller, if the C-FIFO and TX-FIFO preload is configured (TMODE = 1 in the I3C_CFGR register), the software must wait for TXFEF = 1 and CFEF = 1 before starting a new frame transfer.

64.16.14 I3C interrupt enable register (I3C_IER)

Address offset: 0x054

Reset value: 0x0000 0000

This register is used to enable/disable, at bit level, an interrupt, for each of the following event/flag.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GRPIE	DEFIE	INTUPDIE	ASUPDIE	RSTIE	MRLUPDIE	MWLUPDIE	DAUPDIE	STAIE	GETIE	WKPIE	Res.	HJIE	CRUPDIE	CRIE	IBIENDIE
r	r	r	r	r	r	r	r	r	r	r		r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IBIIE	Res.	Res.	Res.	ERRIE	RXTGTENDIE	FCIE	Res.	Res.	Res.	RXFNEIE	TXFNFIE	SFNEIE	CFNFIE	Res.	Res.
r				r	r	r				r	r	r	r		

Bit 31 **GRPIE**: DEFGRPA CCC interrupt enable (when the I3C acts as target)

0: interrupt disabled

1: interrupt enabled

Bit 30 **DEFIE**: DEFTGTS CCC interrupt enable (when the I3C acts as target)

0: interrupt disabled

1: interrupt enabled

Bit 29 **INTUPDIE**: ENEC/DISEC CCC interrupt enable (when the I3C acts as target)

0: interrupt disabled

1: interrupt enabled

Bit 28 **ASUPDIE**: ENTASx CCC interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 27 **RSTIE**: reset pattern interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 26 **MRLUPDIE**: SETMRL CCC interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 25 **MWLUPDIE**: SETMWL CCC interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 24 **DAUPDIE**: ENTDAA/RSTDAA/SETNEWDA CCC interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 23 **STAIE**: format 1 GETSTATUS CCC interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 22 **GETIE**: GETxxx CCC interrupt enable (except GETSTATUS of format 1) (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 21 **WKPIE**: Wake-up interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 20 Reserved, must be kept at reset value.

Bit 19 **HJIE**: Hot-join interrupt enable (when the I3C acts as controller)
0: interrupt disabled
1: interrupt enabled

Bit 18 **CRUPDIE**: Controller-role update interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 17 **CRIE**: Controller-role request interrupt enable (when the I3C acts as controller)
0: interrupt disabled
1: interrupt enabled

Bit 16 **IBIENDIE**: IBI end interrupt enable (when the I3C acts as target)
0: interrupt disabled
1: interrupt enabled

Bit 15 **IBIIE**: IBI request interrupt enable (when the I3C acts as controller)
0: interrupt disabled
1: interrupt enabled

Bits 14:12 Reserved, must be kept at reset value.

Bit 11 **ERRIE**: error interrupt enable (whatever the I3C acts as controller/target)
0: interrupt disabled
1: interrupt enabled

Bit 10 **RXTGTENDIE**: target-initiated read end interrupt enable (when the I3C acts as controller)
0: interrupt disabled
1: interrupt enabled

Bit 9 **FCIE**: frame complete interrupt enable (whatever the I3C acts as controller/target)
0: interrupt disabled
1: interrupt enabled

Bits 8:6 Reserved, must be kept at reset value.

Bit 5 **RXFNEIE**: RX-FIFO not empty interrupt enable (whatever the I3C acts as controller/target)
0: interrupt disabled
1: interrupt enabled

Bit 4 **TXFNFIE**: TX-FIFO not full interrupt enable (whatever the I3C acts as controller/target)
0: interrupt disabled
1: interrupt enabled

Bit 3 **SFNEIE**: S-FIFO not empty interrupt enable when the I3C acts as controller
0: interrupt disabled
1: interrupt enabled

Bit 2 **CFNFIE**: C-FIFO not full interrupt enable when the I3C acts as controller
0: interrupt disabled
1: interrupt enabled

Bits 1:0 Reserved, must be kept at reset value.

64.16.15 I3C clear event register (I3C_CEVR)

Address offset: 0x058

Reset value: 0x0000 0000

This write register is used to clear individually, at bit level, the corresponding event flag of the I3C_EVR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CGRPF	CDEFF	CINTUPDF	CASUPDF	CRSTF	CMRLUPDF	CMWLUPDF	CDAUPDF	CSTAF	CGETF	CWKPF	Res.	CHJF	CCRUPDF	CCRF	CIBIENDF
w	w	w	w	w	w	w	w	w	w	w		w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CIBIF	Res.	Res.	Res.	CERRF	CRXTGTENDF	CFCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w				w	w	w									

Bit 31 **CGRPF**: Clear DEFGRPA CCC flag (when the I3C acts as target)
0: no effect
1: clear GRPF

Bit 30 **CDEFF**: Clear DEFTGTS CCC flag (when the I3C acts as target)
0: no effect
1: clear DEFF

Bit 29 **CINTUPDF**: Clear ENEC/DISEC CCC flag (when the I3C acts as target)
0: no effect
1: clear CINTUPDF

Bit 28 **CASUPDF**: Clear ENTASx CCC flag (when the I3C acts as target)
0: no effect
1: clear ASUPDF

Bit 27 **CRSTF**: Clear reset pattern flag (when the I3C acts as target)
0: no effect
1: clear RSTF

Bit 26 **CMRLUPDF**: Clear SETMRL CCC flag (when the I3C acts as target)
0: no effect
1: clear MRLUPDF

Bit 25 **CMWLUPDF**: Clear SETMWL CCC flag (when the I3C acts as target)
0: no effect
1: clear MWLUPDF

Bit 24 **CDAUPDF**: Clear ENTDAA/RSTDAA/SETNEWDA CCC flag (when the I3C acts as target)
0: no effect
1: clear DAUPDF

Bit 23 **CSTAF**: Clear format 1 GETSTATUS CCC flag (when the I3C acts as target)
0: no effect
1: clear STAF

Bit 22 **CGETF**: Clear GETxxx CCC flag (except GETSTATUS of format 1) (when the I3C acts as target)
0: no effect
1: clear GETF

Bit 21 **CWKPF**: Clear wake-up flag (when the I3C acts as target)
0: no effect
1: clear WKPF

Bit 20 Reserved, must be kept at reset value.

Bit 19 **CHJF**: Clear hot-join flag (when the I3C acts as controller)
0: no effect
1: clear HJF

Bit 18 **CCRUPDF**: Clear controller-role update flag (when the I3C acts as target)
0: no effect
1: clear CRUPDF

Bit 17 **CCRF**: Clear controller-role request flag (when the I3C acts as controller)
0: no effect
1: clear CRF

Bit 16 **CIBIENDF**: Clear IBI end flag (when the I3C acts as target)
0: no effect
1: clear IBIENDF

Bit 15 **CIBIF**: Clear IBI request flag (when the I3C acts as controller)
0: no effect
1: clear IBIF

Bits 14:12 Reserved, must be kept at reset value.

Bit 11 **CERRF**: Clear error flag (whatever the I3C acts as controller/target)
0: no effect
1: clear ERRF

Bit 10 **CRXTGTENDF**: Clear target-initiated read end flag (when the I3C acts as controller)
0: no effect
1: clear RXTGTENDF

Bit 9 **CFCF**: Clear frame complete flag (whatever the I3C acts as controller/target)
0: no effect
1: clear FCF

Bits 8:0 Reserved, must be kept at reset value.

64.16.16 I3C own device characteristics register (I3C_DEVR0)

Address offset: 0x060

Reset value: 0x0000 0000

When the I3C peripheral acts as target, this register is used to write or read its own device characteristics.

When the I3C peripheral acts as controller, the field DA[6:0] is used to write and store its own dynamic address.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTVAL	RSTACT[1:0]		AS[1:0]		HJEN	Res.	CREN	IBIEN
							r	r	r	r	r	rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DA[6:0]							DAVAL
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **RSTVAL**: Reset action is valid (when the I3C acts as target)
This bit is asserted by hardware to indicate that the RSTACT[1:0] field has been updated on the reception of a broadcast or direct write RSTACT CCC (target reset action) and is valid.
This bit is cleared by hardware when the target receives a frame start.
When the device is not in Stop mode:
– If RSTVAL = 1: when RSTF in the I3C_EVR register is asserted (and/or the corresponding interrupt if enabled), RSTACT[1:0] in the I3C_DEVR0 register dictates the reset action to be performed by the software, if any.
– If RSTVAL = 0: when RSTF is asserted (and/or the corresponding interrupt if enabled), the software must issue an I3C reset after a first detected reset pattern, and a system reset on the second one.
When in Stop mode, the corresponding interrupt can be used to wake up the device.

Bits 23:22 **RSTACT[1:0]**: Reset action/level on received reset pattern (when the I3C acts as target)

This read field is used by hardware on the reception of a direct read RSTACT CCC in order to return the corresponding data byte on the I3C bus.

This read field is updated by hardware on the reception of a broadcast or direct write RSTACT CCC (target reset action).

Only the defining bytes 0x00, 0x01 and 0x02 are mapped, and RSTACT[1:0] = Defining Byte[1:0].

00: no reset action

01: first level of reset: the application software must either:

a) partially reset the I3C peripheral, by a write and clear of the enable bit of the I3C configuration register (write EN = 0). This resets the I3C bus interface and the I3C kernel sub-parts, without modifying the content of the I3C APB registers (except the EN bit).

b) fully reset the I3C peripheral, including all its registers, via a write and set of the I3C reset control bit of the RCC (reset and clock controller) register.

10: second level of reset: the application software must issue a warm reset, also known as a system reset:

– the software writes and sets the SYSRESETREQ control bit of the AIRCR register, when the device is controlled by a Cortex®-M.

11: no reset action

Bits 21:20 **AS[1:0]**: Activity state (when the I3C acts as target)

This read field is updated by hardware on the reception of a ENTASx CCC (enter activity state, with x = 0-3):

00: activity state 0

01: activity state 1

10: activity state 2

11: activity state 3

Bit 19 **HJEN**: Hot-join request enable (when the I3C acts as target)

This bit is initially written by software when EN = 0, and is updated by hardware on the reception of DISEC CCC with DISHJ= 1 (cleared) and the reception of ENEC CCC with ENHJ= 1 (set). This bit can only be written by software when EN = 0 in the I3C_CFGR register.

0: hot-join request disabled

1: hot-join request enabled

Bit 18 Reserved, must be kept at reset value.

Bit 17 **CREN**: Controller-role request enable (when the I3C acts as target)

This bit is initially written by software when EN = 0, and is updated by hardware on the reception of DISEC CCC with DISCR = 1 (cleared) and the reception of ENEC CCC with ENCR = 1 (set). This bit can only be written by software when EN = 0 in the I3C_CFGR register.

0: controller-role request disabled

1: controller-role request enabled

Bit 16 **IBIEN**: IBI request enable (when the I3C acts as target)

This bit is initially written by software when EN = 0, and is updated by hardware on the reception of DISEC CCC with DISINT = 1 (cleared) and the reception of ENEC CCC with ENINT = 1 (set). This bit can only be written by software when EN = 0 in the I3C_CFGR register.

0: IBI request disabled

1: IBI request enabled

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:1 **DA[6:0]**: 7-bit dynamic address

When the I3C acts as controller, this field can be written by software, for defining its own dynamic address.

When the I3C acts as target, this field is updated by hardware on the reception of either the broadcast ENTDAA CCC or the direct SETNEWDA CCC.

Bit 0 **DAVAL**: Dynamic address is valid (when the I3C acts as target)

When the I3C acts as controller, this bit can be written by software, for validating its own dynamic address, for example before a controller-role hand-off.

When the I3C acts as target, this bit is asserted by hardware on the acknowledge of the broadcast ENTDAA CCC or the direct SETNEWDA CCC, and this field is cleared by hardware on the acknowledge of the broadcast RSTDAA CCC.

64.16.17 I3C device x characteristics register (I3C_DEVRx)

Address offset: 0x060 + 0x4 * x, (x = 1 to 4)

Reset value: 0x0000 0000

When the I3C peripheral acts as controller, this register is used to define and store some characteristics of a device target x with their related management from the controller, to communicate accordingly with any of this target x over the I3C bus. Then, the hardware can autonomously identify and acknowledge an allowed IBI or/and controller-role request from a target x, receive the expected IBI payload data if any, and notify the software via the corresponding flag IBIF/CRF (and the corresponding interrupt if enabled) in the I3C_EVR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUSP	IBIDEN	CRACK	IBIACK
r												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DA[6:0]							Res.
								rw	rw	rw	rw	rw	rw	rw	

Bit 31 **DIS**: Disables writes to DA[6:0] (when the I3C acts as controller)

When the I3C acts as controller, once the software sets IBIACK= 1 or CRACK= 1, this read bit is set by hardware (DIS= 1) to lock the configured DA[6:0] and IBIDEN values.

Then, to be able to modify DA[6:0], IBIDEN or SUSP, the software must wait for DIS to be de-asserted by hardware (polling on DIS= 0) before modifying these three assigned values to the target x. Indeed, the target can request an IBI or a controller-role while the controller intends to modify DA[6:0], IBIDEN or SUSP.

0: write to DA[7:0] and to IBIDEN in the I3C_DEVRx register is allowed

1: write to DA[7:0] and to IBIDEN is disabled/locked

Bits 30:20 Reserved, must be kept at reset value.

Bit 19 **SUSP**: Suspend/stop I3C transfer on received IBI (when the I3C acts as controller)

When the I3C acts as controller, this bit can be used to receive an IBI from target x with pending read notification feature (received MDB[7:5] = 3'b101).

If this bit is set, when an IBI is received and completed (IBIF = 1 in the I3C_EVR register), a stop is emitted on the I3C bus and both C-FIFO and TX-FIFO are automatically flushed by hardware. The controller execution flow is stopped, even if a next control message is programmed. When the IBI is completed, the controller software can issue a new control word, such as a private read, to the target device that initiated the IBI request.

0: C-FIFO and TX-FIFO are not flushed after an IBI request from target x is acknowledged and completed, and depending on the presence or absence of a next control word, a repeated start or a stop is emitted

1: I3C transfer is stopped and both C-FIFO and TX-FIFO are flushed after receiving an IBI request from target x

Bit 18 **IBIDEN**: IBI data enable (when the I3C acts as controller)

When the I3C acts as controller, this bit must be written by software to store the BCR[2] bit as received from the target x during broadcast ENTDAA or direct GETBCR CCC via the received I3C_RDR.

Writing to this field has no impact when the DIS = 1 in the I3C_DEVRx register.

0: no data byte follows the acknowledged IBI from target x

1: the mandatory data byte MDB[7:0] follows the acknowledged IBI from target x

Bit 17 **CRACK**: Controller-role request acknowledge (when the I3C acts as controller)

When the I3C acts as controller, this bit is written by software to define the acknowledge policy to be applied on the I3C bus on the reception of a controller-role request from target x:

0: a controller-role request from target x must be NACK-ed

After the NACK, the message continues as initially programmed (the target is aware of the NACK and can emit another controller-role request later on)

1: a controller-role request (with 7-bit dynamic address DA[6:0]) from target x must be ACKed

- The field DIS is asserted by hardware to protect DA[6:0] from being modified by software meanwhile the hardware can store internally the current DA[6:0] into the kernel clock domain.

- After the ACK, the message continues as initially programmed. The software is notified by the controller-role request flag (CRF = 1 in the I3C_EVR register) and/or the corresponding interrupt if enabled; For effectively granting the controller-role to the requesting secondary controller, software must issue a GETACCCR (formerly known as GETACCMST), followed by a stop.

- Independently of CRACK configuration for this or other devices, further controller-role request(s) are NACK-ed until controller-role request flag (CRF) and IBI flag (IBIF) in the I3C_EVR register are both cleared.

Bit 16 **IBIACK**: IBI request acknowledge (when the I3C acts as controller)

When the I3C acts as controller, this bit is written by software to define the acknowledge policy to be applied on the I3C bus on the reception of an IBI request from target x:

0: an IBI request from target x must be NACK-ed

- After the NACK, the message continues as initially programmed (the target is aware of the NACK and can emit another IBI request later on)

1: an IBI request (with 7-bit dynamic address DA[6:0]) from target x must be ACKed

- The field DIS is asserted by hardware to protect DA[6:0] from being modified by software meanwhile the hardware can store internally the current DA[6:0] into the kernel clock domain.

- After the ACK, the controller logs the IBI payload data, if any, depending on I3C_DEVRx.IBIDEN.

- The software is notified by the IBI flag (IBIF = 1) and/or the corresponding interrupt if enabled;

- Independently from IBIACK configuration for this or other devices, further IBI request(s) are NACK-ed until IBI request flag (IBIF) and controller-role request flag (CRF) are both cleared.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:1 **DA[6:0]**: Assigned I3C dynamic address to target x (when the I3C acts as controller)

When the I3C acts as controller, this field must be written by software to store the 7-bit dynamic address that the controller sends via a broadcast ENTDAA or a direct SETNEWDA CCC acknowledged by the target x.

Writing to this field has no impact when the read field DIS = 1 in the I3C_DEVRx register.

Bit 0 Reserved, must be kept at reset value.

64.16.18 I3C maximum read length register (I3C_MAXRLR)

Address offset: 0x090

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set or get the maximum read length value exchanged with the controller during respectively GETMRL or SETMRL CCC. This register is also used to set the IBI data payload size.

This register can be written by the software when EN = 0 in the I3C_CFGR register.

When receiving a private read message, the target ends the data transmission (by driving T-bit = 0) when the count of transmitted data reaches MRL[15:0] in the I3C_MAXRLR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IBIP[2:0]		
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MRL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **IBIP[2:0]**: IBI payload data maximum size, in bytes (when I3C acts as target)
This field is initially written by software when EN = 0 to set the maximum number of data bytes to be sent to the controller after an IBI request has been acknowledged.This field can be updated by hardware on the reception of SETMRL command (which potentially also updated IBIP[2:0]).
Software is notified of an MRL update by MRLUPF in the I3C_EVR register and the corresponding interrupt, if enabled.
000: null payload data size (only allowed when BCR2 = 0 in the I3C_BCR register)
001: 1 byte (mandatory data byte MDB[7:0]
010: 2 bytes (including first MDB[7:0])
011: 3 bytes (including first MDB[7:0])
100: 4 bytes (including first MDB[7:0])
others: same as 100

Bits 15:0 **MRL[15:0]**: Maximum data read length (when I3C acts as target)
This field is initially written by software when EN = 0 and updated by hardware on the reception of SETMRL command (with potentially also updated IBIP[2:0]).
Software is notified of an MRL update by MRLUPF and the corresponding interrupt, if enabled.
This field is used by hardware to return the value on the I3C bus when the target receives a GETMRL CCC.

64.16.19 I3C maximum write length register (I3C_MAXWLR)

Address offset: 0x094

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set or get the maximum write length value exchanged with the controller during respectively GETMWL or SETMWL CCC.

This register can be written by the software when EN = 0 in the I3C_CFGR register.

On receiving a private write message, the target stops the data reception (extra received data are not written into RX-FIFO) when the count of received data reaches MWL[15:0] in the I3C_MAXWLR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MWL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MWL[15:0]**: Maximum data write length (when I3C acts as target)
This field is initially written by software when EN = 0 and updated by hardware on the reception of SETMWL command.
Software is notified of an MWL update by MWLUPF in the I3C_EVR register and the corresponding interrupt, if enabled.
This field is used by hardware to return the value on the I3C bus when the target receives a GETMWL CCC.

64.16.20 I3C timing register 0 (I3C_TIMINGR0)

Address offset: 0x0A0

Reset value: 0x0000 0000

When the I3C acts as controller, this register is used to configure the SCL clock signal waveform.

This register can be written by the software when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SCLH_I2C[7:0]								SCLL_OD[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCLH_I3C[7:0]								SCLL_PP[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **SCLH_I2C[7:0]**: SCL high duration, used for legacy I^2C messages, in number of kernel clocks cycles:

$t_{SCLH_I2C} = (SCLH_I2C + 1) \times t_{I3CCLK}$

Note: SCLH_I2C is used to generate t_{DIG_H} (I^2C) timing when communicating with I^2C devices.

Note: With I^2C Fm+ device t_{DIG_Hmin} = 260 ns, with I^2C Fm+ device t_{DIG_Hmin} = 600 ns.

Bits 23:16 **SCLL_OD[7:0]**: SCL low duration in open-drain phases, used for legacy I^2C messages and for I3C open-drain phases (address phase following a start, ACK phase during controller-initiated messages, and T bit phase during direct/private/IBI payload), in number of kernel clocks cycles:

$t_{SCLL_OD} = (SCLL_OD + 1) \times t_{I3CCLK}$

Note: SCLL_OD is used to generate both t_{DIG_L} (I^2C) and $t_{DIG_OD_L}$(I3C) timings.

Note: With I^2C Fm+ device t_{DIG_Lmin} = 500 ns, with I^2C Fm device t_{DIG_Lmin} = 1320 ns.

Note: I3C messages: $t_{DIG_OD_Lmin}$ = 200 ns.

Note: If a single I3C frame is gathering I^2C and I3C messages, the SCL low duration during I3C open-drain phases is increased to fit I^2C timings.

Bits 15:8 **SCLH_I3C[7:0]**: SCL high duration, used for I3C messages (both in push-pull and open-drain phases), in number of kernel clocks cycles:

$t_{SCLH_I3C} = (SCLH_I3C + 1) \times t_{I3CCLK}$

Note: SCLH_I3C is used to generate both t_{DIG_H} (I3C) and $t_{DIG_H_MIXED}$ timings.

Note: For mixed bus (with at least one I^2C target): $t_{DIG_H_MIXEDmin}$ = 32 ns and $t_{DIG_H_MIXEDmax}$ = 45 ns (due to I^2C 50 ns spike filter).

Note: For pure I3C bus (with no I^2C targets): t_{DIG_Hmin} = 32 ns.

Bits 7:0 **SCLL_PP[7:0]**: SCL low duration in I3C push-pull phases, in number of kernel clocks cycles:

$t_{SCLL_PP} = (SCLL_PP + 1) \times t_{I3CCLK}$

Note: SCLL_PP is used to generate t_{DIG_L} (I3C in PP) timing.

Note: t_{DIG_Lmin} = 32 ns (max 40/60 duty cycle at 12.5 MHz).

64.16.21 I3C timing register 1 (I3C_TIMINGR1)

Address offset: 0x0A4

Reset value: 0x0000 0000

When the I3C acts as controller, this register is used to configure some I3C timing settings.

When the I3C acts as target and is controller-capable, this register is used to configure a timing for the controller-role hand-off procedure.

This register can be written by the software when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	SDA_HD	Res.	Res.	Res.	Res.	Res.	FREE[6:0]						
			rw						rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	ASNCR[1:0]		AVAL[7:0]							
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **SDA_HD**: SDA hold time (when the I3C acts as controller), in number of kernel clocks cycles (refer to MIPI timing SDA hold time in push-pull t_{HD_PP}):

SDA hold time = (SDA_HD + 0.5) x t_{I3CCLK}

Note: when controller: t_{HD_PPmiN} = min (t_{CR} , t_{CF}) + 3 ns.

Bits 27:23 Reserved, must be kept at reset value.

Bits 22:16 **FREE[6:0]**: Number of kernel clocks cycles that is used to set some MIPI timings like bus free condition time (when the I3C acts as controller)

When the I3C acts as controller:

1. for I3C start timing: it must wait for (bus free condition) time to be elapsed after a stop and before a start, refer to MIPI timings (I3C) t_{CAS} and (I^2C) t_{BUF}. These timings are defined by: t_{BUF} = t_{CAS} = [(FREE[6:0] + 1) x 2 - (0.5 + SDA_HD)] x t_{I3CCLK}

Note: For pure I3C bus: t_{CASmin} = 38.4 ns, and t_{CASmax} = 1 µs, 100 µs, 2 ms, 50 ms for, respectively, ENTAS0, 1, 2, and 3.

Note: For mixed bus with I^2C Fm+ device t_{BUFmin} = 0.5 µs, for mixed bus with I^2C Fm device t_{BUFmin} = 1.3 µs.

2. for I3C repeated start timing: must wait for time to be elapsed after a repeated start (SDA is de-asserted) and before driving SCL low, refer to. MIPI timing t_{CASr}. This timing is defined by: t_{CASr} = [(FREE[6:0] + 1) x 2 - (0.5 + SDA_HD)] x t_{I3CCLK}.

Note: $t_{CASr,\ miN}$ = 19.2 ns.

3. for I3C stop timing: must wait for time to be elapsed after that the SCL clock is driven high, and before the stop condition (SDA is asserted). This timing is defined by: t_{CBP} = (FREE[6:0] + 1) x t_{I3CCLK}.

Note: t_{CBPmiN} = 19.2 ns.

4. for I3C repeated start timing (T-bit when controller ends read with repeated start followed by stop): must wait for time to be elapsed after that the SCL clock is driven high, and before the repeated start condition (SDA is de-asserted). This timing is defined by: t_{CBSr} = (FREE[6:0] + 1) x t_{I3CCLK}.

Note: $t_{CBSr,\ miN}$ = 19.2 ns.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **ASNCR[1:0]**: Activity state of the new controller (when I3C acts as active controller)

This field indicates the time to wait before being accessed as new target, refer AVAL[7:0].

This field can be modified only when the I3C acts as controller.

Bits 7:0 **AVAL[7:0]**: Number of kernel clock cycles to set a time unit of 1 µs, whatever I3C acts as controller or target.

This time unit is then used by the hardware to build some internal timers, corresponding to the following MIPI I3C timings:

When the I3C acts as target:

1. for bus available condition time: it must wait for (bus available condition) time to be elapsed after a stop and before issuing a start request for an IBI or a controller-role request (bus free condition is sustained for at least t_{AVAL}). Refer to MIPI timing t_{AVAL} = 1 µs. This timing is defined by: t_{AVAL} = (AVAL[7:0] + 2) x t_{I3CCLK}
2. for bus idle condition time: it must wait for (bus idle condition) time to be elapsed after that both SDA and SCL are continuously high and stable before issuing a hot-join event. Refer to MIPI v1.1 timing t_{IDLE} = 200 µs . This timing is defined by: t_{IDLE} = (AVAL[7:0] + 2) x 200 x t_{I3CCLK}

When the I3C acts as controller, it cannot stall the clock beyond a maximum stall time (stall the SCL clock low), as follows:

1. on first bit of assigned address during dynamic address assignment: it cannot stall the clock beyond the MIPI timing $t_{STALLDAA}$ = 15 ms. This timing is defined by: $t_{STALLDAAmax}$ = (AVAL[7:0] + 1) x 15000 x t_{I3CCLK}
2. on ACK/NACK phase of I3C/I^2C transfer, on parity bit of write data transfer, on transition bit of I3C read transfer: it cannot stall the clock beyond the MIPI timing t_{STALL} = 100 µs. This timing is defined by: $t_{STALLmax}$ = (AVAL[7:0] + 1) x 100 x t_{I3CCLK}

Whatever the I3C acts as controller or as (controller-capable) target, during a controller-role hand-off procedure:

1. The new controller must wait for $t_{NEWCRLock}$ before pulling SDA low (issuing a start) after a completed GETACCR CCC. Then the new controller, within t_{CAS}, can pull SCL low to activate SCL clock. The active controller must wait for the same $t_{NEWCRLock}$ time, or at least 100 µs, before testing if the new controller has gained control of the bus by pulling SDA low. The time to wait depends upon the value of ANSCR[1:0] in the I3C_TIMINGR1 register:
   - ASNCR[1:0] = 00: $t_{NEWCRLock}$ = (AVAL[7:0] + 1) x t_{I3CCLK}
   - ASNCR[1:0] = 01: $t_{NEWCRLock}$ = (AVAL[7:0] + 1) x 100 x t_{I3CCLK}
   - ASNCR[1:0] = 10: $t_{NEWCRLock}$ = (AVAL[7:0] + 1) x 2000 x t_{I3CCLK}
   - ASNCR[1:0] = 11: $t_{NEWCRLock}$ = (AVAL[7:0] + 1) x 50000 x t_{I3CCLK}

64.16.22 I3C timing register 2 (I3C_TIMINGR2)

Address offset: 0x0A8

Reset value: 0x0000 0000

When the I3C acts as controller, this register is used to configure and enable SCL clock stalling, to enable SCL clock low stalling, if needed by the addressed I3C or legacy I^2C target(s).

This register can be written only when the I3C acts as controller.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STALL[7:0]								Res.	Res.	Res.	Res.	STALLA	STALLC	STALLD	STALLT
rw	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **STALL[7:0]**: Controller clock stall time, in number of kernel clock cycles

t_{SCLL_STALL} = STALL x t_{I3CCLK}

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **STALLA:** Controller clock stall enable on ACK phase

The SCL is stalled (during t_{SCLL_STALL}as defined by STALL) in the address ACK/NACK phase (before the ninth bit). This allows the target to prepare data. The ACK driven by the controller itself on a target-initiated request (IBI/HJ/CR) is not impacted by this control bit.

0: no stall

1: stall enabled

Bit 2 **STALLC:** Controller clock stall enable on PAR phase of CCC

The SCL is stalled during STALL x t_{SCLL_PP} in the T-bit phase of common command code (before the ninth bit). This allows the target to decode the command.

0: no stall

1: stall enabled

Note: t_{SCLL_PP} = (I3C_TIMINGR0.SCLL_PP[7:0] + 1) x t_{I3CCLK}

Bit 1 **STALLD:** Controller clock stall enable on PAR phase of Data

The SCL is stalled during STALL x t_{SCLL_PP} in the T-bit phase (before the ninth bit). This allows the target to read received data.

0: no stall

1: stall enabled

Note: t_{SCLL_PP} = (I3C_TIMINGR0.SCLL_PP[7:0] + 1) x t_{I3CCLK}

Bit 0 **STALLT:** Controller clock stall enable on T-bit phase of data (and on the ACK/NACK phase of data byte of a legacy I^2C read)

The SCL is stalled during STALL x t_{SCLL_PP} in the T-bit phase (before the ninth bit). This allows the target to prepare the data to be sent.

0: no stall

1: stall enabled

Note: t_{SCLL_PP} = (I3C_TIMINGR0.SCLL_PP[7:0] + 1) x t_{I3CCLK}

64.16.23 I3C bus characteristics register (I3C_BCR)

Address offset: 0x0C0

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to configure three bits used by hardware to return the data byte BCR[7:0] on reception of GETBCR or ENTDAA CCC. The returned byte BCR[7:0] on the I3C bus is then as follows:

- BCR[7] = 0 (reserved)
- BCR[6] = I3C_BCR6 (controller capable)
- BCR[5] = 1 (advanced capabilities, use GETCAPS CCC to determine which ones)
- BCR[4] = 0 (not a virtual target)
- BCR[3] = 1 (offline capable)
- BCR[2] = I3C_BCR2 (IBI payload)
- BCR[1] = 1 (IBI request capable)
- BCR[0] = I3C_BCR0 (max data speed limitation)

This register can be written by software only when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BCR6	Res.	Res.	Res.	BCR2	Res.	BCR0
									rw				rw		rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **BCR6**: Controller capable
0: I3C target (no controller capable)
1: I3C controller capable

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **BCR2**: in-band interrupt (IBI) payload
0: no data byte follows the accepted IBI
1: at least one mandatory data byte follows the accepted IBI (and at most 4 data bytes)

Bit 1 Reserved, must be kept at reset value.

Bit 0 **BCR0**: max data speed limitation
0: no limitation
1: limitation, as described by I3C_GETMXDSR.

64.16.24 I3C device characteristics register (I3C_DCR)

Address offset: 0x0C4

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to configure the device characteristics ID, which is used by hardware to return the data byte DCR[7:0] on reception of GETDCR, ENTDAA, or DEFTGTS CCC.

This register can be written by software only when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCR[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **DCR[7:0]**: device characteristics ID
0x00: generic device (for v1.0 devices)
others: ID to describe the type of the I3C sensor/device

Note: The latest MIPI DCR ID assignments are available on https://www.mipi.org.

64.16.25 I3C get capability register (I3C_GETCAPR)

Address offset: 0x0C8

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set the IBI MDB support for pending read notification support, and is used by hardware to return the GETCAP3 byte on reception of GETCAPS CCC, format 1.

The returned byte GETCAP1[7:0] on the I3C bus is then as follows:

- GETCAP1[7:0] = 0 (no HDR)

The returned byte GETCAP2[7:0] on the I3C bus is then as follows:

- GETCAP2[7:6] = 00 (no HDR)
- GETCAP2[5:4] = 00 (no group addressing)
- GETCAP2[3:0] = 0001 (compliant with MIPI specification v1.1)

The returned byte GETCAP3[7:0] on the I3C bus is then as follows:

- GETCAP3[7] = 0 (reserved)
- GETCAP3[6] = CAPPEND in the I3C_GETCAPR register (IBI MDB support for pending read notification)
- GETCAP3[5] = 0 (no HDR)
- GETCAP3[4] = 1 (defining byte support in GETSTATUS)
- GETCAP3[3] = 1 (defining byte support in GETCAPS)
- GETCAP3[2] = 0 (no device-to-device transfer)
- GETCAP3[1] = 0 (no device-to-device transfer)
- GETCAP3[0] = 0 (no multi-lane data transfer)

This register can be written by software only when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CAPPEND	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw														

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **CAPPEND**: IBI MDB support for pending read notification

This bit is written by software during bus initialization (EN = 0), and indicates the support (or not) of the pending read notification via the IBI MDB[7:0] value.

This bit is used to return the GETCAP3 byte in response to the GETCAPS CCC format 1.

0: this I3C when acting as target sends an IBI request without a mandatory data byte value indicating a pending read notification

1: this I3C when acting as target sends an IBI request with a mandatory data byte value (MDB[7:5] = 101), indicating a pending read notification

Bits 13:0 Reserved, must be kept at reset value.

64.16.26 I3C controller-role capability register (I3C_CRCAPR)

Address offset: 0x0CC

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set features the I3C supports as a secondary controller after controller-role hand-off, and is used by hardware to return the CRCAP1 byte and the CRCAP2 byte on reception of GETCAPS CCC, format 2 with the defining byte CRCAPS (0x91).

The returned CRCAP1[7:0] on the I3C bus is then as follows:

- CRCAP1[7:3] = 00000 (reserved)
- CRCAP1[2] = 0 (no multi-lane)
- CRCAP1[1] = CAPGRP in the I3C_CRCAPR register (group management)
- CRCAP1[0] = 1 (hot-join)

The returned CRLCAP2[7:0] on the I3C bus is then as follows:

- CRCAP2[7:4] = 0000 (reserved)
- CRCAP2[3] = CAPDHOFF in the I3C_CRCAPR register (delayed controller-role handoff)
- CRCAP2[2] = 1 (deep sleep capable)
- CRCAP2[1] = 0 (no automatic controller-role pass-back)
- CRCAP2[0] = 1 (IBI ack capable)

This register can be written by software only when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CAP GRP	Res.	Res.	Res.	Res.	Res.	CAP DHOFF	Res.	Res.	Res.
						rw						rw			

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **CAPGRP**: group management support (when acting as controller)

This bit is written by software during bus initialization (EN = 0), and indicates if the I3C is able to support group management when it acts as a controller (after controller-role hand-off) via emitted DEFGRPA, RSTGRPA, and SETGRPA CCC.

This bit is used to return the CRCAP1 byte in response to the GETCAPS CCC format 2.

0: this I3C does not support group address capabilities

1: this I3C supports group address capabilities (when becoming controller)

Bits 8:4 Reserved, must be kept at reset value.

Bit 3 **CAPDHOFF**: delayed controller-role hand-off

This bit is written by software during bus initialization (EN = 0), and indicates if this target I3C needs additional time to process a controller-role hand-off requested by the current controller.

This bit is used to return the CRCAP2 byte in response to the GETCAPS CCC format 2.

0: this I3C does not needs additional time to process a controller-role hand-off

1: this I3C needs additional time to process a controller-role hand-off

Bits 2:0 Reserved, must be kept at reset value.

64.16.27 I3C get max data speed register (I3C_GETMXDSR)

Address offset: 0x0D0

Reset value: 0x0000 0000

When the I3C acts as target, this register is used to set its capabilities and limitations if any. This register is used by hardware to return data byte(s) on reception of GETMXDS CCC, with format 1 (2 bytes, without maxRdTurn), format 2 (5 bytes, with MaxRdTurn), or format 3 (with defining byte WRRDTURN = 0x00: same 5 bytes as format 2, or with defining byte CRHDLY = 0x91: single byte CRHDLY1).

The returned byte MaxWr[7:0] on the I3C bus is then as follows:

- MaxWr[7:4] = 0000 (reserved)
- MaxWr[3] = 1 (defining byte WRRDTURN and CRHDLY support)
- MaxWr[2:0] = 000 (max sustained data rate for non-CCC messages sent by the controller to the target is designed to operate at 12.5 MHz)

The returned byte MaxRd[7:0] on the I3C bus is then as follows:

- MaxRd[7] = 0 (reserved)
- MaxRd[6] = 1 (stop is allowed between write and read)
- MaxRd[5:3] = 100 if TSCO = 0 (clock to data turnaround time t_{SCO} ≤ 12 ns) in the I3C_GETMXDSR register, else 111 (t_{SCO} > 12 ns, refer to the datasheet for more details)
- MaxRd[2:0] = 000 (max sustained data rate for non-CCC messages sent by the target to the controller is designed to operate at 12.5 MHz)

The returned 3-byte MaxRdTurn[23:0], if FMT[1:0] = 00 in the I3C_GETMXDSR register, on the I3C bus is then as follows:

- MaxRdTurn[23:0], with either the MSB (between 65 ms and 16 s), the middle byte (between 256 µs and 65 ms), or the LSB (less than 256 µs) from RDTURN[7:0] in the I3C_GETMXDSR register (others bits are 0), and depending upon FMT[1:0] in the same register.

The returned byte CRHDLY1[7:0] on the I3C bus is then as follows:

- CRHDLY1[7:3] = 00000 (reserved)
- CRHDLY1[2] = 0 if HOFFAS[1:0] = 00 in the I3C_GETMXDSR register, else 1 (set bus activity state)
- CRHDLY1[1:0] = HOFFAS[1:0] (controller-role hand-off activity state)

This register can be written by software only when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSCO	RDTURN[7:0]							
							rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	FMT[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	HOFFAS[1:0]	
						rw	rw							rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **TSCO**: clock-to-data turnaround time (t_{SCO})

This bit is written by software during bus initialization (EN = 0 in the I3C_CFGR register) and is used to specify the clock-to-data turnaround time t_{SCO} (vs. the value of 12 ns). This bit is used by the hardware in response to the GETMXDS CCC to return the encoded clock-to-data turnaround time via the returned MaxRd[5:3] bits.

0: $t_{SCO} \leq 12$ ns

1: $t_{SCO} > 12$ ns (refer to the datasheet for more details)

Bits 23:16 **RDTURN[7:0]**: programmed byte of the 3-byte MaxRdTurn (maximum read turnaround byte)

This bit is written by software during bus initialization (EN = 0) and writes the value of the selected byte (via the FMT[1:0] field) of the 3-byte MaxRdTurn, which is returned in response to the GETMXDS CCC format 2 to encode the maximum read turnaround time.

Bits 15:10 Reserved, must be kept at reset value.

Bits 9:8 **FMT[1:0]**: GETMXDS CCC format

This field is written by software during bus initialization (EN = 0) and indicates how is returned the GETMXDS format 1 (without MaxRdTurn) and format 2 (with MaxRdTurn).

This field is used to return the 2-byte format 1 (MaxWr, MaxRd) or 5-byte format 2 (MaxWr, MaxRd, 3-byte MaxRdTurn) byte in response to the GETCAPS CCC.

00: format 1 (2 bytes with MaxWr with no defining byte, MaxRd)

01: format 2: (5 bytes w.ith MaxWr with no defining byte, MaxRd, MaxRdTurn)

- 3-byte MaxRdTurn is returned with MSB = 0, middle byte = 0 and LSB = RDTURN[7:0].
- Max read turnaround time is less than 256 µs.

10: format 2 (5 bytes with MaxWr with no defining byte, MaxRd, and middle byte of MaxRdTurn)

- 3-byte MaxRdTurn is returned with MSB = 0, middle byte = RDTURN[7:0] and LSB = 0.
- Max read turnaround time is between 256 and 65535 µs

11: format 2 (5 bytes with MaxWr with no defining byte, MaxRd, MSB of MaxRdTurn)

- 3-byte MaxRdTurn is returned with MSB = RDTURN[7:0], middle byte = 0 and LSB = 0.
- Max read turnaround time is between 65535 µs and 16 s.

Bits 7:2 Reserved, must be kept at reset value.

Bits 1:0 **HOFFAS[1:0]**: Controller hand-off activity state

This field is written by software during bus initialization (EN = 0), and indicates in which initial activity state the (other) current controller must expect the I3C bus after a controller-role hand-off to this controller-capable I3C, when returning the defining byte CRHDLY (0x91) to a GETMXDS CCC.

This 2-bit field is used to return the CRHDLY1 byte in response to the GETCAPS CCC format 3, to state the activity state of this I3C when becoming controller after a controller-role hand-off, and consequently the time the former controller must wait before testing this I3C to be confirmed its ownership.

00: activity state 0 is the initial activity state of this I3C before and when becoming controller

01: activity state 1 is the initial activity state of this I3C when becoming controller

10: activity state 2 is the initial activity state of this I3C when becoming controller

11: activity state 3 is the initial activity state of this I3C when becoming controller

64.16.28 I3C extended provisioned ID register (I3C_EPIDR)

Address offset: 0xD4

Reset value: 0x0208 0000

When the I3C acts as target, this register is used to set the 4-bit MIPI instance ID by software, and some other constant bits used for the 48-bit provisioned ID. It is also used by hardware to return the six bytes for the 48-bit provisioned ID on reception of GETPID and ENTDAA CCC.

The 48-bit provisioned ID on the I3C bus is then returned as:

- provisioned ID [47:32] = 0x0208
- provisioned ID [31:16] = 0x1381
- provisioned ID [15:0] = {MIPIID[3:0], 0x000}

This register can be written by software only when EN = 0 in the I3C_CFGR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MIPIMID[14:0]															IDTSEL
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MIPIID[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw												

Bits 31:17 **MIPIMID[14:0]**: 15-bit MIPI manufacturer ID

This read field is the 15-bit STMicroelectronics MIPI ID (0x0104).

This field represents bits[47:33] of the 48-bit provisioned ID.

Bit 16 **IDTSEL**: provisioned ID type selector

This field is set as 0 (vendor fixed value).

This field represents bit[32] of the 48-bit provisioned ID.

Note: Bits[31:16] of the provisioned ID can be 0.

Bits 15:12 **MIPIID[3:0]**: 4-bit MIPI Instance ID

This field is written by software to set and identify individually each instance of this I3C IP with a specific number on a single I3C bus.

This field represents bits[15:12] of the 48-bit provisioned ID.

Note: Bits[11:0] of the provisioned ID can be 0.

Bits 11:0 Reserved, must be kept at reset value.

64.16.29 I3C register map

Table 635. I3C register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	I3C_CR	MEND	MTYPE[3:0]				Res.	Res.	Res.	ADD[6:0] / CCC[7:0]							RNW	DCNT[15:0]																
	Reset value	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	I3C_CFGR	Res.	TSFSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CFLUSH	CDMAEN	TMODE	SMODE	SFLUSH	SDMAEN	Res.	TXTHRES	TXFLUSH	TXDMAEN	Res.	RXTHRES	RXFLUSH	RXDMAEN	HJACK	Res.	HKSDAEN	EXITPTRN	RSTPTRN	NOARBH	CRINIT	EN
	Reset value		0									0	0	0	0	0	0		0	0	0		0	0	0	0		0	0	0	0	0	0
0x010	I3C_RDR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	RDB0[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x014	I3C_RDWR	RDB3[7:0]								RDB2[7:0]								RDB1[7:0]								RDB0[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	I3C_TDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDB0[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x01C	I3C_TDWR	TDB3[7:0]								TDB2[7:0]								TDB1[7:0]								TDB0[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	I3C_IBIDR	IBIDB3[7:0]								IBIDB2[7:0]								IBIDB1[7:0]								IBIDB0[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	I3C_TGTTDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRELOAD	TGTTDCNT[15:0]															
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	I3C_SR	MID[7:0]								Res.	Res.	Res.	Res.	Res.	DIR	ABT	Res.	XDCNT[15:0]															
	Reset value	0	0	0	0	0	0	0	0						0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	I3C_SER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DERR	DNACK	ANACK	COVR	DOVR	STALL	PERR	CODERR[3:0]			
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x040	I3C_RMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RADD[6:0]							s_e_TP	RCODE[7:0]								Res.	Res.	Res.	Res.	Res.	IBIRDCNT[2:0]		
	Reset value									0	0	0	0	0	0	0		0	0	0	0	0	0	0	0						0	0	0

Table 635. I3C register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x050	I3C_EVR	GRPF	DEFF	INTUPDF	ASUPDF	RSTF	MRLUPDF	MWLUPDF	DAUPDF	STAF	GETF	WKPF	Res.	HJF	CRUPDF	CRF	IBIENDF	IBIF	Res.	Res.	Res.	ERRF	RXTGTENDF	FCF	Res.	RXLASTF	TXLASTF	RXFNEF	TXFNFF	SFNEF	CFNFF	TXFEF	CFEF
	Reset value	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0				0	0	0		0	0	0	0	0	0	1	1
0x054	I3C_IER	GRPIE	DEFIE	INTUPDIE	ASUPDIE	RSTIE	MRLUPDIE	MWLUPDIE	DAUPDIE	STAIE	GETIE	WKPIE	Res.	HJIE	CRUPDIE	CRIE	IBIENDIE	IBIIE	Res.	Res.	Res.	ERRIE	RXTGTENDIE	FCIE	Res.	Res.	Res.	RXFNEIE	TXFNFIE	SFNEIE	CFNFIE	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0				0	0	0				0	0	0	0		
0x058	I3C_CEVR	CGRPF	CDEFF	CINTUPDF	CASUPDF	CRSTF	CMRLUPDF	CMWLUPDF	CDAUPDF	CSTAF	CGETF	CWKPF	Res.	CHJF	CRRUPDF	CCRF	CIBIENDF	CIBIF	Res.	Res.	Res.	CERRF	CRXTGTENDF	CFCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0				0	0	0									
0x05C	Reserved																																
0x060	I3C_DEVR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSTVAL	RSTACT [1:0]		AS [1:0]		HJEN	Res.	CREN	IBIEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DA[6:0]							DAVAL
	Reset value								0	0	0	0	0	0		0	0									0	0	0	0	0	0	0	0
0x060 + 4 * x, (x = 1 to 4)	I3C_DEVRx	DIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUSP	IBIDEN	CRACK	IBIACK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DA[6:0]							Res.
	Reset value	0												0	0	0	0									0	0	0	0	0	0	0	
0x090	I3C_MAXRLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IBIP [2:0]			MRL15:0]															
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x094	I3C_MAXWLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MWL15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A0	I3C_TIMINGR0	SCLH_I2C[7:0]								SCLL_OD[7:0]								SCLH_I3C[7:0]								SCLL_PP[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A4	I3C_TIMINGR1	Res.	Res.	Res.	SDA_HD	Res.	Res.	Res.	Res.	Res.	FREE[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	ASNCR[1:0]		AVAL[7:0]							
	Reset value				0						0	0	0	0	0	0	0							0	0	0	0	0	0	0	0	0	0
0x0A8	I3C_TIMINGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AVAL[7:0]								Res.	Res.	Res.	Res.	STALLA	STALLC	STALLD	STALLT
	Reset value																	0	0	0	0	0	0	0	0					0	0	0	0
0x0C0	I3C_BCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BCR6	Res.	Res.	Res.	BCR2	Res.	BCR0
	Reset value																										0				0		0

Table 635. I3C register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0C4	I3C_DCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCR[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x0C8	I3C_GETCAPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CAPPEND	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																		0														
0x0CC	I3C_CRCAPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CAPGR	Res.	Res.	Res.	Res.	Res.	CAPDHOFF	Res.	Res.	Res.
	Reset value																							0						0			
0x0D0	I3C_GETMXDSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSCO	RDTURN[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	FMT[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	HOFFAS [1:0]	
	Reset value								0	0	0	0	0	0	0	0	0							0	0							0	0
0x0D4	I3C_EPIDR	MIPIMID[14:0]															IDTSEL	MIPIID[3:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0												

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

65 Universal synchronous/asynchronous receiver transmitter (USART/UART)

65.1 USART introduction

The USART offers a flexible means to perform full-duplex data exchange with external equipments requiring an industry standard NRZ asynchronous serial data format. A very wide range of baud rates can be achieved through a fractional baud rate generator.

The USART supports both synchronous one-way and half-duplex single-wire communications, as well as LIN (local interconnection network), smartcard protocol, IrDA (infrared data association) SIR ENDEC specifications, and modem operations (CTS/RTS). Multiprocessor communications are also supported.

High-speed data communications are possible by using the DMA (direct memory access) for multibuffer configuration.

65.2 USART main features

- Full-duplex asynchronous communication
- NRZ standard format (mark/space)
- Configurable oversampling method by 16 or 8 to achieve the best compromise between speed and clock tolerance
- Baud rate generator systems
- Two internal FIFOs for transmit and receive data

  Each FIFO can be enabled/disabled by software and come with a status flag.
- A common programmable transmit and receive baud rate
- Dual clock domain with dedicated kernel clock for peripherals independent from PCLK
- Auto baud rate detection
- Programmable data word length (7, 8 or 9 bits)
- Programmable data order with MSB-first or LSB-first shifting
- Configurable stop bits (1 or 2 stop bits)
- Synchronous master/slave mode and clock output/input for synchronous communications
- SPI slave transmission underrun error flag
- Single-wire half-duplex communications
- Continuous communications using DMA
- Received/transmitted bytes are buffered in reserved SRAM using centralized DMA.
- Separate enable bits for transmitter and receiver
- Separate signal polarity control for transmission and reception
- Swappable Tx/Rx pin configuration
- Hardware flow control for modem and RS-485 transceiver
- Communication control/error detection flags
- Parity control:
  - Transmits parity bit

  - Checks parity of received data byte
- Interrupt sources with flags
- Multiprocessor communications: wake-up from mute mode by idle line detection or address mark detection
- Wake-up from Stop mode

65.3 USART extended features

- LIN master synchronous break send capability and LIN slave break detection capability
  - 13-bit break generation and 10/11 bit break detection when USART is hardware configured for LIN
- IrDA SIR encoder decoder supporting 3/16 bit duration for normal mode
- Smartcard mode
  - Supports the T = 0 and T = 1 asynchronous protocols for smartcards as defined in the ISO/IEC 7816-3 standard
  - 0.5 and 1.5 stop bits for smartcard operation
- Support for Modbus communication
  - Timeout feature
  - CR/LF character recognition

65.4 USART implementation

The following tables describe the USART implementation. LPUART is included for comparison.

Table 636. Instance implementation on STM32N6x5/x7xx

Instance	STM32N6xx
USART1	Full
USART2	Full
USART3	Full
USART6	Full
USART10	Full
UART4	Basic
UART5	Basic
UART7	Basic
UART8	Basic
UART9	Basic
LPUART1	Low-power

Table 637. USART/LPUART features

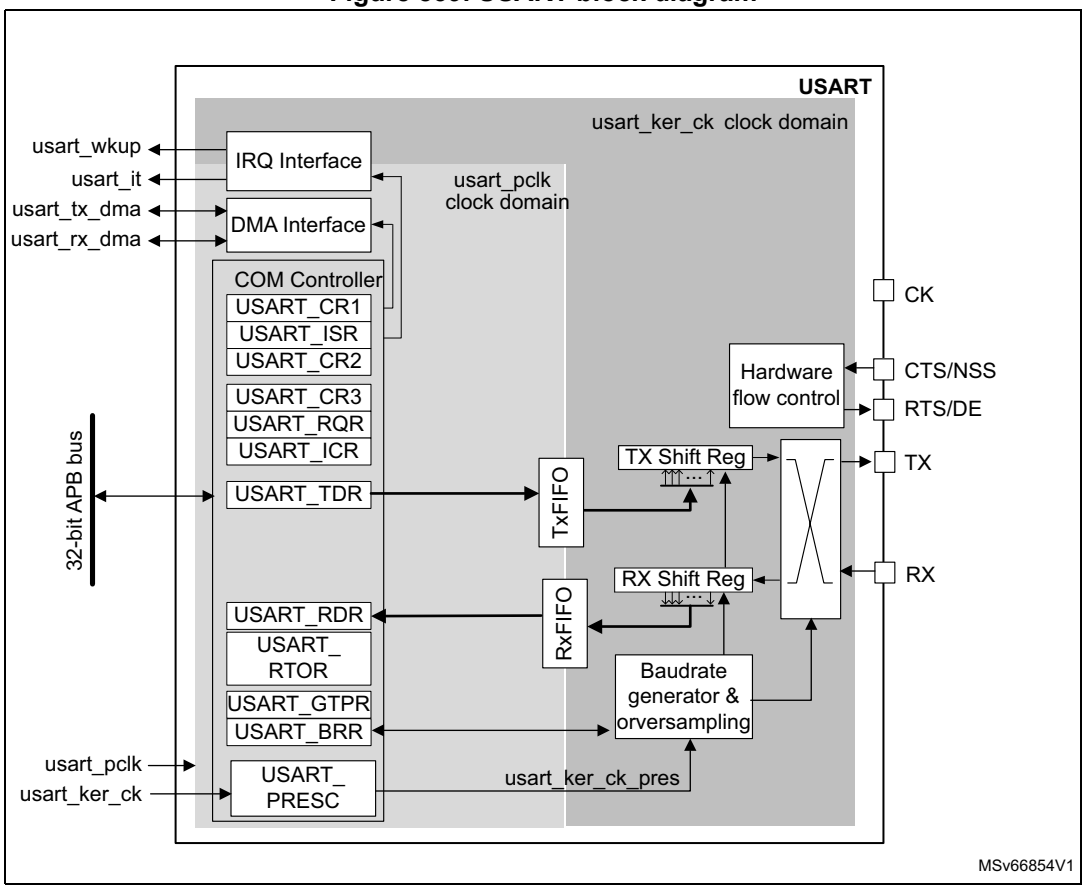
Modes/features[1]	Full feature	Basic feature	Low-power feature
Hardware flow control for modem	X	X	X
Continuous communication using DMA	X	X	X
Multiprocessor communication	X	X	X
Synchronous mode (master/slave)	X	-	-
Smartcard mode	X	-	-
Single-wire half-duplex communication	X	X	X
IrDA SIR ENDEC block	X	X	-
LIN mode	X	X	-
Dual-clock domain	X	X	X
Receiver timeout interrupt	X	X	-
Modbus communication	X	X	-
Auto baud rate detection	X	X	-
Driver enable	X	X	X
USART data length	7, 8 and 9 bits		
Tx/Rx FIFO	X	X	X
Tx/Rx FIFO size (bytes)	16		
Wake-up from low-power mode	X[2]	X[2]	X[2]

1. “X” = supported, “-” = not supported.
2. Wake-up supported from Stop mode.

65.5 USART functional description

65.5.1 USART block diagram

Figure 839. USART block diagram



65.5.2 USART pins and internal signals

Description of USART input/output pins

- USART bidirectional communications

  USART bidirectional communications require a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX):

  – **RX** (Receive Data Input)

    RX is the serial data input. Oversampling techniques are used for data recovery. They discriminate between valid incoming data and noise.

  – **TX** (Transmit Data Output)

    When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and no data needs to be transmitted, the TX pin is High. In single-wire and smartcard modes, this I/O is used to transmit and receive data.

- RS232 hardware flow control mode

  The following pins are required in RS232 hardware flow control mode:

  – **CTS** (Clear To Send)

    When driven high, this signal blocks the data transmission at the end of the current transfer.

  – **RTS** (Request To Send)

    When it is low, this signal indicates that the USART is ready to receive data.

- RS485 hardware control mode

  The **DE** (Driver Enable) pin is required in RS485 hardware control mode. This signal activates the transmission mode of the external transceiver.

- Synchronous SPI master/slave mode and smartcard mode

  The following pins are required in synchronous master/slave mode and smartcard mode:

  – **CK**

    This pin acts as clock output in synchronous SPI master and smartcard modes.

    It acts as clock input in synchronous SPI slave mode.

    In synchronous master mode, this pin outputs the transmitter data clock for synchronous transmission corresponding to SPI master mode (no clock pulses on start bit and stop bit, and a software option to send a clock pulse on the last data bit). In parallel, data can be received synchronously on RX pin. This mechanism can be used to control peripherals featuring shift registers (such as LCD drivers). The clock phase and polarity are software programmable.

    In smartcard mode, CK output provides the clock to the smartcard.

  – **NSS**

    This pin acts as Slave Select input in synchronous slave mode.

Refer to *Table 638* and *Table 639* for the list of USART input/output pins and internal signals.

Table 638. USART/UART input/output pins

Pin name	Signal type	Description
USART_RX/UART_RX	Input	Serial data receive input
USART_TX/UART_TX	Output	Transmit data output
USART_CTS/UART_CTS	Input	Clear to send
USART_RTS/UART_RTS	Output	Request to send
USART_DE(1)/UART_DE(2)	Output	Driver enable
USART_CK	Output	Clock output in synchronous master and smartcard modes.
USART_NSS(3)	Input	Slave select input in synchronous slave mode.

1. USART_DE and USART_RTS share the same pin.
2. UART_DE and UART_RTS share the same pin.
3. USART_NSS and USART_CTS share the same pin.

Description of USART input/output signals

Table 639. USART internal input/output signals

Signal name	Signal type	Description
usart_pclk	Input	APB clock
usart_ker_ck	Input	USART kernel clock
usart_wkup	Output	USART provides a wake-up interrupt
usart_it	Output	USART global interrupt
usart_tx_dma	Input/output	USART transmit DMA request
usart_rx_dma	Input/output	USART receive DMA request

65.5.3 USART clocks

The simplified block diagram given in *Section 65.5.1: USART block diagram* shows two fully-independent clock domains:

- The **usart_pclk** clock domain

  The **usart_pclk** clock signal feeds the peripheral bus interface. It must be active when accesses to the USART registers are required.
- The **usart_ker_ck** kernel clock domain.

  The **usart_ker_ck** is the USART clock source. It is independent from **usart_pclk** and delivered by the RCC. The USART registers can consequently be written/read even when the **usart_ker_ck** clock is stopped.

  When the dual clock domain feature is not supported, the **usart_ker_ck** clock is the same as the **usart_pclk** clock.

There is no constraint between **usart_pclk** and **usart_ker_ck**: **usart_ker_ck** can be faster or slower than **usart_pclk**. The only limitation is the software ability to manage the communication fast enough.

When the USART operates in SPI slave mode, it handles data flow using the serial interface clock derived from the external CK signal provided by the external master SPI device. The **usart_ker_ck** clock must be at least 3 times faster than the clock on the CK input.

65.5.4 USART character description

The word length can be set to 7, 8 or 9 bits, by programming the M bits (M0: bit 12 and M1: bit 28) in the USART_CR1 register (see *Figure 840*):

- 7-bit character length: M[1:0] = 10
- 8-bit character length: M[1:0] = 00
- 9-bit character length: M[1:0] = 01

Note: In 7-bit data length mode, the smartcard mode, LIN master mode and Auto baud rate (0x7F and 0x55 frames detection) are not supported.

By default, the signal (TX or RX) is in low state during the start bit. It is in high state during the stop bit.

These values can be inverted, separately for each signal, through polarity configuration control.

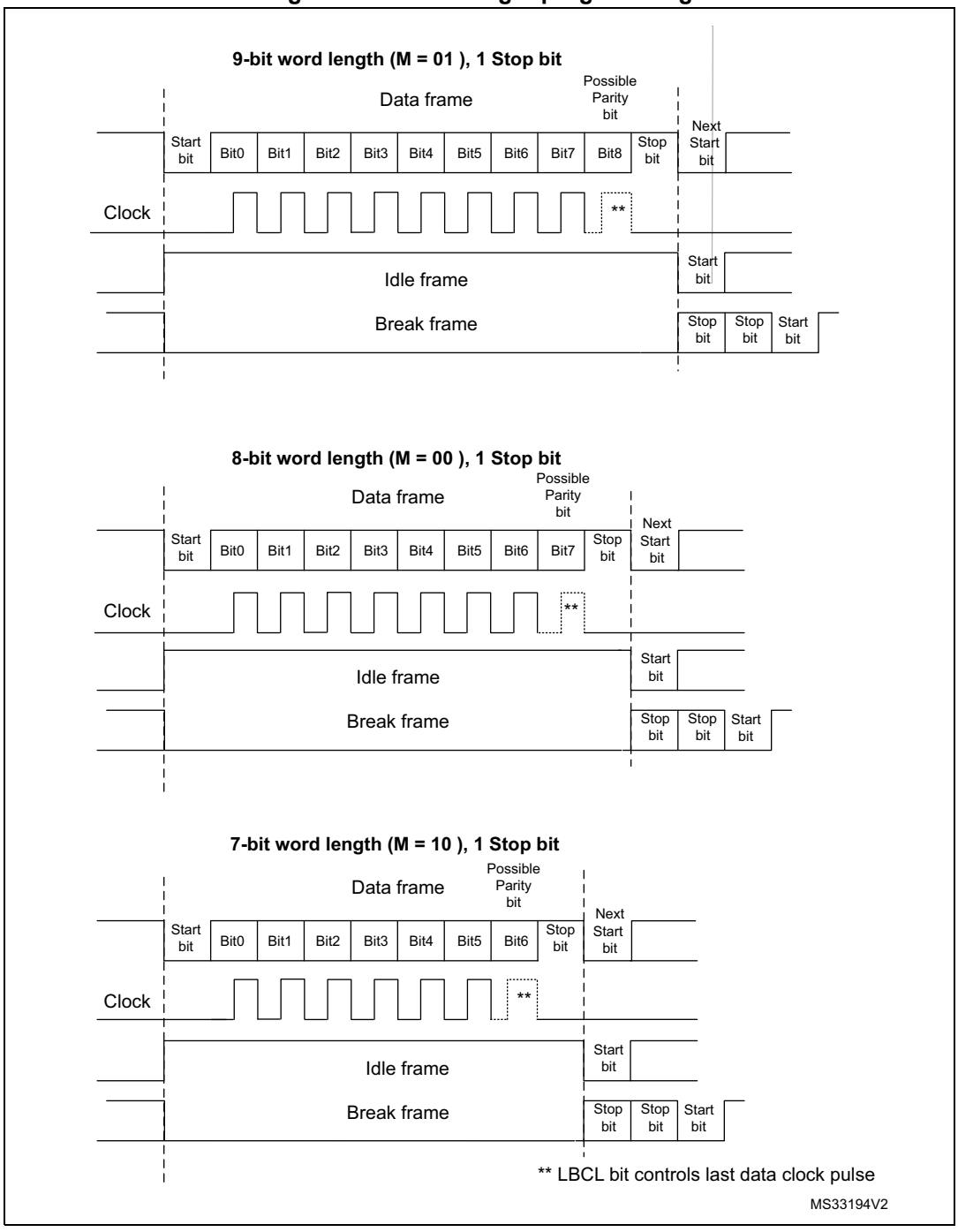
An ***Idle character*** is interpreted as an entire frame of “1”s (the number of “1”s includes the number of stop bits).

A ***Break character*** is interpreted on receiving “0”s for a frame period. At the end of the break frame, the transmitter inserts 2 stop bits.

Transmission and reception are driven by a common baud rate generator. The transmission and reception clock are generated when the enable bit is set for the transmitter and receiver, respectively.

A detailed description of each block is given below.

Figure 840. Word length programming



65.5.5 USART FIFOs and thresholds

The USART can operate in FIFO mode.

The USART comes with a Transmit FIFO (TXFIFO) and a Receive FIFO (RXFIFO). The FIFO mode is enabled by setting FIFOEN in USART_CR1 register (bit 29). This mode is supported only in UART, SPI and smartcard modes.

Since the maximum data word length is 9 bits, the TXFIFO is 9-bit wide. However the RXFIFO default width is 12 bits. This is due to the fact that the receiver does not only store the data in the FIFO, but also the error flags associated to each character (Parity error, Noise error and Framing error flags).

Note: The received data is stored in the RXFIFO together with the corresponding flags. However, only the data are read when reading the RDR.

The status flags are available in the USART_ISR register.

It is possible to configure the TXFIFO and RXFIFO levels at which the Tx and RX interrupts are triggered. These thresholds are programmed through the RXFTCFG[2:0] and TXFTCFG[2:0] bitfields of the USART_CR3 control register.

In this case:

- The Rx interrupt is generated when the number of received data in the RXFIFO reaches the threshold programmed in the RXFTCFG[2:0] bitfield.

  In this case, the RXFT flag is set in the USART_ISR register. This means that RXFTCFG[2:0] data have been received: 1 data in USART_RDR and (RXFTCFG[2:0] – 1) data in the RXFIFO. As an example, when RXFTCFG[2:0] is programmed to 101, the RXFT flag is set when a number of data corresponding to the FIFO size has been received (FIFO size – 1 data in the RXFIFO and 1 data in the USART_RDR). As a result, the next received data does not set the overrun flag.
- The Tx interrupt is generated when the number of empty locations in the TXFIFO is greater than the threshold programmed in the TXFTCFG[2:0] bitfield of the USART_CR3 register.

65.5.6 USART transmitter

The transmitter can send data words of either 7 or 8 or 9 bits, depending on the M bit status. The Transmit Enable bit (TE) must be set in order to activate the transmitter function. The data in the transmit shift register is output on the TX pin while the corresponding clock pulses are output on the CK pin.

Character transmission

During an USART transmission, data shifts out the least significant bit first (default configuration) on the TX pin. In this mode, the USART_TDR register consists of a buffer (TDR) between the internal bus and the transmit shift register.

When FIFO mode is enabled, the data written to the transmit data register (USART_TDR) are queued in the TXFIFO.

Every character is preceded by a start bit which corresponds to a low logic level for one bit period. The character is terminated by a configurable number of stop bits.

The number of stop bits can be configured to 0.5, 1, 1.5 or 2.

Note: The TE bit must be set before writing the data to be transmitted to the USART_TDR.

The TE bit must not be reset during data transmission. Resetting the TE bit during the transmission corrupts the data on the TX pin as the baud rate counters get frozen. The current data being transmitted are then lost.

An idle frame is sent when the TE bit is enabled.

Configurable stop bits

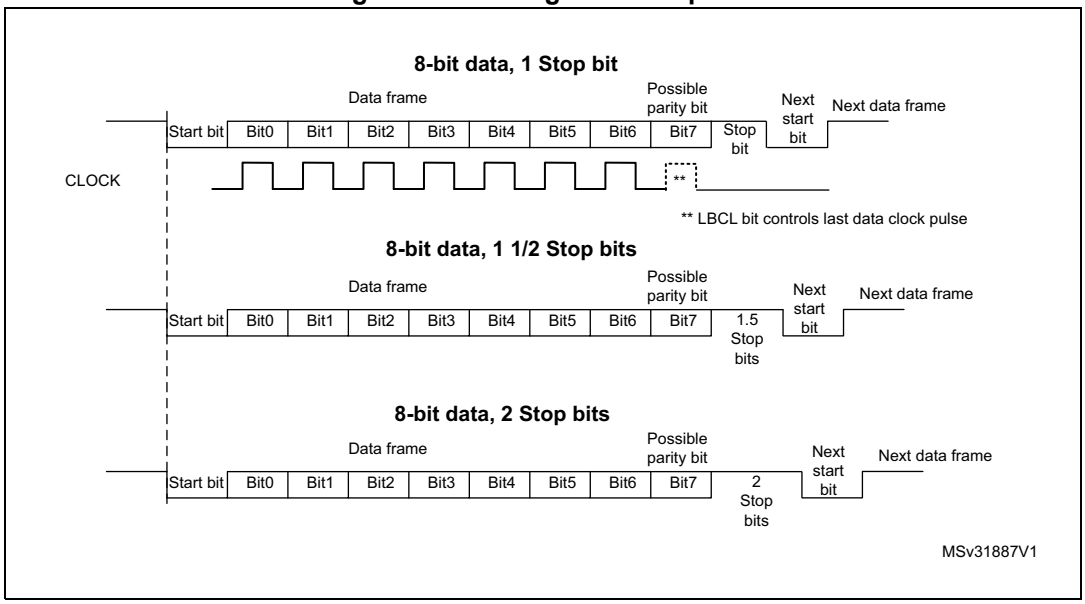
The number of stop bits to be transmitted with every character can be programmed in USART_CR2, bits 13,12.

- ***1 stop bit***: This is the default value of number of stop bits.
- ***2 stop bits***: This is supported by normal USART, single-wire and modem modes.
- ***1.5 stop bits***: To be used in smartcard mode.

An idle frame transmission includes the stop bits.

A break transmission is 10 low bits (when M[1:0] = 00) or 11 low bits (when M[1:0] = 01) or 9 low bits (when M[1:0] = 10) followed by 2 stop bits (see *Section 65.5.1: USART block diagram*). It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

Figure 841. Configurable stop bits



Character transmission procedure

To transmit a character, follow the sequence below:

1. Program the M bits in USART_CR1 to define the word length.
2. Select the desired baud rate using the USART_BRR register.
3. Program the number of stop bits in USART_CR2.
4. Enable the USART by writing the UE bit in USART_CR1 register to 1.
5. Select DMA enable (DMAT) in USART_CR3 if multibuffer communication must take place. Configure the DMA register as explained in *Section 65.5.20: Continuous communication using USART and DMA*.
6. Set the TE bit in USART_CR1 to send an idle frame as first transmission.

7. Write the data to send in the USART_TDR register. Repeat this for each data to be transmitted in case of single buffer.
   – When FIFO mode is disabled, writing a data to the USART_TDR clears the TXE flag.
   – When FIFO mode is enabled, writing a data to the USART_TDR adds one data to the TXFIFO. Write operations to the USART_TDR are performed when TXFNF flag is set. This flag remains set until the TXFIFO is full.
8. When the last data is written to the USART_TDR register, wait until TC = 1.
   – When FIFO mode is disabled, this indicates that the transmission of the last frame has completed.
   – When FIFO mode is enabled, this indicates that both TXFIFO and shift register are empty.

   This check is required to avoid corrupting the last transmission when the USART is disabled or enters Halt mode.

Single byte communication

- When FIFO mode is disabled

  Writing to the transmit data register always clears the TXE bit. The TXE flag is set by hardware. It indicates that:
  – the data have been moved from the USART_TDR register to the shift register and the data transmission has started;
  – the USART_TDR register is empty;
  – the next data can be written to the USART_TDR register without overwriting the previous data.

  This flag generates an interrupt if the TXEIE bit is set.

  When a transmission is ongoing, a write instruction to the USART_TDR register stores the data in the TDR buffer. It is then copied in the shift register at the end of the current transmission.

  When no transmission is ongoing, a write instruction to the USART_TDR register places the data in the shift register, the data transmission starts, and the TXE bit is set.
- When FIFO mode is enabled, the TXFNF (TXFIFO not full) flag is set by hardware to indicate that:
  – the TXFIFO is not full;
  – the USART_TDR register is empty;
  – the next data can be written to the USART_TDR register without overwriting the previous data. When a transmission is ongoing, a write operation to the USART_TDR register stores the data in the TXFIFO. Data are copied from the TXFIFO to the shift register at the end of the current transmission.

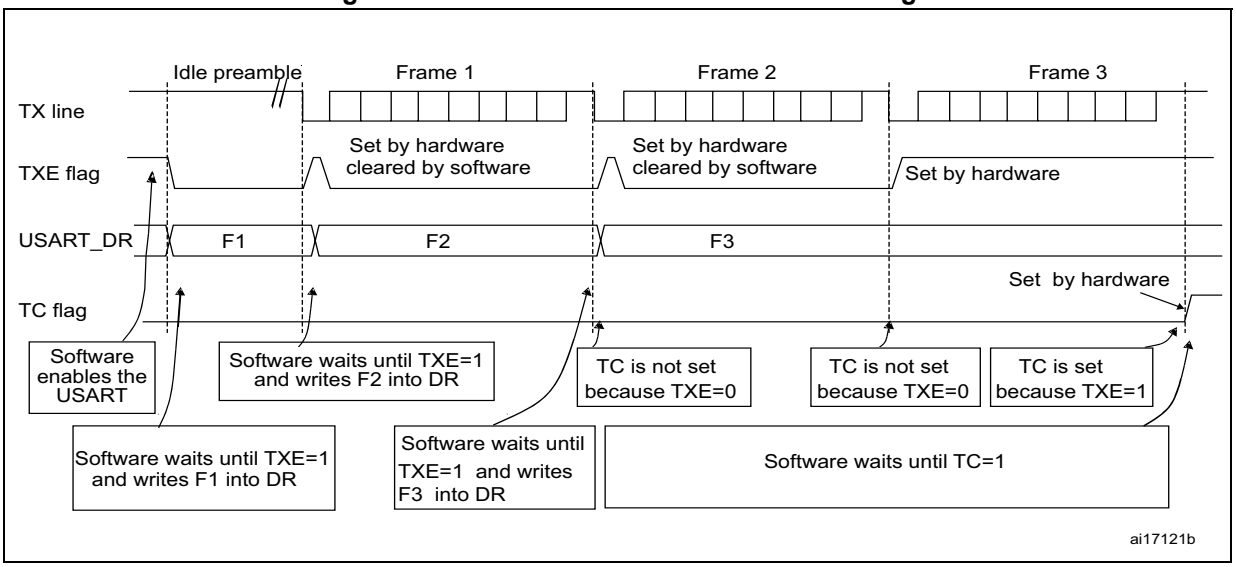
  When the TXFIFO is not full, the TXFNF flag stays at 1 even after a write operation to USART_TDR register. It is cleared when the TXFIFO is full. This flag generates an interrupt if the TXFNFIE bit is set.

  Alternatively, interrupts can be generated and data can be written to the FIFO when the TXFIFO threshold is reached. In this case, the CPU can write a block of data defined by the programmed trigger level.

  If a frame is transmitted (after the stop bit) and the TXE flag (TXFE in case of FIFO mode) is set, the TC flag goes high. An interrupt is generated if the TCIE bit is set in the USART_CR1 register.

After writing the last data to the USART_TDR register, it is mandatory to wait until TC is set before disabling the USART or causing the microcontroller to enter the low-power mode (see *Figure 842*).

Figure 842. TC/TXE behavior when transmitting



Note: When FIFO management is enabled, the TXFNF flag is used for data transmission.

Break characters

Setting the SBKRQ bit transmits a break character. The break frame length depends on the M bit (see *Figure 840*).

If a 1 is written to the SBKRQ bit, a break character is sent on the TX line after completing the current character transmission. The SBKF bit is set by the write operation and it is reset by hardware when the break character is complete (during the stop bits after the break character). The USART inserts a logic 1 signal (stop) for the duration of 2 bits at the end of the break frame to guarantee the recognition of the start bit of the next frame.

When the SBKRQ bit is set, the break character is sent at the end of the current transmission.

When FIFO mode is enabled, sending the break character has priority on sending data even if the TXFIFO is full.

Idle characters

Setting the TE bit drives the USART to send an idle frame before the first data frame.

65.5.7 USART receiver

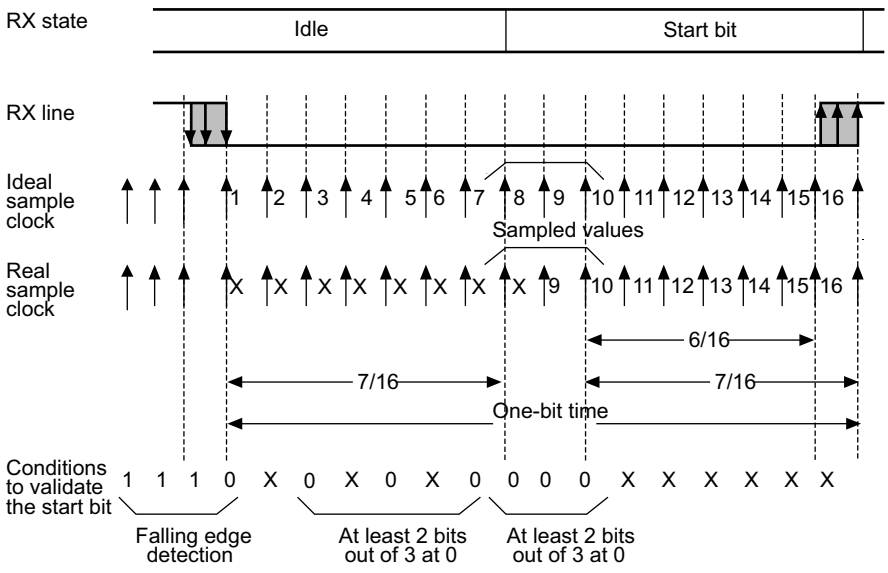
The USART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the USART_CR1 register.

Start bit detection

The start bit detection sequence is the same when oversampling by 16 or by 8.

In the USART, the start bit is detected when a specific sequence of samples is recognized. This sequence is: 1 1 1 0 X 0 X 0X 0X 0 X 0X 0.

Figure 843. Start bit detection when oversampling by 16 or 8



Note: If the sequence has not completed, the start bit detection aborts and the receiver returns to the idle state (no flag is set), where it waits for a falling edge.

The start bit is confirmed (RXNE flag set and interrupt generated if RXNEIE = 1, or RXFNE flag set and interrupt generated if RXFNEIE = 1 if FIFO mode enabled) if the 3 sampled bits are at 0 (first sampling on the 3rd, 5th and 7th bits finds the 3 bits at 0 and second sampling on the 8th, 9th and 10th bits also finds the 3 bits at 0).

The start bit is validated but the NE noise flag is set if,

- for both samplings, 2 out of the 3 sampled bits are at 0 (sampling on the 3rd, 5th and 7th bits and sampling on the 8th, 9th and 10th bits), or
- for one of the samplings (sampling on the 3rd, 5th and 7th bits or sampling on the 8th, 9th and 10th bits), 2 out of the 3 bits are found at 0.

If neither of the above conditions are met, the start detection aborts and the receiver returns to the idle state (no flag is set).

Character reception

During an USART reception, data are shifted out least significant bit first (default configuration) through the RX pin.

Character reception procedure

To receive a character, follow the sequence below:

1. Program the M bits in USART_CR1 to define the word length.
2. Select the desired baud rate using the baud rate register USART_BRR
3. Program the number of stop bits in USART_CR2.
4. Enable the USART by writing the UE bit in USART_CR1 register to 1.
5. Select DMA enable (DMAR) in USART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in *Section 65.5.20: Continuous communication using USART and DMA*.
6. Set the RE bit USART_CR1. This enables the receiver which begins searching for a start bit.

When a character is received:

- When FIFO mode is disabled, the RXNE bit is set to indicate that the content of the shift register is transferred to the RDR. In other words, data have been received and can be read (as well as their associated error flags).
- When FIFO mode is enabled, the RXFNE bit is set to indicate that the RXFIFO is not empty. Reading the USART_RDR returns the oldest data entered in the RXFIFO. When a data is received, it is stored in the RXFIFO together with the corresponding error bits.
- An interrupt is generated if the RXNEIE (RXFNEIE when FIFO mode is enabled) bit is set.
- The error flags can be set if a frame error, noise, parity or an overrun error was detected during reception.
- In multibuffer communication mode:
  - When FIFO mode is disabled, the RXNE flag is set after every byte reception. It is cleared when the DMA reads the Receive data Register.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every DMA request, a data is retrieved from the RXFIFO. A DMA request is triggered when the RXFIFO is not empty, that is when there are data to be read from the RXFIFO.
- In single-buffer mode:
  - When FIFO mode is disabled, clearing the RXNE flag is done by performing a software read from the USART_RDR register. The RXNE flag can also be cleared by programming RXFRQ bit to 1 in the USART_RQR register. The RXNE flag must be cleared before the end of the reception of the next character to avoid an overrun error.
  - When FIFO mode is enabled, the RXFNE is set when the RXFIFO is not empty. After every read operation from USART_RDR, a data is retrieved from the RXFIFO. When the RXFIFO is empty, the RXFNE flag is cleared. The RXFNE flag can also be cleared by programming RXFRQ bit to 1 in USART_RQR. When the RXFIFO is full, the first entry in the RXFIFO must be read before the end of the reception of the next character, to avoid an overrun error. The RXFNE flag generates an interrupt if the RXFNEIE bit is set. Alternatively, interrupts can be

generated and data can be read from RXFIFO when the RXFIFO threshold is reached. In this case, the CPU can read a block of data defined by the programmed threshold.

Break character

When a break character is received, the USART handles it as a framing error.

Idle character

When an idle frame is detected, it is handled in the same way as a data character reception except that an interrupt is generated if the IDLEIE bit is set.

Overrun error

- FIFO mode disabled

  An overrun error occurs if a character is received and RXNE has not been reset.

  Data can not be transferred from the shift register to the RDR register until the RXNE bit is cleared. The RXN E flag is set after every byte reception.

  An overrun error occurs if RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:

  – the ORE bit is set;
  – the RDR content is not lost. The previous data is available by reading the USART_RDR register.
  – the shift register is overwritten. After that, any data received during overrun is lost.
  – an interrupt is generated if either the RXNEIE or the EIE bit is set.

- FIFO mode enabled

  An overrun error occurs when the shift register is ready to be transferred and the receive FIFO is full.

  Data can not be transferred from the shift register to the USART_RDR register until there is one free location in the RXFIFO. The RXFNE flag is set when the RXFIFO is not empty.

  An overrun error occurs if the RXFIFO is full and the shift register is ready to be transferred. When an overrun error occurs:

  – The ORE bit is set.
  – The first entry in the RXFIFO is not lost. It is available by reading the USART_RDR register.
  – The shift register is overwritten. After that point, any data received during overrun is lost.
  – An interrupt is generated if either the RXFNEIE or EIE bit is set.

The ORE bit is reset by setting the ORECF bit in the USART_ICR register.

Note: The ORE bit, when set, indicates that at least 1 data has been lost.

When the FIFO mode is disabled, there are two possibilities

- *if RXNE = 1, then the last valid data is stored in the receive register (RDR) and can be read,*
- *if RXNE = 0, the last valid data has already been read and there is nothing left to be read in the RDR register. This case can occur when the last valid data is read in the RDR register at the same time as the new (and lost) data is received.*

Selecting the clock source and the appropriate oversampling method

The choice of the clock source is done through the Clock Control system (see *Section : Reset and Clock Control (RCC)*). The clock source must be selected through the UE bit before enabling the USART.

The clock source must be selected according to two criteria:

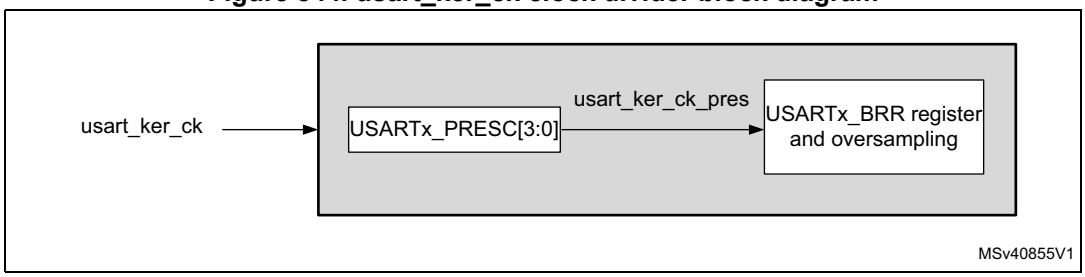
- Possible use of the USART in low-power mode
- Communication speed.

The clock source frequency is usart_ker_ck.

When the dual clock domain and the wake-up from low-power mode features are supported, the usart_ker_ck clock source can be configurable in the RCC (see *Section : Reset and Clock Control (RCC)*). Otherwise the usart_ker_ck clock is the same as usart_pclk.

The usart_ker_ck clock can be divided by a programmable factor, defined in the USART_PRESC register.

Figure 844. usart_ker_ck clock divider block diagram



Some usart_ker_ck sources enable the USART to receive data while the MCU is in low-power mode. Depending on the received data and wake-up mode selected, the USART wakes up the MCU, when needed, in order to transfer the received data, by performing a software read to the USART_RDR register or by DMA.

For the other clock sources, the system must be active to enable USART communications.

The communication speed range (specially the maximum communication speed) is also determined by the clock source.

The receiver implements different user-configurable oversampling techniques (except in synchronous mode) for data recovery by discriminating between valid incoming data and noise. This enables obtaining the best a trade-off between the maximum communication speed and noise/clock inaccuracy immunity.

The oversampling method can be selected by programming the OVER8 bit in the USART_CR1 register either to 16 or 8 times the baud rate clock (see *Figure 845* and *Figure 846*).

Depending on the application:

- select oversampling by 8 (OVER8 = 1) to achieve higher speed (up to usart_ker_ck_pres/8). In this case the maximum receiver tolerance to clock deviation is reduced (refer to *Section 65.5.9: Tolerance of the USART receiver to clock deviation*)
- select oversampling by 16 (OVER8 = 0) to increase the tolerance of the receiver to clock deviations. In this case, the maximum speed is limited to maximum usart_ker_ck_pres/16 (where usart_ker_ck_pres is the USART input clock divided by a prescaler).

Programming the ONEBIT bit in the USART_CR3 register selects the method used to evaluate the logic level. Two options are available:

- The majority vote of the three samples in the center of the received bit. In this case, when the 3 samples used for the majority vote are not equal, the NE bit is set.
- A single sample in the center of the received bit

  Depending on the application:

  – select the three sample majority vote method (ONEBIT = 0) when operating in a noisy environment and reject the data when a noise is detected (refer to *Table 640*) because this indicates that a glitch occurred during the sampling.

  – select the single sample method (ONEBIT = 1) when the line is noise-free to increase the receiver tolerance to clock deviations (see *Section 65.5.9: Tolerance of the USART receiver to clock deviation*). In this case the NE bit is never set.

When noise is detected in a frame:

- The NE bit is set at the rising edge of the RXNE bit (RXFNE in case of FIFO mode enabled).
- The invalid data is transferred from the Shift register to the USART_RDR register.
- No interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit (RXFNE in case of FIFO mode enabled) which itself generates an interrupt. In case of multibuffer communication an interrupt is issued if the EIE bit is set in the USART_CR3 register.

The NE bit is reset by setting NFCF bit in ICR register.

Note: Noise error is not supported in SPI and IrDA modes.

Oversampling by 8 is not available in the smartcard, IrDA and LIN modes. In those modes, the OVER8 bit is forced to 0 by hardware.

Figure 845. Data sampling when oversampling by 16

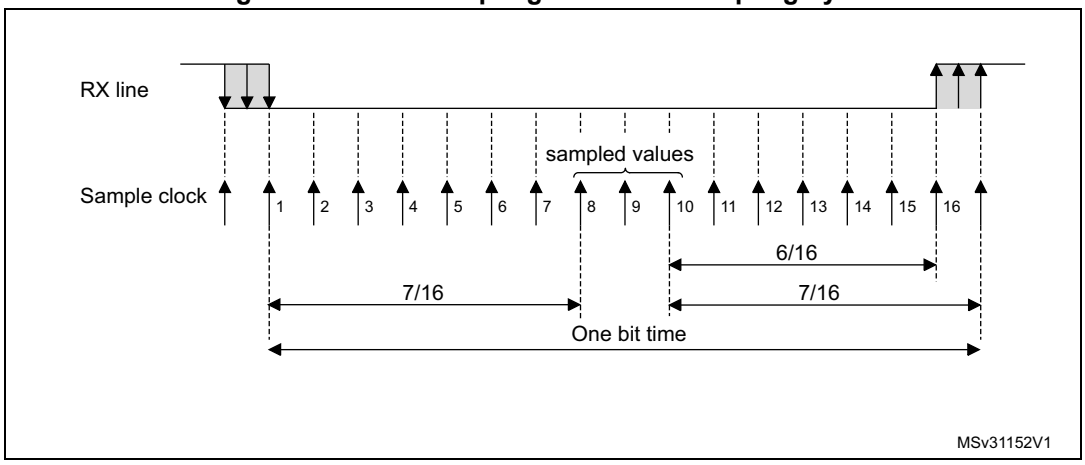


Figure 846. Data sampling when oversampling by 8

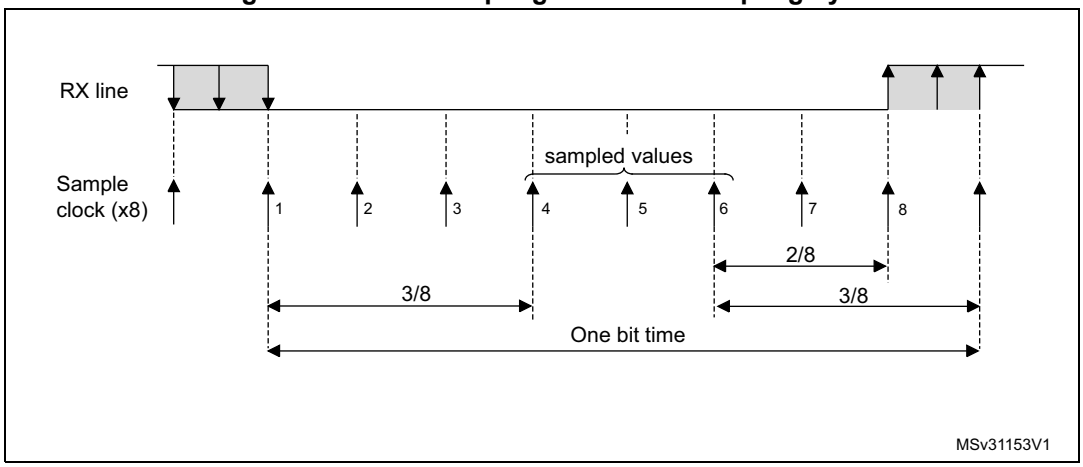


Table 640. Noise detection from sampled data

Sampled value	NE status	Received bit value
000	0	0
001	1	0
010	1	0
011	1	1
100	1	0
101	1	1
110	1	1
111	0	1

Framing error

A framing error is detected when the stop bit is not recognized on reception at the expected time, following either a de-synchronization or excessive noise.

When the framing error is detected:

- the FE bit is set by hardware.
- the invalid data is transferred from the Shift register to the USART_RDR register (RXFIFO in case FIFO mode is enabled).
- no interrupt is generated in case of single byte communication. However this bit rises at the same time as the RXNE bit (RXFNE in case FIFO mode is enabled) which itself generates an interrupt. In case of multibuffer communication an interrupt is issued if the EIE bit is set in the USART_CR3 register.

The FE bit is reset by writing 1 to the FECF in the USART_ICR register.

Note: Framing error is not supported in SPI mode.

Configurable stop bits during reception

The number of stop bits to be received can be configured through the control bits of USART_CR: it can be either 1 or 2 in normal mode and 0.5 or 1.5 in smartcard mode.

- **0.5 stop bit (reception in smartcard mode)**: no sampling is done for 0.5 stop bit. As a consequence, no framing error and no break frame can be detected when 0.5 stop bit is selected.
- **1 stop bit**: sampling for 1 stop bit is done on the 8th, 9th and 10th samples.
- **1.5 stop bits (smartcard mode)**

  When transmitting in smartcard mode, the device must check that the data are correctly sent. The receiver block must consequently be enabled (RE = 1 in USART_CR1) and the stop bit is checked to test if the smartcard has detected a parity error.

  In the event of a parity error, the smartcard forces the data signal low during the sampling (NACK signal), which is flagged as a framing error. The FE flag is then set through RXNE flag (RXFNE if the FIFO mode is enabled) at the end of the 1.5 stop bit. Sampling for 1.5 stop bits is done on the 16th, 17th and 18th samples (1 baud clock period after the beginning of the stop bit). The 1.5 stop bit can be broken into 2 parts: one 0.5 baud clock period during which nothing happens, followed by 1 normal stop bit period during which sampling occurs halfway through (refer to *Section 65.5.17: USART receiver timeout* for more details).
- **2 stop bits**

  Sampling for 2 stop bits is done on the 8th, 9th and 10th samples of the first stop bit.

  The framing error flag is set if a framing error is detected during the first stop bit.

  The second stop bit is not checked for framing error. The RXNE flag (RXFNE if the FIFO mode is enabled) is set at the end of the first stop bit.

65.5.8 USART baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the value programmed in the USART_BRR register.

Equation 1: baud rate for standard USART (SPI mode included) (OVER8 = 0 or 1)

In case of oversampling by 16, the baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{\text{usart_ker_ck_pres}}{\text{USARTDIV}}$$

In case of oversampling by 8, the baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{2 \times \text{usart_ker_ck_pres}}{\text{USARTDIV}}$$

Equation 2: baud rate in smartcard, LIN and IrDA modes (OVER8 = 0)

The baud rate is given by the following formula:

$$\text{Tx/Rx baud} = \frac{\text{usart_ker_ck_pres}}{\text{USARTDIV}}$$

USARTDIV is an unsigned fixed point number that is coded on the USART_BRR register.

- When OVER8 = 0, BRR = USARTDIV.
- When OVER8 = 1
  - BRR[2:0] = USARTDIV[3:0] shifted 1 bit to the right.
  - BRR[3] must be kept cleared.
  - BRR[15:4] = USARTDIV[15:4]

Note: The baud counters are updated to the new value in the baud registers after a write operation to USART_BRR. Hence the baud rate register value must not be changed during communication.

In case of oversampling by 16 and 8, USARTDIV must be greater than or equal to 16.

How to derive USARTDIV from USART_BRR register values

Example 1

To obtain 9600 bauds with usart_ker_ck_pres= 8 MHz:

- In case of oversampling by 16:
  USARTDIV = 8 000 000/9600
  BRR[3:0] = USARTDIV = 0d833 = 0x0341
- In case of oversampling by 8:
  USARTDIV = 2 * 8 000 000/9600
  USARTDIV = 1666,66 (0d1667 = 0x683)
  BRR[3:0] = 0x3 >>1 = 0x1
  BRR[3:0] = 0x681

Example 2

To obtain 921.6 kbauds with usart_ker_ck_pres = 48 MHz:

- In case of oversampling by 16:
  USARTDIV = 48 000 000/921 600
  BRR[3:0] = USARTDIV = 52 = 0x34
- In case of oversampling by 8:
  USARTDIV = 2 * 48 000 000/921 600
  USARTDIV = 104 (0d104 = 0x68)
  BRR[3:0] = USARTDIV[3:0] >> 1 = 0x8 >> 1 = 0x4
  BRR[3:0] = 0x64

65.5.9 Tolerance of the USART receiver to clock deviation

The USART asynchronous receiver operates correctly only if the total clock system deviation is less than the tolerance of the USART receiver.

The causes which contribute to the total deviation are:

- DTRA: deviation due to the transmitter error (which also includes the deviation of the transmitter's local oscillator)
- DQUANT: error due to the baud rate quantization of the receiver
- DREC: deviation of the receiver local oscillator
- DTCL: deviation due to the transmission line (generally due to the transceivers which can introduce an asymmetry between the low-to-high transition timing and the high-to-low transition timing)

$$DTRA + DQUANT + DREC + DTCL + DWU < USART\ receiver\ tolerance$$

where

DWU is the error due to sampling point deviation when the wake-up from low-power mode is used.

when M[1:0] = 01:

$$DWU = \frac{t_{WUUSART}}{11 \times Tbit}$$

when M[1:0] = 00:

$$DWU = \frac{t_{WUUSART}}{10 \times Tbit}$$

when M[1:0] = 10:

$$DWU = \frac{t_{WUUSART}}{9 \times Tbit}$$

$t_{WUUSART}$ is the time between the detection of the start bit falling edge and the instant when the clock (requested by the peripheral) is ready and reaching the peripheral, and the regulator is ready.

The USART receiver can receive data correctly at up to the maximum tolerated deviation specified in *Table 641*, *Table 642*, depending on the following settings:

- 9-, 10- or 11-bit character length defined by the M bits in the USART_CR1 register
- Oversampling by 8 or 16 defined by the OVER8 bit in the USART_CR1 register
- Bits BRR[3:0] of USART_BRR register are equal to or different from 0000.
- Use of 1 bit or 3 bits to sample the data, depending on the value of the ONEBIT bit in the USART_CR3 register.

Table 641. Tolerance of the USART receiver when BRR [3:0] = 0000

M bits	OVER8 bit = 0		OVER8 bit = 1	
	ONEBIT = 0	ONEBIT = 1	ONEBIT = 0	ONEBIT = 1
00	3.75%	4.375%	2.50%	3.75%
01	3.41%	3.97%	2.27%	3.41%
10	4.16%	4.86%	2.77%	4.16%

Table 642. Tolerance of the USART receiver when BRR[3:0] is different from 0000

M bits	OVER8 bit = 0		OVER8 bit = 1	
	ONEBIT = 0	ONEBIT = 1	ONEBIT = 0	ONEBIT = 1
00	3.33%	3.88%	2%	3%
01	3.03%	3.53%	1.82%	2.73%
10	3.7%	4.31%	2.22%	3.33%

Note: The data specified in Table 641 and Table 642 may slightly differ in the special case when the received frames contain some Idle frames of exactly 10-bit times when M bits = 00 (11-bit times when M= 01 or 9- bit times when M = 10).

65.5.10 USART auto baud rate detection

The USART can detect and automatically set the USART_BRR register value based on the reception of one character. Automatic baud rate detection is useful under two circumstances:

- The communication speed of the system is not known in advance.
- The system is using a relatively low accuracy clock source and this mechanism enables the correct baud rate to be obtained without measuring the clock deviation.

The clock source frequency must be compatible with the expected communication speed.

- When oversampling by 16, the baud rate ranges from usart_ker_ck_pres/65535 and usart_ker_ck_pres/16.
- When oversampling by 8, the baud rate ranges from usart_ker_ck_pres/32763 and usart_ker_ck_pres/8.

Before activating the auto baud rate detection, the auto baud rate detection mode must be selected through the ABRMOD[1:0] field in the USART_CR2 register. There are four modes based on different character patterns. In these auto baud rate modes, the baud rate is measured several times during the synchronization data reception and each measurement is compared to the previous one.

These modes are the following:

- **Mode 0**: Any character starting with a bit at 1.

  In this case the USART measures the duration of the start bit (falling edge to rising edge).
- **Mode 1:** Any character starting with a 10xx bit pattern.

  In this case, the USART measures the duration of the Start and of the 1st data bit. The measurement is done falling edge to falling edge, to ensure a better accuracy in the case of slow signal slopes.
- **Mode 2**: A 0x7F character frame (it may be a 0x7F character in LSB first mode or a 0xFE in MSB first mode).

  In this case, the baud rate is updated first at the end of the start bit (BRs), then at the end of bit 6 (based on the measurement done from falling edge to falling edge: BR6). Bit0 to bit6 are sampled at BRs while further bits of the character are sampled at BR6.
- **Mode 3**: A 0x55 character frame.

  In this case, the baud rate is updated first at the end of the start bit (BRs), then at the end of bit0 (based on the measurement done from falling edge to falling edge: BR0), and finally at the end of bit6 (BR6). Bit 0 is sampled at BRs, bit 1 to bit 6 are sampled at BR0, and further bits of the character are sampled at BR6. In parallel, another check is performed for each intermediate RX line transition. An error is generated if the transitions on RX are not sufficiently synchronized with the receiver (the receiver being based on the baud rate calculated on bit 0).

Prior to activating the auto baud rate detection, the USART_BRR register must be initialized by writing a non-zero baud rate value.

The automatic baud rate detection is activated by setting the ABREN bit in the USART_CR2 register. The USART then waits for the first character on the RX line. The auto baud rate operation completion is indicated by the setting of the ABRF flag in the USART_ISR register. If the line is noisy, the correct baud rate detection cannot be guaranteed. In this case the BRR value may be corrupted and the ABRE error flag is set. This also happens if the communication speed is not compatible with the automatic baud rate detection range (bit duration not between 16 and 65536 clock periods (oversampling by 16) and not between 8 and 65536 clock periods (oversampling by 8)).

The auto baud rate detection can be re-launched later by resetting the ABRF flag (by writing a 0).

When FIFO management is disabled and an auto baud rate error occurs, the ABRE flag is set through RXNE and FE bits.

When FIFO management is enabled and an auto baud rate error occurs, the ABRE flag is set through RXFNE and FE bits.

If the FIFO mode is enabled, the auto baud rate detection must be made using the data on the first RXFIFO location. So, prior to launching the auto baud rate detection, make sure that the RXFIFO is empty by checking the RXFNE flag in USART_ISR register.

Note: The BRR value might be corrupted if the USART is disabled (UE = 0) during an auto baud rate operation.

65.5.11 USART multiprocessor communication

It is possible to perform USART multiprocessor communications (with several USARTs connected in a network). For instance one of the USARTs can be the master with its TX output connected to the RX inputs of the other USARTs, while the others are slaves with their respective TX outputs logically ANDed together and connected to the RX input of the master.

In multiprocessor configurations, it is often desirable that only the intended message recipient actively receives the full message contents, thus reducing redundant USART service overhead for all non addressed receivers.

The non-addressed devices can be placed in mute mode by means of the muting function. To use the mute mode feature, the MME bit must be set in the USART_CR1 register.

Note: When FIFO management is enabled and MME is already set, MME bit must not be cleared and then set again quickly (within two usart_ker_ck cycles), otherwise mute mode might remain active.

When the mute mode is enabled:

- none of the reception status bits can be set;
- all the receive interrupts are inhibited;
- the RWU bit in USART_ISR register is set to 1. RWU can be controlled automatically by hardware or by software, through the MMRQ bit in the USART_RQR register, under certain conditions.

The USART can enter or exit from mute mode using one of two methods, depending on the WAKE bit in the USART_CR1 register:

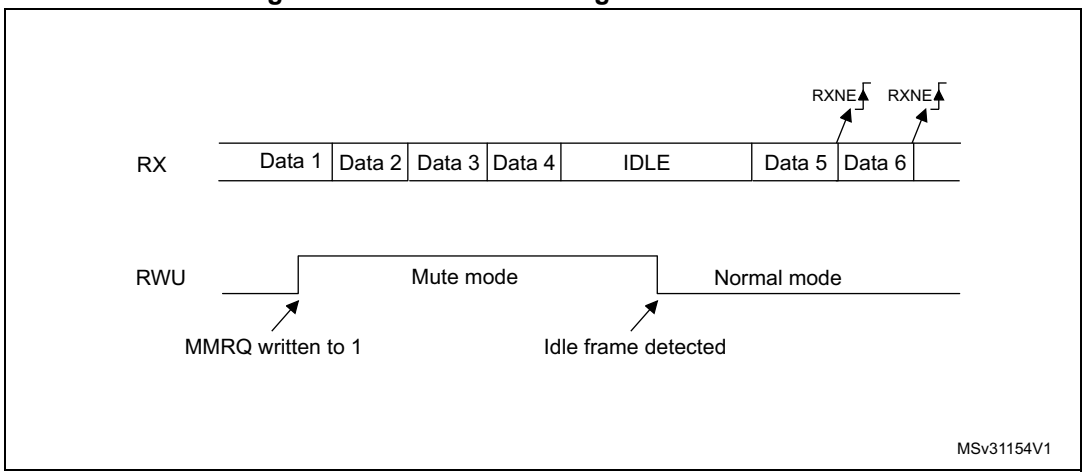
- Idle line detection if the WAKE bit is reset,
- Address mark detection if the WAKE bit is set.

Idle line detection (WAKE = 0)

The USART enters mute mode when the MMRQ bit is written to 1 and the RWU is automatically set.

The USART wakes up when an Idle frame is detected. The RWU bit is then cleared by hardware but the IDLE bit is not set in the USART_ISR register. An example of mute mode behavior using Idle line detection is given in *Figure 847*.

Figure 847. Mute mode using Idle line detection



Note: If the MMRQ is set while the IDLE character has already elapsed, mute mode is not entered (RWU is not set).

If the USART is activated while the line is idle, the idle state is detected after the duration of one IDLE frame (not only after the reception of one character frame).

4-bit/7-bit address mark detection (WAKE = 1)

In this mode, bytes are recognized as addresses if their MSB is a 1, otherwise they are considered as data. In an address byte, the address of the targeted receiver is put in the 4 or 7 LSBs. The choice of 7 or 4 bit address detection is done using the ADDM7 bit. This 4-bit/7-bit word is compared by the receiver with its own address which is programmed in the ADD bits in the USART_CR2 register.

Note: In 7-bit and 9-bit data modes, address detection is done on 6-bit and 8-bit addresses (ADD[5:0] and ADD[7:0]) respectively.

The USART enters mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is not set for this address byte and no interrupt or DMA request is issued when the USART enters mute mode. When FIFO management is enabled, the software must ensure that there is at least one empty location in the RXFIFO before entering mute mode.

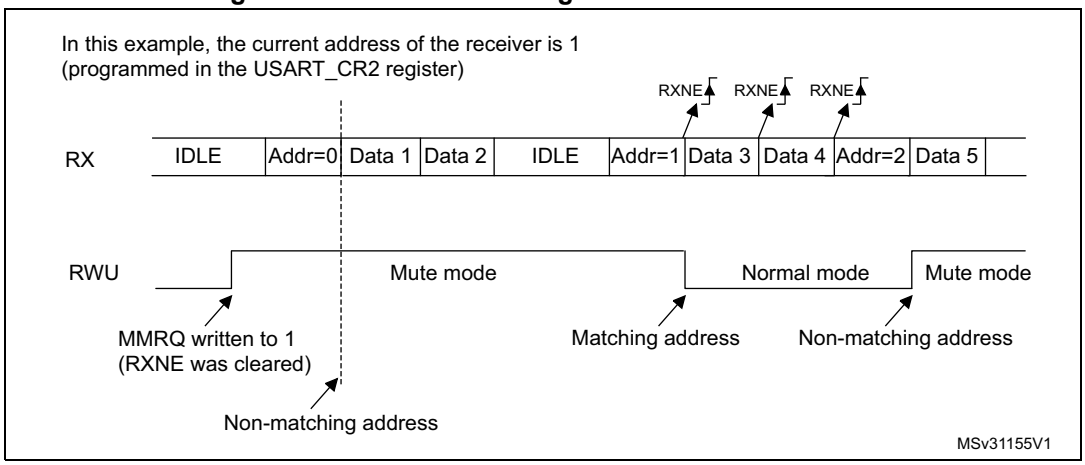
The USART also enters mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

The USART exits from mute mode when an address character is received which matches the programmed address. Then the RWU bit is cleared and subsequent bytes are received normally. The RXNE/RXFNE bit is set for the address character since the RWU bit has been cleared.

Note: When FIFO management is enabled, when MMRQ is set while the receiver is sampling last bit of a data, this data may be received before effectively entering in mute mode

An example of mute mode behavior using address mark detection is given in *Figure 848*.

Figure 848. Mute mode using address mark detection



65.5.12 USART Modbus communication

The USART offers basic support for the implementation of Modbus/RTU and Modbus/ASCII protocols. Modbus/RTU is a half-duplex, block-transfer protocol. The control part of the protocol (address recognition, block integrity control and command interpretation) must be implemented in software.

The USART offers basic support for the end of the block detection, without software overhead or other resources.

Modbus/RTU

In this mode, the end of one block is recognized by a “silence” (idle line) for more than 2 character times. This function is implemented through the programmable timeout function.

The timeout function and interrupt must be activated, through the RTOEN bit in the USART_CR2 register and the RTOIE in the USART_CR1 register. The value corresponding to a timeout of 2 character times (for example 22 x bit time) must be programmed in the RTO register. When the receive line is idle for this duration, after the last stop bit is received, an interrupt is generated, informing the software that the current block reception has not completed.

Modbus/ASCII

In this mode, the end of a block is recognized by a specific (CR/LF) character sequence. The USART manages this mechanism using the character match function.

By programming the LF ASCII code in the ADD[7:0] field and by activating the character match interrupt (CMIE = 1), the software is informed when a LF has been received and can check the CR/LF in the DMA buffer.

65.5.13 USART parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the USART_CR1 register. Depending on the frame length defined by the M bits, the possible USART frame formats are as listed in *Table 643*.

Table 643. USART frame formats

M bits	PCE bit	USART frame(1)
00	0	| SB | 8 bit data | STB |
00	1	| SB | 7-bit data | PB | STB |
01	0	| SB | 9-bit data | STB |
01	1	| SB | 8-bit data PB | STB |
10	0	| SB | 7bit data | STB |
10	1	| SB | 6-bit data | PB | STB |

1. Legends: SB: start bit, STB: stop bit, PB: parity bit. In the data register, the PB is always taking the MSB position (8th or 7th, depending on the M bit value).

Even parity

The parity bit is calculated to obtain an even number of “1s” inside the frame of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data = 00110101, and 4 bits are set, then the parity bit is equal to 0 if even parity is selected (PS bit in USART_CR1 = 0).

Odd parity

The parity bit is calculated to obtain an odd number of “1s” inside the frame made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data = 00110101 and 4 bits set, then the parity bit is equal to 1 if odd parity is selected (PS bit in USART_CR1 = 1).

Parity checking in reception

If the parity check fails, the PE flag is set in the USART_ISR register and an interrupt is generated if PEIE is set in the USART_CR1 register. The PE flag is cleared by software writing 1 to the PECF in the USART_ICR register.

Parity generation in transmission

If the PCE bit is set in USART_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of “1s” if even parity is selected (PS = 0) or an odd number of “1s” if odd parity is selected (PS = 1)).

65.5.14 USART LIN (local interconnection network) mode

This section is relevant only when LIN mode is supported. Refer to *Section 65.4: USART implementation*.

The LIN mode is selected by setting the LINEN bit in the USART_CR2 register. In LIN mode, the following bits must be kept cleared:

- STOP[1:0] and CLKEN in the USART_CR2 register,
- SCEN, HDSEL and IREN in the USART_CR3 register.

LIN transmission

The procedure described in *Section 65.5.5* has to be applied for LIN master transmission. It must be the same as for normal USART transmission with the following differences:

- Clear the M bit to configure 8-bit word length.
- Set the LINEN bit to enter LIN mode. In this case, setting the SBKRQ bit sends 13 zero bits as a break character. Then 2 bits of value ‘1 are sent to enable the next start detection.

LIN reception

When LIN mode is enabled, the break detection circuit is activated. The detection is totally independent from the normal USART receiver. A break can be detected whenever it occurs, during Idle state or during a frame.

When the receiver is enabled (RE = 1 in USART_CR1), the circuit looks at the RX input for a start signal. The method for detecting start bits is the same when searching break characters or data. After a start bit has been detected, the circuit samples the next bits exactly like for the data (on the 8th, 9th and 10th samples). If 10 (when the LBDL = 0 in USART_CR2) or 11 (when LBDL = 1 in USART_CR2) consecutive bits are detected as 0, and are followed by a delimiter character, the LBDF flag is set in USART_ISR. If the LBDIE bit = 1, an interrupt is generated. Before validating the break, the delimiter is checked for as it signifies that the RX line has returned to a high level.

If a 1 is sampled before the 10 or 11 have occurred, the break detection circuit cancels the current detection and searches for a start bit again.

If the LIN mode is disabled (LINEN = 0), the receiver continues working as normal USART, without taking into account the break detection.

If the LIN mode is enabled (LINEN = 1), as soon as a framing error occurs (that is, stop bit detected at 0, which is the case for any break frame), the receiver stops until the break detection circuit receives either a ‘1, if the break word was not complete, or a delimiter character if a break has been detected.

The behavior of the break detector state machine and the break flag is shown in *Figure 849*.

Examples of break frames are given in *Figure 850*.

Figure 849. Break detection in LIN mode (11-bit break length - LBDL bit is set)

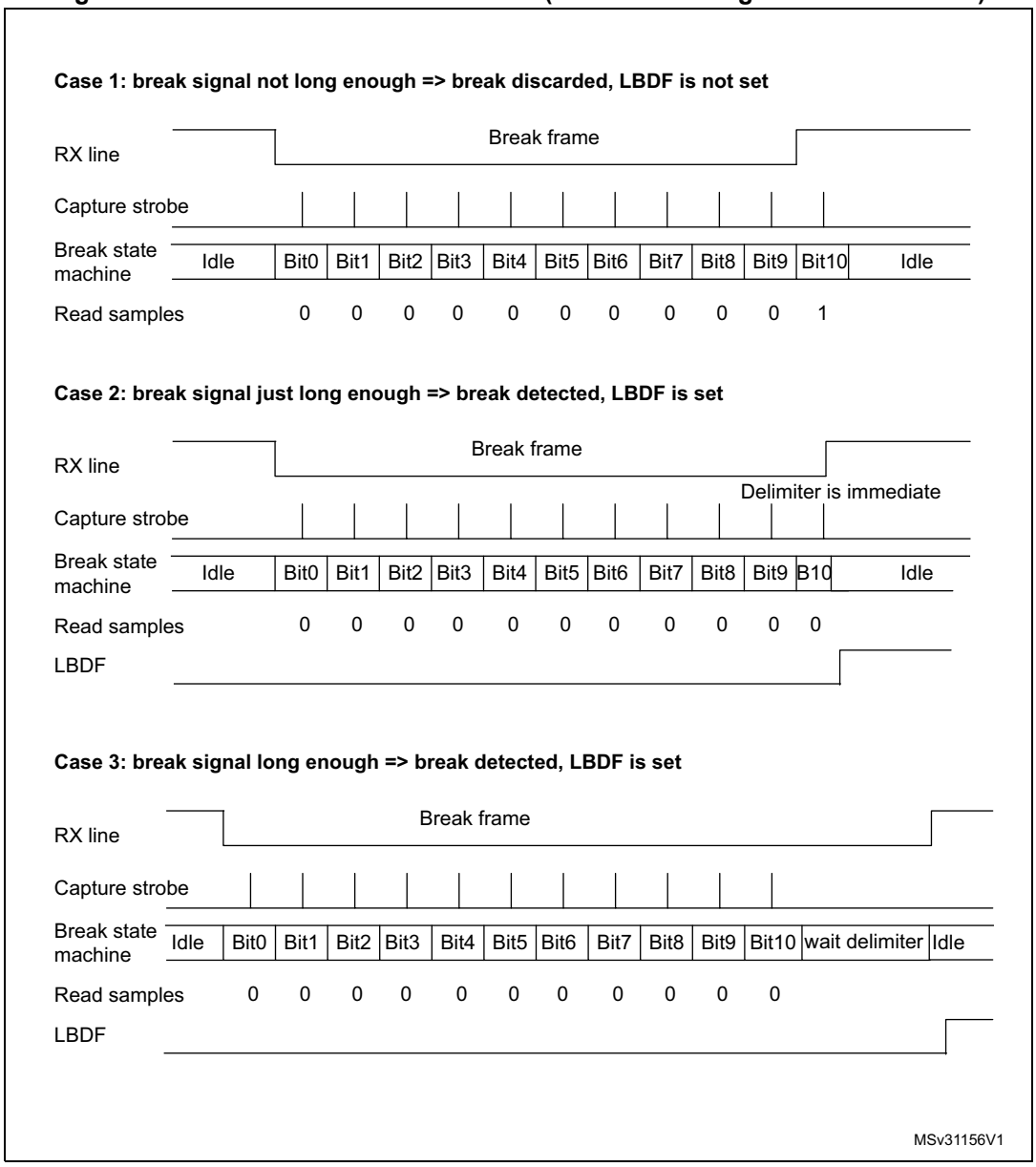
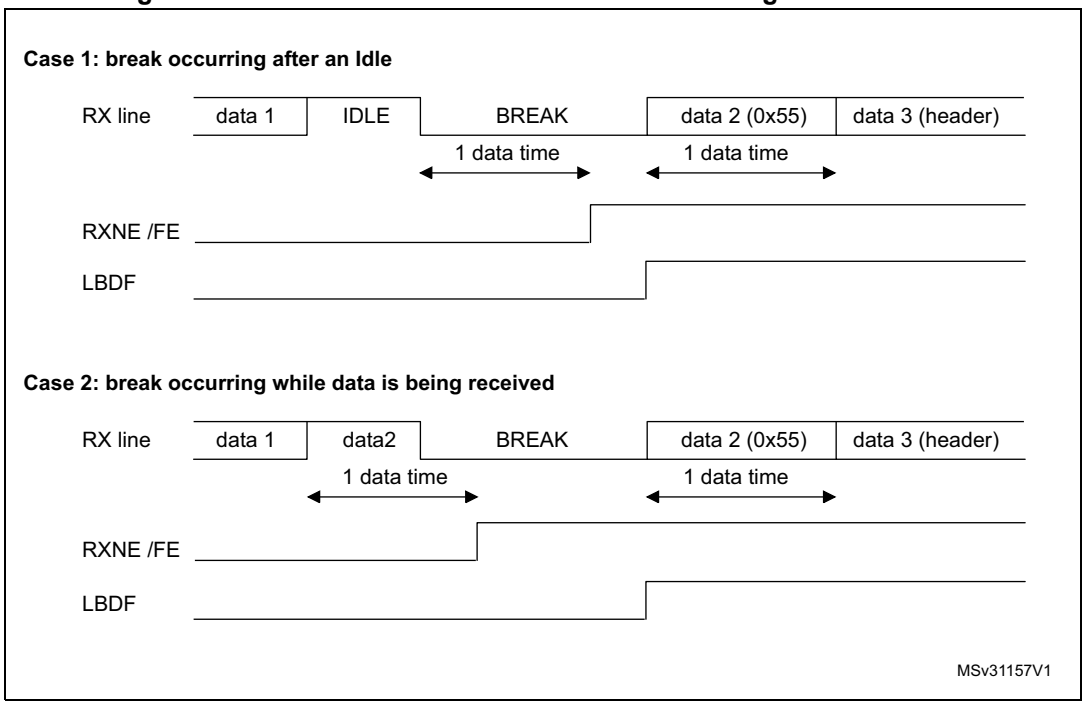


Figure 850. Break detection in LIN mode vs. Framing error detection



65.5.15 USART synchronous mode

Master mode

The synchronous master mode is selected by programming the CLKEN bit in the USART_CR2 register to 1. In synchronous mode, the following bits must be kept cleared:

- LINEN bit in the USART_CR2 register,
- SCEN, HDSEL and IREN bits in the USART_CR3 register.

In this mode, the USART can be used to control bidirectional synchronous serial communications in master mode. The CK pin is the output of the USART transmitter clock. No clock pulses are sent to the CK pin during start bit and stop bit. Depending on the state of the LBCL bit in the USART_CR2 register, clock pulses are, or are not, generated during the last valid data bit (address mark). The CPOL bit in the USART_CR2 register is used to select the clock polarity, and the CPHA bit in the USART_CR2 register is used to select the phase of the external clock (see *Figure 851*, *Figure 852* and *Figure 853*).

During the Idle state, preamble and send break, the external CK clock is not activated.

In synchronous master mode, the USART transmitter operates exactly like in asynchronous mode. However, since CK is synchronized with TX (according to CPOL and CPHA), the data on TX is synchronous.

In synchronous master mode, the USART receiver operates in a different way compared to asynchronous mode. If RE is set to 1, the data are sampled on CK (rising or falling edge, depending on CPOL and CPHA), without any oversampling. A given setup and a hold time must be respected (which depends on the baud rate: 1/16 bit time).

Note: In master mode, the CK pin operates in conjunction with the TX pin. Thus, the clock is provided only if the transmitter is enabled (TE = 1) and data are being transmitted (USART_TDR data register written). This means that it is not possible to receive synchronous data without transmitting data.

Figure 851. USART example of synchronous master transmission

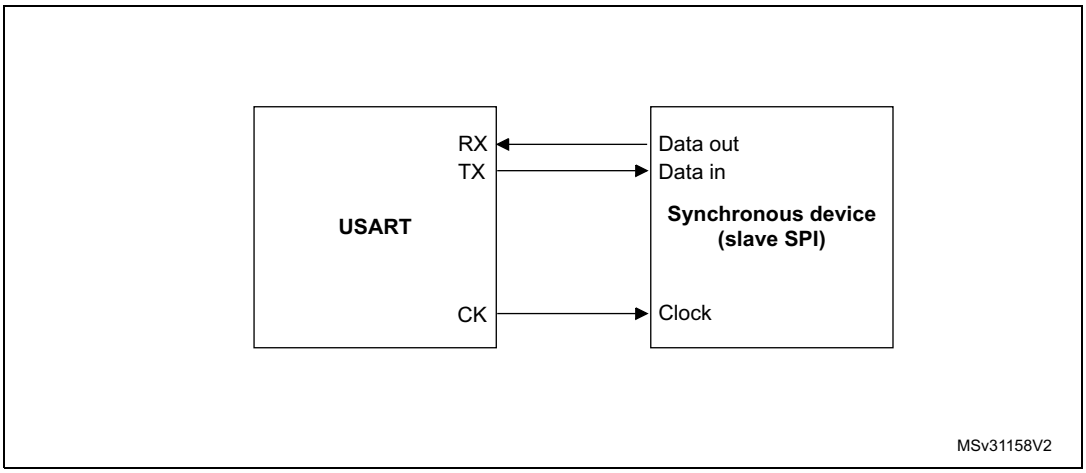


Figure 852. USART data clock timing diagram in synchronous master mode (M bits = 00)

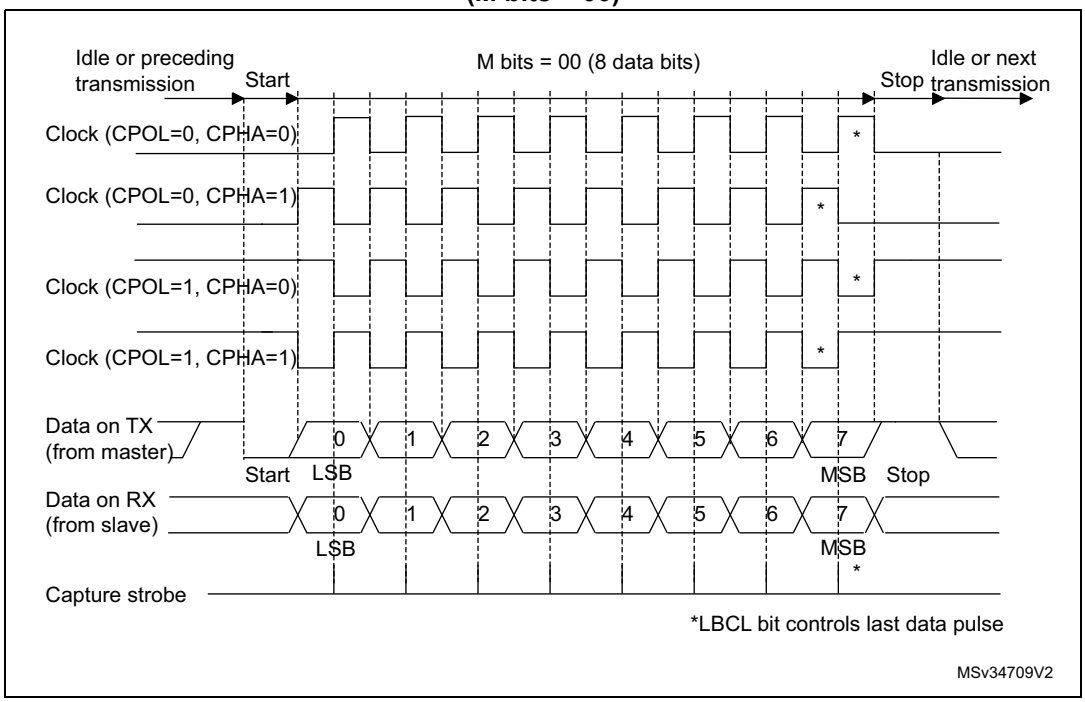
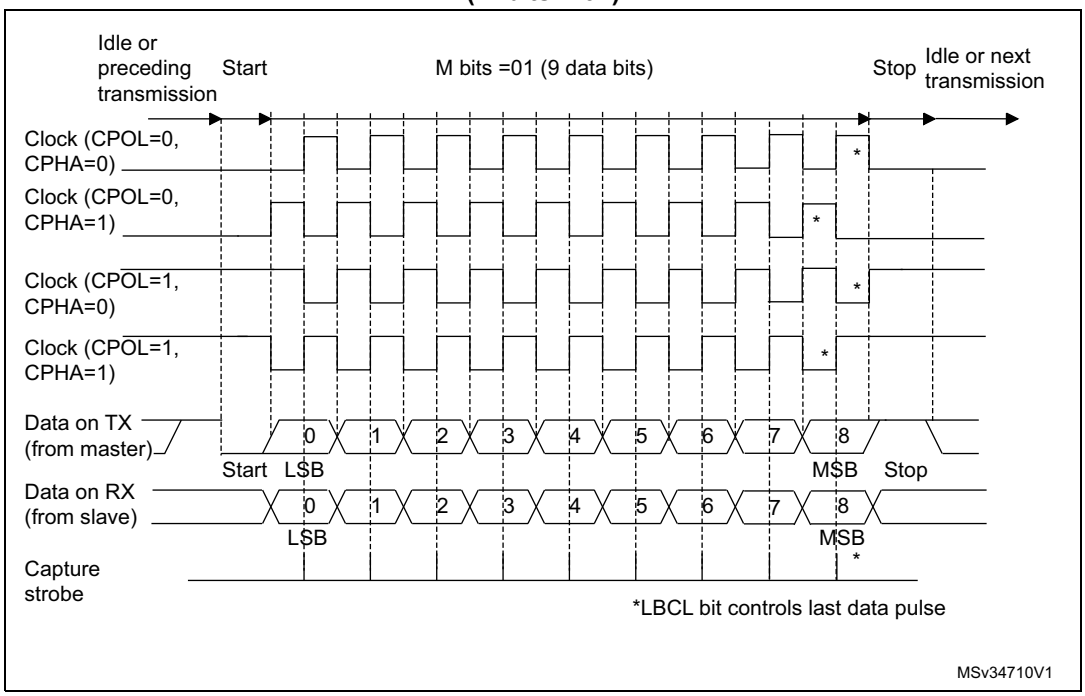


Figure 853. USART data clock timing diagram in synchronous master mode (M bits = 01)



Slave mode

The synchronous slave mode is selected by programming the SLVEN bit in the USART_CR2 register to 1. In synchronous slave mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the USART_CR2 register,
- SCEN, HDSEL and IREN bits in the USART_CR3 register.

In this mode, the USART can be used to control bidirectional synchronous serial communications in slave mode. The CK pin is the input of the USART in slave mode.

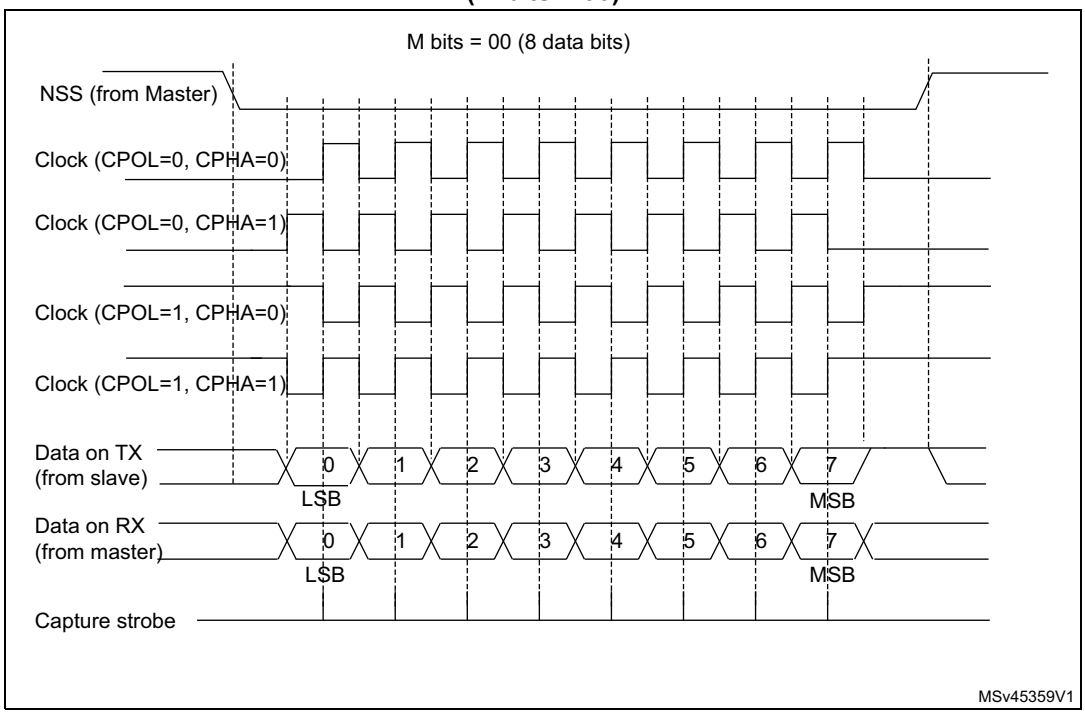
*Note: When the peripheral is used in SPI slave mode, the frequency of peripheral clock source (*usart_ker_ck_pres*) must be greater than 3 times the CK input frequency.*

The CPOL bit and the CPHA bit in the USART_CR2 register are used to select the clock polarity and the phase of the external clock, respectively (see *Figure 854*).

An underrun error flag is available in slave transmission mode. This flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value to USART_TDR.

The slave supports the hardware and software NSS management.

Figure 854. USART data clock timing diagram in synchronous slave mode (M bits = 00)



Slave Select (NSS) pin management

The hardware or software slave select management can be set through the DIS_NSS bit in the USART_CR2 register:

- Software NSS management (DIS_NSS = 1)

  SPI slave is always selected and NSS input pin is ignored.

  The external NSS pin remains free for other application uses.
- Hardware NSS management (DIS_NSS = 0)

  The SPI slave selection depends on NSS input pin. The slave is selected when NSS is low and deselected when NSS is high.

Note: The LBCL (used only on SPI master mode), CPOL and CPHA bits have to be selected when the USART is disabled (UE = 0) to ensure that the clock pulses function correctly.

In SPI slave mode, the USART must be enabled before starting the master communications (or between frames while the clock is stable). Otherwise, if the USART slave is enabled while the master is in the middle of a frame, it becomes desynchronized with the master. The data register of the slave needs to be ready before the first edge of the communication clock or before the end of the ongoing communication, otherwise the SPI slave transmits zeros.

SPI slave underrun error

When an underrun error occurs, the UDR flag is set in the USART_ISR register, and the SPI slave goes on sending the last data until the underrrun error flag is cleared by software.

The underrun flag is set at the beginning of the frame. An underrun error interrupt is triggered if EIE bit is set in the USART_CR3 register.

The underrun error flag is cleared by setting bit UDRCF in the USART_ICR register.

In case of underrun error, it is still possible to write to the TDR register. Clearing the underrun error enables sending new data.

If an underrun error occurred and there is no new data written in TDR, then the TC flag is set at the end of the frame.

Note: An underrun error may occur if the moment the data is written to the USART_TDR is too close to the first CK transmission edge. To avoid this underrun error, the USART_TDR must be written 3 usart_ker_ck *cycles before the first CK edge.*

65.5.16 USART single-wire half-duplex communication

Single-wire half-duplex mode is selected by setting the HDSEL bit in the USART_CR3 register. In this mode, the following bits must be kept cleared:

- LINEN and CLKEN bits in the USART_CR2 register,
- SCEN and IREN bits in the USART_CR3 register.

The USART can be configured to follow a single-wire half-duplex protocol where the TX and RX lines are internally connected. The selection between half- and full-duplex communication is made with a control bit HDSEL in USART_CR3.

As soon as HDSEL is written to 1:

- The TX and RX lines are internally connected.
- The RX pin is no longer used.
- The TX pin is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. It means that the I/O must be configured so that TX is configured as alternate function open-drain with an external pull-up.

Apart from this, the communication protocol is similar to normal USART mode. Any conflict on the line must be managed by software (for instance by using a centralized arbiter). In particular, the transmission is never blocked by hardware and continues as soon as data are written in the data register while the TE bit is set.

65.5.17 USART receiver timeout

The receiver timeout feature is enabled by setting the RTOEN bit in the USART_CR2 control register.

The timeout duration is programmed using the RTO bitfields in the USART_RTOR register.

The receiver timeout counter starts counting:

- from the end of the stop bit if STOP = 00 or STOP = 11
- from the end of the second stop bit if STOP = 10.
- from the beginning of the stop bit if STOP = 01.

When the timeout duration has elapsed, the RTOF flag in the USART_ISR register is set. A timeout is generated if RTOIE bit in USART_CR1 register is set.

65.5.18 USART smartcard mode

This section is relevant only when smartcard mode is supported. Refer to *Section 65.4: USART implementation*.

Smartcard mode is selected by setting the SCEN bit in the USART_CR3 register. In smartcard mode, the following bits must be kept cleared:

- LINEN bit in the USART_CR2 register,
- HDSEL and IREN bits in the USART_CR3 register.

The CLKEN bit can also be set to provide a clock to the smartcard.

The smartcard interface is designed to support asynchronous smartcard protocol as defined in the ISO 7816-3 standard. Both T = 0 (character mode) and T = 1 (block mode) are supported.

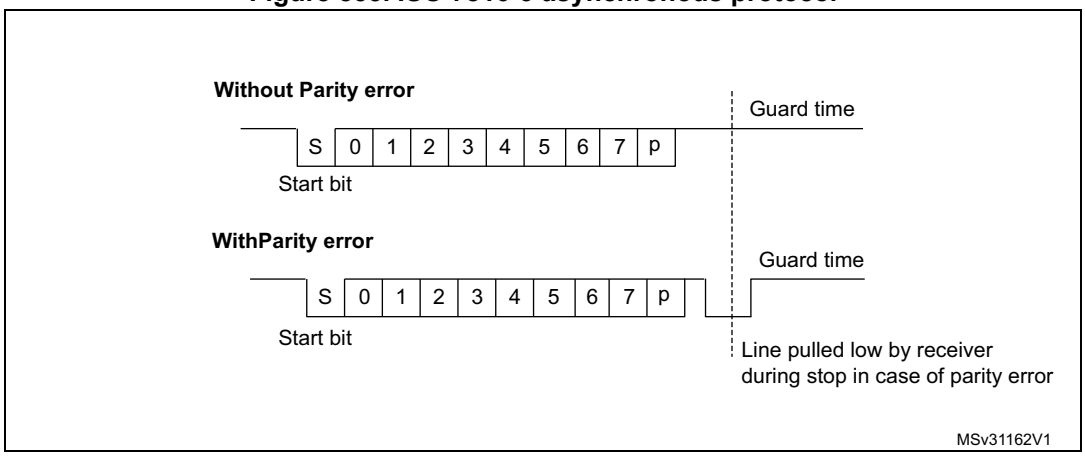
The USART must be configured as:

- 8 bits plus parity: M = 1 and PCE = 1 in the USART_CR1 register
- 1.5 stop bits when transmitting and receiving data: STOP = 11 in the USART_CR2 register. It is also possible to choose 0.5 stop bit for reception.

In T = 0 (character) mode, the parity error is indicated at the end of each character during the guard time period.

Figure 855 shows examples of what can be seen on the data line with and without parity error.

Figure 855. ISO 7816-3 asynchronous protocol



When connected to a smartcard, the TX output of the USART drives a bidirectional line that is also driven by the smartcard. The TX pin must be configured as open drain.

Smartcard mode implements a single wire half duplex communication protocol.

- Transmission of data from the transmit shift register is guaranteed to be delayed by a minimum of 1/2 baud clock. In normal operation a full transmit shift register starts shifting on the next baud clock edge. In smartcard mode this transmission is further delayed by a guaranteed 1/2 baud clock.
- In transmission, if the smartcard detects a parity error, it signals this condition to the USART by driving the line low (NACK). This NACK signal (pulling transmit line low for 1 baud clock) causes a framing error on the transmitter side (configured with 1.5 stop bits). The USART can handle automatic re-sending of data according to the protocol.

The number of retries is programmed in the SCARCNT bitfield. If the USART continues receiving the NACK after the programmed number of retries, it stops transmitting and signals the error as a framing error. The TXE bit (TXFNF bit in case FIFO mode is enabled) may be set using the TXFRQ bit in the USART_RQR register.

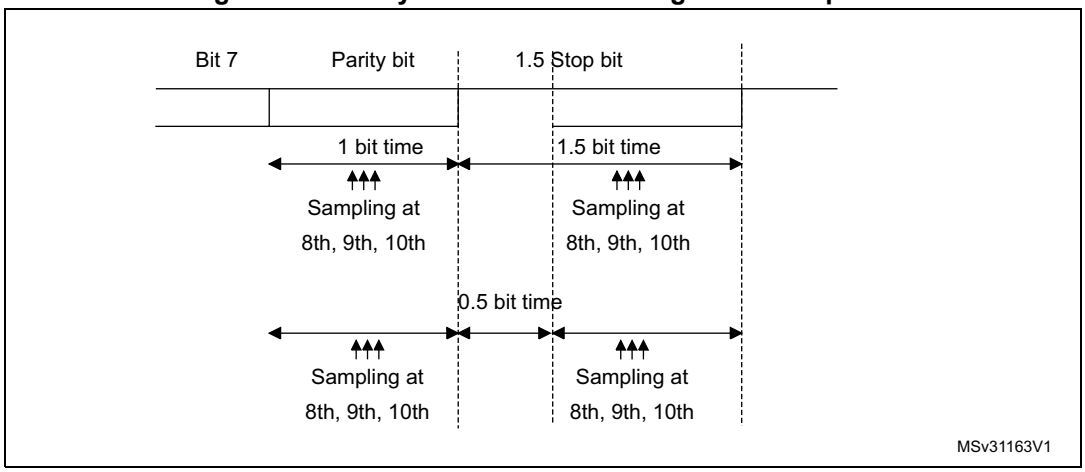
- Smartcard auto-retry in transmission: A delay of 2.5 baud periods is inserted between the NACK detection by the USART and the start bit of the repeated character. The TC bit is set immediately at the end of reception of the last repeated character (no guard time). If the software wants to repeat it again, it must ensure the minimum 2-baud periods required by the standard.
- If a parity error is detected during reception of a frame programmed with a 1.5 stop bit period, the transmit line is pulled low for a baud clock period after the completion of the receive frame. This is to indicate to the smartcard that the data transmitted to the USART has not been correctly received. A parity error is NACKed by the receiver if the NACK control bit is set, otherwise a NACK is not transmitted (to be used in T = 1 mode). If the received character is erroneous, the RXNE (RXFNE in case FIFO mode is enabled)/receive DMA request is not activated. According to the protocol specification, the smartcard must resend the same character. If the received character is still erroneous after the maximum number of retries specified in the SCARCNT bitfield, the USART stops transmitting the NACK and signals the error as a parity error.
- Smartcard auto-retry in reception: the BUSY flag remains set if the USART NACKs the card but the card doesn't repeat the character.
- In transmission, the USART inserts the guard time (as programmed in the guard time register) between two successive characters. As the guard time is measured after the stop bit of the previous character, the GT[7:0] register must be programmed to the desired CGT character guard time, as defined by the 7816-3 specification) minus 12 (the duration of one character).
- The assertion of the TC flag can be delayed by programming the guard time register. In normal operation, TC is asserted when the transmit shift register is empty and no further transmit requests are outstanding. In smartcard mode an empty transmit shift register triggers the guard time counter to count up to the programmed value in the guard time register. TC is forced low during this time. When the guard time counter reaches the programmed value TC is asserted high. The TCBGT flag can be used to detect the end of data transfer without waiting for guard time completion. This flag is set just after the end of frame transmission and if no NACK has been received from the card.
- The de-assertion of TC flag is unaffected by smartcard mode.
- If a framing error is detected on the transmitter end (due to a NACK from the receiver), the NACK is not detected as a start bit by the receive block of the transmitter. According to the ISO protocol, the duration of the received NACK can be 1 or 2 baud clock periods.
- On the receiver side, if a parity error is detected and a NACK is transmitted the receiver does not detect the NACK as a start bit.

Note: Break characters are not significant in smartcard mode. A 0x00 data with a framing error is treated as data and not as a break.

No Idle frame is transmitted when toggling the TE bit. The Idle frame (as defined for the other configurations) is not defined by the ISO protocol.

Figure 856 shows how the NACK signal is sampled by the USART. In this example the USART is transmitting data and is configured with 1.5 stop bits. The receiver part of the USART is enabled in order to check the integrity of the data and the NACK signal.

Figure 856. Parity error detection using the 1.5 stop bits



The USART can provide a clock to the smartcard through the CK output. In smartcard mode, CK is not associated to the communication but is simply derived from the internal peripheral input clock through a 5-bit prescaler. The division ratio is configured in the USART_GTPR register. CK frequency can be programmed from usart_ker_ck_pres/2 to usart_ker_ck_pres/62, where usart_ker_ck_pres is the peripheral input clock divided by a programmed prescaler.

Block mode (T = 1)

In T = 1 (block) mode, the parity error transmission can be deactivated by clearing the NACK bit in the USART_CR3 register.

When requesting a read from the smartcard, in block mode, the software must program the RTOR register to the BWT (block wait time) – 11 value. If no answer is received from the card before the expiration of this period, a timeout interrupt is generated. If the first character is received before the expiration of the period, it is signaled by the RXNE/RXFNE interrupt.

Note: The RXNE/RXFNE interrupt must be enabled even when using the USART in DMA mode to read from the smartcard in block mode. In parallel, the DMA must be enabled only after the first received byte.

After the reception of the first character (RXNE/RXFNE interrupt), the RTO register must be programmed to the CWT (character wait time – 11 value), in order to enable the automatic check of the maximum wait time between two consecutive characters. This time is expressed in baud time units. If the smartcard does not send a new character in less than the CWT period after the end of the previous character, the USART signals it to the software through the RTOF flag and interrupt (when RTOIE bit is set).

Note: As in the smartcard protocol definition, the BWT/CWT values must be defined from the beginning (start bit) of the last character. The RTO register must be programmed to BWT – 11 or CWT – 11, respectively, taking into account the length of the last character itself.

A block length counter is used to count all the characters received by the USART. This counter is reset when the USART is transmitting. The length of the block is communicated by the smartcard in the third byte of the block (prologue field). This value must be programmed to the BLEN field in the USART_RTOR register. When using DMA mode, before the start of the block, this register field must be programmed to the minimum value

(0x0). With this value, an interrupt is generated after the 4th received character. The software must read the LEN field (third byte), its value must be read from the receive buffer.

In interrupt driven receive mode, the length of the block may be checked by software or by programming the BLEN value. However, before the start of the block, the maximum value of BLEN (0xFF) may be programmed. The real value is programmed after the reception of the third character.

If the block is using the LRC longitudinal redundancy check (1 epilogue byte), the BLEN=LEN. If the block is using the CRC mechanism (2 epilog bytes), BLEN=LEN+1 must be programmed. The total block length (including prologue, epilogue and information fields) equals BLEN+4. The end of the block is signaled to the software through the EOBF flag and interrupt (when EOBIE bit is set).

In case of an error in the block length, the end of the block is signaled by the RTO interrupt (Character Wait Time overflow).

Note: The error checking code (LRC/CRC) must be computed/verified by software.

Direct and inverse convention

The smartcard protocol defines two conventions: direct and inverse.

The direct convention is defined as: LSB first, logical bit value of 1 corresponds to a H state of the line and parity is even. In order to use this convention, the following control bits must be programmed: MSBFIRST = 0, DATAINV = 0 (default values).

The inverse convention is defined as: MSB first, logical bit value 1 corresponds to an L state on the signal line and parity is even. In order to use this convention, the following control bits must be programmed: MSBFIRST = 1, DATAINV =1.

Note: When logical data values are inverted (0 = H, 1 = L), the parity bit is also inverted in the same way.

In order to recognize the card convention, the card sends the initial character, TS, as the first character of the ATR (Answer To Reset) frame. The two possible patterns for the TS are: LHHL LLL LLH and LHHL HHH LLH.

- (H) LHHL LLL LLH sets up the inverse convention: state L encodes value 1 and moment 2 conveys the most significant bit (MSB first). When decoded by inverse convention, the conveyed byte is equal to '3F'.
- (H) LHHL HHH LLH sets up the direct convention: state H encodes value 1 and moment 2 conveys the least significant bit (LSB first). When decoded by direct convention, the conveyed byte is equal to '3B'.

Character parity is correct when there is an even number of bits set to 1 in the nine moments 2 to 10.

As the USART does not know which convention is used by the card, it needs to be able to recognize either pattern and act accordingly. The pattern recognition is not done in hardware, but through a software sequence. Moreover, supposing that the USART is configured in direct convention (default) and the card answers with the inverse convention, TS = LHHL LLL LLH ≥ the USART received character is equal to 03 and the parity is odd.

Therefore, two methods are available for TS pattern recognition:

Method 1

The USART is programmed in standard smartcard mode/direct convention. In this case, the TS pattern reception generates a parity error interrupt and error signal to the card.

- The parity error interrupt informs the software that the card did not answer correctly in direct convention. Software then reprograms the USART for inverse convention
- In response to the error signal, the card retries the same TS character, and it is correctly received this time, by the reprogrammed USART

Alternatively, in answer to the parity error interrupt, the software may decide to reprogram the USART and to also generate a new reset command to the card, then wait again for the TS.

Method 2

The USART is programmed in 9-bit/no-parity mode, no bit inversion. In this mode it receives any of the two TS patterns as:

(H) LHHL LLL LLH = 0x103: inverse convention to be chosen

(H) LHHL HHH LLH = 0x13B: direct convention to be chosen

The software checks the received character against these two patterns and, if any of them match, then programs the USART accordingly for the next character reception.

If none of the two is recognized, a card reset may be generated in order to restart the negotiation.

65.5.19 USART IrDA SIR ENDEC block

This section is relevant only when IrDA mode is supported. Refer to *Section 65.4: USART implementation*.

IrDA mode is selected by setting the IREN bit in the USART_CR3 register. In IrDA mode, the following bits must be kept cleared:

- LINEN, STOP and CLKEN bits in the USART_CR2 register,
- SCEN and HDSEL bits in the USART_CR3 register.

The IrDA SIR physical layer specifies the use of a return-to-zero, inverted (RZI) modulation scheme that represents logic 0 as an infrared light pulse (see *Figure 857*).

The encoding and decoding of data in IrDA mode is handled by hardware blocks.

Even if IrDA is a half-duplex protocol, it requires two pins for transmission and reception (USART_TX and USART_RX).

The SIR transmit encoder modulates the non return-to-zero (NRZ) transmit bit stream output from USART. The output pulse stream is transmitted to an external output driver and infrared LED. USART supports only bit rates up to 115.2 kbauds for the SIR ENDEC. In normal mode the transmitted pulse width is specified as 3/16 of a bit period.

The SIR receive decoder demodulates the return-to-zero bit stream from the infrared detector and outputs the received NRZ serial bit stream to the USART. The decoder input is normally high (marking state) in the Idle state. The transmit encoder output has the opposite polarity to the decoder input. A start bit is detected when the decoder input is low.

- IrDA is a half duplex communication protocol. If the Transmitter is busy (when the USART is sending data to the IrDA encoder), any data on the IrDA receive line is

ignored by the IrDA decoder and if the Receiver is busy (when the USART is receiving decoded data from the USART), data on the TX from the USART to IrDA is not encoded. While receiving data, transmission must be avoided as the data to be transmitted may be corrupted.

- A 0 is transmitted as a high pulse and a 1 is transmitted as a 0. The width of the pulse is specified as 3/16th of the selected bit period in normal mode (see *Figure 858*).
- The SIR decoder converts the IrDA compliant receive signal into a bit stream for USART.
- The SIR receive logic interprets a high state as a logic one and low pulses as logic zeros.
- The transmit encoder output has the opposite polarity to the decoder input. The SIR output is in low state when Idle.
- The IrDA specification requires the acceptance of pulses greater than 1.41 µs. The acceptable pulse width is programmable. Glitch detection logic on the receiver end filters out pulses of width less than two periods, where the period equals PSC / usart_ker_ck_pres (PSC being the prescaler value programmed in the PSC[7:0] bitfield of the USART_GTPR register). Pulses of width less than one period are always rejected, but those of width greater than one and less than two periods may be accepted or rejected, those greater than two periods are accepted as a pulse. The IrDA encoder/decoder does not work when PSC = 0.
- The receiver can communicate with a low-power transmitter.
- In IrDA mode, the stop bits in the USART_CR2 register must be configured to 1 stop bit.

IrDA low-power mode

- Transmitter

  In low-power mode, the pulse width is not maintained at 3 / 16 of the bit period. Instead, the width of the pulse is three times the IrDA low-power period. The IrDA low-power frequency minimum value is 1.42 MHz. Generally, this value is 1.8432 MHz (1.42 MHz < IrDA low-power frequency < 2.12 MHz). A low-power mode programmable divisor divides the system clock to achieve this value.
- Receiver

  Receiving in low-power mode is similar to receiving in normal mode. For glitch detection the USART must discard pulses of duration shorter than one IrDA low-power period. A valid low is accepted only if its duration is greater than two IrDA low-power periods.

Note: IrDA low-power period = PSC / usart_ker_ck_pres, where PSC corresponds to the value programmed in the PSC[7:0] bitfield of *the USART_GTPR register.*

A pulse of width less than two and greater than one IrDA low-power period may or may not be rejected.

The receiver set-up time must be managed by software. The IrDA physical layer specification specifies a minimum of 10 ms delay between transmission and reception.

Figure 857. IrDA SIR ENDEC block diagram

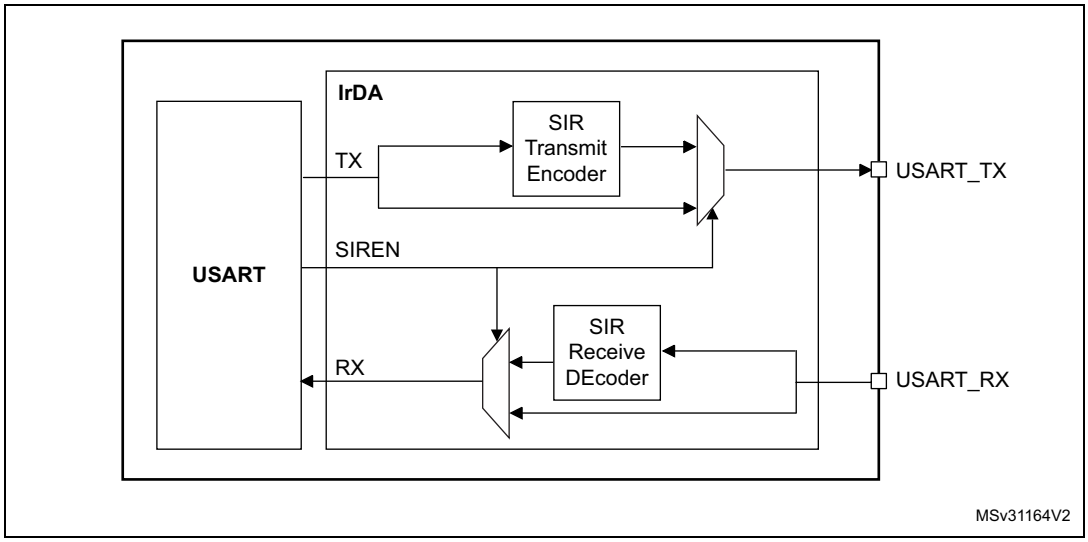
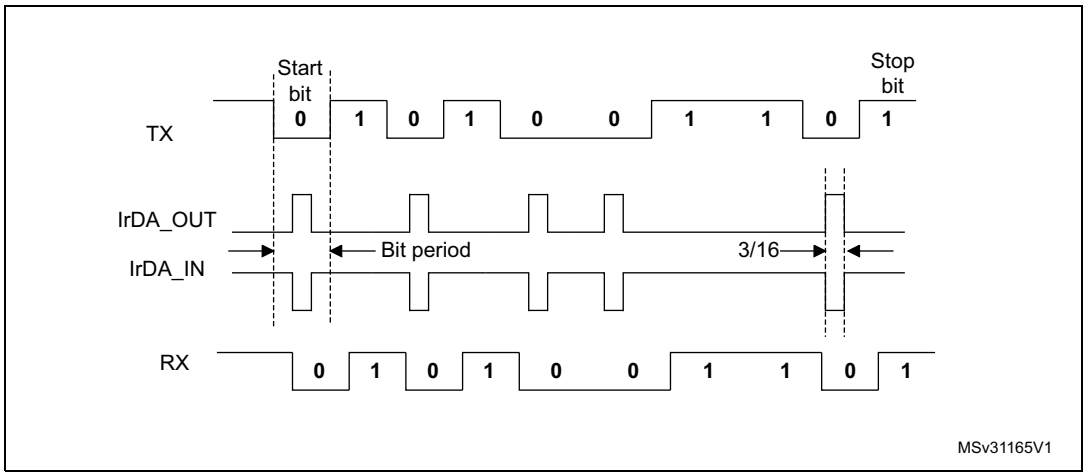


Figure 858. IrDA data modulation (3/16) - normal mode



65.5.20 Continuous communication using USART and DMA

The USART is capable of performing continuous communications using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

Note: Refer to Section 65.4: USART implementation to determine if the DMA mode is supported. If DMA is not supported, use the USART as explained in Section 65.5.7. To perform continuous communications when the FIFO is disabled, clear the TXE/ RXNE flags in the USART_ISR register.

Transmission using DMA

DMA mode can be enabled for transmission by setting the DMAT bit in the USART_CR3 register. Data are loaded from an SRAM area configured using the DMA peripheral (refer to *section direct memory access controller (DMA)*) to the USART_TDR register whenever the TXE flag (TXFNF flag if FIFO mode is enabled) is set. To map a DMA channel for USART transmission, use the following procedure (x denotes the channel number):

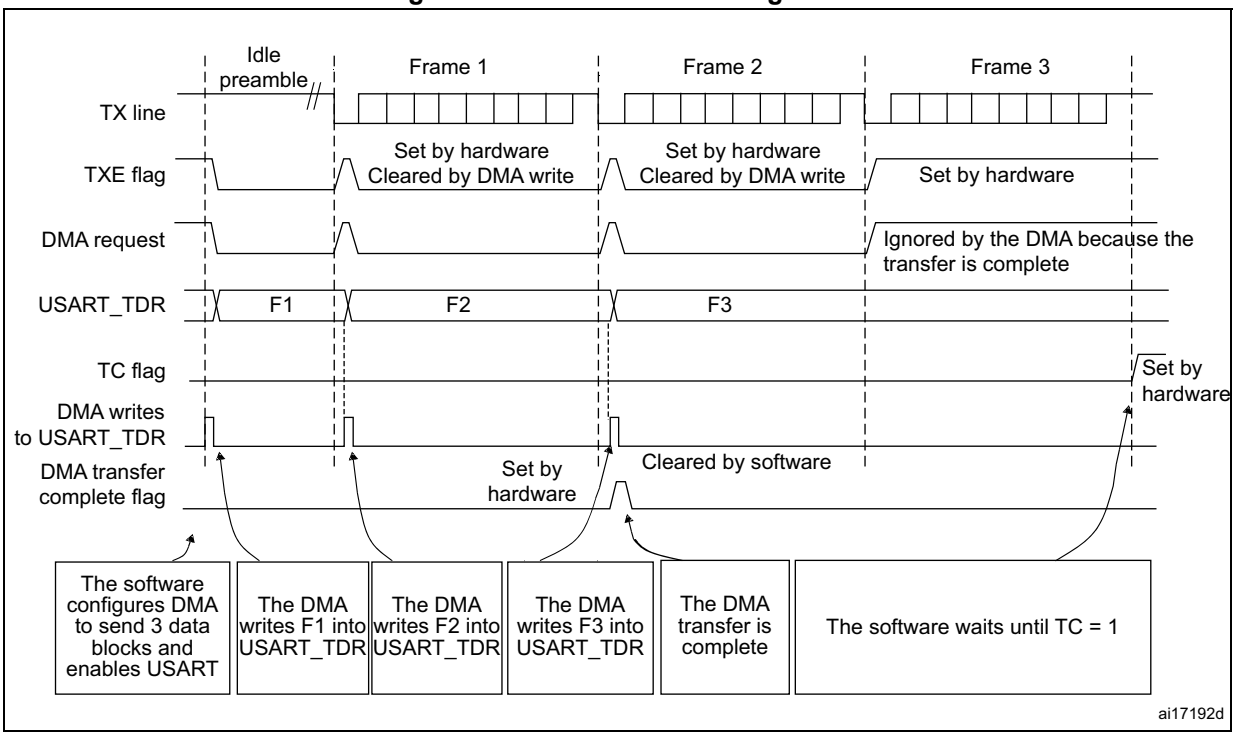
1. Write the USART_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data is loaded into the USART_TDR register from this memory area after each TXE (or TXFNF if FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA register
5. Configure DMA interrupt generation after half/ full transfer as required by the application.
6. Clear the TC flag in the USART_ISR register by setting the TCCF bit in the USART_ICR register.
7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In transmission mode, once the DMA has written all the data to be transmitted (DMA transfer complete), the TC flag can be monitored to make sure that the USART communication has completed. This is required to avoid corrupting the last transmission before disabling the USART or before the system enters a low-power mode when the peripheral clock is disabled. Software must wait until TC = 1. The TC flag remains cleared during all data transfers and it is set by hardware at the end of transmission of the last frame.

Note: The DMAT bit must not be cleared before the DMA end of transfer.

Figure 859. Transmission using DMA



Note: When FIFO management is enabled, the DMA request is triggered by transmit FIFO not full (that is, TXFNF = 1).

Reception using DMA

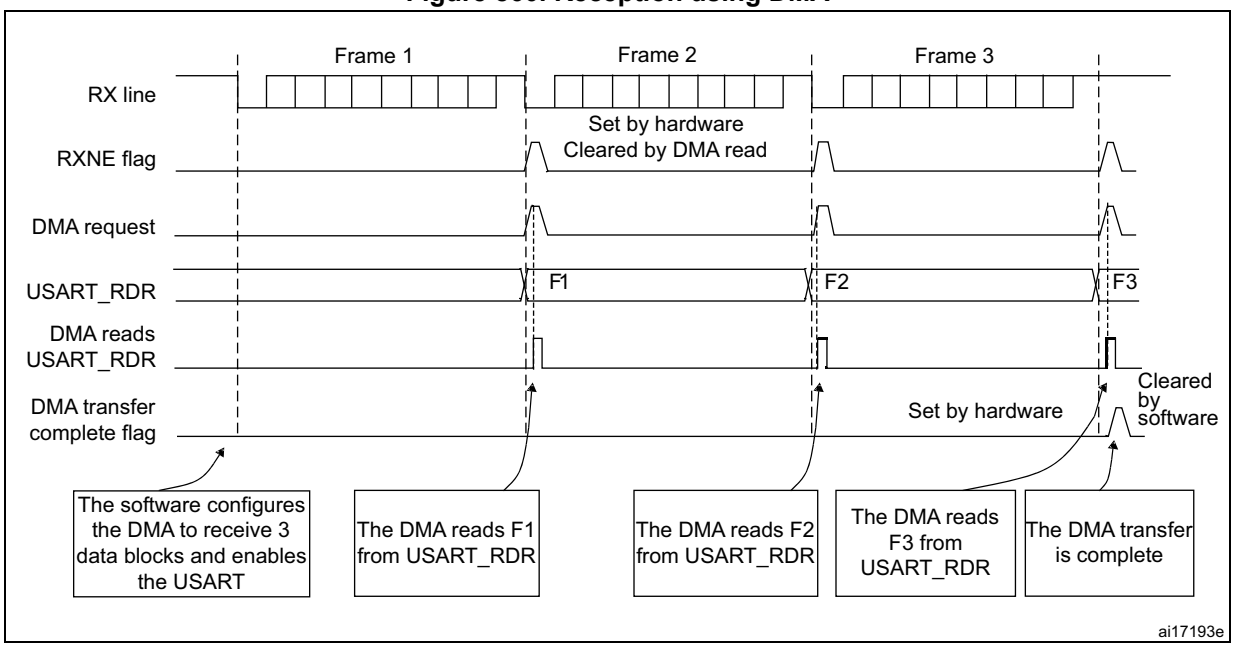
DMA mode can be enabled for reception by setting the DMAR bit in the USART_CR3 register. Data are loaded from the USART_RDR register to an SRAM area configured using the DMA peripheral (refer to *section direct memory access controller (DMA)*) whenever a data byte is received. To map a DMA channel for USART reception, use the following procedure:

1. Write the USART_RDR register address in the DMA control register to configure it as the source of the transfer. The data is moved from this address to the memory after each RXNE (RXFNE in case FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data is loaded from USART_RDR to this memory area after each RXNE (RXFNE in case FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA control register
5. Configure interrupt generation after half/ full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

Note: The DMAR bit must not be cleared before the DMA end of transfer.

Figure 860. Reception using DMA



Note: When FIFO management is enabled, the DMA request is triggered by receive FIFO not empty (that is, RXFNE = 1).

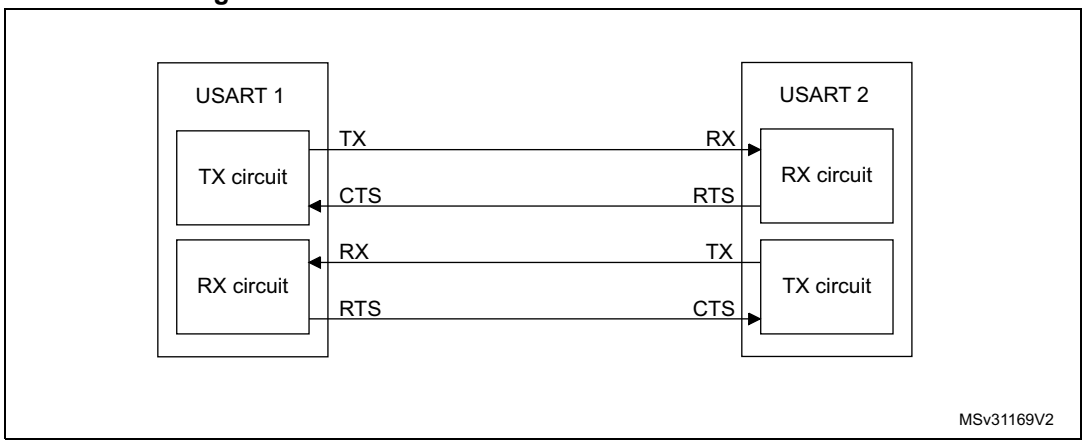
Error flagging and interrupt generation in multibuffer communication

If any error occurs during a transaction in multibuffer communication mode, the error flag is asserted after the current byte. An interrupt is generated if the interrupt enable flag is set. For framing error, overrun error and noise flag, which are asserted with RXNE (RXFNE in case FIFO mode is enabled) in single byte reception, there is a separate error flag interrupt enable bit (EIE bit in the USART_CR3 register), which, if set, enables an interrupt after the current byte if any of these errors occur.

65.5.21 RS232 hardware flow control and RS485 driver enable

It is possible to control the serial data flow between two devices by using the CTS input and the RTS output. The *Figure 861* shows how to connect two devices in this mode:

Figure 861. Hardware flow control between two USARTs

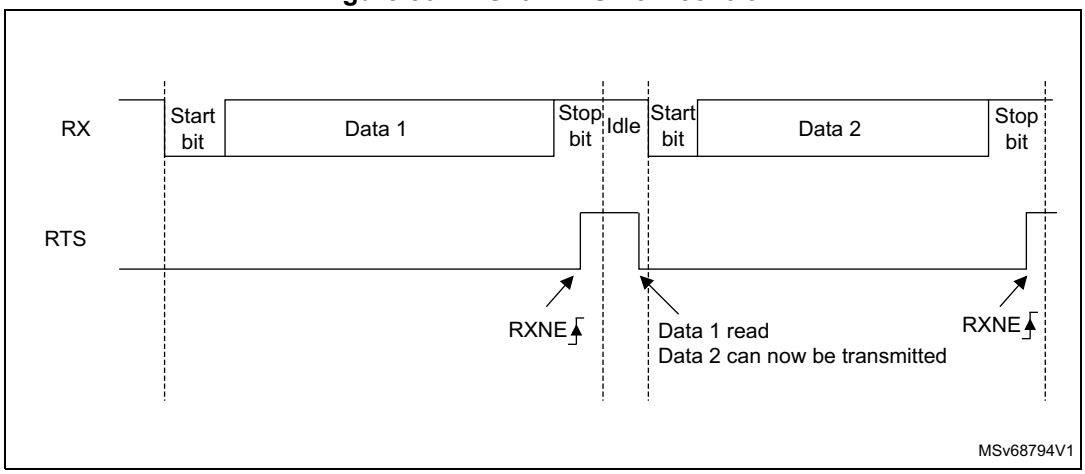


RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits to 1 in the USART_CR3 register.

RS232 RTS flow control

If the RTS flow control is enabled (RTSE = 1), then RTS is deasserted (tied low) as long as the USART receiver is ready to receive a new data. When the receive register is full, RTS is asserted, indicating that the transmission is expected to stop at the end of the current frame. *Figure 862* shows an example of communication with RTS flow control enabled.

Figure 862. RS232 RTS flow control



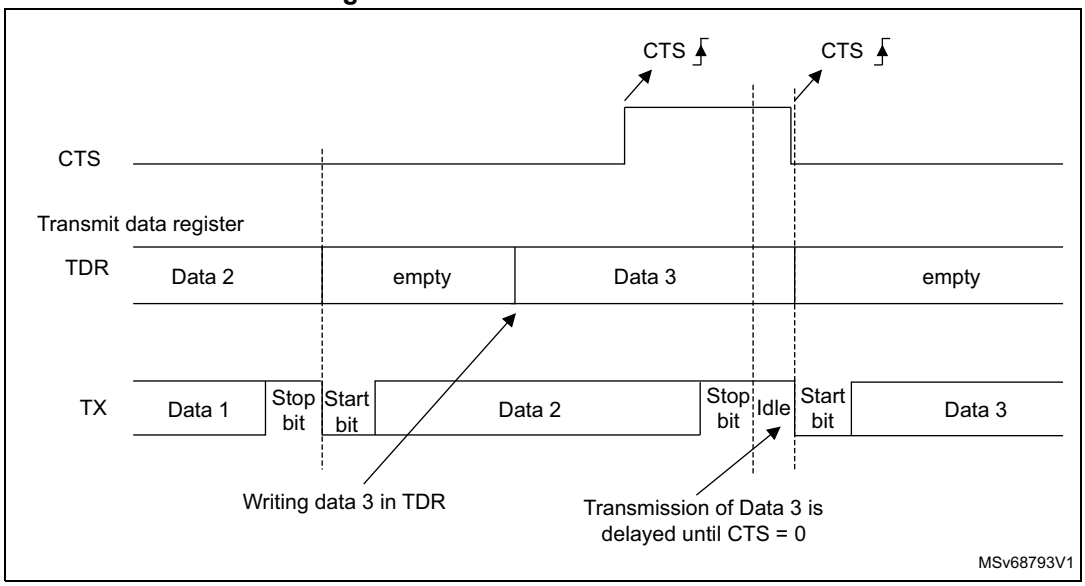
Note: When FIFO mode is enabled, RTS is asserted only when RXFIFO is full.

RS232 CTS flow control

If the CTS flow control is enabled (CTSE = 1), then the transmitter checks the CTS input before transmitting the next frame. If CTS is deasserted (tied low), then the next data is transmitted (assuming that data is to be transmitted, in other words, if TXE/TXFE = 0), else the transmission does not occur. When CTS is asserted during a transmission, the current transmission completes before the transmitter stops.

When CTSE = 1, the CTSIF status bit is automatically set by hardware as soon as the CTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the USART_CR3 register is set. *Figure 863* shows an example of communication with CTS flow control enabled.

Figure 863. RS232 CTS flow control



Note: For correct behavior, CTS must be deasserted at least three USART clock source periods before the end of the current character. In addition, it must be noted that the CTSCF flag may not be set for pulses shorter than 2 x PCLK periods.

RS485 driver enable

The driver enable feature is enabled by setting bit DEM in the USART_CR3 control register. This enables the user to activate the external transceiver control, through the DE (driver enable) signal. The deassertion time is the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE signal. It is programmed using the DEDT [4:0] bitfields in the USART_CR1 control register. The polarity of the DE signal can be configured using the DEP bit in the USART_CR3 control register.

In USART, the DEAT and DEDT are expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

65.5.22 USART low-power management

The USART has advanced low-power mode functions, that enables transferring properly data even when the usart_pclk clock is disabled.

The USART is able to wake up the MCU from low-power mode when the UESM bit is set.

When the usart_pclk is gated, the USART provides a wake-up interrupt (**usart_wkup**) if a specific action requiring the activation of the **usart_pclk** clock is needed:

- If FIFO mode is disabled

  usart_pclk clock has to be activated to empty the USART data register.

  In this case, the usart_wkup interrupt source is RXNE set to 1. The RXNEIE bit must be set before entering low-power mode.
- If FIFO mode is enabled

  usart_pclk clock has to be activated to:
  - To fill the TXFIFO, or
  - To empty the RXFIFO

  In this case, the usart_wkup interrupt source can be:
  - RXFIFO not empty. In this case, the RXFNEIE bit must be set before entering low-power mode.
  - RXFIFO full. In this case, the RXFFIE bit must be set before entering low-power mode, the number of received data corresponds to the RXFIFO size, and the RXFF flag is not set.
  - TXFIFO empty. In this case, the TXFEIE bit must be set before entering low-power mode.

  This enables sending/receiving the data in the TXFIFO/RXFIFO during low-power mode.

  To avoid overrun/underrun errors and transmit/receive data in low-power mode, the usart_wkup interrupt source can be one of the following events:
  - TXFIFO threshold reached. In this case, the TXFTIE bit must be set before entering low-power mode.
  - RXFIFO threshold reached. In this case, the RXFTIE bit must be set before entering low-power mode.

  For example, the application can set the threshold to the maximum RXFIFO size if the wake-up time is less than the time required to receive a single byte across the line.

  Using the RXFIFO full, TXFIFO empty, RXFIFO not empty and RXFIFO/TXFIFO threshold interrupts to wake up the MCU from low-power mode enables doing as many USART transfers as possible during low-power mode with the benefit of optimizing consumption.

Alternatively, a specific **usart_wkup** interrupt can be selected through the WUS bitfields.

When the wake-up event is detected, the WUF flag is set by hardware and a **usart_wkup** interrupt is generated if the WUFIE bit is set.

Note: Before entering low-power mode, make sure that no USART transfers are ongoing. Checking the BUSY flag cannot ensure that low-power mode is never entered when data reception is ongoing.

The WUF flag is set when a wake-up event is detected, independently of whether the MCU is in low-power or active mode.

When entering low-power mode just after having initialized and enabled the receiver, the REACK bit must be checked to make sure the USART is enabled.

When DMA is used for reception, it must be disabled before entering low-power mode and reenabled when exiting from low-power mode.

When the FIFO is enabled, waking up from low-power mode on address match is only possible when mute mode is enabled.

Using mute mode with low-power mode

If the USART is put into mute mode before entering low-power mode:

- Wake-up from mute mode on idle detection must not be used, because idle detection cannot work in low-power mode.
- If the wake-up from mute mode on address match is used, then the low-power mode wake-up source must also be the address match. If the RXNE flag was set when entering the low-power mode, the interface remains in mute mode upon address match and wake up from low-power mode.

Note: When FIFO management is enabled, mute mode can be used with wake-up from low-power mode without any constraints (that is, the two points mentioned above about mute and low-power mode are valid only when FIFO management is disabled).

Wake-up from low-power mode when USART kernel clock (usart_ker_ck) is OFF in low-power mode

If during low-power mode, the usart_ker_ck clock is switched OFF when a falling edge on the USART receive line is detected, the USART interface requests the usart_ker_ck clock to be switched ON thanks to the usart_ker_ck_req signal. usart_ker_ck is then used for the frame reception.

If the wake-up event is verified, the MCU wakes up from low-power mode and data reception goes on normally.

If the wake-up event is not verified, usart_ker_ck is switched OFF again, the MCU is not woken up and remains in low-power mode, and the kernel clock request is released.

The example below shows the case of a wake-up event programmed to “address match detection” and FIFO management disabled.

Figure 864 shows the USART behavior when the wake-up event is verified.

Figure 864. Wake-up event verified (wake-up event = address match, FIFO disabled)

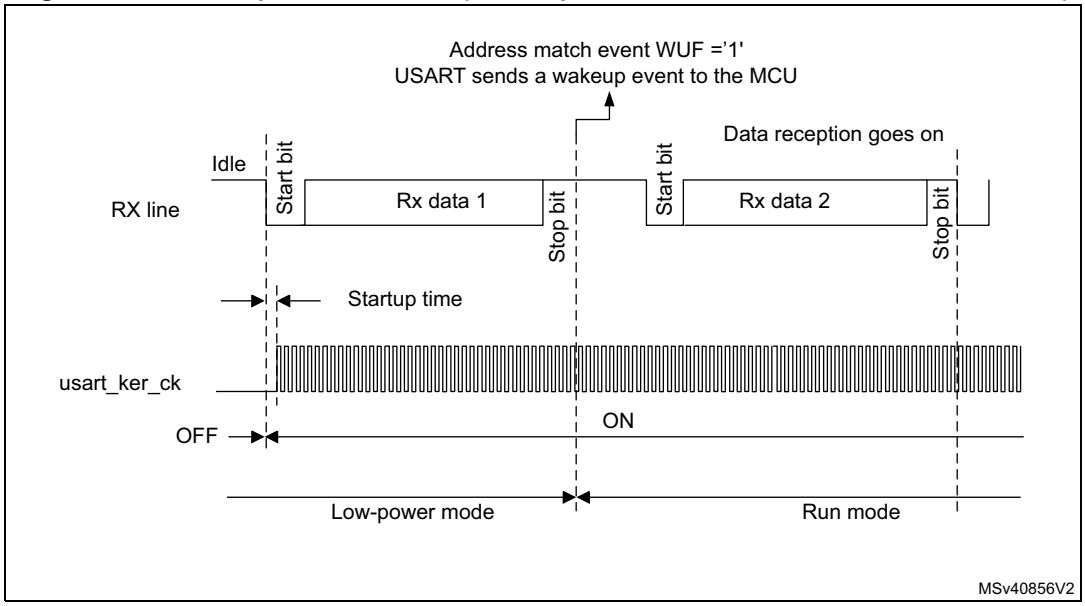
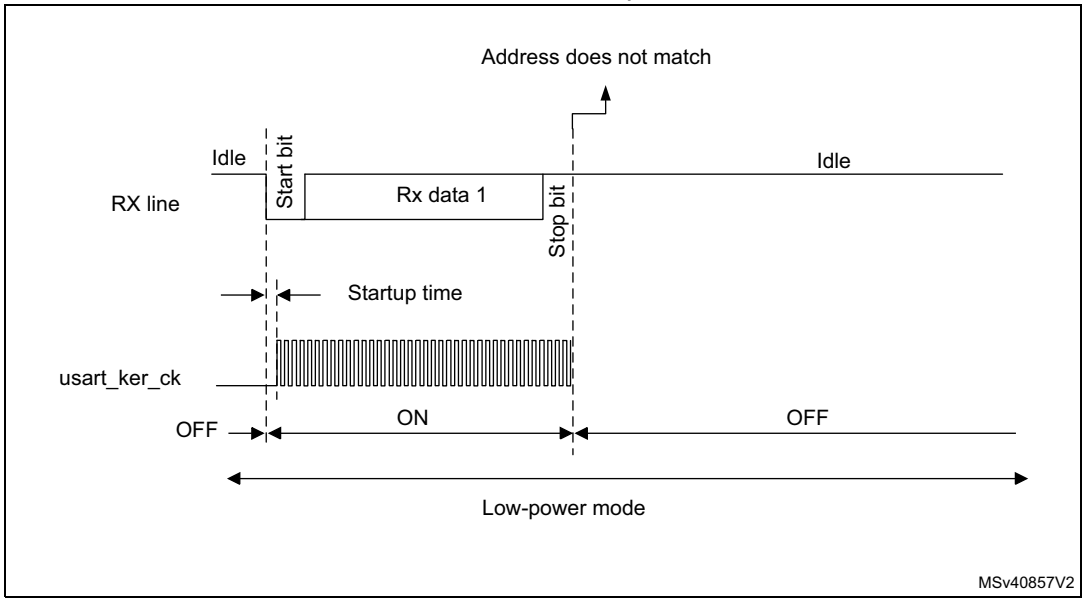


Figure 865 shows the USART behavior when the wake-up event is not verified.

Figure 865. Wake-up event not verified (wake-up event = address match, FIFO disabled)



Note: The figures above are valid when address match or any received frame is used as a wake-up event. If the wake-up event is the start bit detection, the USART sends the wake-up event to the MCU at the end of the start bit.

Determining the maximum USART baud rate that enables to correctly wake up the microcontroller from low-power mode

The maximum baud rate that enables to correctly wake up the microcontroller from low-power mode depends on the wake-up time parameter (refer to the device datasheet) and on the USART receiver tolerance (see *Section 65.5.9: Tolerance of the USART receiver to clock deviation*).

Let us take the example of OVER8 = 0, M bits = 01, ONEBIT = 0 and BRR [3:0] = 0000.

In these conditions, according to *Table 641: Tolerance of the USART receiver when BRR [3:0] = 0000*, the USART receiver tolerance equals 3.41%.

DTRA + DQUANT + DREC + DTCL + DWU < USART receiver tolerance

$D_{WUmax} = t_{WUUSART} / (11 \times T_{bitmin})$

$T_{bitmin} = t_{WUUSART} / (11 \times D_{WUmax})$

where $t_{WUUSART}$ is the wake-up time from low-power mode.

If we consider the ideal case where DTRA, DQUANT, DREC, and DTCL parameters are at 0%, the maximum value of DWU is 3.41%. In fact, we need to consider at least the usart_ker_ck inaccuracy (DREC).

For example, if HSI is used as usart_ker_ck, and the HSI inaccuracy is of 1%, then we obtain:

$t_{WUUSART}$ = 3 µs (values provided only as examples; for correct values, refer to the device datasheet).

D_{WUmax} = USART receiver tolerance – DREC = 3.41% – 1% = 2.41%

T_{bitmin} = 3 µs/ (11 x 2.41%) = 11.32 µs.

As a result, the maximum baud rate enables to wake up correctly from low-power mode is: 1/11.32 µs = 88.36 kbauds.

65.6 USART in low-power modes

Table 644. Effect of low-power modes on the USART

Mode	Description
Sleep	No effect. USART interrupts cause the device to exit Sleep mode.
Stop(1)	The content of the USART registers is kept. The USART is able to wake up the microcontroller from Stop mode when the USART is clocked by an oscillator available in Stop mode.
Standby	The USART peripheral is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 65.4: USART implementation* to know if the wake-up from Stop mode is supported for a given peripheral instance. If an instance is not functional in a given Stop mode, it must be disabled before entering this Stop mode.

65.7 USART interrupts

Refer to *Table 645* for a detailed description of all USART interrupt requests.

Table 645. USART interrupt requests

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop(1) modes	Exit from Standby mode
USART or UART	Transmit data register empty	TXE	TXEIE	Write TDR	Yes	No	No
	Transmit FIFO not full	TXFNF	TXFNFIE	TXFIFO full		No	
	Transmit FIFO empty	TXFE	TXFEIE	Write TDR or write 1 in TXFRQ		Yes	
	Transmit FIFO threshold reached	TXFT	TXFTIE	Write TDR(2)		Yes	
	CTS interrupt	CTSIF	CTSIE	Write 1 in CTSCF		No	
	Transmission complete	TC	TCIE	Write TDR or write 1 in TCCF		No	
	Transmission complete before guard time	TCBGT	TCBGTIE	Write TDR or write 1 in TCBGT		No	

Table 645. USART interrupt requests (continued)

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop(1) modes	Exit from Standby mode
USART or UART	Receive data register not empty (data ready to be read)	RXNE	RXNEIE	Read RDR or write 1 in RXFRQ	Yes	Yes	No
	Receive FIFO not empty	RXFNE	RXFNEIE	Read RDR until RXFIFO empty or write 1 in RXFRQ		Yes	
	Receive FIFO full	RXFF(3)	RXFFIE	Read RDR		Yes	
	Receive FIFO threshold reached	RXFT	RXFTIE	Read RDR		Yes	
	Overrun error detected	ORE	RX-NEIE/RX FNEIE	Write 1 in ORECF		No	
	Idle line detected	IDLE	IDLEIE	Write 1 in IDLECF		No	
	Parity error	PE	PEIE	Write 1 in PECF		No	
	LIN break	LBDF	LBDIE	Write 1 in LBDCF		No	
	Noise error in multibuffer communication.	NE	EIE	Write 1 in NFCF		No	
	Overrun error in multibuffer communication.	ORE(4)		Write 1 in ORECF		No	
	Framing error in multibuffer communication.	FE		Write 1 in FECF		No	
	Character match	CMF	CMIE	Write 1 in CMCF		No	
	Receiver timeout	RTOF	RTOFIE	Write 1 in RTOCCF		No	
	End of Block	EOBF	EOBIE	Write 1 in EOBCF		No	
	Wake-up from low-power mode	WUF	WUFIE	Write 1 in WUC		Yes	
	SPI slave underrun error	UDR	EIE	Write 1 in UDRCF		No	

1. The USART can wake up the device from Stop mode only if the peripheral instance supports the wake-up from Stop mode feature. Refer to *Section 65.4: USART implementation* for the list of supported Stop modes.
2. Writing to TDR clears the TXFT flag only if the number of empty locations is less than the value programmed in TXFTCFG[2:0].
3. RXFF flag is asserted if the USART receives n+1 data (n being the RXFIFO size): n data in the RXFIFO and 1 data in USART_RDR. In Stop mode, USART_RDR is not clocked. As a result, this register is not written and once n data are received and written in the RXFIFO, the RXFF interrupt is asserted (RXFF flag is not set).
4. When OVRDIS = 0.

65.8 USART registers

Refer to *Section 1.2* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32 bits).

65.8.1 USART control register 1 (USART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enable, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXFFIE	TXFEIE	FIFOEN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXFNFIE	TCIE	RXFNEIE	IDLEIE	TE	RE	UESM	UE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RXFFIE**: RXFIFO Full interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when RXFF = 1 in the USART_ISR register

Bit 30 **TXFEIE**: TXFIFO empty interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when TXFE = 1 in the USART_ISR register

Bit 29 **FIFOEN**: FIFO mode enable
This bit is set and cleared by software.
0: FIFO mode is disabled.
1: FIFO mode is enabled.
This bitfield can only be written when the USART is disabled (UE = 0).
Note: FIFO mode can be used on standard UART communication, in SPI master/slave mode and in smartcard modes only. It must not be enabled in IrDA and LIN modes.

Bit 28 **M1**: Word length
This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.
M[1:0] = 00: 1 start bit, 8 Data bits, n stop bits
M[1:0] = 01: 1 start bit, 9 Data bits, n stop bits
M[1:0] = 10: 1 start bit, 7 Data bits, n stop bits
This bit can only be written when the USART is disabled (UE = 0).
Note: In 7-bits data length mode, the smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.

Bit 27 **EOBIE**: End of block interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the EOBF flag is set in the USART_ISR register

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

Bit 26 **RTOIE**: Receiver timeout interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the RTOF bit is set in the USART_ISR register.

Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Section 65.4: USART implementation.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

If the USART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

Bit 15 **OVER8**: Oversampling mode

0: Oversampling by 16

1: Oversampling by 8

This bit can only be written when the USART is disabled (UE = 0).

Note: In LIN, IrDA and smartcard modes, this bit must be kept cleared.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the CMF bit is set in the USART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit enables the USART mute mode function. When set, the USART can switch between active and mute mode, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1)description).

This bit can only be written when the USART is disabled (UE = 0).

Bit 11 **WAKE**: Receiver wake-up method
This bit determines the USART wake-up method from mute mode. It is set or cleared by software.
0: Idle line
1: Address mark
This bitfield can only be written when the USART is disabled (UE = 0).

Bit 10 **PCE**: Parity control enable
This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M=0) and the parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).
0: Parity control disabled
1: Parity control enabled
This bitfield can only be written when the USART is disabled (UE = 0).

Bit 9 **PS**: Parity selection
This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.
0: Even parity
1: Odd parity
This bitfield can only be written when the USART is disabled (UE = 0).

Bit 8 **PEIE**: PE interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever PE = 1 in the USART_ISR register

Bit 7 **TXFNFIE**: TXFIFO not full interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever TXFNF = 1 in the USART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever TC = 1 in the USART_ISR register

Bit 5 **RXFNEIE**: RXFIFO not empty interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever ORE = 1 or RXFNE = 1 in the USART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever IDLE = 1 in the USART_ISR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. It is set and cleared by software.

0: Transmitter is disabled

1: Transmitter is enabled

Note: When the USART acts as a transmitter, a low pulse on the TE bit (0 followed by 1) sends a preamble (idle line) after the current word, except in smartcard mode. In order to generate an idle character, the TE must not be immediately written to 1. To ensure the required duration, the software can poll the TEACK bit in the USART_ISR register.

In smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: USART enable in low-power mode

When this bit is cleared, the USART cannot wake up the MCU from low-power mode.

When this bit is set, the USART can wake up the MCU from low-power mode.

This bit is set and cleared by software.

0: USART not able to wake up the MCU from low-power mode.

1: USART able to wake up the MCU from low-power mode.

Note: It is recommended to set the UESM bit just before entering low-power mode, and clear it when exiting low-power mode.

Bit 0 **UE**: USART enable

When this bit is cleared, the USART prescalers and outputs are stopped immediately, and all current operations are discarded. The USART configuration is kept, but all the USART_ISR status flags are reset. This bit is set and cleared by software.

0: USART prescaler and outputs disabled, low-power mode

1: USART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be previously reset and the software must wait for the TC bit in the USART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

In smartcard mode, (SCEN = 1), the CK is always available when CLKEN = 1, regardless of the UE bit value.

65.8.2 USART control register 1 [alternate] (USART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	FIFO EN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]				
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **FIFOEN**: FIFO mode enable
This bit is set and cleared by software.
0: FIFO mode is disabled.
1: FIFO mode is enabled.
This bitfield can only be written when the USART is disabled (UE = 0).
Note: FIFO mode can be used on standard UART communication, in SPI master/slave mode and in smartcard modes only. It must not be enabled in IrDA and LIN modes.

Bit 28 **M1**: Word length
This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.
M[1:0] = 00: 1 start bit, 8 Data bits, n stop bits
M[1:0] = 01: 1 start bit, 9 Data bits, n stop bits
M[1:0] = 10: 1 start bit, 7 Data bits, n stop bits
This bit can only be written when the USART is disabled (UE = 0).
Note: In 7-bits data length mode, the smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.

Bit 27 **EOBIE**: End of block interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when the EOBF flag is set in the USART_ISR register
Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

Bit 26 **RTOIE**: Receiver timeout interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when the RTOF bit is set in the USART_ISR register.
Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Section 65.4: USART implementation.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal. It is expressed in sample time units (1/8 or 1/16 bit time, depending on the oversampling rate).

If the USART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

Bit 15 **OVER8**: Oversampling mode

0: Oversampling by 16

1: Oversampling by 8

This bit can only be written when the USART is disabled (UE = 0).

Note: In LIN, IrDA and smartcard modes, this bit must be kept cleared.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated when the CMF bit is set in the USART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit enables the USART mute mode function. When set, the USART can switch between active and mute mode, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1)description).

This bit can only be written when the USART is disabled (UE = 0).

Bit 11 **WAKE**: Receiver wake-up method

This bit determines the USART wake-up method from mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M = 0) and the parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 9 **PS**: Parity selection
This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.
0: Even parity
1: Odd parity
This bitfield can only be written when the USART is disabled (UE = 0).

Bit 8 **PEIE**: PE interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever PE = 1 in the USART_ISR register

Bit 7 **TXEIE**: Transmit data register empty
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever TXE = 1 in the USART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever TC = 1 in the USART_ISR register

Bit 5 **RXNEIE**: Receive data register not empty interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: USART interrupt is generated whenever ORE = 1 or RXNE = 1 in the USART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever IDLE = 1 in the USART_ISR register

Bit 3 **TE**: Transmitter enable
This bit enables the transmitter. It is set and cleared by software.
0: Transmitter is disabled
1: Transmitter is enabled
Note: When the USART acts as a transmitter, a low pulse on the TE bit (0 followed by 1) sends a preamble (idle line) after the current word, except in smartcard mode. In order to generate an idle character, the TE must not be immediately written to 1. To ensure the required duration, the software can poll the TEACK bit in the USART_ISR register.
In smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable
This bit enables the receiver. It is set and cleared by software.
0: Receiver is disabled
1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: USART enable in low-power mode
When this bit is cleared, the USART cannot wake up the MCU from low-power mode.
When this bit is set, the USART can wake up the MCU from low-power mode.
This bit is set and cleared by software.
0: USART not able to wake up the MCU from low-power mode.
1: USART able to wake up the MCU from low-power mode.
Note: It is recommended to set the UESM bit just before entering low-power mode, and clear it when exiting low-power mode.

Bit 0 **UE**: USART enable

When this bit is cleared, the USART prescalers and outputs are stopped immediately, and all current operations are discarded. The USART configuration is kept, but all the USART_ISR status flags are reset. This bit is set and cleared by software.

0: USART prescaler and outputs disabled, low-power mode

1: USART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be previously reset and the software must wait for the TC bit in the USART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

In smartcard mode, (SCEN = 1), the CK is always available when CLKEN = 1, regardless of the UE bit value.

65.8.3 USART control register 2 (USART_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD[7:0]								RTOEN	ABRMOD[1:0]		ABREN	MSBFIRST	DATAINV	TXINV	RXINV
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWAP	LINEN	STOP[1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	ADDM7	DIS_NSS	Res.	Res.	SLVEN
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw			rw

Bits 31:24 **ADD[7:0]**: Address of the USART node

These bits give the address of the USART node in mute mode or a character code to be recognized in low-power or Run mode:

- In mute mode: they are used in multiprocessor communication to wake up from mute mode with 4-bit/7-bit address mark detection. The MSB of the character sent by the transmitter is equal to 1. In 4-bit address mark detection, only ADD[3:0] bits are used.
- In low-power mode: they are used for wake up from low-power mode on character match. When WUS[1:0] is programmed to 0b00 (WUF active on address match), the wake-up from low-power mode is performed when the received character corresponds to the character programmed through ADD[6:0] or ADD[3:0] bitfield (depending on ADDM7 bit), and WUF interrupt is enabled by setting WUFIE bit. The MSB of the character sent by transmitter is equal to 1.
- In Run mode with mute mode inactive (for example, end-of-block detection in ModBus protocol): the whole received character (8 bits) is compared to ADD[7:0] value and CMF flag is set on match. An interrupt is generated if the CMIE bit is set.

These bits can only be written when the USART is disabled (UE = 0).

Bit 23 **RTOEN**: Receiver timeout enable

This bit is set and cleared by software.

0: Receiver timeout feature disabled.

1: Receiver timeout feature enabled.

When this feature is enabled, the RTOF flag in the USART_ISR register is set if the RX line is idle (no reception) for the duration programmed in the RTOR (receiver timeout register).

Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

Bits 22:21 **ABRMOD[1:0]**: Auto baud rate mode

These bits are set and cleared by software.

00: Measurement of the start bit is used to detect the baud rate.

01: Falling edge to falling edge measurement (the received frame must start with a single bit = 1 -> Frame = Start10xxxxxx)

10: 0x7F frame detection.

11: 0x55 frame detection

This bitfield can only be written when ABREN = 0 or the USART is disabled (UE = 0).

Note: If DATAINV = 1 and/or MSBFIRST = 1 the patterns must be the same on the line, for example 0xAA for MSBFIRST)

If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

Bit 20 **ABREN**: Auto baud rate enable

This bit is set and cleared by software.

0: Auto baud rate detection is disabled.

1: Auto baud rate detection is enabled.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

Bit 19 **MSBFIRST**: Most significant bit first

This bit is set and cleared by software.

0: data is transmitted/received with data bit 0 first, following the start bit.

1: data is transmitted/received with the MSB (bit 7/8) first, following the start bit.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 18 **DATAINV:** Binary data inversion

This bit is set and cleared by software.

0: Logical data from the data register are send/received in positive/direct logic. (1 = H, 0 = L)

1: Logical data from the data register are send/received in negative/inverse logic. (1 = L, 0 = H).

The parity bit is also inverted.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 17 **TXINV:** TX pin active level inversion

This bit is set and cleared by software.

0: TX pin signal works using the standard logic levels (V_{DD} = 1 / idle, Gnd = 0 / mark)

1: TX pin signal values are inverted. ((V_{DD} = 0 / mark, Gnd = 1 / idle).

This enables the use of an external inverter on the TX line.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 16 **RXINV:** RX pin active level inversion

This bit is set and cleared by software.

0: RX pin signal works using the standard logic levels (V_{DD} = 1 / idle, Gnd = 0 / mark)

1: RX pin signal values are inverted. ((V_{DD} = 0 / mark, Gnd = 1 / idle).

This enables the use of an external inverter on the RX line.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 15 **SWAP:** Swap TX/RX pins

This bit is set and cleared by software.

0: TX/RX pins are used as defined in standard pinout

1: The TX and RX pins functions are swapped. This enables to work in the case of a cross-wired connection to another UART.

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 14 **LINEN**: LIN mode enable

This bit is set and cleared by software.

0: LIN mode disabled

1: LIN mode enabled

The LIN mode enables the capability to send LIN synchronous breaks (13 low bits) using the SBKRQ bit in the USART_CR1 register, and to detect LIN Sync breaks.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support LIN mode, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

Bits 13:12 **STOP[1:0]**: Stop bits

These bits are used for programming the stop bits.

00: 1 stop bit

01: 0.5 stop bit.

10: 2 stop bits

11: 1.5 stop bits

This bitfield can only be written when the USART is disabled (UE = 0).

Bit 11 **CLKEN**: Clock enable

This bit enables the user to enable the CK pin.

0: CK pin disabled

1: CK pin enabled

Note: If neither synchronous mode nor smartcard mode is supported, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

In smartcard mode, in order to provide correctly the CK clock to the smartcard, the steps below must be respected:
UE = 0
SCEN = 1
GTPR configuration
CLKEN= 1
UE = 1

Bit 10 **CPOL**: Clock polarity

This bit enables the user to select the polarity of the clock output on the CK pin in synchronous mode. It works in conjunction with the CPHA bit to produce the desired clock/data relationship

0: Steady low value on CK pin outside transmission window

1: Steady high value on CK pin outside transmission window

This bit can only be written when the USART is disabled (UE = 0).

Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

Bit 9 **CPHA**: Clock phase

This bit is used to select the phase of the clock output on the CK pin in synchronous mode. It works in conjunction with the CPOL bit to produce the desired clock/data relationship (see *Figure 845* and *Figure 846*)

0: The first clock transition is the first data capture edge

1: The second clock transition is the first data capture edge

This bit can only be written when the USART is disabled (UE = 0).

Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

Bit 8 **LBCL**: Last bit clock pulse

This bit is used to select whether the clock pulse associated with the last data bit transmitted (MSB) has to be output on the CK pin in synchronous mode.

0: The clock pulse of the last data bit is not output to the CK pin

1: The clock pulse of the last data bit is output to the CK pin

Caution: The last bit is the 7th or 8th or 9th data bit transmitted depending on the 7 or 8 or 9 bit format selected by the M bit in the USART_CR1 register.

This bit can only be written when the USART is disabled (UE = 0).

Note: If synchronous mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **LBDIE**: LIN break detection interrupt enable

Break interrupt mask (break detection using break delimiter).

0: Interrupt is inhibited

1: An interrupt is generated whenever LBDF = 1 in the USART_ISR register

Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

Bit 5 **LBDL**: LIN break detection length

This bit is for selection between 11 bit or 10 bit break detection.

0: 10-bit break detection

1: 11-bit break detection

This bit can only be written when the USART is disabled (UE = 0).

Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

Bit 4 **ADDM7**: 7-bit address detection/4-bit address detection

This bit is for selection between 4-bit address detection or 7-bit address detection.

0: 4-bit address detection

1: 7-bit address detection (in 8-bit data mode)

This bit can only be written when the USART is disabled (UE = 0)

Note: In 7-bit and 9-bit data modes, the address detection is done on 6-bit and 8-bit address (ADD[5:0] and ADD[7:0]) respectively.

Bit 3 **DIS_NSS**:

When the DIS_NSS bit is set, the NSS pin input is ignored.

0: SPI slave selection depends on NSS input pin.

1: SPI slave is always selected and NSS input pin is ignored.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: When SPI slave mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **SLVEN**: Synchronous slave mode enable
When the SLVEN bit is set, the synchronous slave mode is enabled.
0: Slave mode disabled.
1: Slave mode enabled.
This bitfield can only be written when the USART is disabled (UE = 0).
Note: When SPI slave mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

Note: The CPOL, CPHA, and LBCL bits must not be written while the transmitter is enabled.

65.8.4 USART control register 3 (USART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

FIFO mode enabled, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXFTCFG[2:0]			RXF TIE	RXFTCFG[2:0]			TCBG TIE	TXFTIE	WUFIE	WUS1	WUS0	SCARCNT[2:0]			Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVR DIS	ONE BIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HD SEL	IRLP	IREN	EIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 **TXFTCFG[2:0]:** TXFIFO threshold configuration
000:TXFIFO reaches 1/8 of its depth
001:TXFIFO reaches 1/4 of its depth
010:TXFIFO reaches 1/2 of its depth
011:TXFIFO reaches 3/4 of its depth
100:TXFIFO reaches 7/8 of its depth
101:TXFIFO becomes empty
Others: Reserved, must not be used
This bitfield can only be written when the USART is disabled (UE = 0).

Bit 28 **RXFTIE**: RXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when Receive FIFO reaches the threshold programmed in RXFTCFG[2:0].

Bits 27:25 **RXFTCFG[2:0]:** Receive FIFO threshold configuration
000:Receive FIFO reaches 1/8 of its depth
001:Receive FIFO reaches 1/4 of its depth
010:Receive FIFO reaches 1/2 of its depth
011:Receive FIFO reaches 3/4 of its depth
100:Receive FIFO reaches 7/8 of its depth
101:Receive FIFO becomes full
Others: Reserved, must not be used
This bitfield can only be written when the USART is disabled (UE = 0).

Bit 24 **TCBGTIE**: Transmission Complete before guard time, interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever TCBGT = 1 in the USART_ISR register
Note: If the USART does not support the smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

Bit 23 **TXFTIE**: TXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated when TXFIFO reaches the threshold programmed in TXFTCFG[2:0].

Bit 22 **WUFIE**: Wake-up from low-power mode interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: USART interrupt generated whenever WUF = 1 in the USART_ISR register
Note: WUFIE must be set before entering in low-power mode.
If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

Bits 21:20 **WUS[1:0]**: Wake-up from low-power mode interrupt flag selection
This bitfield specifies the event which activates the WUF (wake-up from low-power mode flag).
00: WUF active on address match (as defined by ADD[7:0] and ADDM7)
01: Reserved.
10: WUF active on start bit detection
11: WUF active on RXNE/RXFNE.
This bitfield can only be written when the USART is disabled (UE = 0).
Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

Bits 19:17 **SCARCNT[2:0]**: Smartcard auto-retry count
This bitfield specifies the number of retries for transmission and reception in smartcard mode.
In transmission mode, it specifies the number of automatic retransmission retries, before generating a transmission error (FE bit set).
In reception mode, it specifies the number or erroneous reception trials, before generating a reception error (RXNE/RXFNE and PE bits set).
This bitfield must be programmed only when the USART is disabled (UE = 0).
When the USART is enabled (UE = 1), this bitfield may only be written to 0x0, in order to stop retransmission.
0x0: retransmission disabled - No automatic retransmission in transmission mode.
0x1 to 0x7: number of automatic retransmission attempts (before signaling error)
Note: If smartcard mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection
0: DE signal is active high.
1: DE signal is active low.
This bit can only be written when the USART is disabled (UE = 0).
Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

Bit 14 **DEM**: Driver enable mode

This bit enables the user to activate the external transceiver control, through the DE signal.

0: DE function is disabled.

1: DE function is enabled. The DE signal is output on the RTS pin.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Section 65.4: USART implementation on page 3281.

Bit 13 **DDRE**: DMA Disable on reception Error

0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred. (used for smartcard mode)

1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE(RXFNE is case FIFO mode is enabled) before clearing the error flag.

This bit can only be written when the USART is disabled (UE = 0).

Note: The reception errors are: parity error, framing error or noise error.

Bit 12 **OVRDIS**: Overrun Disable

This bit is used to disable the receive overrun detection.

0: Overrun Error Flag, ORE, is set when received data is not read before receiving new data.

1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the USART_RDR register. When FIFO mode is enabled, the RXFIFO is bypassed and data are written directly in USART_RDR register. Even when FIFO management is enabled, the RXNE flag is to be used.

This bit can only be written when the USART is disabled (UE = 0).

Note: This control bit enables *checking the communication flow w/o reading the data*

Bit 11 **ONEBIT**: One sample bit method enable

This bit enables the user to select the sample method. When the one sample bit method is selected the noise detection flag (NE) is disabled.

0: Three sample bit method

1: One sample bit method

This bit can only be written when the USART is disabled (UE = 0).

Bit 10 **CTSIE**: CTS interrupt enable

0: Interrupt is inhibited

1: An interrupt is generated whenever CTSIF = 1 in the USART_ISR register

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

Bit 9 **CTSE**: CTS enable

0: CTS hardware flow control disabled

1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0). If the CTS input is asserted while data is being transmitted, then the transmission completes before stopping. If data is written into the data register while CTS is asserted, the transmission is postponed until CTS is deasserted.

This bit can only be written when the USART is disabled (UE = 0)

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

Bit 8 **RTSE**: RTS enable

0: RTS hardware flow control disabled

1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The RTS output is deasserted (pulled to 0) when data can be received.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

Bit 7 **DMAT**: DMA enable transmitter

This bit is set/reset by software

1: DMA mode is enabled for transmission

0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver

This bit is set/reset by software

1: DMA mode is enabled for reception

0: DMA mode is disabled for reception

Bit 5 **SCEN**: Smartcard mode enable

This bit is used for enabling smartcard mode.

0: Smartcard mode disabled

1: Smartcard mode enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

Bit 4 **NACK**: Smartcard NACK enable

0: NACK transmission in case of parity error is disabled

1: NACK transmission during parity error is enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

Bit 3 **HDSEL**: Half-duplex selection

Selection of single-wire half-duplex mode

0: Half-duplex mode is not selected

1: Half-duplex mode is selected

This bit can only be written when the USART is disabled (UE = 0).

Bit 2 **IRLP**: IrDA low-power

This bit is used for selecting between normal and low-power IrDA modes

0: Normal mode

1: Low-power mode

This bit can only be written when the USART is disabled (UE = 0).

Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

Bit 1 **IREN**: IrDA mode enable

This bit is set and cleared by software.

0: IrDA disabled

1: IrDA enabled

This bit can only be written when the USART is disabled (UE = 0).

Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

Bit 0 **EIE**: Error interrupt enable

Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error noise flag or SPI slave underrun error (FE = 1 or ORE = 1 or NE = 1 or UDR = 1 in the USART_ISR register).

0: Interrupt inhibited

1: interrupt generated when FE = 1 or ORE = 1 or NE = 1 or UDR = 1 (in SPI slave mode) in the USART_ISR register.

65.8.5 USART control register 3 [alternate] (USART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCBGTIE	Res.	WUFIE	WUS1	WUS0	SCARCNT[2:0]			Res.
							rw		rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVRDIS	ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **TCBGTIE**: Transmission Complete before guard time, interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever TCBGT = 1 in the USART_ISR register

Note: If the USART does not support the smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

Bit 23 Reserved, must be kept at reset value.

Bit 22 **WUFIE**: Wake-up from low-power mode interrupt enable

This bit is set and cleared by software.

0: Interrupt inhibited

1: USART interrupt generated whenever WUF = 1 in the USART_ISR register

Note: WUFIE must be set before entering in low-power mode.

If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

Bits 21:20 **WUS[1:0]**: Wake-up from low-power mode interrupt flag selection

This bitfield specifies the event which activates the WUF (wake-up from low-power mode flag).

00: WUF active on address match (as defined by ADD[7:0] and ADDM7)

01: Reserved.

10: WUF active on start bit detection

11: WUF active on RXNE/RXFNE.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

Bits 19:17 **SCARCNT[2:0]**: Smartcard auto-retry count

This bitfield specifies the number of retries for transmission and reception in smartcard mode.

In transmission mode, it specifies the number of automatic retransmission retries, before generating a transmission error (FE bit set).

In reception mode, it specifies the number or erroneous reception trials, before generating a reception error (RXNE/RXFNE and PE bits set).

This bitfield must be programmed only when the USART is disabled (UE = 0).

When the USART is enabled (UE = 1), this bitfield may only be written to 0x0, in order to stop retransmission.

0x0: retransmission disabled - No automatic retransmission in transmission mode.

0x1 to 0x7: number of automatic retransmission attempts (before signaling error)

Note: If smartcard mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection

0: DE signal is active high.

1: DE signal is active low.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

Bit 14 **DEM**: Driver enable mode

This bit enables the user to activate the external transceiver control, through the DE signal.

0: DE function is disabled.

1: DE function is enabled. The DE signal is output on the RTS pin.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the Driver Enable feature is not supported, this bit is reserved and must be kept at reset value. Section 65.4: USART implementation.

Bit 13 **DDRE**: DMA Disable on reception Error

0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred. (used for smartcard mode)

1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE(RXFNE is case FIFO mode is enabled) before clearing the error flag.

This bit can only be written when the USART is disabled (UE = 0).

Note: The reception errors are: parity error, framing error or noise error.

Bit 12 **OVRDIS**: Overrun Disable

This bit is used to disable the receive overrun detection.

0: Overrun Error Flag, ORE, is set when received data is not read before receiving new data.

1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the USART_RDR register. When FIFO mode is enabled, the RXFIFO is bypassed and data are written directly in USART_RDR register. Even when FIFO management is enabled, the RXNE flag is to be used.

This bit can only be written when the USART is disabled (UE = 0).

Note: This control bit enables *checking the communication flow w/o reading the data*

Bit 11 **ONEBIT**: One sample bit method enable

This bit enables the user to select the sample method. When the one sample bit method is selected the noise detection flag (NE) is disabled.

0: Three sample bit method

1: One sample bit method

This bit can only be written when the USART is disabled (UE = 0).

Bit 10 **CTSIE**: CTS interrupt enable

0: Interrupt is inhibited

1: An interrupt is generated whenever CTSIF = 1 in the USART_ISR register

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

Bit 9 **CTSE**: CTS enable

0: CTS hardware flow control disabled

1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0). If the CTS input is asserted while data is being transmitted, then the transmission completes before stopping. If data is written into the data register while CTS is asserted, the transmission is postponed until CTS is deasserted.

This bit can only be written when the USART is disabled (UE = 0)

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

Bit 8 **RTSE**: RTS enable

0: RTS hardware flow control disabled

1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The RTS output is deasserted (pulled to 0) when data can be received.

This bit can only be written when the USART is disabled (UE = 0).

Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

Bit 7 **DMAT**: DMA enable transmitter

This bit is set/reset by software

1: DMA mode is enabled for transmission

0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver

This bit is set/reset by software

1: DMA mode is enabled for reception

0: DMA mode is disabled for reception

Bit 5 **SCEN**: Smartcard mode enable

This bit is used for enabling smartcard mode.

0: Smartcard mode disabled

1: Smartcard mode enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

Bit 4 **NACK**: Smartcard NACK enable

0: NACK transmission in case of parity error is disabled

1: NACK transmission during parity error is enabled

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

Bit 3 **HDSEL**: Half-duplex selection

Selection of single-wire half-duplex mode

0: Half-duplex mode is not selected

1: Half-duplex mode is selected

This bit can only be written when the USART is disabled (UE = 0).

Bit 2 **IRLP**: IrDA low-power

This bit is used for selecting between normal and low-power IrDA modes

0: Normal mode

1: Low-power mode

This bit can only be written when the USART is disabled (UE = 0).

Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

Bit 1 **IREN**: IrDA mode enable

This bit is set and cleared by software.

0: IrDA disabled

1: IrDA enabled

This bit can only be written when the USART is disabled (UE = 0).

Note: If IrDA mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

Bit 0 **EIE**: Error interrupt enable

Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error noise flag or SPI slave underrun error (FE = 1 or ORE = 1 or NE = 1or UDR = 1 in the USART_ISR register).

0: Interrupt inhibited

1: interrupt generated when FE = 1 or ORE = 1 or NE = 1 or UDR = 1 (in SPI slave mode) in the USART_ISR register.

65.8.6 USART baud rate register (USART_BRR)

This register can only be written when the USART is disabled (UE = 0). It may be automatically updated by hardware in auto baud rate detection mode.

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BRR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **BRR[15:0]:** USART baud rate

BRR[15:4]

BRR[15:4] correspond to USARTDIV[15:4]

BRR[3:0]

When OVER8 = 0, BRR[3:0] = USARTDIV[3:0].

When OVER8 = 1:

BRR[2:0] = USARTDIV[3:0] shifted 1 bit to the right.

BRR[3] must be kept cleared.

65.8.7 USART guard time and prescaler register (USART_GTPR)

Address offset: 0x10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GT[7:0]								PSC[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **GT[7:0]**: Guard time value

This bitfield is used to program the Guard time value in terms of number of baud clock periods.

This is used in smartcard mode. The transmission complete flag is set after this guard time value.

This bitfield can only be written when the USART is disabled (UE = 0).

Note: If smartcard mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

Bits 7:0 **PSC[7:0]**: Prescaler value

Condition: IrDA low-power and normal IrDA mode

PSC[7:0] = IrDA normal and Low-power baud rate

This bitfield is used for programming the prescaler for dividing the USART source clock to achieve the low-power frequency:

The source clock is divided by the value given in the register (8 significant bits):

00000000: Reserved - do not program this value

00000001: divides the source clock by 1

00000010: divides the source clock by 2

...

Condition: Smartcard mode

PSC[4:0]: Prescaler value

This bitfield is used for programming the prescaler for dividing the USART source clock to provide the smartcard clock.

The value given in the register (5 significant bits) is multiplied by 2 to give the division factor of the source clock frequency:

00000: Reserved - do not program this value

00001: divides the source clock by 2

00010: divides the source clock by 4

00011: divides the source clock by 6

...

This bitfield can only be written when the USART is disabled (UE = 0).

Note: Bits [7:5] must be kept cleared if smartcard mode is used.

This bitfield is reserved and forced by hardware to 0 when the smartcard and IrDA modes are not supported. Refer to Section 65.4: USART implementation.

65.8.8 USART receiver timeout register (USART_RTOR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BLEN[7:0]								RTO[23:16]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTO[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **BLEN[7:0]**: Block Length

This bitfield gives the block length in smartcard T = 1 reception. Its value equals the number of information characters + the length of the Epilogue Field (1 – LEC / 2 – CRC) – 1.

Examples:

BLEN = 0 -> 0 information characters + LEC

BLEN = 1 -> 0 information characters + CRC

BLEN = 255 -> 254 information characters + CRC (total 256 characters))

In smartcard mode, the block length counter is reset when TXE = 0 (TXFE = 0 in case FIFO mode is enabled).

This bitfield can be used also in other modes. In this case, the block length counter is reset when RE = 0 (receiver disabled) and/or when the EOBCF bit is written to 1.

Note: This value can be programmed after the start of the block reception (using the data from the LEN character in the Prologue Field). It must be programmed only once per received block.

Bits 23:0 **RTO[23:0]**: Receiver timeout value

This bitfield gives the Receiver timeout value in terms of number of bit duration.

In Standard mode, the RTOF flag is set if, after the last received character, no new start bit is detected for more than the RTO value.

In smartcard mode, this value is used to implement the CWT and BWT. See smartcard chapter for more details. In the standard, the CWT/BWT measurement is done starting from the start bit of the last received character.

Note: This value must only be programmed once per received character.

Note: RTOR can be written on-the-fly. If the new value is lower than or equal to the counter, the RTOF flag is set.

This register is reserved and forced by hardware to "0x00000000" when the receiver timeout feature is not supported. Refer to Section 65.4: USART implementation.

65.8.9 USART request register (USART_RQR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	ABRRQ
											w	w	w	w	w

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **TXFRQ**: Transmit data flush request

When FIFO mode is disabled, writing 1 to this bit sets the TXE flag. This enables to discard the transmit data. This bit must be used only in smartcard mode, when data have not been sent due to errors (NACK) and the FE flag is active in the USART_ISR register. If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value.

When FIFO is enabled, TXFRQ bit is set to flush the whole FIFO. This sets the TXFE flag (Transmit FIFO empty, bit 23 in the USART_ISR register). Flushing the Transmit FIFO is supported in both UART and smartcard modes.

Note: In FIFO mode, the TXFNF flag is reset during the flush request until TxFIFO is empty in order to ensure that no data are written in the data register.

Bit 3 **RXFRQ**: Receive data flush request

Writing 1 to this bit empties the entire receive FIFO, that is clears the bit RXFNE.

This enables to discard the received data without reading them, and avoid an overrun condition.

Bit 2 **MMRQ**: Mute mode request

Writing 1 to this bit puts the USART in mute mode and resets the RWU flag.

Bit 1 **SBKRQ**: Send break request

Writing 1 to this bit sets the SBKF flag and request to send a BREAK on the line, as soon as the transmit machine is available.

Note: When the application needs to send the break character following all previously inserted data, including the ones not yet transmitted, the software must wait for the TXE flag assertion before setting the SBKRQ bit.

Bit 0 **ABRRQ**: Auto baud rate request

Writing 1 to this bit resets the ABRF and ABRE flags in the USART_ISR and requests an automatic baud rate measurement on the next received data frame.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

65.8.10 USART interrupt and status register (USART_ISR)

Address offset: 0x1C

Reset value: 0x0XX0 00C0

XX = 28 if FIFO/smartcard mode supported

XX = 08 if FIFO supported and smartcard mode not supported

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enabled, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TXFT	RXFT	TCBGT	RXFF	TXFE	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXFNF	TC	RXFNE	IDLE	ORE	NE	FE	PE
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **TXFT**: TXFIFO threshold flag

This bit is set by hardware when the number of empty locations in the TXFIFO is greater than the threshold programmed in the TXFTCFG[2:0] bitfield of USART_CR3 register.
An interrupt is generated if the TXFTIE bit (bit 31) is set in the USART_CR3 register.
0: TXFIFO does not reach the programmed threshold.
1: TXFIFO reached the programmed threshold.

Bit 26 **RXFT**: RXFIFO threshold flag

This bit is set by hardware when the threshold programmed in the RXFTCFG[2:0] bitfield of the USART_CR3 register is reached. This means that there are (RXFTCFG[2:0] – 1) data in the Receive FIFO and one data in the USART_RDR register. An interrupt is generated if the RXFTIE bit = 1 (bit 28) in the USART_CR3 register.
0: Receive FIFO does not reach the programmed threshold.
1: Receive FIFO reached the programmed threshold.

Note: When the RXFTCFG[2:0] threshold is configured to 101, the RXFT flag is set if RXFIFO size data are available, that is, (RXFIFO size – 1) data in the RXFIFO and 1 data in the USART_RDR. Consequently, the (RXFIFO size + 1) th received data does not cause an overrun error. The overrun error occurs after receiving the (RXFIFO size + 2) th data.

Bit 25 **TCBGT:** Transmission complete before guard time flag

This bit is set when the last data written in the USART_TDR has been transmitted correctly out of the shift register.
It is set by hardware in smartcard mode, if the transmission of a frame containing data has completed and if the smartcard did not send back any NACK. An interrupt is generated if TCBGTIE = 1 in the USART_CR3 register.
This bit is cleared by software, by writing 1 to the TCBGTCF in the USART_ICR register or by a write to the USART_TDR register.
0: Transmission has not completed, or transmission has completed unsuccessfully (that is, a NACK is received from the card)
1: Transmission has completed successfully (before Guard time completion and there is no NACK from the smart card).

Note: If the USART does not support the smartcard mode, this bit is reserved and kept at reset value. If the USART supports the smartcard mode and the smartcard mode is enabled, the TCBGT reset value is 1. Refer to Section 65.4: USART implementation.

Bit 24 **RXFF**: RXFIFO Full

This bit is set by hardware when the number of received data corresponds to RXFIFO size + 1 (RXFIFO full + 1 data in the USART_RDR register.
An interrupt is generated if the RXFFIE bit = 1 in the USART_CR1 register.
0: RXFIFO not full.
1: RXFIFO Full.

Bit 23 **TXFE**: TXFIFO Empty

This bit is set by hardware when TXFIFO is Empty. When the TXFIFO contains at least one data, this flag is cleared. The TXFE flag can also be set by writing 1 to the bit TXFRQ (bit 4) in the USART_RQR register.
An interrupt is generated if the TXFEIE bit = 1 (bit 30) in the USART_CR1 register.
0: TXFIFO not empty.
1: TXFIFO empty.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the USART.

It can be used to verify that the USART is ready for reception before entering low-power mode.

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 65.4: USART implementation.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the USART.

It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the USART_CR1 register, in order to respect the TE = 0 minimum period.

Bit 20 **WUF**: Wake-up from low-power mode flag

This bit is set by hardware, when a wake-up event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the USART_ICR register.

An interrupt is generated if WUFIE = 1 in the USART_CR3 register.

Note: When UESM is cleared, WUF flag is also cleared.

If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 65.4: USART implementation.

Bit 19 **RWU**: Receiver wake-up from mute mode

This bit indicates if the USART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the USART_CR1 register.

When wake-up on idle mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the USART_RQR register.

0: Receiver in active mode

1: Receiver in mute mode

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 65.4: USART implementation.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the USART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: No break character transmitted

1: Break character transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the USART_ICR register.

An interrupt is generated if CMIE = 1in the USART_CR1 register.

0: No Character match detected

1: Character match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: USART is idle (no reception)

1: Reception ongoing

Bit 15 **ABRF:** Auto baud rate flag

This bit is set by hardware when the automatic baud rate has been set (RXFNE is also set, generating an interrupt if RXFNEIE = 1) or when the auto baud rate operation has completed without success (ABRE = 1) (ABRE, RXFNE and FE are also set in this case)

It is cleared by software, in order to request a new auto baud rate detection, by writing 1 to the ABRRQ in the USART_RQR register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 14 **ABRE**: Auto baud rate error

This bit is set by hardware if the baud rate measurement failed (baud rate out of range or character comparison failed)

It is cleared by software, by writing 1 to the ABRRQ bit in the USART_RQR register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 13 **UDR**: SPI slave underrun error flag

In slave transmission mode, this flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value into USART_TDR. This flag is reset by setting UDRCF bit in the USART_ICR register.

0: No underrun error

1: underrun error

Note: If the USART does not support the SPI slave mode, this bit is reserved and kept at reset value. Refer to Section 65.4: USART implementation.

Bit 12 **EOBF**: End of block flag

This bit is set by hardware when a complete block has been received (for example T = 1 smartcard mode). The detection is done when the number of received bytes (from the start of the block, including the prologue) is equal or greater than BLEN + 4.

An interrupt is generated if EOBIE = 1 in the USART_CR1 register.

It is cleared by software, writing 1 to EOBCF in the USART_ICR register.

0: End of block not reached

1: End of block (number of characters) reached

Note: If smartcard mode is not supported, this bit is reserved and kept at reset value. Refer to Section 65.4: USART implementation.

Bit 11 **RTOF**: Receiver timeout

This bit is set by hardware when the timeout value, programmed in the RTOR register has lapsed, without any communication. It is cleared by software, writing 1 to the RTOCF bit in the USART_ICR register.

An interrupt is generated if RTOIE = 1 in the USART_CR2 register.

In smartcard mode, the timeout corresponds to the CWT or BWT timings.

0: Timeout value not reached

1: Timeout value reached without any data reception

Note: If a time equal to the value programmed in RTOR register separates 2 characters, RTOF is not set. If this time exceeds this value + 2 sample times (2/16 or 2/8, depending on the oversampling method), RTOF flag is set.

The counter counts even if RE = 0 but RTOF is set only when RE = 1. If the timeout has already elapsed when RE is set, then RTOF is set.

If the USART does not support the Receiver timeout feature, this bit is reserved and kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.

0: CTS line set

1: CTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the USART_ICR register.

An interrupt is generated if CTSIE = 1 in the USART_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 **LBDF**: LIN break detection flag

This bit is set by hardware when the LIN break is detected. It is cleared by software, by writing 1 to the LBDCF in the USART_ICR.

An interrupt is generated if LBDIE = 1 in the USART_CR2 register.

0: LIN Break not detected

1: LIN break detected

Note: If the USART does not support LIN mode, this bit is reserved and kept at reset value. Refer to Section 65.4: USART implementation.

Bit 7 **TXFNF:** TXFIFO not full

TXFNF is set by hardware when TXFIFO is not full meaning that data can be written in the USART_TDR. Every write operation to the USART_TDR places the data in the TXFIFO. This flag remains set until the TXFIFO is full. When the TXFIFO is full, this flag is cleared indicating that data can not be written into the USART_TDR.

An interrupt is generated if the TXFNFIE bit = 1 in the USART_CR1 register.

0: Transmit FIFO is full

1: Transmit FIFO is not full

Note: The TXFNF is kept reset during the flush request until TXFIFO is empty. After sending the flush request (by setting TXFRQ bit), the flag TXFNF must be checked prior to writing in TXFIFO (TXFNF and TXFE is set at the same time).

This bit is used during single buffer transmission.

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the USART_TDR has been transmitted out of the shift register.

It is set by hardware when the transmission of a frame containing data has completed, and the TXFE bit is set.

An interrupt is generated if TCIE = 1 in the USART_CR1 register.

The TC bit is cleared by software, by writing 1 to the TCCF of the USART_ICR register, or by a write to the USART_TDR register.

0: Transmission has not completed

1: Transmission has completed

Note: If the TE bit is reset and no transmission is ongoing, the TC bit is immediately set.

Bit 5 **RXFNE**: RXFIFO not empty

RXFNE bit is set by hardware when the RXFIFO is not empty, meaning that data can be read from the USART_RDR register. Every read operation from the USART_RDR frees a location in the RXFIFO.

RXFNE is cleared when the RXFIFO is empty. The RXFNE flag can also be cleared by writing 1 to the RXFRQ in the USART_RQR register.

An interrupt is generated if RXFNEIE = 1 in the USART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE = 1 in the USART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the USART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit is not set again until the RXFNE bit has been set (that is, a new idle line occurs).

If mute mode is enabled (MME = 1), IDLE is set if the USART is not mute (RWU = 0), whatever the mute mode selected by the WAKE bit. If RWU = 1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the USART_RDR register while RXFF = 1. It is cleared by a software, writing 1 to the ORECF, in the USART_ICR register.

An interrupt is generated if RXFNEIE = 1 in the USART_CR1 register, or EIE = 1 in the USART_CR3 register.

0: No overrun error

1: Overrun error is detected

Note: When this bit is set, the USART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the USART_CR3 register.

Bit 2 **NE**: Noise detection flag

This bit is set by hardware when noise is detected on a received frame. It is cleared by software, writing 1 to the NFCF bit in the USART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXFNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

When the line is noise-free, the NE flag can be disabled by programming the ONEBIT bit to 1 to increase the USART tolerance to deviations (Refer to Section 65.5.9: Tolerance of the USART receiver to clock deviation).

This error is associated with the character in the USART_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the USART_ICR register.
When transmitting data in smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).
An interrupt is generated if EIE = 1 in the USART_CR3 register.
0: No Framing error is detected
1: Framing error or break character is detected

Note: This error is associated with the character in the USART_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in reception mode. It is cleared by software, writing 1 to the PECF in the USART_ICR register.
An interrupt is generated if PEIE = 1 in the USART_CR1 register.
0: No parity error
1: Parity error

Note: This error is associated with the character in the USART_RDR.

65.8.11 USART interrupt and status register [alternate] (USART_ISR)

Address offset: 0x1C

Reset value: 0x0XX0 00C0

XX = 28 if FIFO/smartcard mode supported

XX = 08 if FIFO supported and smartcard mode not supported)

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	TCBGT	Res.	Res.	RE ACK	TE ACK	WUF	RWU	SBKF	CMF	BUSY
						r			r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **TCBGT:** Transmission complete before guard time flag

This bit is set when the last data written in the USART_TDR has been transmitted correctly out of the shift register.

It is set by hardware in smartcard mode, if the transmission of a frame containing data has completed, and if the smartcard did not send back any NACK. An interrupt is generated if TCBGTIE = 1 in the USART_CR3 register.

This bit is cleared by software, by writing 1 to the TCBGTCF in the USART_ICR register or by a write to the USART_TDR register.

0: Transmission has not completed or transmission has completed unsuccessfully (that is, a NACK is received from the card)

1: Transmission has not completed successfully (before Guard time completion and there is no NACK from the smart card).

Note: If the USART does not support the smartcard mode, this bit is reserved and kept at reset value. If the USART supports the smartcard mode and the smartcard mode is enabled, the TCBGT reset value is 1. Refer to Section 65.4: USART implementation.

Bits 24:23 Reserved, must be kept at reset value.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the USART.

It can be used to verify that the USART is ready for reception before entering low-power mode.

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 65.4: USART implementation.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the USART.

It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the USART_CR1 register, in order to respect the TE = 0 minimum period.

Bit 20 **WUF**: Wake-up from low-power mode flag

This bit is set by hardware, when a wake-up event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the USART_ICR register.

An interrupt is generated if WUFIE = 1 in the USART_CR3 register.

Note: When UESM is cleared, WUF flag is also cleared.

If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 65.4: USART implementation.

Bit 19 **RWU**: Receiver wake-up from mute mode

This bit indicates if the USART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the USART_CR1 register.

When wake-up on idle mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the USART_RQR register.

0: Receiver in active mode

1: Receiver in mute mode

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 65.4: USART implementation.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the USART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: No break character transmitted

1: Break character transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the USART_ICR register.

An interrupt is generated if CMIE = 1 in the USART_CR1 register.

0: No Character match detected

1: Character match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: USART is idle (no reception)

1: Reception ongoing

Bit 15 **ABRF:** Auto baud rate flag

This bit is set by hardware when the automatic baud rate has been set (RXNE is also set, generating an interrupt if RXNEIE = 1) or when the auto baud rate operation has completed without success (ABRE = 1) (ABRE, RXNE and FE are also set in this case)

It is cleared by software, in order to request a new auto baud rate detection, by writing 1 to the ABRRQ in the USART_RQR register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 14 **ABRE**: Auto baud rate error

This bit is set by hardware if the baud rate measurement failed (baud rate out of range or character comparison failed)

It is cleared by software, by writing 1 to the ABRRQ bit in the USART_RQR register.

Note: If the USART does not support the auto baud rate feature, this bit is reserved and kept at reset value.

Bit 13 **UDR**: SPI slave underrun error flag

In slave transmission mode, this flag is set when the first clock pulse for data transmission appears while the software has not yet loaded any value into USART_TDR. This flag is reset by setting UDRCF bit in the USART_ICR register.

0: No underrun error

1: underrun error

Note: If the USART does not support the SPI slave mode, this bit is reserved and kept at reset value. Refer to Section 65.4: USART implementation.

Bit 12 **EOBF**: End of block flag

This bit is set by hardware when a complete block has been received (for example T = 1 smartcard mode). The detection is done when the number of received bytes (from the start of the block, including the prologue) is equal or greater than BLEN + 4.

An interrupt is generated if EOBIE = 1 in the USART_CR1 register.

It is cleared by software, writing 1 to EOBCF in the USART_ICR register.

0: End of block not reached

1: End of block (number of characters) reached

Note: If smartcard mode is not supported, this bit is reserved and kept at reset value. Refer to Section 65.4: USART implementation.

Bit 11 **RTOF**: Receiver timeout

This bit is set by hardware when the timeout value, programmed in the RTOR register has lapsed, without any communication. It is cleared by software, writing 1 to the RTOCF bit in the USART_ICR register.

An interrupt is generated if RTOIE = 1 in the USART_CR2 register.

In smartcard mode, the timeout corresponds to the CWT or BWT timings.

0: Timeout value not reached

1: Timeout value reached without any data reception

Note: If a time equal to the value programmed in RTOR register separates 2 characters, RTOF is not set. If this time exceeds this value + 2 sample times (2/16 or 2/8, depending on the oversampling method), RTOF flag is set.

The counter counts even if RE = 0 but RTOF is set only when RE = 1. If the timeout has already elapsed when RE is set, then RTOF is set.

If the USART does not support the Receiver timeout feature, this bit is reserved and kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.

0: CTS line set

1: CTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the USART_ICR register.

An interrupt is generated if CTSIE = 1 in the USART_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 **LBDF**: LIN break detection flag

This bit is set by hardware when the LIN break is detected. It is cleared by software, by writing 1 to the LBDCF in the USART_ICR.

An interrupt is generated if LBDIE = 1 in the USART_CR2 register.

0: LIN Break not detected

1: LIN break detected

Note: If the USART does not support LIN mode, this bit is reserved and kept at reset value. Refer to Section 65.4: USART implementation.

Bit 7 **TXE:** Transmit data register empty

TXE is set by hardware when the content of the USART_TDR register has been transferred into the shift register. It is cleared by writing to the USART_TDR register. The TXE flag can also be set by writing 1 to the TXFRQ in the USART_RQR register, in order to discard the data (only in smartcard T = 0 mode, in case of transmission failure).

An interrupt is generated if the TXEIE bit = 1 in the USART_CR1 register.

0: Data register full

1: Data register empty

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the USART_TDR has been transmitted out of the shift register. The TC flag is set when the transmission of a frame containing data has completed and when TXE is set.

An interrupt is generated if TCIE = 1 in the USART_CR1 register.

TC bit is cleared by software by writing 1 to the TCCF in the USART_ICR register or by writing to the USART_TDR register.

Bit 5 **RXNE**: Read data register not empty

RXNE bit is set by hardware when the content of the USART_RDR shift register has been transferred to the USART_RDR register. It is cleared by reading from the USART_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the USART_RQR register.

An interrupt is generated if RXNEIE = 1 in the USART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE = 1 in the USART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the USART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit is not set again until the RXNE bit has been set (that is, a new idle line occurs).

If mute mode is enabled (MME = 1), IDLE is set if the USART is not mute (RWU = 0), whatever the mute mode selected by the WAKE bit. If RWU = 1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the USART_RDR register while RXNE = 1. It is cleared by a software, writing 1 to the ORECF, in the USART_ICR register.

An interrupt is generated if RXNEIE = 1 in the USART_CR1 register, or EIE = 1 in the USART_CR3 register.

1: Overrun error is detected

Note: When this bit is set, the USART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the USART_CR3 register.

Bit 2 **NE**: Noise detection flag

This bit is set by hardware when noise is detected on a received frame. It is cleared by software, writing 1 to the NFCF bit in the USART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

When the line is noise-free, the NE flag can be disabled by programming the ONEBIT bit to 1 to increase the USART tolerance to deviations (Refer to Section 65.5.9: Tolerance of the USART receiver to clock deviation).

This error is associated with the character in the USART_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the USART_ICR register.

When transmitting data in smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).

An interrupt is generated if EIE = 1 in the USART_CR3 register.

0: No Framing error is detected

1: Framing error or break character is detected

Note: This error is associated with the character in the USART_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in reception mode. It is cleared by software, writing 1 to the PECF in the USART_ICR register.

An interrupt is generated if PEIE = 1 in the USART_CR1 register.

0: No parity error

1: Parity error

Note: This error is associated with the character in the USART_RDR.

65.8.12 USART interrupt flag clear register (USART_ICR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.
											w			w	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	UDRCF	EOBCF	RTOCF	Res.	CTSCF	LBDCF	TCBGTCF	TCCF	TXFECF	IDLECF	ORECF	NECF	FECF	PECF
		w	w	w		w	w	w	w	w	w	w	w	w	w

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **WUCF**: Wake-up from low-power mode clear flag

Writing 1 to this bit clears the WUF flag in the USART_ISR register.

Note: If the USART does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CMCF**: Character match clear flag

Writing 1 to this bit clears the CMF flag in the USART_ISR register.

Bits 16:14 Reserved, must be kept at reset value.

Bit 13 **UDRCF**:SPI slave underrun clear flag

Writing 1 to this bit clears the UDRF flag in the USART_ISR register.

Note: If the USART does not support SPI slave mode, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation

Bit 12 **EOBCF**: End of block clear flag

Writing 1 to this bit clears the EOBF flag in the USART_ISR register.

Note: If the USART does not support smartcard mode, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

Bit 11 **RTOCF**: Receiver timeout clear flag
Writing 1 to this bit clears the RTOF flag in the USART_ISR register.
Note: If the USART does not support the Receiver timeout feature, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **CTSCF**: CTS clear flag
Writing 1 to this bit clears the CTSIF flag in the USART_ISR register.
Note: If the hardware flow control feature is not supported, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

Bit 8 **LBDCF**: LIN break detection clear flag
Writing 1 to this bit clears the LBDF flag in the USART_ISR register.
Note: If LIN mode is not supported, this bit is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

Bit 7 **TCBGTCF**: Transmission complete before Guard time clear flag
Writing 1 to this bit clears the TCBGT flag in the USART_ISR register.

Bit 6 **TCCF**: Transmission complete clear flag
Writing 1 to this bit clears the TC flag in the USART_ISR register.

Bit 5 **TXFECF**: TXFIFO empty clear flag
Writing 1 to this bit clears the TXFE flag in the USART_ISR register.

Bit 4 **IDLECF**: Idle line detected clear flag
Writing 1 to this bit clears the IDLE flag in the USART_ISR register.

Bit 3 **ORECF**: Overrun error clear flag
Writing 1 to this bit clears the ORE flag in the USART_ISR register.

Bit 2 **NECF**: Noise detected clear flag
Writing 1 to this bit clears the NE flag in the USART_ISR register.

Bit 1 **FECF**: Framing error clear flag
Writing 1 to this bit clears the FE flag in the USART_ISR register.

Bit 0 **PECF**: Parity error clear flag
Writing 1 to this bit clears the PE flag in the USART_ISR register.

65.8.13 USART receive data register (USART_RDR)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
							r	r	r	r	r	r	r	r	r

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **RDR[8:0]**: Receive data value

Contains the received data character.

The RDR register provides the parallel interface between the input shift register and the internal bus (see *Section 65.5.1: USART block diagram*).

When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

65.8.14 USART transmit data register (USART_TDR)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **TDR[8:0]**: Transmit data value

Contains the data character to be transmitted.

The USART_TDR register provides the parallel interface between the internal bus and the output shift register (see *Section 65.5.1: USART block diagram*).

When transmitting with the parity enabled (PCE bit set to 1 in the USART_CR1 register), the value written in the MSB (bit 7 or bit 8 depending on the data length) has no effect because it is replaced by the parity.

Note: This register must be written only when TXE/TXFNF = 1.

65.8.15 USART prescaler register (USART_PRESC)

This register can only be written when the USART is disabled (UE = 0).

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PRESCALER[3:0]**: Clock prescaler

The USART input clock can be divided by a prescaler factor:

0000: input clock not divided
0001: input clock divided by 2
0010: input clock divided by 4
0011: input clock divided by 6
0100: input clock divided by 8
0101: input clock divided by 10
0110: input clock divided by 12
0111: input clock divided by 16
1000: input clock divided by 32
1001: input clock divided by 64
1010: input clock divided by 128
1011: input clock divided by 256
Others: Reserved, must not be used

Note: When PRESCALER is programmed with a value different of the allowed ones, programmed prescaler value is equal to 1011 that is, input clock divided by 256.

If the prescaler is not supported, this bitfield is reserved and must be kept at reset value. Refer to Section 65.4: USART implementation.

65.8.16 USART register map

Table 646. USART register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	USART_CR1 FIFO mode enabled	RXFFIE	TXFEIE	FIFOEN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]					OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXFNFIE	TCIE	RXFNEIE	IDLEIE	TE	RE	UESM	UE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00	USART_CR1 FIFO mode disabled	Res.	Res.	FIFOEN	M1	EOBIE	RTOIE	DEAT[4:0]					DEDT[4:0]					OVER8	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	USART_CR2	ADD[7:0]								RTOEN	ABRMOD[1:0]		ABREN	MSBFIRST	DATAINV	TXINV	RXINV	SWAP	LINEN	STOP [1:0]		CLKEN	CPOL	CPHA	LBCL	Res.	LBDIE	LBDL	ADDM7	DIS_NSS	Res.	Res.	SLVEN
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0			0
0x08	USART_CR3 FIFO mode enabled	TXFTCFG[2:0]			RXFTIE	RXFTCFG2:0]			TCBGTIE	TXFTIE	WUFIE	WUS 1	WUS 0	SCAR CNT2:0]			Res.	DEP	DEM	DDRE	OVRDIS	ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	USART_CR3 FIFO mode disabled	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCBGTIE	Res.	WUFIE	WUS 1	WUS 0	SCAR CNT2:0]			Res.	DEP	DEM	DDRE	OVRDIS	ONEBIT	CTSIE	CTSE	RTSE	DMAT	DMAR	SCEN	NACK	HDSEL	IRLP	IREN	EIE
	Reset value								0		0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 646. USART register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0C	USART_BRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BRR[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	USART_ GTPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GT[7:0]								PSC[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	USART_ RTOR	BLEN[7:0]								RTO[23:0]																							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	USART_RQR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	ABRRQ
	Reset value																												0	0	0	0	0
0x1C	USART_ISR FIFO mode enabled	Res.	Res.	Res.	Res.	TXFT	RXFT	TCBGT	RXFF	TXFE	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY	ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXFNF	TC	RXFNE	IDLE	ORE	NE	FE	PE
	Reset value					0	0	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0x1C	USART_ISR FIFO mode disabled	Res.	Res.	Res.	Res.	Res.	Res.	TCBGT	Res.	Res.	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY	ABRF	ABRE	UDR	EOBF	RTOF	CTS	CTSIF	LBDF	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
	Reset value							0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0x20	USART_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.	Res.	Res.	UDRCF	EOBCF	RTOCF	Res.	CTSCF	LBDCF	TCBGTCF	TCCF	TXFECF	IDLECF	ORECF	NECF	FECF	PECF
	Reset value												0			0				0	0	0		0	0	0	0	0	0	0	0	0	0
0x24	USART_RDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0
0x28	USART_TDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0
0x2C	USART_ PRESC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[3:0]			
	Reset value																													0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

66 Low-power universal asynchronous receiver transmitter (LPUART)

66.1 LPUART introduction

The LPUART is an UART, which enables bidirectional UART communications with a limited power consumption. Only a 32.768 kHz LSE clock is required to enable UART communications up to 9600 bauds. Higher baud rates can be reached when the LPUART is clocked by clock sources different from the LSE clock.

Even when the microcontroller is in low-power mode, the LPUART can wait for an incoming UART frame while having an extremely low energy consumption. The LPUART includes all necessary hardware support to make asynchronous serial communications possible with minimum power consumption.

It supports half-duplex single-wire communications and modem operations (CTS/RTS).

It also supports multiprocessor communications.

DMA (direct memory access) can be used for data transmission/reception.

66.2 LPUART main features

- Full-duplex asynchronous communications
- NRZ standard format (mark/space)
- Programmable baud rate
- From 300 bauds to 9600 bauds using a 32.768 kHz clock source.
- Higher baud rates can be achieved by using a higher frequency clock source
- Two internal FIFOs to transmit and receive data

  Each FIFO can be enabled/disabled by software and come with status flags for FIFOs states.
- Dual clock domain with dedicated kernel clock for peripherals independent from PCLK.
- Programmable data word length (7 or 8 or 9 bits)
- Programmable data order with MSB-first or LSB-first shifting
- Configurable stop bits (1 or 2 stop bits)
- Single-wire half-duplex communications
- Continuous communications using DMA
- Received/transmitted bytes are buffered in reserved SRAM using centralized DMA.
- Separate enable bits for transmitter and receiver
- Separate signal polarity control for transmission and reception
- Swappable Tx/Rx pin configuration
- Hardware flow control for modem and RS485 transceiver
- Transfer detection flags:
  - Receive buffer full
  - Transmit buffer empty
  - Busy and end of transmission flags

- Parity control:
  - Transmits parity bit
  - Checks parity of received data byte
- Four error detection flags:
  - Overrun error
  - Noise detection
  - Frame error
  - Parity error
- Interrupt sources with flags
- Multiprocessor communications: wake-up from mute mode by idle line detection or address mark detection
- Wake-up from Stop mode

66.3 LPUART implementation

The tables below describe the LPUART implementation. USARTs and UARTs are included for comparison.

Table 647. Instance implementation on STM32N6x5/x7xx

Instance	STM32N6xx
USART1	Full
USART2	Full
USART3	Full
USART6	Full
USART10	Full
UART4	Basic
UART5	Basic
UART7	Basic
UART8	Basic
UART9	Basic
LPUART1	Low-power

Table 648. USART/LPUART features

Modes/features(1)	Full feature	Basic feature	Low-power feature
Hardware flow control for modem	X	X	X
Continuous communication using DMA	X	X	X
Multiprocessor communication	X	X	X
Synchronous mode (master/slave)	X	-	-

Table 648. USART/LPUART features (continued)

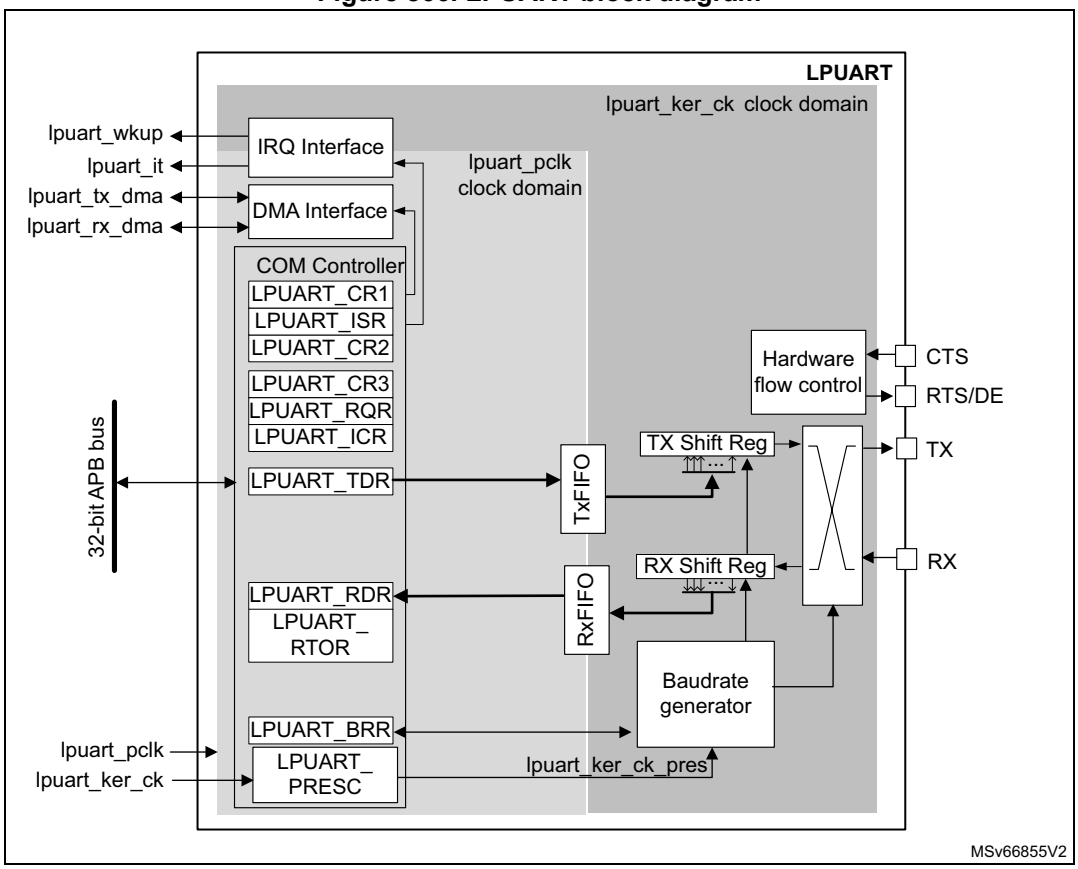
Modes/features[1]	Full feature	Basic feature	Low-power feature
Smartcard mode	X	-	-
Single-wire half-duplex communication	X	X	X
IrDA SIR ENDEC block	X	X	-
LIN mode	X	X	-
Dual-clock domain	X	X	X
Receiver timeout interrupt	X	X	-
Modbus communication	X	X	-
Auto baud rate detection	X	X	-
Driver enable	X	X	X
USART data length	7, 8 and 9 bits		
Tx/Rx FIFO	X	X	X
Tx/Rx FIFO size (bytes)	16		
Wake-up from low-power mode	X[2]	X[2]	X[2]

1. X = supported.
2. Wake-up supported from Stop mode.

66.4 LPUART functional description

66.4.1 LPUART block diagram

Figure 866. LPUART block diagram



66.4.2 LPUART pins and internal signals

Description LPUART input/output pins

- LPUART bidirectional communications

  LPUART bidirectional communications require a minimum of two pins: Receive Data In (RX) and Transmit Data Out (TX):

  – **RX** (Receive Data Input):

    RX is the serial data input.

  – **TX (**Transmit Data Output)

    When the transmitter is disabled, the output pin returns to its I/O port configuration. When the transmitter is enabled and nothing is to be transmitted, the TX pin is at high level. In single-wire mode, this I/O is used to transmit and receive the data.

- RS232 hardware flow control mode

  The RS232 hardware flow control mode requires the following pins:

  – **CTS (**Clear To Send)

    When driven high, this signal blocks the data transmission at the end of the current transfer.

  – **RTS** (Request to send)

    When it is low, this signal indicates that the LPUART is ready to receive data.

- RS485 hardware flow control mode

  The **DE** (Driver Enable) pin is required in RS485 hardware control mode. This signal activates the transmission mode of the external transceiver.

Refer to *Table 649* and *Table 650* for the list of LPUART input/output pins and internal signals.

Table 649. LPUART input/output pins

Pin name	Signal type	Description
LPUART_RX	Input	Serial data receive input.
LPUART_TX	Output	Transmit data output.
LPUART_CTS	Input	Clear to send
LPUART_RTS	Output	Request to send
LPUART_DE(1)	Output	Driver enable

1. LPUART_DE and LPUART_RTS share the same pin.

Description LPUART input/output signals

Table 650. LPUART internal input/output signals

Signal name	Signal type	Description
lpuart_pclk	Input	APB clock
lpuart_ker_ck	Input	LPUART kernel clock
lpuart_wkup	Output	LPUART provides a wake-up interrupt

Table 650. LPUART internal input/output signals (continued)

Signal name	Signal type	Description
lpuart_it	Output	LPUART global interrupt
lpuart_tx_dma	Input/output	LPUART transmit DMA request
lpuart_rx_dma	Input/output	LPUART receive DMA request

66.4.3 LPUART clocks

The simplified block diagram given in *Section 66.4.1: LPUART block diagram* shows two fully independent clock domains:

- The **lpuart_pclk** clock domain

  The **lpuart_pclk** clock signal feeds the peripheral bus interface. It must be active when accesses to the LPUART registers are required.
- The **lpuart_ker_ck** kernel clock domain

  The **lpuart_ker_ck** is the LPUART clock source. It is independent of the **lpuart_pclk** and delivered by the RCC. So, the LPUART registers can be written/read even when the **lpuart_ker_ck** is stopped.

  When the dual clock domain feature is not supported, the **lpuart_ker_ck i**s the same as the lpuart_pclk clock.

There is no constraint between **lpuart_pclk** and **lpuart_ker_ck**: **lpuart_ker_ck** can be faster or slower than **lpuart_pclk**, with no more limitation than the ability for the software to manage the communication fast enough.

66.4.4 LPUART character description

The word length can be set to 7 or 8 or 9 bits, by programming the M bits (M0: bit 12 and M1: bit 28) in the LPUART_CR1 register (see *Figure 840*).

- 7-bit character length: M[1:0] = 10
- 8-bit character length: M[1:0] = 00
- 9-bit character length: M[1:0] = 01

By default, the signal (TX or RX) is in low state during the start bit. It is in high state during the stop bit.

These values can be inverted, separately for each signal, through polarity configuration control.

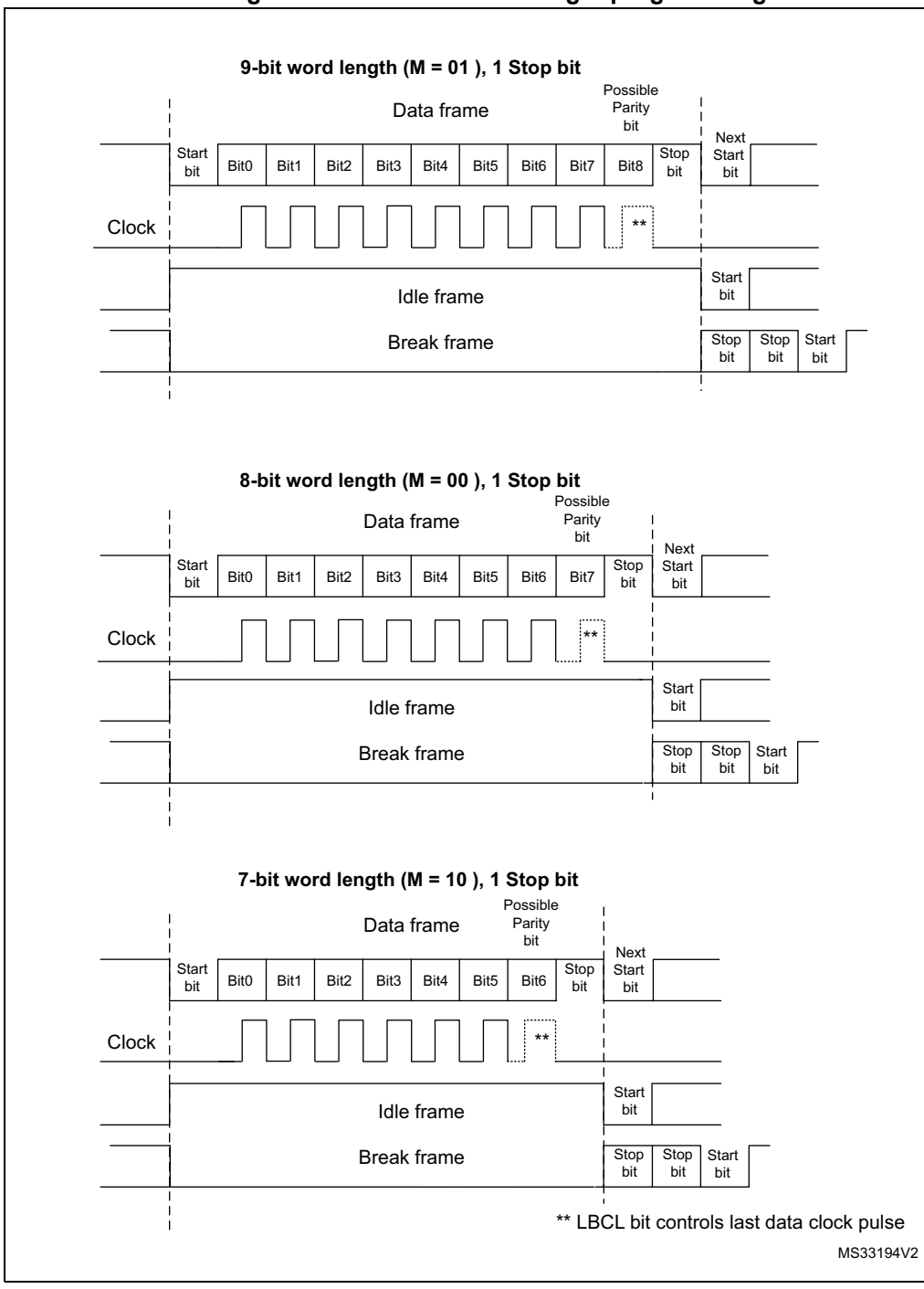
An ***idle character*** is interpreted as an entire frame of “1”s (the number of “1”s includes the number of stop bits).

A ***break character*** is interpreted on receiving “0”s for a frame period. At the end of the break frame, the transmitter inserts 2 stop bits.

Transmission and reception are driven by a common baud rate generator. The transmission and reception clocks are generated when the enable bit is set for the transmitter and receiver, respectively.

The details of each block is given below.

Figure 867. LPUART word length programming



66.4.5 LPUART FIFOs and thresholds

The LPUART can operate in FIFO mode.

The LPUART comes with a transmit FIFO (TXFIFO) and a Receive FIFO (RXFIFO). The FIFO mode is enabled by setting FIFOEN bit (bit 29) in the LPUART_CR1 register.

Since 9 bits the maximum data word length is 9 bits, the TXFIFO is 9-bit wide. However, the RXFIFO default width is 12 bits. This is due to the fact that the receiver does not only store the data in the FIFO, but also the error flags associated to each character (Parity error, Noise error and Framing error flags).

Note: The received data is stored in the RXFIFO together with the corresponding flags. However, only the data are read when reading the RDR.

The status flags are available in the LPUART_ISR register.

It is possible to define the TXFIFO and RXFIFO levels at which the Tx and RX interrupts are triggered. These thresholds are programmed through the RXFTCFG[2:0] and TXFTCFG[2:0] bitfields of the LPUART_CR3 control register.

In this case:

- The Rx interrupt is generated when the number of received data in the RXFIFO reaches the threshold programmed in the RXFTCFG[2:0] bitfield.

  In this case, the RXFT flag is set in the LPUART_ISR register. This means that RXFTCFG[2:0] data have been received: 1 data in LPUART_RDR and (RXFTCFG[2:0] – 1) data in the RXFIFO. As an example, when the RXFTCFG[2:0] is programmed to 101, the RXFT flag is set when a number of data corresponding to the FIFO size has been received: FIFO size – 1 data in the RXFIFO and 1 data in the LPUART_RDR. As a result, the next received data does not set the overrun flag.
- The Tx interrupt is generated when the number of empty locations in the TXFIFO is greater than the threshold programmed in the TXFTCFG[2:0] bitfield of the LSART_CR3 register.

66.4.6 LPUART transmitter

The transmitter can send data words of either 7 or 8 or 9 bits, depending on the M bit status. The transmit enable bit (TE) must be set in order to activate the transmitter function. The data in the transmit shift register is output on the TX pin.

Character transmission

During an LPUART transmission, data shifts out least significant bit first (default configuration) on the TX pin. In this mode, the LPUART_TDR register consists of a buffer (TDR) between the internal bus and the transmit shift register (see *Section 66.4.1: LPUART block diagram*).

When FIFO mode is enabled, the data written to the LPUART_TDR register are queued in the TXFIFO.

Every character is preceded by a start bit bit, which corresponds to a low logic level for one bit period. The character is terminated by a configurable number of stop bits.

The number of stop bits can be 1 or 2.

Note: The TE bit must be set before writing the data to be transmitted to the LPUART_TDR.

The TE bit must not be reset during data transmission. Resetting the TE bit during the transmission corrupts the data on the TX pin as the baud rate counters is frozen. The current data being transmitted are lost.

An idle frame is sent after the TE bit is enabled.

Configurable stop bits

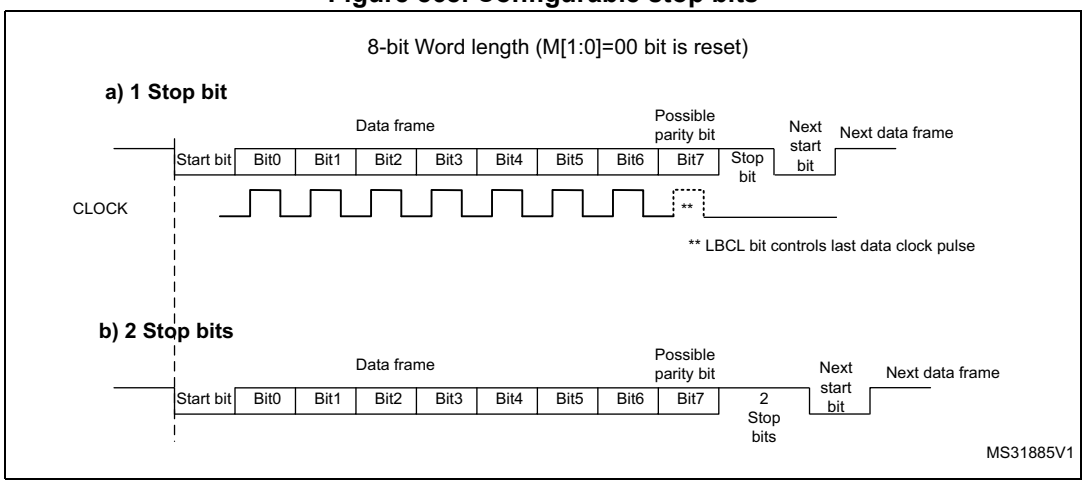
The number of stop bits to be transmitted with every character can be programmed in LPUART_CR2 (bits 13, 12).

- ***1 stop bit***: This is the default value of the number of stop bits.
- ***2 stop bits***: This is supported by normal LPUART, single-wire, and modem modes.

An idle frame transmission includes the stop bits.

A break transmission is 10 low bits (when M[1:0] = 00) or 11 low bits (when M[1:0] = 01) or 9 low bits (when M[1:0] = 10) followed by 2 stop bits. It is not possible to transmit long breaks (break of length greater than 9/10/11 low bits).

Figure 868. Configurable stop bits



Character transmission procedure

To transmit a character, follow the sequence below:

1. Program the M bits in LPUART_CR1 to define the word length.
2. Select the desired baud rate using the LPUART_BRR register.
3. Program the number of stop bits in LPUART_CR2.
4. Enable the LPUART by writing the UE bit in LPUART_CR1 register to 1.
5. Select DMA enable (DMAT) in LPUART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in *Section 66.4.13: Continuous communication using DMA and LPUART*.
6. Set the TE bit in LPUART_CR1 to send an idle frame as first transmission.

7. Write the data to send in the LPUART_TDR register. Repeat this operation for each data to be transmitted in case of single buffer.
   – When FIFO mode is disabled, writing a data in the LPUART_TDR clears the TXE flag.
   – When FIFO mode is enabled, writing a data in the LPUART_TDR adds one data to the TXFIFO. Write operations to the LPUART_TDR are performed when the TXFNF flag is set. This flag remains set until the TXFIFO is full.
8. When the last data is written to the LPUART_TDR register, wait until TC = 1. This indicates that the transmission of the last frame has been completed.
   – When FIFO mode is disabled, this indicates that the transmission of the last frame has been completed.
   – When FIFO mode is enabled, this indicates that both TXFIFO and shift register are empty.

   This check is required to avoid corrupting the last transmission when the LPUART is disabled or enters Halt mode.

Single byte communication

- When FIFO mode disabled:

  Writing to the transmit data register always clears the TXE bit. The TXE flag is set by hardware to indicate that:
  – the data have been moved from the LPUART_TDR register to the shift register and data transmission has started;
  – the LPUART_TDR register is empty;
  – the next data can be written to the LPUART_TDR register without overwriting the previous data.

  The TXE flag generates an interrupt if the TXEIE bit is set.

  When a transmission is ongoing, a write instruction to the LPUART_TDR register stores the data in the TDR register, which is copied to the shift register at the end of the current transmission.

  When no transmission is ongoing, a write instruction to the LPUART_TDR register places the data in the shift register, the data transmission starts, and the TXE bit is set.
- When FIFO mode is enabled, the TXFNF (TXFIFO not full) flag is set by hardware to indicate that:
  – The TXFIFO is not full;
  – The LPUART_TDR register is empty;
  – The next data can be written to the LPUART_TDR register without overwriting the previous data. When a transmission is ongoing, a write operation to the

LPUART_TDR register stores the data in the TXFIFO. Data are copied from the TXFIFO to the shift register at the end of the current transmission.

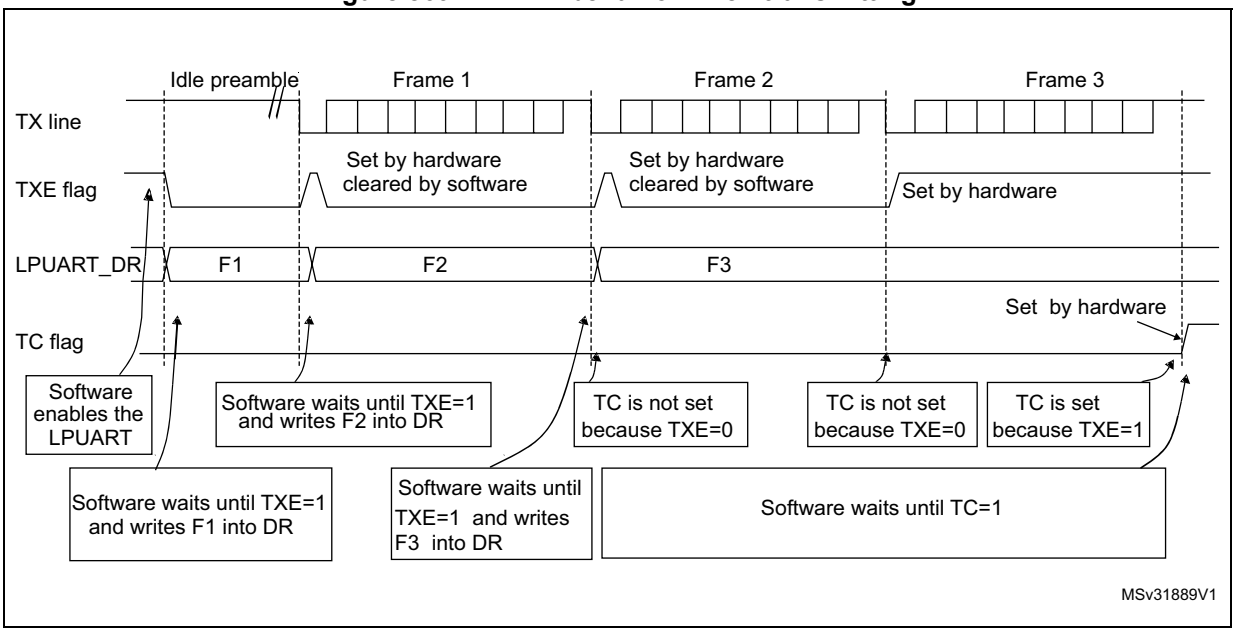
When the TXFIFO is not full, the TXFNF flag stays at 1 even after a write in LPUART_TDR. It is cleared when the TXFIFO is full. This flag generates an interrupt if the TXFNEIE bit is set.

Alternatively, interrupts can be generated and data can be written to the TXFIFO when the TXFIFO threshold is reached. In this case, the CPU can write a block of data defined by the programmed threshold.

If a frame is transmitted (after the stop bit) and the TXE flag (TXFE in case of FIFO mode) is set, the TC bit goes high. An interrupt is generated if the TCIE bit is set in the LPUART_CR1 register.

After writing the last data in the LPUART_TDR register, it is mandatory to wait for TC = 1 before disabling the LPUART or causing the microcontroller to enter the low-power mode (see *Figure 869: TC/TXE behavior when transmitting*).

Figure 869. TC/TXE behavior when transmitting



Note: When FIFO management is enabled, the TXFNF flag is used for data transmission.

Break characters

Setting the SBKRQ bit transmits a break character. The break frame length depends on the M bits (see *Figure 867*).

If a 1 is written to the SBKRQ bit, a break character is sent on the TX line after completing the current character transmission. The SBKF bit is set by the write operation and it is reset by hardware when the break character is complete (during the stop bits after the break character). The LPUART inserts a logic 1 signal (STOP) for the duration of 2 bits at the end of the break frame to guarantee the recognition of the start bit of the next frame.

When the SBKRQ bit is set, the break character is sent at the end of the current transmission.

When FIFO mode is enabled, sending the break character has priority on sending data even if the TXFIFO is full.

Idle characters

Setting the TE bit drives the LPUART to send an idle frame before the first data frame.

66.4.7 LPUART receiver

The LPUART can receive data words of either 7 or 8 or 9 bits depending on the M bits in the LPUART_CR1 register.

Start bit detection

In the LPUART, the start bit is detected when a falling edge occurs on the Rx line, followed by a sample taken in the middle of the start bit to confirm that it is still 0. If the start sample is at 1, then the noise error flag (NE) is set, then the start bit is discarded and the receiver waits for a new start bit. Else, the receiver continues to sample all incoming bits normally.

Character reception

During an LPUART reception, data are shifted with the least significant bit first (default configuration) through the RX pin. In this mode, the LPUART_RDR register consists of a buffer (RDR) between the internal bus and the received shift register.

Character reception procedure

To receive a character, follow the sequence below:

1. Program the M bits in LPUART_CR1 to define the word length.
2. Select the desired baud rate using the baud rate register LPUART_BRR
3. Program the number of stop bits in LPUART_CR2.
4. Enable the LPUART by writing the UE bit in LPUART_CR1 register to 1.
5. Select DMA enable (DMAR) in LPUART_CR3 if multibuffer communication is to take place. Configure the DMA register as explained in *Section 66.4.13: Continuous communication using DMA and LPUART*.
6. Set the RE bit LPUART_CR1. This enables the receiver, which begins searching for a start bit.

When a character is received

- When FIFO mode is disabled, the RXNE bit is set. It indicates that the content of the shift register is transferred to the RDR. In other words, data has been received and can be read (as well as its associated error flags).
- When FIFO mode is enabled, the RXFNE bit is set indicating that the RXFIFO is not empty. Reading the LPUART_RDR returns the oldest data entered in the RXFIFO.

When a data is received, it is stored in the RXFIFO, together with the corresponding error bits.

- An interrupt is generated if the RXNEIE (RXFNEIE in case of FIFO mode) bit is set.
- The error flags can be set if a frame error, noise, or an overrun error has been detected during reception.
- In multibuffer communication mode:
  - When FIFO mode is disabled, the RXNE flag is set after every byte received and is cleared by the DMA read of the receive data register.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every DMA request, a data is retrieved from the RXFIFO. DMA request is triggered by RXFIFO is not empty, that is, there is a data in the RXFIFO to be read.
- In single-buffer mode:
  - When FIFO mode is disabled, clearing the RXNE flag is done by performing a software read from the LPUART_RDR register. The RXNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register. The RXNE bit must be cleared before the end of the reception of the next character to avoid an overrun error.
  - When FIFO mode is enabled, the RXFNE flag is set when the RXFIFO is not empty. After every read operation from the LPUART_RDR register, a data is retrieved from the RXFIFO. When the RXFIFO is empty, the RXFNE flag is cleared. The RXFNE flag can also be cleared by writing 1 to the RXFRQ bit in the LPUART_RQR register. When the RXFIFO is full, the first entry in the RXFIFO must be read before the end of the reception of the next character to avoid an overrun error. The RXFNE flag generates an interrupt if the RXFNEIE bit is set. Alternatively, interrupts can be generated and data can be read from RXFIFO when the RXFIFO threshold is reached. In this case, the CPU can read a block of data defined by the programmed threshold.

Break character

When a break character is received, the LPUART handles it as a framing error.

Idle character

When an idle frame is detected, it is handled in the same way as a data character reception except that an interrupt is generated if the IDLEIE bit is set.

Overrun error

- FIFO mode disabled

  An overrun error occurs when a character is received when RXNE has not been reset.

  Data cannot be transferred from the shift register to the RDR register until the RXNE bit is cleared. The RXNE flag is set after every byte received.

  An overrun error occurs if the RXNE flag is set when the next data is received or the previous DMA request has not been serviced. When an overrun error occurs:

  – The ORE bit is set;
  – The RDR content is not lost. The previous data is available when a read to LPUART_RDR is performed.
  – The shift register is overwritten. After that, any data received during overrun is lost.
  – An interrupt is generated if either the RXNEIE bit or EIE bit is set.

- FIFO mode enabled

  An overrun error occurs when the shift register is ready to be transferred when the receive FIFO is full.

  Data cannot be transferred from the shift register to the LPUART_RDR register until there is one free location in the RXFIFO. The RXFNE flag is set when the RXFIFO is not empty.

  An overrun error occurs if the RXFIFO is full and the shift register is ready to be transferred. When an overrun error occurs:

  – The ORE bit is set;
  – The first entry in the RXFIFO is not lost. It is available when a read to LPUART_RDR is performed.
  – The shift register is overwritten. After that, any data received during overrun is lost.
  – An interrupt is generated if either the RXFNEIE bit or EIE bit is set.

The ORE bit is reset by setting the ORECF bit in the ICR register.

Note: The ORE bit, when set, indicates that at least 1 data has been lost. T

When the FIFO mode is disabled, there are two possibilities

- *If RXNE = 1, then the last valid data is stored in the receive register (RDR) and can be read,*
- *If RXNE = 0, then the last valid data has already been read and there is nothing left to be read in the RDR. This case can occur when the last valid data is read in the RDR at the same time as the new (and lost) data is received.*

Selecting the clock source

The choice of the clock source is done through the clock control system (see *Section Reset and clock controller (RCC)*). The clock source must be selected through the UE bit, before enabling the LPUART.

The clock source must be selected according to two criteria:

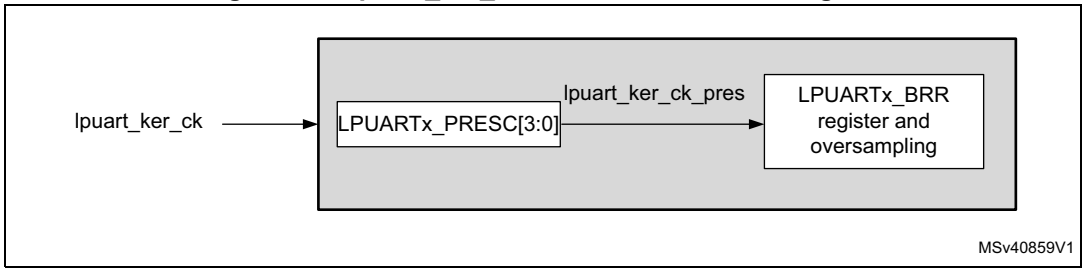
- Possible use of the LPUART in low-power mode
- Communication speed.

The clock source frequency is lpuart_ker_ck.

When the dual clock domain and the wake-up from low-power mode features are supported, the lpuart_ker_ck clock source can be configured in the RCC (see *Section Reset and clock controller (RCC)*). Otherwise, the lpuart_ker_ck is the same as lpuart_pclk.

The lpuart_ker_ck can be divided by a programmable factor in the LPUART_PRESC register.

Figure 870. lpuart_ker_ck clock divider block diagram



Some lpuart_ker_ck sources enable the LPUART to receive data while the MCU is in low-power mode. Depending on the received data and wake-up mode selection, the LPUART wakes up the MCU, when needed, in order to transfer the received data by software reading the LPUART_RDR register or by DMA.

For the other clock sources, the system must be active to enable LPUART communications.

The communication speed range (specially the maximum communication speed) is also determined by the clock source.

The receiver samples each incoming bit as close as possible to the middle of the bit-period. Only a single sample is taken of each of the incoming bits.

Note: There is no noise detection for data.

Framing error

A framing error is detected when the stop bit is not recognized on reception at the expected time, following either a desynchronization or excessive noise.

When the framing error is detected:

- The FE bit is set by hardware.
- The invalid data is transferred from the Shift register to the LPUART_RDR register.
- No interrupt is generated in case of single byte communication. However, this bit rises at the same time as the RXNE bit which itself generates an interrupt. In case of multibuffer communication, an interrupt is issued if the EIE bit is set in the LPUART_CR3 register.

The FE bit is reset by writing 1 to the FECF in the LPUART_ICR register.

Configurable stop bits during reception

The number of stop bits to be received can be configured through the control bits of LPUART_CR2: it can be either 1 or 2 in normal mode.

- **1 stop bit**: sampling for 1 stop bit is done on the 8th, 9th, and 10th samples.
- **2 stop bits**: sampling for the 2 stop bits is done in the middle of the second stop bit. The RXNE and FE flags are set just after this sample, that is, during the second stop bit. The first stop bit is not checked for framing error.

66.4.8 LPUART baud rate generation

The baud rate for the receiver and transmitter (Rx and Tx) are both set to the value programmed in the LPUART_BRR register.

$$\text{Tx/Rx baud} = \frac{256 \times \text{lpuart_ker_ck_pres}}{\text{LPUARTDIV}}$$

LPUARTDIV is defined in the LPUART_BRR register.

Note: The baud counters are updated to the new value in the baud registers after a write operation to LPUART_BRR. Hence, the baud rate register value must not be changed during communication.

It is forbidden to write values lower than 0x300 in the LPUART_BRR register.

lpuart_ker_ck_pres must range from 3 x baud rate to 4096 x baud rate.

The maximum baud rate that can be reached when the LPUART clock source is the LSE, is 9600 bauds. Higher baud rates can be reached when the LPUART is clocked by clock sources different from the LSE clock.

Table 651. Error calculation for programmed baud rates at lpuart_ker_ck_pres= 32.768 kHz

Baud rate		lpuart_ker_ck_pres= 32.768 kHz		
S.No	**Desired**	**Actual**	**Value programmed in the baud rate register**	**% Error = (Calculated – Desired) B.rate / Desired B.rate**
1	0.3 kbaud	300 baud	0x6D3A	0
2	0.6 kbaud	600 baud	0x369D	0
3	1200 bauds	1200.087 bauds	0x1B4E	0.007
4	2400 bauds	2400.17 bauds	0xDA7	0.007
5	4800 bauds	4801.72 bauds	0x6D3	0.035
6	9600 kbauds	9608.94 bauds	0x369	0.093

66.4.9 Tolerance of the LPUART receiver to clock deviation

The asynchronous receiver of the LPUART works correctly only if the total clock system deviation is less than the tolerance of the LPUART receiver. The causes that contribute to the total deviation are:

- DTRA: deviation due to the transmitter error (which also includes the deviation of the transmitter's local oscillator)
- DQUANT: error due to the baud rate quantization of the receiver
- DREC: deviation of the receiver local oscillator
- DTCL: deviation due to the transmission line (generally due to the transceivers, which can introduce an asymmetry between the low-to-high transition timing and the high-to-low transition timing)

$$DTRA + DQUANT + DREC + DTCL + DWU < \text{LPUART receiver tolerance}$$

where

DWU is the error due to sampling point deviation when the wake-up from low-power mode is used.

when M[1:0] = 01:

$$DWU = \frac{t_{WULPUART}}{11 \times Tbit}$$

when M[1:0] = 00:

$$DWU = \frac{t_{WULPUART}}{10 \times Tbit}$$

when M[1:0] = 10:

$$DWU = \frac{t_{WULPUART}}{9 \times Tbit}$$

$t_{WULPUART}$ is the time between the detection of the start bit falling edge and the instant when the clock (requested by the peripheral) is ready and reaching the peripheral, and the regulator is ready.

The LPUART receiver can receive data correctly at up to the maximum tolerated deviation specified in *Table 652*:

- Number of stop bits defined through STOP[1:0] bits in the LPUART_CR2 register
- LPUART_BRR register value.

Table 652. Tolerance of the LPUART receiver

M bits	768 < BRR < 1024	1024 < BRR < 2048	2048 < BRR < 4096	4096 ≤ BRR
8 bits (M = 00), 1 stop bit	1.82%	2.56%	3.90%	4.42%
9 bits (M = 01), 1 stop bit	1.69%	2.33%	2.53%	4.14%
7 bits (M = 10), 1 stop bit	2.08%	2.86%	4.35%	4.42%
8 bits (M = 00), 2 stop bits	2.08%	2.86%	4.35%	4.42%
9 bits (M = 01), 2 stop bits	1.82%	2.56%	3.90%	4.42%
7 bits (M = 10), 2 stop bits	2.34%	3.23%	4.92%	4.42%

Note: The data specified in Table 652 may slightly differ in the special case when the received frames contain some idle frames of exactly 10-bit times when M bits = 00 (11-bit times when M = 01 or 9- bit times when M = 10).

66.4.10 LPUART multiprocessor communication

It is possible to perform LPUART multiprocessor communications (with several LPUARTs connected in a network). For instance one of the LPUARTs can be the master, with its TX output connected to the RX inputs of the other LPUARTs. The others are slaves, with their respective TX outputs are logically ANDed together and connected to the RX input of the master.

In multiprocessor configurations it is often desirable that only the intended message recipient actively receives the full message contents, thus reducing redundant LPUART service overhead for all nonaddressed receivers.

The nonaddressed devices can be placed in mute mode by means of the muting function. To use the mute mode feature, the MME bit must be set in the LPUART_CR1 register.

Note: When FIFO management is enabled and MME is already set, MME bit must not be cleared and then set again quickly (within two lpuart_ker_ck *cycles), otherwise mute mode might remain active.*

When the mute mode is enabled:

- none of the reception status bits can be set;
- all the receive interrupts are inhibited;
- the RWU bit in the LPUART_ISR register is set to 1. RWU can be controlled automatically by hardware or by software, through the MMRQ bit in the LPUART_RQR register, under certain conditions.

The LPUART can enter or exit from mute mode using one of two methods, depending on the WAKE bit in the LPUART_CR1 register:

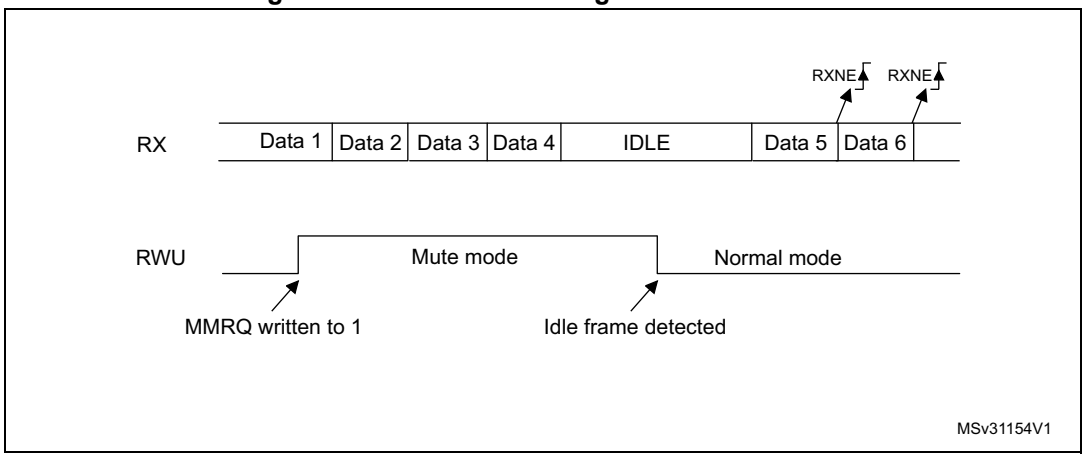
- Idle Line detection if the WAKE bit is reset,
- Address mark detection if the WAKE bit is set.

Idle line detection (WAKE = 0)

The LPUART enters mute mode when the MMRQ bit is written to 1 and the RWU is automatically set.

The LPUART wakes up when an Idle frame is detected. The RWU bit is then cleared by hardware but the IDLE bit is not set in the LPUART_ISR register. An example of mute mode behavior using Idle line detection is given in *Figure 871*.

Figure 871. Mute mode using Idle line detection



Note: If the MMRQ is set while the IDLE character has already elapsed, mute mode is not entered (RWU is not set).

If the LPUART is activated while the line is IDLE, the idle state is detected after the duration of one IDLE frame (not only after the reception of one character frame).

4-bit/7-bit address mark detection (WAKE = 1)

In this mode, bytes are recognized as addresses if their MSB is a 1 otherwise they are considered as data. In an address byte, the address of the targeted receiver is put in the 4 or 7 LSBs. The choice of 7- or 4-bit address detection is done using the ADDM7 bit. This 4-bit/7-bit word is compared by the receiver with its own address, which is programmed in the ADD bits in the LPUART_CR2 register.

Note: In 7-bit and 9-bit data modes, address detection is done on 6-bit and 8-bit addresses (ADD[5:0] and ADD[7:0]) respectively.

The LPUART enters mute mode when an address character is received which does not match its programmed address. In this case, the RWU bit is set by hardware. The RXNE flag is not set for this address byte and no interrupt or DMA request is issued when the LPUART enters mute mode.

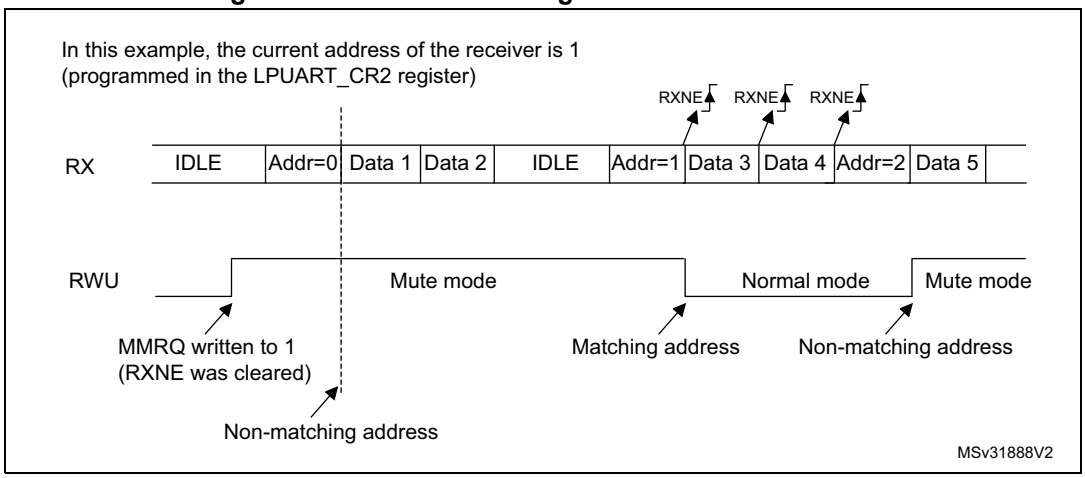
The LPUART also enters mute mode when the MMRQ bit is written to 1. The RWU bit is also automatically set in this case.

The LPUART exits from mute mode when an address character is received which matches the programmed address. Then the RWU bit is cleared and subsequent bytes are received normally. The RXNE/RXFNE bit is set for the address character since the RWU bit has been cleared.

Note: When FIFO management is enabled, when MMRQ bit is set while the receiver is sampling the last bit of a data, this data may be received before effectively entering in mute mode.

An example of mute mode behavior using address mark detection is given in *Figure 872*.

Figure 872. Mute mode using address mark detection



66.4.11 LPUART parity control

Parity control (generation of parity bit in transmission and parity checking in reception) can be enabled by setting the PCE bit in the LPUART_CR1 register. Depending on the frame length defined by the M bits, the possible LPUART frame formats are as listed in *Table 653*.

Table 653: LPUART frame formats

M bits	PCE bit	LPUART frame(1)
00	0	| SB | 8 bit data | STB |
00	1	| SB | 7-bit data | PB | STB |
01	0	| SB | 9-bit data | STB |
01	1	| SB | 8-bit data PB | STB |
10	0	| SB | 7bit data | STB |
10	1	| SB | 6-bit data | PB | STB |

1. Legends: SB: start bit, STB: stop bit, PB: parity bit.
2. In the data register, the PB is always taking the MSB position (8th or 7th, depending on the M bit value).

Even parity

The parity bit is calculated to obtain an even number of “1s” inside the frame, which is made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data = 00110101, and 4 bits are set, then the parity bit is equal to 0 if even parity is selected (PS bit in LPUART_CR1 = 0).

Odd parity

The parity bit is calculated to obtain an odd number of “1s” inside the frame made of the 6, 7 or 8 LSB bits (depending on M bit values) and the parity bit.

As an example, if data = 00110101 and 4 bits set, then the parity bit is equal to 1 if odd parity is selected (PS bit in LPUART_CR1 = 1).

Parity checking in reception

If the parity check fails, the PE flag is set in the LPUART_ISR register and an interrupt is generated if PEIE is set in the LPUART_CR1 register. The PE flag is cleared by software writing 1 to the PECF in the LPUART_ICR register.

Parity generation in transmission

If the PCE bit is set in LPUART_CR1, then the MSB bit of the data written in the data register is transmitted but is changed by the parity bit (even number of “1s” if even parity is selected (PS = 0) or an odd number of “1s” if odd parity is selected (PS = 1)).

66.4.12 LPUART single-wire half-duplex communication

Single-wire half-duplex mode is selected by setting the HDSEL bit in the LPUART_CR3 register.

The LPUART can be configured to follow a single-wire half-duplex protocol where the TX and RX lines are internally connected. The selection between half- and full-duplex communication is made with a control bit HDSEL in LPUART_CR3.

As soon as HDSEL is written to 1:

- The TX and RX lines are internally connected.
- The RX pin is no longer used
- The TX pin is always released when no data is transmitted. Thus, it acts as a standard I/O in idle or in reception. It means that the I/O must be configured so that TX is configured as alternate function open-drain with an external pull-up.

Apart from this, the communication protocol is similar to normal LPUART mode. Any conflict on the line must be managed by software (for instance by using a centralized arbiter). In particular, the transmission is never blocked by hardware and continues as soon as data is written in the data register while the TE bit is set.

Note: In LPUART communications, in the case of 1-stop bit configuration, the RXNE flag is set in the middle of the stop bit.

66.4.13 Continuous communication using DMA and LPUART

The LPUART is capable of performing continuous communication using the DMA. The DMA requests for Rx buffer and Tx buffer are generated independently.

Note: Refer to Section 66.3: LPUART implementation to determine if the DMA mode is supported. If DMA is not supported, use the LPUSRT as explained in Section 66.4.7. To perform continuous communication. When FIFO is disabled, clear the TXE/ RXNE flags in the LPUART_ISR register.

Transmission using DMA

DMA mode can be enabled for transmission by setting the DMAT bit in the LPUART_CR3 register. Data are loaded from an SRAM area configured using the DMA peripheral (refer to Section *direct memory access controller*) to the LPUART_TDR register whenever the TXE flag (TXFNF flag if FIFO mode is enabled) is set. To map a DMA channel for LPUART transmission, use the following procedure (x denotes the channel number):

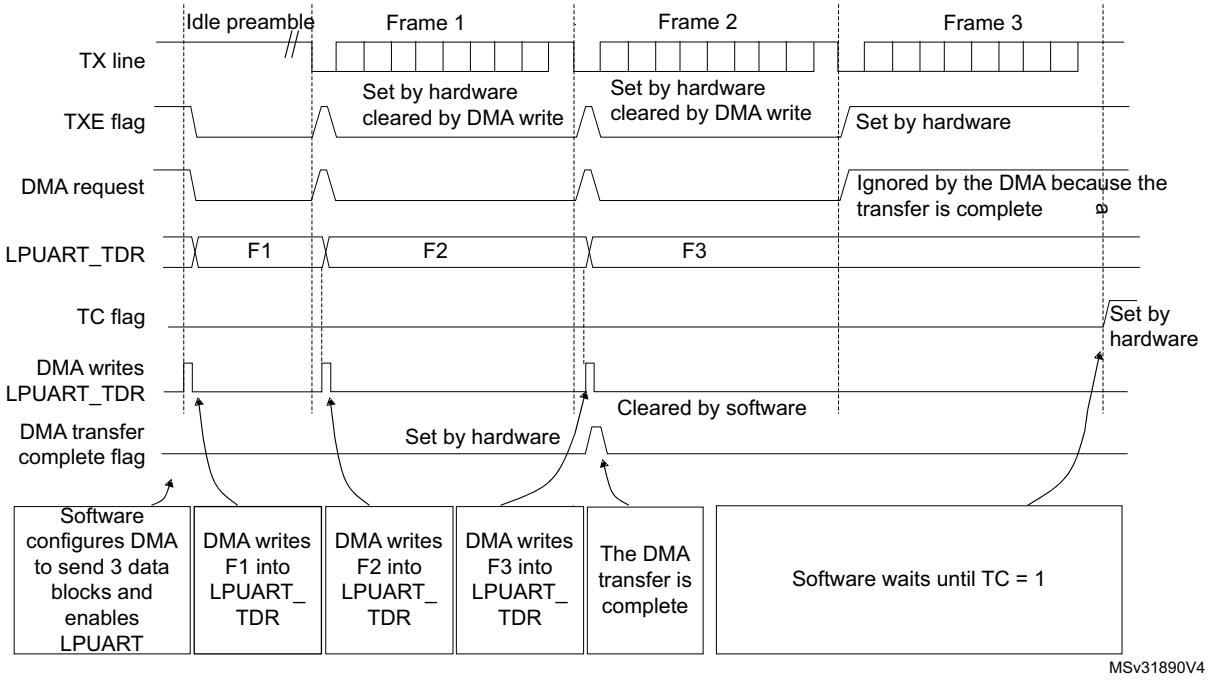
1. Write the LPUART_TDR register address in the DMA control register to configure it as the destination of the transfer. The data is moved to this address from memory after each TXE (or TXFNF if FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the source of the transfer. The data is loaded into the LPUART_TDR register from this memory area after each TXE (or TXFNF if FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA register
5. Configure DMA interrupt generation after half/ full transfer as required by the application.
6. Clear the TC flag in the LPUART_ISR register by setting the TCCF bit in the LPUART_ICR register.
7. Activate the channel in the DMA register.

When the number of data transfers programmed in the DMA controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

In transmission mode, once the DMA has written all the data to be transmitted (DMA transfer complete), the TC flag can be monitored to make sure that the LPUART communication has completed. This is required to avoid corrupting the last transmission before disabling the LPUART or entering low-power mode. Software must wait until TC = 1. The TC flag remains cleared during all data transfers and it is set by hardware at the end of transmission of the last frame.

Note: The DMAT bit must not be cleared before the DMA end of transfer.

Figure 873. Transmission using DMA



Note: When FIFO management is enabled, the DMA request is triggered by transmit FIFO not full (that is, TXFNF = 1).

Reception using DMA

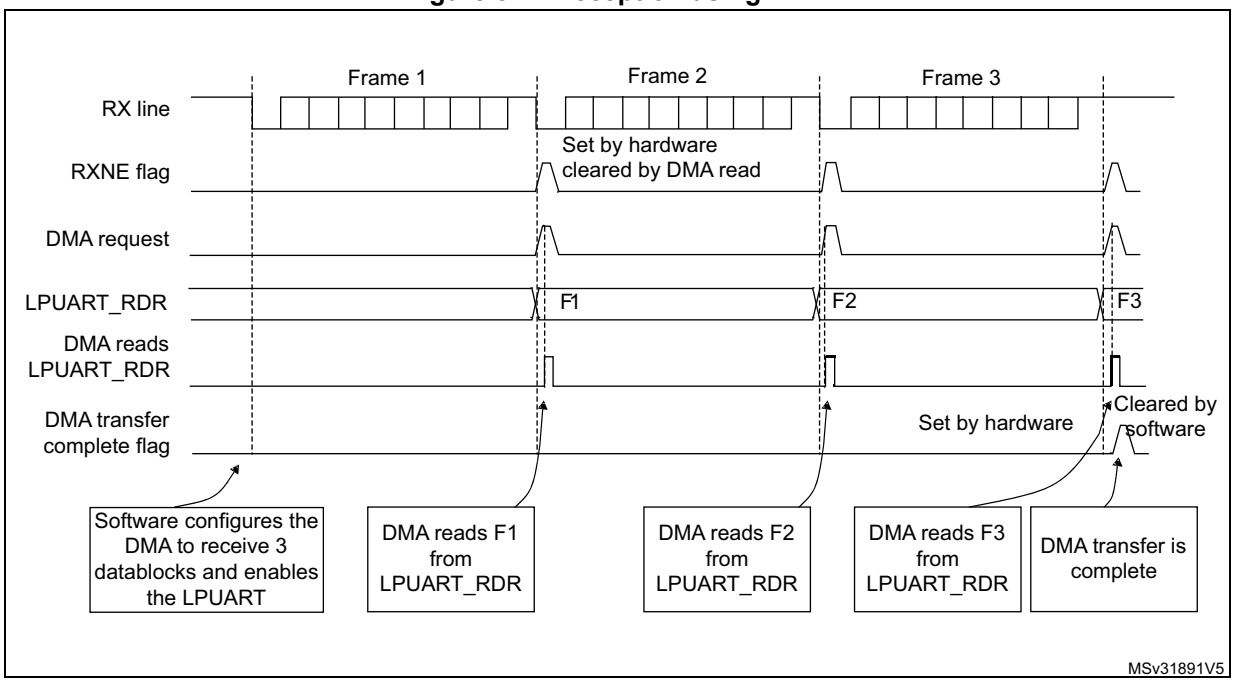
DMA mode can be enabled for reception by setting the DMAR bit in the LPUART_CR3 register. Data are loaded from the LPUART_RDR register to a SRAM area configured using the DMA peripheral (refer to section *direct memory access controller (DMA)*) whenever a data byte is received. To map a DMA channel for LPUART reception, use the following procedure:

1. Write the LPUART_RDR register address in the DMA control register to configure it as the source of the transfer. The data is moved from this address to the memory after each RXNE (RXFNE in case FIFO mode is enabled) event.
2. Write the memory address in the DMA control register to configure it as the destination of the transfer. The data is loaded from LPUART_RDR to this memory area after each RXNE (RXFNE in case FIFO mode is enabled) event.
3. Configure the total number of bytes to be transferred to the DMA control register.
4. Configure the channel priority in the DMA control register
5. Configure interrupt generation after half/ full transfer as required by the application.
6. Activate the channel in the DMA control register.

When the number of data transfers programmed in the DMA controller is reached, the DMA controller generates an interrupt on the DMA channel interrupt vector.

Note: The DMAR bit must not be cleared before the DMA end of transfer.

Figure 874. Reception using DMA



Note: When FIFO management is enabled, the DMA request is triggered by receive FIFO not empty (that is, RXFNE = 1).

Error flagging and interrupt generation in multibuffer communication

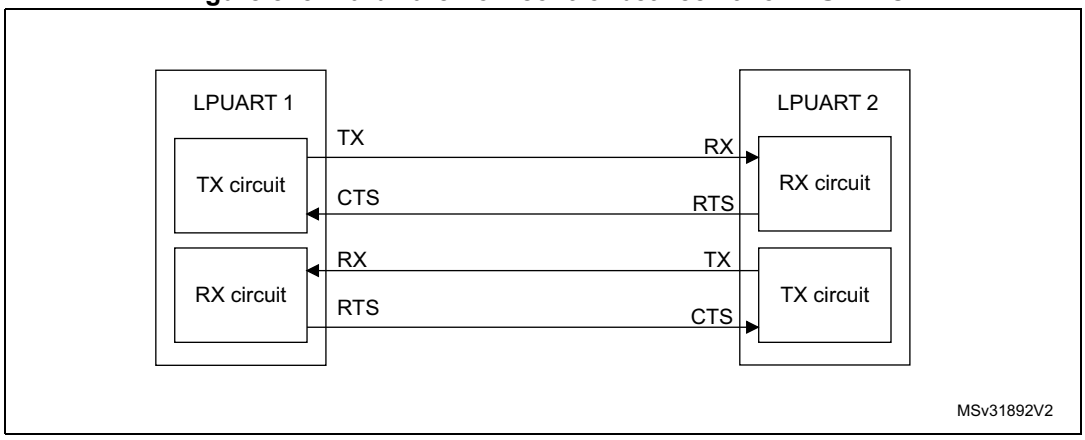
If any error occurs during a transaction In multibuffer communication mode, the error flag is asserted after the current byte. An interrupt is generated if the interrupt enable flag is set.

For framing error, overrun error and noise flag, which are asserted with RXNE (RXFNE in case FIFO mode is enabled) in single byte reception, there is a separate error flag interrupt enable bit (EIE bit in the LPUART_CR3 register), which, if set, enables an interrupt after the current byte if any of these errors occur.

66.4.14 RS232 hardware flow control and RS485 driver enable

It is possible to control the serial data flow between two devices by using the CTS input and the RTS output. *Figure 875* shows how to connect two devices in this mode.

Figure 875. Hardware flow control between two LPUARTs

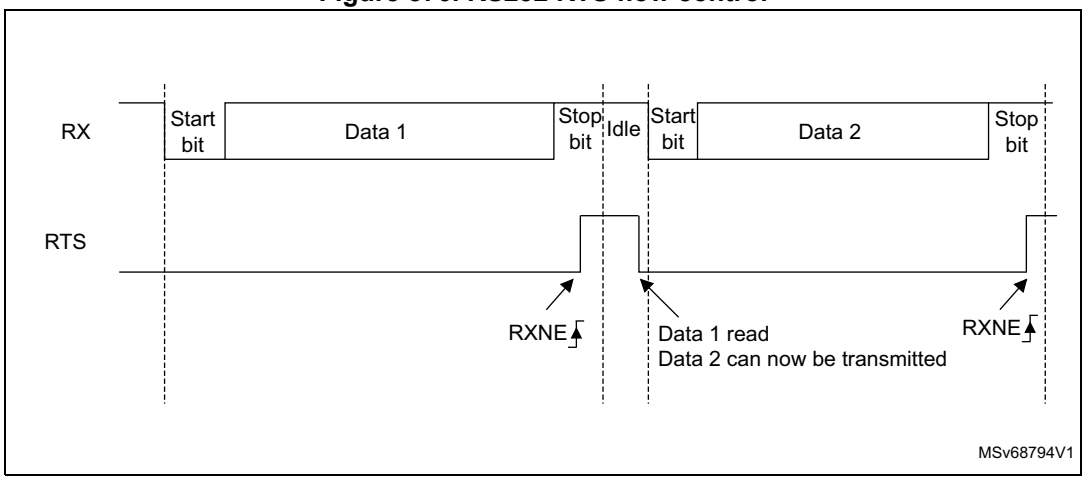


RS232 RTS and CTS flow control can be enabled independently by writing the RTSE and CTSE bits respectively to 1 (in the LPUART_CR3 register).

RS232 RTS flow control

If the RTS flow control is enabled (RTSE = 1), then RTS is deasserted (tied low) as long as the LPUART receiver is ready to receive a new data. When the receive register is full, RTS is asserted, indicating that the transmission is expected to stop at the end of the current frame. *Figure 876* shows an example of communication with RTS flow control enabled.

Figure 876. RS232 RTS flow control



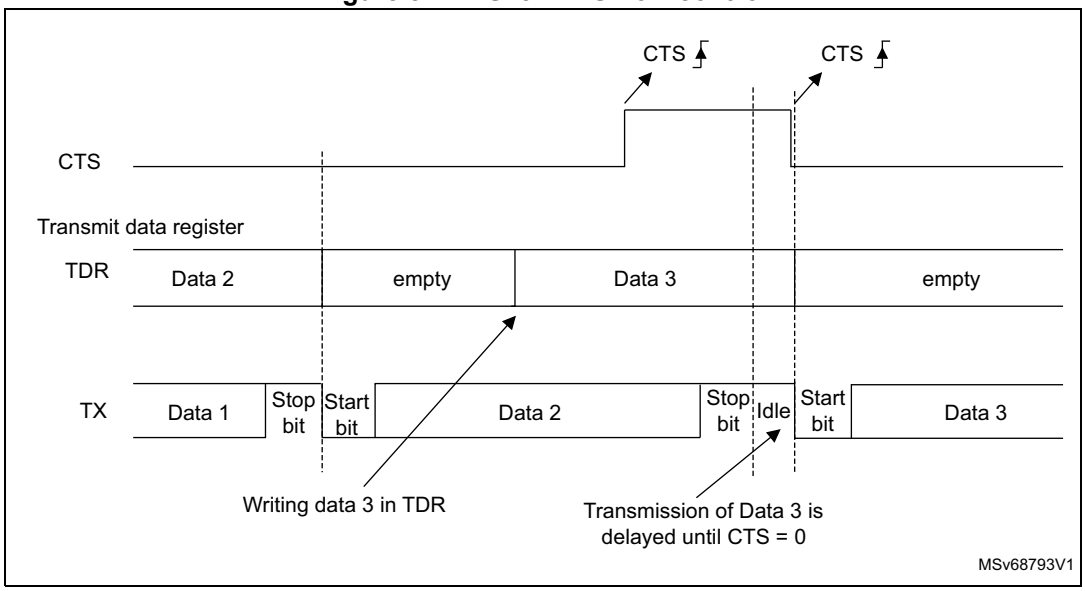
Note: When FIFO mode is enabled, RTS is asserted only when RXFIFO is full.

RS232 CTS flow control

If the CTS flow control is enabled (CTSE = 1), then the transmitter checks the CTS input before transmitting the next frame. If CTS is deasserted (tied low), then the next data is transmitted (assuming that data is to be transmitted, in other words, if TXE/TXFE = 0), else the transmission does not occur. When CTS is asserted during a transmission, the current transmission completes before the transmitter stops.

When CTSE = 1, the CTSIF status bit is automatically set by hardware as soon as the CTS input toggles. It indicates when the receiver becomes ready or not ready for communication. An interrupt is generated if the CTSIE bit in the LPUART_CR3 register is set. *Figure 877* shows an example of communication with CTS flow control enabled.

Figure 877. RS232 CTS flow control



Note: For correct behavior, CTS must be deasserted at least 3 LPUART clock source periods before the end of the current character. In addition it must be noted that the CTSCF flag may not be set for pulses shorter than 2 x PCLK periods.

RS485 driver enable

The driver enable feature is enabled by setting bit DEM in the LPUART_CR3 control register. This enables activating the external transceiver control, through the DE (Driver Enable) signal. The assertion time is the time between the activation of the DE signal and the beginning of the start bit. It is programmed using the DEAT [4:0] bitfields in the LPUART_CR1 control register. The deassertion time is the time between the end of the last stop bit, in a transmitted message, and the deactivation of the DE signal. It is programmed using the DEDT [4:0] bitfields in the LPUART_CR1 control register. The polarity of the DE signal can be configured using the DEP bit in the LPUART_CR3 control register.

The LPUART DEAT and DEDT are expressed in LPUART clock source ($f_{lpuart_ker_ck_pres}$) cycles:

- The driver enable assertion time equals
  - (1 + (DEAT x P)) x lpuart_ker_ck_pres, if P # 0
  - (1 + DEAT) x lpuart_ker_ck_pres, if P = 0
- The driver enable deassertion time equals
  - (1 + (DEDT x P)) x lpuart_ker_ck_pres, if P # 0
  - (1 + DEDT) x lpuart_ker_ck_pres, if P = 0

where P = BRR[20:11]

66.4.15 LPUART low-power management

The LPUART has advanced low-power mode functions that enable it to transfer properly data even when the lpuart_pclk clock is disabled.
The LPUART is able to wake up the MCU from low-power mode when the UESM bit is set.
When the lpuart_pclk is gated, the LPUART provides a wake-up interrupt (**lpuart_wkup**) if a specific action requiring the activation of the **lpuart_pclk** clock is needed:

- If FIFO mode is disabled

  luart_pclk clock has to be activated to empty the LPUART data register.

  In this case, the lpuart_wkup interrupt source is the RXNE set to 1. The RXNEIE bit must be set before entering low-power mode.

- If FIFO mode is enabled

  luart_pclk clock has to be activated
  - To fill the TXFIFO, or
  - To empty the RXFIFO

  In this case, the lpuart_wkup interrupt source can be:
  - RXFIFO not empty. In this case, the RXFNEIE bit must be set before entering low-power mode.
  - RXFIFO full. In this case, the RXFFIE bit must be set before entering low-power mode, the number of received data corresponds to the RXFIFO size, and the RXFF flag is not set.
  - TXFIFO empty. In this case, the TXFEIE bit must be set before entering low-power mode.

  This enables sending/receiving the data in the TXFIFO/RXFIFO during low-power mode.

  To avoid overrun/underrun errors and transmit/receive data in low-power mode, the lpuart_wkup interrupt source can be one of the following events:
  - TXFIFO threshold reached. In this case, the TXFTIE bit must be set before entering low-power mode.
  - RXFIFO threshold reached. In this case, the RXFTIE bit must be set before entering low-power mode.

  For example, the application can set the threshold to the maximum RXFIFO size if the wake-up time is less than the time to receive a single byte across the line.

  Using the RXFIFO full, TXFIFO empty, RXFIFO not empty and RXFIFO/TXFIFO threshold interrupts to wake up the MCU from low-power mode enables doing as many

LPUART transfers as possible during low-power mode with the benefit of optimizing consumption.

Alternatively, a specific **lpuart_wkup** interrupt may be selected through the WUS bitfields.

When the wake-up event is detected, the WUF flag is set by hardware and the **lpuart_wkup** interrupt is generated if the WUFIE bit is set.

Note: Before entering low-power mode, make sure that no LPUART transfer is ongoing. Checking the BUSY flag cannot ensure that low-power mode is never entered when data reception is ongoing.

The WUF flag is set when a wake-up event is detected, independently of whether the MCU is in low-power or in an active mode.

When entering low-power mode just after having initialized and enabled the receiver, the REACK bit must be checked to ensure that the LPUART is actually enabled.

When DMA is used for reception, it must be disabled before entering low-power mode and reenabled upon exit from low-power mode.

When FIFO is enabled, the wake-up from low-power mode on address match is only possible when mute mode is enabled.

Using mute mode with low-power mode

If the LPUART is put into mute mode before entering low-power mode:

- Wake-up from mute mode on idle detection must not be used, because idle detection cannot work in low-power mode.
- If the wake-up from mute mode on address match is used, then the low-power mode wake-up source from must also be the address match. If the RXNE flag was set when entering the low-power mode, the interface remains in mute mode upon address match and wake up from low-power mode.

Note: When FIFO management is enabled, mute mode is used with wake-up from low-power mode without any constraints (that is, the two points mentioned above about mute and low-power mode are valid only when FIFO management is disabled).

Wake-up from low-power mode when LPUART kernel clock lpuart_ker_ck is OFF in low-power mode

If during low-power mode, the lpuart_ker_ck clock is switched OFF, when a falling edge on the LPUART receive line is detected, the LPUART interface requests the lpuart_ker_ck clock to be switched ON thanks to the lpuart_ker_ck_req signal. The lpuart_ker_ck is then used for the frame reception.

If the wake-up event is verified, the MCU wakes up from low-power mode and data reception goes on normally.

If the wake-up event is not verified, the lpuart_ker_ck is switched OFF again, the MCU is not woken up and stays in low-power mode and the kernel clock request is released.

The example below shows the case of a wake-up event programmed to “address match detection” and FIFO management disabled.

Figure 878 shows the behavior when the wake-up event is verified.

Figure 878. Wake-up event verified (wake-up event = address match, FIFO disabled)

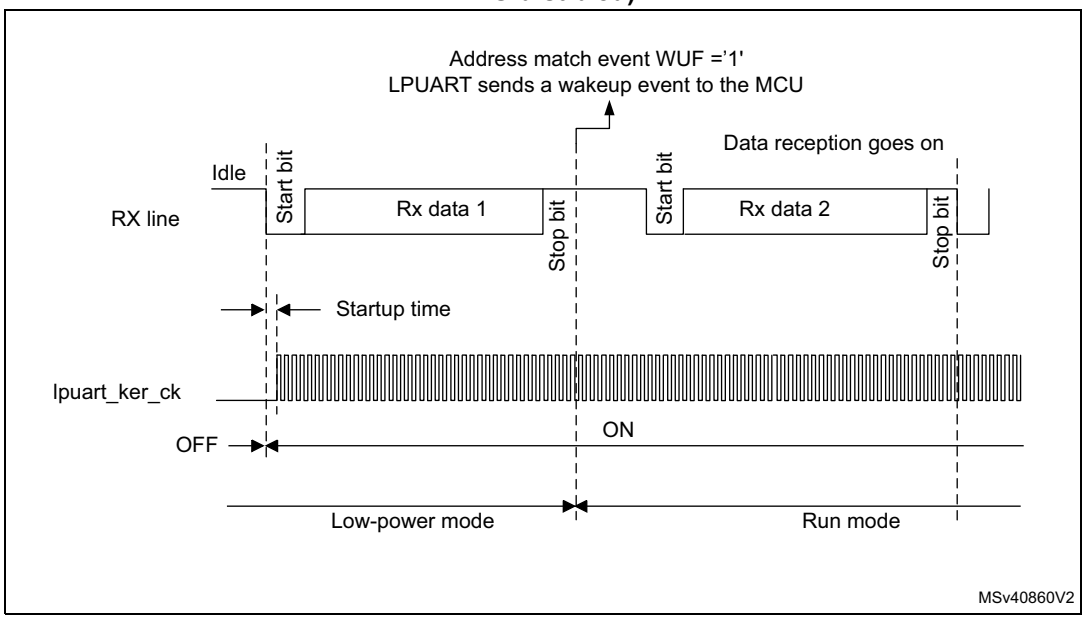
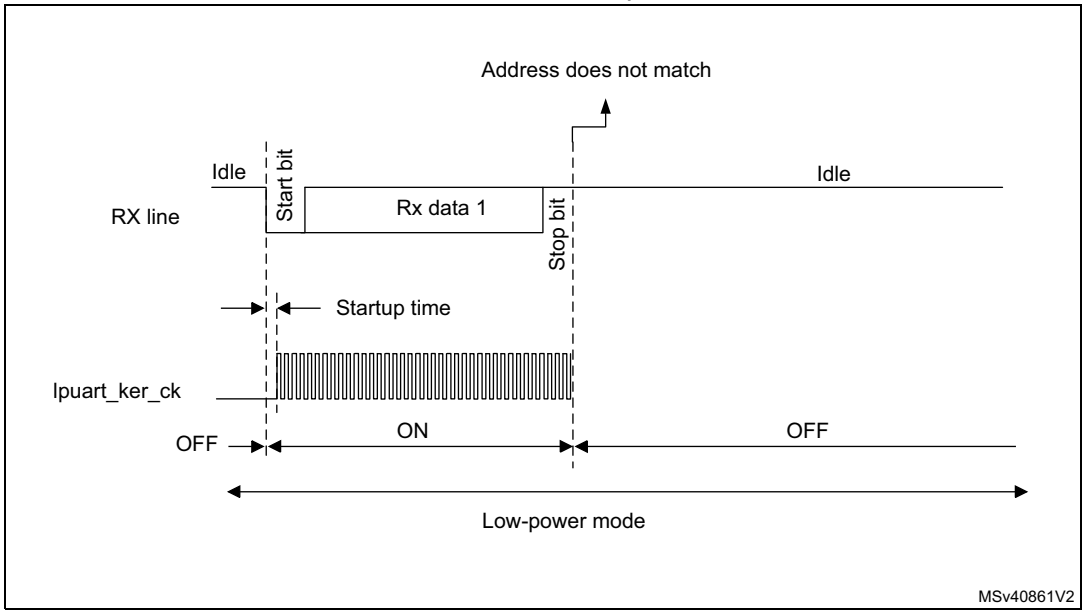


Figure 879 shows the behavior when the wake-up event is not verified.

Figure 879. Wake-up event not verified (wake-up event = address match, FIFO disabled)



Note: The above figures are valid when address match or any received frame is used as wake-up event. In the case the wake-up event is the start bit detection, the LPUART sends the wake-up event to the MCU at the end of the start bit.

Determining the maximum LPUART baud rate that enables to correctly wake up the MCU from low-power mode

The maximum baud rate that enables to correctly wake up the MCU from low-power mode depends on the wake-up time parameter (refer to the device datasheet) and on the LPUART receiver tolerance (see *Section 66.4.9: Tolerance of the LPUART receiver to clock deviation*).

Let us take the example of OVER8 = 0, M bits = 01, ONEBIT = 0 and BRR [3:0] = 0000.

In these conditions, according to *Table 652: Tolerance of the LPUART receiver*, the LPUART receiver tolerance equals 3.41%.

DTRA + DQUANT + DREC + DTCL + DWU < LPUART receiver tolerance

$D_{WUmax} = t_{WULPUART} / (11 \times T_{bit\ Min})$

$T_{bit\ Min} = t_{WULPUART} / (11 \times D_{WUmax})$

where $t_{WULPUART}$ is the wake-up time from low-power mode.

If we consider the ideal case where DTRA, DQUANT, DREC, and DTCL parameters are at 0%, the maximum value of DWU is 3.41%. In reality, we need to consider at least the lpuart_ker_ck inaccuracy.

For example, if HSI is used as lpuart_ker_ck, and the HSI inaccuracy is of 1%, then we obtain:

$t_{WULPUART}$ = 3 µs (values provided only as examples; for correct values, refer to the device datasheet).

D_{WUmax} = 3.41% – 1% = 2.41%

$T_{bit\ min}$ = 3 µs/ (11 x 2.41%) = 11.32 µs.

As a result, the maximum baud rate that enables to wake up correctly from low-power mode is: 1/11.32 µs = 88.36 kbauds.

66.5 LPUART in low-power modes

Table 654. Effect of low-power modes on the LPUART

Mode	Description
Sleep	No effect. LPUART interrupts cause the device to exit Sleep mode.
Stop(1)	The content of the LPUART registers is kept. The LPUART is able to wake up the microcontroller from Stop mode when the LPUART is clocked by an oscillator available in Stop mode.
Standby	The LPUART peripheral is powered down and must be reinitialized after exiting Standby mode.

1. Refer to *Section 66.3: LPUART implementation* to know if the wake-up from Stop mode is supported for a given peripheral instance. If an instance is not functional in a given Stop mode, it must be disabled before entering this Stop mode.

66.6 LPUART interrupts

Refer to *Table 655* for a detailed description of all LPUART interrupt requests.

Table 655. LPUART interrupt requests

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Interrupt clear method	Exit from Sleep mode	Exit from Stop(1) modes	Exit from Standby mode
LPUART	Transmit data register empty	TXE	TXEIE	Write TDR	Yes	No	No
	Transmit FIFO Not Full	TXFNF	TXFNFIE	TXFIFO full		No	
	Transmit FIFO Empty	TXFE	TXFEIE	Write TDR or write 1 in TXFRQ		Yes	
	Transmit FIFO threshold reached	TXFT	TXFTIE	Write TDR		Yes	
	CTS interrupt	CTSIF	CTSIE	Write 1 in CTSCF		No	
	Transmission Complete	TC	TCIE	Write TDR or write 1 in TCCF		No	
	Receive data register not empty (data ready to be read)	RXNE	RXNEIE	Read RDR or write 1 in RXFRQ	Yes	Yes	
	Receive FIFO Not Empty	RXFNE	RXFNEIE	Read RDR until RXFIFO empty or write 1 in RXFRQ		Yes	
	Receive FIFO Full	RXFF(2)	RXFFIE	Read RDR		Yes	
	Receive FIFO threshold reached	RXFT	RXFTIE	Read RDR		Yes	
	Overrun error detected	ORE	RX-NEIE/RX FNEIE	Write 1 in ORECF		No	
	Idle line detected	IDLE	IDLEIE	Write 1 in IDLECF		No	
	Parity error	PE	PEIE	Write 1 in PECF		No	
	Noise error in multibuffer communication.	NE	EIE	Write 1 in NFCF		No	
	Overrun error in multibuffer communication.	ORE(3)		Write 1 in ORECF		No	
	Framing Error in multibuffer communication.	FE		Write 1 in FECF		No	
	Character match	CMF	CMIE	Write 1 in CMCF		No	
	Wake-up from low-power mode	WUF	WUFIE	Write 1 in WUC		Yes	

1. The LPUART can wake up the device from Stop mode only if the peripheral instance supports the wake-up from Stop mode feature. Refer to *Section 66.3: LPUART implementation* for the list of supported Stop modes.
2. RXFF flag is asserted if the LPUART receives n+1 data (n being the RXFIFO size): n data in the RXFIFO and 1 data in LPUART_RDR. In Stop mode, LPUART_RDR is not clocked. As a result, this register is not written and once n data are received and written in the RXFIFO, the RXFF interrupt is asserted (RXFF flag is not set).
3. When OVRDIS = 0.

66.7 LPUART registers

Refer to *Section 1.2: List of abbreviations for registers* for a list of abbreviations used in register descriptions.

The peripheral registers have to be accessed by words (32 bits).

66.7.1 LPUART control register 1 (LPUART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enabled, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXFFIE	TXFEIE	FIFOEN	M1	Res.	Res.	DEAT[4:0]					DEDT[4:0]				
rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXFNFIE	TCIE	RXFNEIE	IDLEIE	TE	RE	UESM	UE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RXFFIE:**RXFIFO Full interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated when RXFF = 1 in the LPUART_ISR register

Bit 30 **TXFEIE:**TXFIFO empty interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated when TXFE = 1 in the LPUART_ISR register

Bit 29 **FIFOEN:**FIFO mode enable
This bit is set and cleared by software.
0: FIFO mode is disabled.
1: FIFO mode is enabled.

Bit 28 **M1**: Word length

This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.

M[1:0] = 00: 1 Start bit, 8 Data bits, n stop bits

M[1:0] = 01: 1 Start bit, 9 Data bits, n stop bits

M[1:0] = ‘10: 1 Start bit, 7 Data bits, n stop bits

This bit can only be written when the LPUART is disabled (UE = 0).

Note: In 7-bit data length mode, the smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in lpuart_ker_ck clock cycles. For more details, refer *Section 65.5.21: RS232 hardware flow control and RS485 driver enable*.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal.It is expressed in lpuart_ker_ck clock cycles. For more details, refer *Section 66.4.14: RS232 hardware flow control and RS485 driver enable*.

If the LPUART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated when the CMF bit is set in the LPUART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit activates the mute mode function of the LPUART. When set, the LPUART can switch between the active and mute modes, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1) description).

This bit can only be written when the LPUART is disabled (UE = 0).

Bit 11 **WAKE**: Receiver wake-up method

This bit determines the LPUART wake-up method from mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M = 0) and parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.

0: Even parity

1: Odd parity

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever PE = 1 in the LPUART_ISR register

Bit 7 **TXFNFIE**: TXFIFO not full interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever TXFNF = 1 in the LPUART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever TC = 1 in the LPUART_ISR register

Bit 5 **RXFNEIE**: RXFIFO not empty interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever ORE = 1 or RXFNE = 1 in the LPUART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever IDLE = 1 in the LPUART_ISR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. It is set and cleared by software.

0: Transmitter is disabled

1: Transmitter is enabled

Note: When the LPUART acts as a transmitter, a low pulse on the TE bit (0 followed by 1) sends a preamble (idle line) after the current word, except in smartcard mode. In order to generate an idle character, the TE must not be immediately written to 1. To ensure the required duration, the software can poll the TEACK bit in the LPUART_ISR register.

In smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: LPUART enable in low-power mode

When this bit is cleared, the LPUART cannot wake up the MCU from low-power mode.

When this bit is set, the LPUART can wake up the MCU from low-power mode.

This bit is set and cleared by software.

0: LPUART not able to wake up the MCU from low-power mode.

1: LPUART able to wake up the MCU from low-power mode.

Note: It is recommended to set the UESM bit just before entering low-power mode, and clear it when exiting low-power mode.

Bit 0 **UE**: LPUART enable

When this bit is cleared, the LPUART prescalers and outputs are stopped immediately, and current operations are discarded. The configuration of the LPUART is kept, but all the status flags, in the LPUART_ISR are reset. This bit is set and cleared by software.

0: LPUART prescaler and outputs disabled, low-power mode

1: LPUART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be reset before and the software must wait for the TC bit in the LPUART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

66.7.2 LPUART control register 1 [alternate] (LPUART_CR1)

Address offset: 0x00

Reset value: 0x0000 0000

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	FIFO EN	M1	Res.	Res.	DEAT[4:0]					DEDT[4:0]				
		rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **FIFOEN:**FIFO mode enable

This bit is set and cleared by software.

0: FIFO mode is disabled.

1: FIFO mode is enabled.

Bit 28 **M1**: Word length

This bit must be used in conjunction with bit 12 (M0) to determine the word length. It is set or cleared by software.

M[1:0] = 00: 1 Start bit, 8 Data bits, n stop bits

M[1:0] = 01: 1 Start bit, 9 Data bits, n stop bits

M[1:0] = ‘10: 1 Start bit, 7 Data bits, n stop bits

This bit can only be written when the LPUART is disabled (UE = 0).

Note: In 7-bit data length mode, the smartcard mode, LIN master mode and auto baud rate (0x7F and 0x55 frames detection) are not supported.

Bits 27:26 Reserved, must be kept at reset value.

Bits 25:21 **DEAT[4:0]**: Driver Enable assertion time

This 5-bit value defines the time between the activation of the DE (Driver Enable) signal and the beginning of the start bit. It is expressed in lpuart_ker_ck clock cycles. For more details, refer *Section 65.5.21: RS232 hardware flow control and RS485 driver enable*.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bits 20:16 **DEDT[4:0]**: Driver Enable deassertion time

This 5-bit value defines the time between the end of the last stop bit, in a transmitted message, and the de-activation of the DE (Driver Enable) signal.It is expressed in lpuart_ker_ck clock cycles. For more details, refer *Section 66.4.14: RS232 hardware flow control and RS485 driver enable*.

If the LPUART_TDR register is written during the DEDT time, the new data is transmitted only when the DEDT and DEAT times have both elapsed.

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 15 Reserved, must be kept at reset value.

Bit 14 **CMIE**: Character match interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated when the CMF bit is set in the LPUART_ISR register.

Bit 13 **MME**: Mute mode enable

This bit activates the mute mode function of the LPUART. When set, the LPUART can switch between the active and mute modes, as defined by the WAKE bit. It is set and cleared by software.

0: Receiver in active mode permanently

1: Receiver can switch between mute mode and active mode.

Bit 12 **M0**: Word length

This bit is used in conjunction with bit 28 (M1) to determine the word length. It is set or cleared by software (refer to bit 28 (M1) description).

This bit can only be written when the LPUART is disabled (UE = 0).

Bit 11 **WAKE**: Receiver wake-up method

This bit determines the LPUART wake-up method from mute mode. It is set or cleared by software.

0: Idle line

1: Address mark

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 10 **PCE**: Parity control enable

This bit selects the hardware parity control (generation and detection). When the parity control is enabled, the computed parity is inserted at the MSB position (9th bit if M = 1; 8th bit if M = 0) and parity is checked on the received data. This bit is set and cleared by software. Once it is set, PCE is active after the current byte (in reception and in transmission).

0: Parity control disabled

1: Parity control enabled

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 9 **PS**: Parity selection

This bit selects the odd or even parity when the parity generation/detection is enabled (PCE bit set). It is set and cleared by software. The parity is selected after the current byte.

0: Even parity

1: Odd parity

This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 8 **PEIE**: PE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever PE = 1 in the LPUART_ISR register

Bit 7 **TXEIE**: Transmit data register empty

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever TXE = 1 in the LPUART_ISR register

Bit 6 **TCIE**: Transmission complete interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever TC = 1 in the LPUART_ISR register

Bit 5 **RXNEIE**: Receive data register not empty interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: A LPUART interrupt is generated whenever ORE = 1 or RXNE = 1 in the LPUART_ISR register

Bit 4 **IDLEIE**: IDLE interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited

1: An LPUART interrupt is generated whenever IDLE = 1 in the LPUART_ISR register

Bit 3 **TE**: Transmitter enable

This bit enables the transmitter. It is set and cleared by software.

0: Transmitter is disabled

1: Transmitter is enabled

Note: When the LPUART acts as a transmitted, a low pulse on the TE bit (0 followed by 1) sends a preamble (idle line) after the current word, except in smartcard mode. In order to generate an idle character, the TE must not be immediately written to '1. To ensure the required duration, the software can poll the TEACK bit in the LPUART_ISR register.

In smartcard mode, when TE is set, there is a 1 bit-time delay before the transmission starts.

Bit 2 **RE**: Receiver enable

This bit enables the receiver. It is set and cleared by software.

0: Receiver is disabled

1: Receiver is enabled and begins searching for a start bit

Bit 1 **UESM**: LPUART enable in low-power mode

When this bit is cleared, the LPUART cannot wake up the MCU from low-power mode.

When this bit is set, the LPUART can wake up the MCU from low-power mode.

This bit is set and cleared by software.

0: LPUART not able to wake up the MCU from low-power mode.

1: LPUART able to wake up the MCU from low-power mode.

Note: It is recommended to set the UESM bit just before entering low-power mode, and clear it when exiting low-power mode.

Bit 0 **UE**: LPUART enable

When this bit is cleared, the LPUART prescalers and outputs are stopped immediately, and current operations are discarded. The configuration of the LPUART is kept, but all the status flags, in the LPUART_ISR are reset. This bit is set and cleared by software.

0: LPUART prescaler and outputs disabled, low-power mode

1: LPUART enabled

Note: To enter low-power mode without generating errors on the line, the TE bit must be reset before and the software must wait for the TC bit in the LPUART_ISR to be set before resetting the UE bit.

The DMA requests are also reset when UE = 0 so the DMA channel must be disabled before resetting the UE bit.

66.7.3 LPUART control register 2 (LPUART_CR2)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD[7:0]								Res.	Res.	Res.	Res.	MSBFIRST	DATAINV	TXINV	RXINV
rw	rw	rw	rw	rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWAP	Res.	STOP[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDM7	Res.	Res.	Res.	Res.
rw		rw	rw								rw				

Bits 31:24 **ADD[7:0]**: Address of the LPUART node

These bits give the address of the LPUART node in mute mode or a character code to be recognized in low-power or Run mode:

- In mute mode: they are used in multiprocessor communication to wake up from mute mode with 4-bit/7-bit address mark detection. The MSB of the character sent by the transmitter is equal to 1. In 4-bit address mark detection, only ADD[3:0] bits are used.
- In low-power mode: they are used for wake up from low-power mode on character match. When WUS[1:0] is programmed to 0b00 (WUF active on address match), the wake-up from low-power mode is performed when the received character corresponds to the character programmed through ADD[6:0] or ADD[3:0] bitfield (depending on ADDM7 bit), and WUF interrupt is enabled by setting WUFIE bit. The MSB of the character sent by transmitter is equal to 1.
- In Run mode with mute mode inactive (for example, end-of-block detection in ModBus protocol): the whole received character (8 bits) is compared to ADD[7:0] value and CMF flag is set on match. An interrupt is generated if the CMIE bit is set.

These bits can only be written when the reception is disabled (RE = 0) or when the LPUART is disabled (UE = 0).

Bits 23:20 Reserved, must be kept at reset value.

Bit 19 **MSBFIRST**: Most significant bit first

This bit is set and cleared by software.
0: data is transmitted/received with data bit 0 first, following the start bit.
1: data is transmitted/received with the MSB (bit 7/8) first, following the start bit.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 18 **DATAINV:** Binary data inversion

This bit is set and cleared by software.
0: Logical data from the data register are send/received in positive/direct logic. (1=H, 0=L)
1: Logical data from the data register are send/received in negative/inverse logic. (1=L, 0=H). The parity bit is also inverted.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 17 **TXINV:** TX pin active level inversion

This bit is set and cleared by software.
0: TX pin signal works using the standard logic levels (V_{DD} = 1 / idle, Gnd = 0 / mark)
1: TX pin signal values are inverted. ((V_{DD} = 0 / mark, Gnd = 1 / idle).
This enables the use of an external inverter on the TX line.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 16 **RXINV:** RX pin active level inversion

This bit is set and cleared by software.
0: RX pin signal works using the standard logic levels (V_{DD} = 1 / idle, Gnd = 0/mark)
1: RX pin signal values are inverted. ((V_{DD} = 0/mark, Gnd = 1 / idle).
This enables the use of an external inverter on the RX line.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 15 **SWAP:** Swap TX/RX pins

This bit is set and cleared by software.
0: TX/RX pins are used as defined in standard pinout
1: The TX and RX pins functions are swapped. This enables to work in the case of a cross-wired connection to another UART.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bit 14 Reserved, must be kept at reset value.

Bits 13:12 **STOP[1:0]**: Stop bits
These bits are used for programming the stop bits.
00: 1 stop bit
01: Reserved.
10: 2 stop bits
11: Reserved
This bitfield can only be written when the LPUART is disabled (UE = 0).

Bits 11:5 Reserved, must be kept at reset value.

Bit 4 **ADDM7**:7-bit Address Detection/4-bit Address Detection
This bit is for selection between 4-bit address detection or 7-bit address detection.
0: 4-bit address detection
1: 7-bit address detection (in 8-bit data mode)
This bit can only be written when the LPUART is disabled (UE = 0)
Note: In 7-bit and 9-bit data modes, the address detection is done on 6-bit and 8-bit address (ADD[5:0] and ADD[7:0]) respectively.

Bits 3:0 Reserved, must be kept at reset value.

66.7.4 LPUART control register 3 (LPUART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

FIFO mode enabled, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXFTCFG[2:0]			RXFTIE	RXFTCFG[2:0]			Res.	TXFTIE	WUFIE	WUS1	WUS0	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVRDIS	Res.	CTSIE	CTSE	RTSE	DMAT	DMAR	Res.	Res.	HDSEL	Res.	Res.	EIE
rw	rw	rw	rw		rw	rw	rw	rw	rw			rw			rw

Bits 31:29 **TXFTCFG[2:0]:** TXFIFO threshold configuration
000:TXFIFO reaches 1/8 of its depth.
001:TXFIFO reaches 1/4 of its depth.
110:TXFIFO reaches 1/2 of its depth.
011:TXFIFO reaches 3/4 of its depth.
100:TXFIFO reaches 7/8 of its depth.
101:TXFIFO becomes empty.
Others: Reserved, must not be used.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 28 **RXFTIE**: RXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: An LPUART interrupt is generated when Receive FIFO reaches the threshold programmed in RXFTCFG[2:0].

Bits 27:25 **RXFTCFG[2:0]:** Receive FIFO threshold configuration
000:Receive FIFO reaches 1/8 of its depth.
001:Receive FIFO reaches 1/4 of its depth.
110:Receive FIFO reaches 1/2 of its depth.
011:Receive FIFO reaches 3/4 of its depth.
100:Receive FIFO reaches 7/8 of its depth.
101:Receive FIFO becomes full.
Others: Reserved, must not be used.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 24 Reserved, must be kept at reset value.

Bit 23 **TXFTIE**: TXFIFO threshold interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited
1: A LPUART interrupt is generated when TXFIFO reaches the threshold programmed in TXFTCFG[2:0].

Bit 22 **WUFIE**: Wake-up from low-power mode interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: LPUART interrupt generated whenever WUF = 1 in the LPUART_ISR register

Note: WUFIE must be set before entering in low-power mode.
If the LPUART *does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 66.3: LPUART implementation.*

Bits 21:20 **WUS[1:0]**: Wake-up from low-power mode interrupt flag selection
This bitfield specifies the event which activates the WUF (wake-up from low-power mode flag).
00: WUF active on address match (as defined by ADD[7:0] and ADDM7)
01: Reserved.
10: WUF active on start bit detection
11: WUF active on RXNE/RXFNE.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Note: If the LPUART *does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 66.3: LPUART implementation.*

Bits 19:16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection
0: DE signal is active high.
1: DE signal is active low.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 14 **DEM**: Driver enable mode
This bit enables the user to activate the external transceiver control, through the DE signal.
0: DE function is disabled.
1: DE function is enabled. The DE signal is output on the RTS pin.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 13 **DDRE**: DMA Disable on reception Error

0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred.

1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE before clearing the error flag.

This bit can only be written when the LPUART is disabled (UE = 0).

Note: The reception errors are: parity error, framing error or noise error.

Bit 12 **OVRDIS**: Overrun Disable

This bit is used to disable the receive overrun detection.

0: Overrun Error Flag, ORE is set when received data is not read before receiving new data.

1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the LPUART_RDR register.

This bit can only be written when the LPUART is disabled (UE = 0).

Note: This control bit enables *checking the communication flow w/o reading the data.*

Bit 11 Reserved, must be kept at reset value.

Bit 10 **CTSIE**: CTS interrupt enable

0: Interrupt is inhibited

1: An interrupt is generated whenever CTSIF = 1 in the LPUART_ISR register

Bit 9 **CTSE**: CTS enable

0: CTS hardware flow control disabled

1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0). If the CTS input is asserted while data is being transmitted, then the transmission completes before stopping. If data is written into the data register while CTS is asserted, the transmission is postponed until CTS is deasserted.

This bit can only be written when the LPUART is disabled (UE = 0)

Bit 8 **RTSE**: RTS enable

0: RTS hardware flow control disabled

1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The RTS output is deasserted (pulled to 0) when data can be received.

This bit can only be written when the LPUART is disabled (UE = 0).

Bit 7 **DMAT**: DMA enable transmitter

This bit is set/reset by software

1: DMA mode is enabled for transmission

0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver

This bit is set/reset by software

1: DMA mode is enabled for reception

0: DMA mode is disabled for reception

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **HDSEL**: Half-duplex selection
Selection of single-wire half-duplex mode
0: Half-duplex mode is not selected
1: Half-duplex mode is selected
This bit can only be written when the LPUART is disabled (UE = 0).

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **EIE**: Error interrupt enable
Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error or noise flag (FE = 1 or ORE = 1 or NE = 1 in the LPUART_ISR register).
0: Interrupt is inhibited
1: An interrupt is generated when FE = 1 or ORE = 1 or NE = 1 in the LPUART_ISR register.

66.7.5 LPUART control register 3 [alternate] (LPUART_CR3)

Address offset: 0x08

Reset value: 0x0000 0000

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUFIE	WUS1	WUS0	Res.	Res.	Res.	Res.
									rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DEP	DEM	DDRE	OVRDIS	Res.	CTSIE	CTSE	RTSE	DMAT	DMAR	Res.	Res.	HDSEL	Res.	Res.	EIE
rw	rw	rw	rw		rw	rw	rw	rw	rw			rw			rw

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **WUFIE**: Wake-up from low-power mode interrupt enable
This bit is set and cleared by software.
0: Interrupt inhibited
1: LPUART interrupt generated whenever WUF = 1 in the LPUART_ISR register

Note: WUFIE must be set before entering in low-power mode.
If the LPUART *does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 66.3: LPUART implementation.*

Bits 21:20 **WUS[1:0]**: Wake-up from low-power mode interrupt flag selection
This bitfield specifies the event which activates the WUF (wake-up from low-power mode flag).
00: WUF active on address match (as defined by ADD[7:0] and ADDM7)
01: Reserved.
10: WUF active on start bit detection
11: WUF active on RXNE/RXFNE.
This bitfield can only be written when the LPUART is disabled (UE = 0).

Note: If the LPUART *does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value. Refer to Section 66.3: LPUART implementation.*

Bits 19:16 Reserved, must be kept at reset value.

Bit 15 **DEP**: Driver enable polarity selection
0: DE signal is active high.
1: DE signal is active low.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 14 **DEM**: Driver enable mode
This bit enables the user to activate the external transceiver control, through the DE signal.
0: DE function is disabled.
1: DE function is enabled. The DE signal is output on the RTS pin.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 13 **DDRE**: DMA Disable on reception Error
0: DMA is not disabled in case of reception error. The corresponding error flag is set but RXNE is kept 0 preventing from overrun. As a consequence, the DMA request is not asserted, so the erroneous data is not transferred (no DMA request), but next correct received data is transferred.
1: DMA is disabled following a reception error. The corresponding error flag is set, as well as RXNE. The DMA request is masked until the error flag is cleared. This means that the software must first disable the DMA request (DMAR = 0) or clear RXNE before clearing the error flag.
This bit can only be written when the LPUART is disabled (UE = 0).
Note: The reception errors are: parity error, framing error or noise error.

Bit 12 **OVRDIS**: Overrun Disable
This bit is used to disable the receive overrun detection.
0: Overrun Error Flag, ORE is set when received data is not read before receiving new data.
1: Overrun functionality is disabled. If new data is received while the RXNE flag is still set the ORE flag is not set and the new received data overwrites the previous content of the LPUART_RDR register.
This bit can only be written when the LPUART is disabled (UE = 0).
Note: This control bit enables *checking the communication flow w/o reading the data.*

Bit 11 Reserved, must be kept at reset value.

Bit 10 **CTSIE**: CTS interrupt enable
0: Interrupt is inhibited
1: An interrupt is generated whenever CTSIF = 1 in the LPUART_ISR register

Bit 9 **CTSE**: CTS enable
0: CTS hardware flow control disabled
1: CTS mode enabled, data is only transmitted when the CTS input is deasserted (tied to 0). If the CTS input is asserted while data is being transmitted, then the transmission completes before stopping. If data is written into the data register while CTS is asserted, the transmission is postponed until CTS is deasserted.
This bit can only be written when the LPUART is disabled (UE = 0)

Bit 8 **RTSE**: RTS enable
0: RTS hardware flow control disabled
1: RTS output enabled, data is only requested when there is space in the receive buffer. The transmission of data is expected to cease after the current character has been transmitted. The RTS output is deasserted (pulled to 0) when data can be received.
This bit can only be written when the LPUART is disabled (UE = 0).

Bit 7 **DMAT**: DMA enable transmitter
This bit is set/reset by software
1: DMA mode is enabled for transmission
0: DMA mode is disabled for transmission

Bit 6 **DMAR**: DMA enable receiver
This bit is set/reset by software
1: DMA mode is enabled for reception
0: DMA mode is disabled for reception

Bits 5:4 Reserved, must be kept at reset value.

Bit 3 **HDSEL**: Half-duplex selection
Selection of single-wire half-duplex mode
0: Half-duplex mode is not selected
1: Half-duplex mode is selected
This bit can only be written when the LPUART is disabled (UE = 0).

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **EIE**: Error interrupt enable
Error Interrupt Enable Bit is required to enable interrupt generation in case of a framing error, overrun error or noise flag (FE = 1 or ORE = 1 or NE = 1 in the LPUART_ISR register).
0: Interrupt is inhibited
1: An interrupt is generated when FE = 1 or ORE = 1 or NE = 1 in the LPUART_ISR register.

66.7.6 LPUART baud rate register (LPUART_BRR)

This register can only be written when the LPUART is disabled (UE = 0). It may be automatically updated by hardware in auto baud rate detection mode.

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BRR[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BRR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **BRR[19:0]:** LPUART baud rate division (LPUARTDIV)

Note: It is forbidden to write values lower than 0x300 in the LPUART_BRR register.

Provided that LPUART_BRR must be ≥ 0x300 and LPUART_BRR is 20 bits, a care must be taken when generating high baud rates using high lpuart_ker_ck_pres values. lpuart_ker_ck_pres must be in the range [3 x baud rate..4096 x baud rate].

66.7.7 LPUART request register (LPUART_RQR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	Res.
											w	w	w	w	

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **TXFRQ**: Transmit data flush request

This bit is used when FIFO mode is enabled. TXFRQ bit is set to flush the whole FIFO. This sets the flag TXFE (TXFIFO empty, bit 23 in the LPUART_ISR register).

Note: In FIFO mode, the TXFNF flag is reset during the flush request until TxFIFO is empty in order to ensure that no data are written in the data register.

Bit 3 **RXFRQ**: Receive data flush request

Writing 1 to this bit clears the RXNE flag.

This enables discarding the received data without reading it, and avoid an overrun condition.

Bit 2 **MMRQ**: Mute mode request

Writing 1 to this bit puts the LPUART in Mute mode and resets the RWU flag.

Bit 1 **SBKRQ**: Send break request

Writing 1 to this bit sets the SBKF flag and request to send a BREAK on the line, as soon as the transmit machine is available.

Note: If the application needs to send the break character following all previously inserted data, including the ones not yet transmitted, the software must wait for the TXE flag assertion before setting the SBKRQ bit.

Bit 0 Reserved, must be kept at reset value.

66.7.8 LPUART interrupt and status register (LPUART_ISR)

Address offset: 0x1C

Reset value: 0x0080 00C0

The same register can be used in FIFO mode enabled (this section) and FIFO mode disabled (next section).

FIFO mode enabled, FIFOEN = 1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TXFT	RXFT	Res.	RXFF	TXFE	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY
				r	r		r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXFNF	TC	RXFNE	IDLE	ORE	NE	FE	PE
					r	r		r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **TXFT**: TXFIFO threshold flag

This bit is set by hardware when the number of empty locations in the TXFIFO is greater than the threshold programmed in the TXFTCFG[2:0] bitfield of LPUART_CR3 register.An interrupt is generated if the TXFTIE bit = 1 (bit 31) in the LPUART_CR3 register.

0: TXFIFO does not reach the programmed threshold.

1: TXFIFO reached the programmed threshold.

Bit 26 **RXFT**: RXFIFO threshold flag

This bit is set by hardware when the RXFIFO reaches the threshold programmed in the RXFTCFG[2:0] bitfield of the LPUART_CR3 register, that is, the Receive FIFO contains RXFTCFG[2:0] data. An interrupt is generated if the RXFTIE bit = 1 (bit 28) in the LPUART_CR3 register.

0: Receive FIFO does not reach the programmed threshold.

1: Receive FIFO reached the programmed threshold.

Note: When the RXFTCFG[2:0] threshold is configured to 101, the RXFT flag is set if RXFIFO size data are available, that is, (RXFIFO size – 1) data in the RXFIFO and 1 data in the LPUART_RDR. Consequently, the (RXFIFO size + 1) th received data does not cause an overrun error. The overrun error occurs after receiving the (RXFIFO size + 2) th data.

Bit 25 Reserved, must be kept at reset value.

Bit 24 **RXFF**: RXFIFO Full

This bit is set by hardware when the number of received data corresponds to RXFIFO size + 1 (RXFIFO full + 1 data in the LPUART_RDR register.

An interrupt is generated if the RXFFIE bit = 1 in the LPUART_CR1 register.

0: RXFIFO is not Full.

1: RXFIFO is Full.

Bit 23 **TXFE**: TXFIFO Empty

This bit is set by hardware when TXFIFO is Empty. When the TXFIFO contains at least one data, this flag is cleared. The TXFE flag can also be set by writing 1 to the bit TXFRQ (bit 4) in the LPUART_RQR register.

An interrupt is generated if the TXFEIE bit = 1 (bit 30) in the LPUART_CR1 register.

0: TXFIFO is not empty.

1: TXFIFO is empty.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the LPUART.

It can be used to verify that the LPUART is ready for reception before entering low-power mode.

Note: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the LPUART.

It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the LPUART_CR1 register, in order to respect the TE = 0 minimum period.

Bit 20 **WUF**: Wake-up from low-power mode flag

This bit is set by hardware, when a wake-up event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the LPUART_ICR register.
An interrupt is generated if WUFIE = 1 in the LPUART_CR3 register.

Note: When UESM is cleared, WUF flag is also cleared.

If the LPUART *does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 66.3: LPUART implementation.*

Bit 19 **RWU**: Receiver wake-up from mute mode

This bit indicates if the LPUART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the LPUART_CR1 register.
When wake-up on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the LPUART_RQR register.
0: Receiver in active mode
1: Receiver in mute mode

Note: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the LPUART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.
0: No break character transmitted
1: Break character transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the LPUART_ICR register.
An interrupt is generated if CMIE = 1in the LPUART_CR1 register.
0: No Character match detected
1: Character match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).
0: LPUART is idle (no reception)
1: reception ongoing

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.
0: CTS line set
1: CTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the LPUART_ICR register.

An interrupt is generated if CTSIE = 1 in the LPUART_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **TXFNF:** TXFIFO not full

TXFNF is set by hardware when TXFIFO is not full, and so data can be written in the LPUART_TDR. Every write in the LPUART_TDR places the data in the TXFIFO. This flag remains set until the TXFIFO is full. When the TXFIFO is full, this flag is cleared indicating that data can not be written into the LPUART_TDR.

The TXFNF is kept reset during the flush request until TXFIFO is empty. After sending the flush request (by setting TXFRQ bit), the flag TXFNF must be checked prior to writing in TXFIFO (TXFNF and TXFE are set at the same time).

An interrupt is generated if the TXFNFIE bit = 1 in the LPUART_CR1 register.

0: Data register is full/Transmit FIFO is full.

1: Data register/Transmit FIFO is not full.

Note: This bit is used during single buffer transmission.

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the LPUART_TDR has been transmitted out of the shift register.

It is set by hardware when the transmission of a frame containing data has completed and the TXFE bit is set.

An interrupt is generated if TCIE = 1 in the LPUART_CR1 register.

The TC bit is cleared by software, by writing 1 to the TCCF of the LPUART_ICR register, or by a write to the LPUART_TDR register.

0: Transmission has not completed

1: Transmission has completed

Note: If the TE bit is reset and no transmission is ongoing, the TC bit is immediately set.

Bit 5 **RXFNE**: RXFIFO not empty

RXFNE bit is set by hardware when the RXFIFO is not empty, and so data can be read from the LPUART_RDR register. Every read of the LPUART_RDR frees a location in the RXFIFO. It is cleared when the RXFIFO is empty.

The RXFNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register.

An interrupt is generated if RXFNEIE = 1 in the LPUART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle line is detected. An interrupt is generated if IDLEIE = 1 in the LPUART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the LPUART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit is not set again until the RXFNE bit has been set (that is, a new idle line occurs).

If mute mode is enabled (MME = 1), IDLE is set if the LPUART is not mute (RWU = 0), whatever the mute mode selected by the WAKE bit. If RWU = 1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the LPUART_RDR register while RXFF = 1. It is cleared by a software, writing 1 to the ORECF, in the LPUART_ICR register.

An interrupt is generated if RXFNEIE = 1 in the LPUART_CR1 register, or EIE = 1 in the LPUART_CR3 register.

1: Overrun error is detected

Note: When this bit is set, the LPUART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the LPUART_CR3 register.

Bit 2 **NE:** Start bit noise detection flag

This bit is set by hardware when noise is detected on the start bit of a received frame. It is cleared by software, writing 1 to the NFCF bit in the LPUART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXFNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

This error is associated with the character in the LPUART_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the LPUART_ICR register.

When transmitting data in smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).

An interrupt is generated if EIE = 1 in the LPUART_CR3 register.

0: No Framing error is detected

1: Framing error or break character is detected

Note: This error is associated with the character in the LPUART_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in reception mode. It is cleared by software, writing 1 to the PECF in the LPUART_ICR register.

An interrupt is generated if PEIE = 1 in the LPUART_CR1 register.

0: No parity error

1: Parity error

Note: This error is associated with the character in the LPUART_RDR.

66.7.9 LPUART interrupt and status register [alternate] (LPUART_ISR)

Address offset: 0x1C

Reset value: 0x0000 00C0

The same register can be used in FIFO mode enabled (previous section) and FIFO mode disabled (this section).

FIFO mode disabled, FIFOEN = 0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY
									r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
					r	r		r	r	r	r	r	r	r	r

Bits 31:23 Reserved, must be kept at reset value.

Bit 22 **REACK**: Receive enable acknowledge flag

This bit is set/reset by hardware, when the Receive Enable value is taken into account by the LPUART.

It can be used to verify that the LPUART is ready for reception before entering low-power mode.

Note: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.

Bit 21 **TEACK**: Transmit enable acknowledge flag

This bit is set/reset by hardware, when the Transmit Enable value is taken into account by the LPUART.

It can be used when an idle frame request is generated by writing TE = 0, followed by TE = 1 in the LPUART_CR1 register, in order to respect the TE = 0 minimum period.

Bit 20 **WUF**: Wake-up from low-power mode flag

This bit is set by hardware, when a wake-up event is detected. The event is defined by the WUS bitfield. It is cleared by software, writing a 1 to the WUCF in the LPUART_ICR register.

An interrupt is generated if WUFIE = 1 in the LPUART_CR3 register.

Note: When UESM is cleared, WUF flag is also cleared.

If the LPUART *does not support the wake-up from Stop feature, this bit is reserved and kept at reset value. Refer to Section 66.3: LPUART implementation.*

Bit 19 **RWU**: Receiver wake-up from mute mode

This bit indicates if the LPUART is in mute mode. It is cleared/set by hardware when a wake-up/mute sequence is recognized. The mute mode control sequence (address or IDLE) is selected by the WAKE bit in the LPUART_CR1 register.

When wake-up on IDLE mode is selected, this bit can only be set by software, writing 1 to the MMRQ bit in the LPUART_RQR register.

0: Receiver in active mode

1: Receiver in mute mode

Note: If the LPUART does not support the wake-up from Stop feature, this bit is reserved and kept at reset value.

Bit 18 **SBKF**: Send break flag

This bit indicates that a send break character was requested. It is set by software, by writing 1 to the SBKRQ bit in the LPUART_CR3 register. It is automatically reset by hardware during the stop bit of break transmission.

0: No break character transmitted

1: Break character transmitted

Bit 17 **CMF**: Character match flag

This bit is set by hardware, when a the character defined by ADD[7:0] is received. It is cleared by software, writing 1 to the CMCF in the LPUART_ICR register.

An interrupt is generated if CMIE = 1in the LPUART_CR1 register.

0: No Character match detected

1: Character match detected

Bit 16 **BUSY**: Busy flag

This bit is set and reset by hardware. It is active when a communication is ongoing on the RX line (successful start bit detected). It is reset at the end of the reception (successful or not).

0: LPUART is idle (no reception)

1: Reception ongoing

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **CTS**: CTS flag

This bit is set/reset by hardware. It is an inverted copy of the status of the CTS input pin.

0: CTS line set

1: CTS line reset

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 9 **CTSIF**: CTS interrupt flag

This bit is set by hardware when the CTS input toggles, if the CTSE bit is set. It is cleared by software, by writing 1 to the CTSCF bit in the LPUART_ICR register.

An interrupt is generated if CTSIE = 1 in the LPUART_CR3 register.

0: No change occurred on the CTS status line

1: A change occurred on the CTS status line

Note: If the hardware flow control feature is not supported, this bit is reserved and kept at reset value.

Bit 8 Reserved, must be kept at reset value.

Bit 7 **TXE:** Transmit data register empty

TXE is set by hardware when the content of the LPUART_TDR register has been transferred into the shift register. It is cleared by a write to the LPUART_TDR register.

An interrupt is generated if the TXEIE bit = 1 in the LPUART_CR1 register.

0: Data register full

1: Data register empty

Note: This bit is used during single buffer transmission.

Bit 6 **TC**: Transmission complete

This bit indicates that the last data written in the LPUART_TDR has been transmitted out of the shift register. The TC flag is set when the transmission of a frame containing data has completed and when TXE is set.

An interrupt is generated if TCIE = 1 in the LPUART_CR1 register.

TC bit is cleared by software by writing 1 to the TCCF in the LPUART_ICR register or by writing to the LPUART_TDR register.

Bit 5 **RXNE**: Read data register not empty

RXNE bit is set by hardware when the content of the LPUART_RDR shift register has been transferred to the LPUART_RDR register. It is cleared by a read to the LPUART_RDR register. The

RXNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register.

The RXFNE flag can also be cleared by writing 1 to the RXFRQ in the LPUART_RQR register.

An interrupt is generated if RXNEIE = 1 in the LPUART_CR1 register.

0: Data is not received

1: Received data is ready to be read.

Bit 4 **IDLE**: Idle line detected

This bit is set by hardware when an Idle Line is detected. An interrupt is generated if IDLEIE = 1 in the LPUART_CR1 register. It is cleared by software, writing 1 to the IDLECF in the LPUART_ICR register.

0: No Idle line is detected

1: Idle line is detected

Note: The IDLE bit is not set again until the RXNE bit has been set (that is, a new idle line occurs).

If mute mode is enabled (MME = 1), IDLE is set if the LPUART is not mute (RWU = 0), whatever the mute mode selected by the WAKE bit. If RWU = 1, IDLE is not set.

Bit 3 **ORE**: Overrun error

This bit is set by hardware when the data currently being received in the shift register is ready to be transferred into the LPUART_RDR register while RXNE = 1 (RXFF = 1 in case FIFO mode is enabled). It is cleared by a software, writing 1 to the ORECF, in the LPUART_ICR register.

An interrupt is generated if RXNEIE = 1 in the LPUART_CR1 register, or EIE = 1 in the LPUART_CR3 register.

1: Overrun error is detected

Note: When this bit is set, the LPUART_RDR register content is not lost but the shift register is overwritten. An interrupt is generated if the ORE flag is set during multi buffer communication if the EIE bit is set.

This bit is permanently forced to 0 (no overrun detection) when the bit OVRDIS is set in the LPUART_CR3 register.

Bit 2 **NE:** Start bit noise detection flag

This bit is set by hardware when noise is detected on the start bit of a received frame. It is cleared by software, writing 1 to the NFCF bit in the LPUART_ICR register.

0: No noise is detected

1: Noise is detected

Note: This bit does not generate an interrupt as it appears at the same time as the RXNE/RXFNE bit which itself generates an interrupt. An interrupt is generated when the NE flag is set during multi buffer communication if the EIE bit is set.

In FIFO mode, this error is associated with the character in the LPUART_RDR.

Bit 1 **FE**: Framing error

This bit is set by hardware when a de-synchronization, excessive noise or a break character is detected. It is cleared by software, writing 1 to the FECF bit in the LPUART_ICR register.
When transmitting data in smartcard mode, this bit is set when the maximum number of transmit attempts is reached without success (the card NACKs the data frame).
An interrupt is generated if EIE = 1 in the LPUART_CR3 register.
0: No Framing error is detected
1: Framing error or break character is detected

Note: In FIFO mode, this error is associated with the character in the LPUART_RDR.

Bit 0 **PE**: Parity error

This bit is set by hardware when a parity error occurs in reception mode. It is cleared by software, writing 1 to the PECF in the LPUART_ICR register.
An interrupt is generated if PEIE = 1 in the LPUART_CR1 register.
0: No parity error
1: Parity error

Note: In FIFO mode, this error is associated with the character in the LPUART_RDR.

66.7.10 LPUART interrupt flag clear register (LPUART_ICR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.
											w			w	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CTSCF	Res.	Res.	TCCF	Res.	IDLECF	ORECF	NECF	FECF	PECF
						w			w		w	w	w	w	w

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **WUCF**: Wake-up from low-power mode clear flag

Writing 1 to this bit clears the WUF flag in the LPUART_ISR register.

Note: If the LPUART *does not support the wake-up from Stop feature, this bit is reserved and must be kept at reset value.* Refer to *Section 66.3: LPUART implementation.*

Bits 19:18 Reserved, must be kept at reset value.

Bit 17 **CMCF**: Character match clear flag

Writing 1 to this bit clears the CMF flag in the LPUART_ISR register.

Bits 16:10 Reserved, must be kept at reset value.

Bit 9 **CTSCF**: CTS clear flag

Writing 1 to this bit clears the CTSIF flag in the LPUART_ISR register.

Bit 8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TCCF**: Transmission complete clear flag

Writing 1 to this bit clears the TC flag in the LPUART_ISR register.

Bit 5 Reserved, must be kept at reset value.

Bit 4 **IDLECF**: Idle line detected clear flag
Writing 1 to this bit clears the IDLE flag in the LPUART_ISR register.

Bit 3 **ORECF**: Overrun error clear flag
Writing 1 to this bit clears the ORE flag in the LPUART_ISR register.

Bit 2 **NECF**: Noise detected clear flag
Writing 1 to this bit clears the NE flag in the LPUART_ISR register.

Bit 1 **FECF**: Framing error clear flag
Writing 1 to this bit clears the FE flag in the LPUART_ISR register.

Bit 0 **PECF**: Parity error clear flag
Writing 1 to this bit clears the PE flag in the LPUART_ISR register.

66.7.11 LPUART receive data register (LPUART_RDR)

Address offset: 0x24

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDR[8:0]								
							r	r	r	r	r	r	r	r	r

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **RDR[8:0]**: Receive data value
Contains the received data character.
The RDR register provides the parallel interface between the input shift register and the internal bus (see *Section 66.4.1: LPUART block diagram*).
When receiving with the parity enabled, the value read in the MSB bit is the received parity bit.

66.7.12 LPUART transmit data register (LPUART_TDR)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:0 **TDR[8:0]**: Transmit data value

Contains the data character to be transmitted.

The TDR register provides the parallel interface between the internal bus and the output shift register (see *Section 66.4.1: LPUART block diagram*).

When transmitting with the parity enabled (PCE bit set to 1 in the LPUART_CR1 register), the value written in the MSB (bit 7 or bit 8 depending on the data length) has no effect because it is replaced by the parity.

Note: This register must be written only when TXE/TXFNF = 1.

66.7.13 LPUART prescaler register (LPUART_PRESC)

This register can only be written when the LPUART is disabled (UE = 0).

Address offset: 0x2C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PRESCALER[3:0]**: Clock prescaler

The LPUART input clock can be divided by a prescaler:

0000: input clock not divided
0001: input clock divided by 2
0010: input clock divided by 4
0011: input clock divided by 6
0100: input clock divided by 8
0101: input clock divided by 10
0110: input clock divided by 12
0111: input clock divided by 16
1000: input clock divided by 32
1001: input clock divided by 64
1010: input clock divided by 128
1011: input clock divided by 256
Others: Reserved, must not be used.

Note: When PRESCALER is programmed with a value different of the allowed ones, programmed prescaler value is equal to 1011, that is, input clock divided by 256.

If the prescaler is not supported, this bitfield is reserved and must be kept at reset value. Refer to Section 66.3: LPUART implementation.

66.7.14 LPUART register map

Table 656. LPUART register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	LPUART_CR1FIFO mode enabled	RXFFIE	TXFEIE	FIFOEN	M1	Res.	Res.	DEAT[4:0]						DEDT[4:0]				Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXFNFIE	TCIE	RXFNEIE	IDLEIE	TE	RE	UESM	UE
	Reset value	0	0	0	0			0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00	LPUART_CR1FIFO mode disabled	Res.	Res.	FIFOEN	M1	Res.	Res.	DEAT[4:0]						DEDT[4:0]				Res.	CMIE	MME	M0	WAKE	PCE	PS	PEIE	TXEIE	TCIE	RXNEIE	IDLEIE	TE	RE	UESM	UE
	Reset value			0	0			0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	LPUART_CR2	ADD[7:0]								Res.	Res.	Res.	Res.	MSBFIRST	DATAINV	TXINV	RXINV	SWAP	Res.	STOP [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDM7	Res.	Res.	Res.	Res.
	Reset value	0	0	0	0	0	0	0	0					0	0	0	0	0		0	0								0				
0x08	LPUART_CR3 FIFO mode enabled	TXFTCFG[2:0]			RXFTIE	RXFTCFG[2:0]			Res.	TXFTIE	WUFIE	WUS 1	WUS 0	Res.	Res.	Res.	Res.	DEP	DEM	DDRE	OVRDIS	Res.	CTSIE	CTSE	RTSE	DMAT	DMAR	Res.	Res.	HDSEL	Res.	Res.	Res.	EIE
	Reset value	0	0	0	0	0	0	0		0	0	0	0					0	0	0	0		0	0	0	0	0			0			0
0x08	LPUART_CR3 FIFO mode disabled	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUFIE	WUS 1	WUS 0	Res.	Res.	Res.	Res.	DEP	DEM	DDRE	OVRDIS	Res.	CTSIE	CTSE	RTSE	DMAT	DMAR	Res.	Res.	HDSEL	Res.	Res.	Res.	EIE
	Reset value										0	0	0					0	0	0	0		0	0	0	0	0			0				0
0x0C	LPUART_BRR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BRR[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10-0x14	Reserved																																
0x18	LPUART_RQR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFRQ	RXFRQ	MMRQ	SBKRQ	Res.
	Reset value																												0	0	0	0	
0x1C	LPUART_ISRFIFO mode enabled	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXFF	TXFF	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY	Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXFNF	TC	RXFNE	IDLE	ORE	NE	FE	PE
	Reset value								0	1	0	0	0	0	0	0	0						0	0		1	0	0	0	0	0	0	0
0x1C	LPUART_ISRFIFO mode disabled	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REACK	TEACK	WUF	RWU	SBKF	CMF	BUSY	Res.	Res.	Res.	Res.	Res.	CTS	CTSIF	Res.	TXE	TC	RXNE	IDLE	ORE	NE	FE	PE
	Reset value										0	0	0	0	0	0	0						0	0		1	0	0	0	0	0	0	0
0x20	LPUART_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUCF	Res.	Res.	CMCF	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTSCF	Res.	Res.	TCCF	Res.	IDLECF	ORECF	NECF	FECF	PECF
	Reset value												0			0								0			0		0	0	0	0	0
0x24	LPUART_RDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res	RDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0

Table 656. LPUART register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x28	LPUART_TDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDR[8:0]								
	Reset value																								0	0	0	0	0	0	0	0	0
0x2C	LPUART_ PRESC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[3:0]			
	Reset value																													0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

67 Serial peripheral interface (SPI)

67.1 SPI introduction

The serial peripheral interface (SPI) can be used to communicate with external devices while using the specific synchronous protocol. The SPI protocol supports half-duplex, full-duplex, and simplex synchronous, serial communication with external devices. The interface can be configured as master or slave and can operate in multislave or multimaster configurations. The device configured as master provides a communication clock (SCK) to the slave device. The slave select (NSS) and ready (RDY) signals can be applied optionally just to set up communication with a concrete slave and to ensure it handles the data flow properly. The Motorola data format is used by default, but some other specific modes are supported as well.

67.2 SPI main features

- Full-duplex synchronous transfers on three lines
- Half-duplex synchronous transfer on two lines (with bidirectional data line)
- Simplex synchronous transfers on two lines (with unidirectional data line)
- From 4-bit up to 32-bit data size selection
- Multimaster or multislave mode capability
- Dual clock domain, the peripheral kernel clock is independent from the APB bus clock
- Baud rate prescaler up to kernel frequency/2 or bypass from RCC in master mode
- Protection of configuration and setting
- Slave select hardware or software management for both master and slave
- Adjustable minimum delays between data and between NSS and data flow
- Configurable NSS signal polarity and timing, MISO x MOSI swap capability
- Programmable clock polarity and phase
- Programmable data order with MSB-first or LSB-first shifting
- Programmable number of data within a transfer to control NSS and CRC
- Dedicated transmission and reception flags with interrupt capability
- SPI Motorola and TI format support
- Hardware CRC can verify the integrity of the communication at the end of a transfer by:
  - Adding CRC value in Tx mode
  - Automatic CRC error checking for Rx mode
- Error detection with interrupt capability in case of data overrun, CRC error, data underrun, the mode fault, and frame error, depending on the operating mode
- Two 8-bit width embedded Rx and Tx FIFOs (FIFO size depends on instances)
- Configurable FIFO thresholds (to handle the data packets)
- Capability to handle data streams by system DMA controller
- Configurable behavior at slave underrun condition (support of cascaded circular buffers)
- Optional status pin RDY signalizing that the slave device is ready to handle the data flow

67.3 SPI implementation

The table below describes the SPI implementation.

Table 657. SPI features

SPI feature	SPI1, SPI2, SPI3, SPI6 (full-featured instances)	SPI4, SPI5 (full-featured instances)
Data and CRC size	Configurable from 4 to 32 bits	Configurable from 4 to 16 bits
CRC computation	CRC polynomial length configurable from 5 to 33 bits	CRC polynomial length configurable from 5 to 17 bits
FIFO size	16 x 8 bits	8 x 8 bits
FIFO threshold	1 - 16 data	1 - 16 data
Number of data control (TSIZE)	Up to 65535	Up to 65535
I2S feature	Yes	No
Autonomous in Stop modes with wake-up capability	No	No
Autonomous in LP-Stop and Standby modes with wake-up capability	No	No

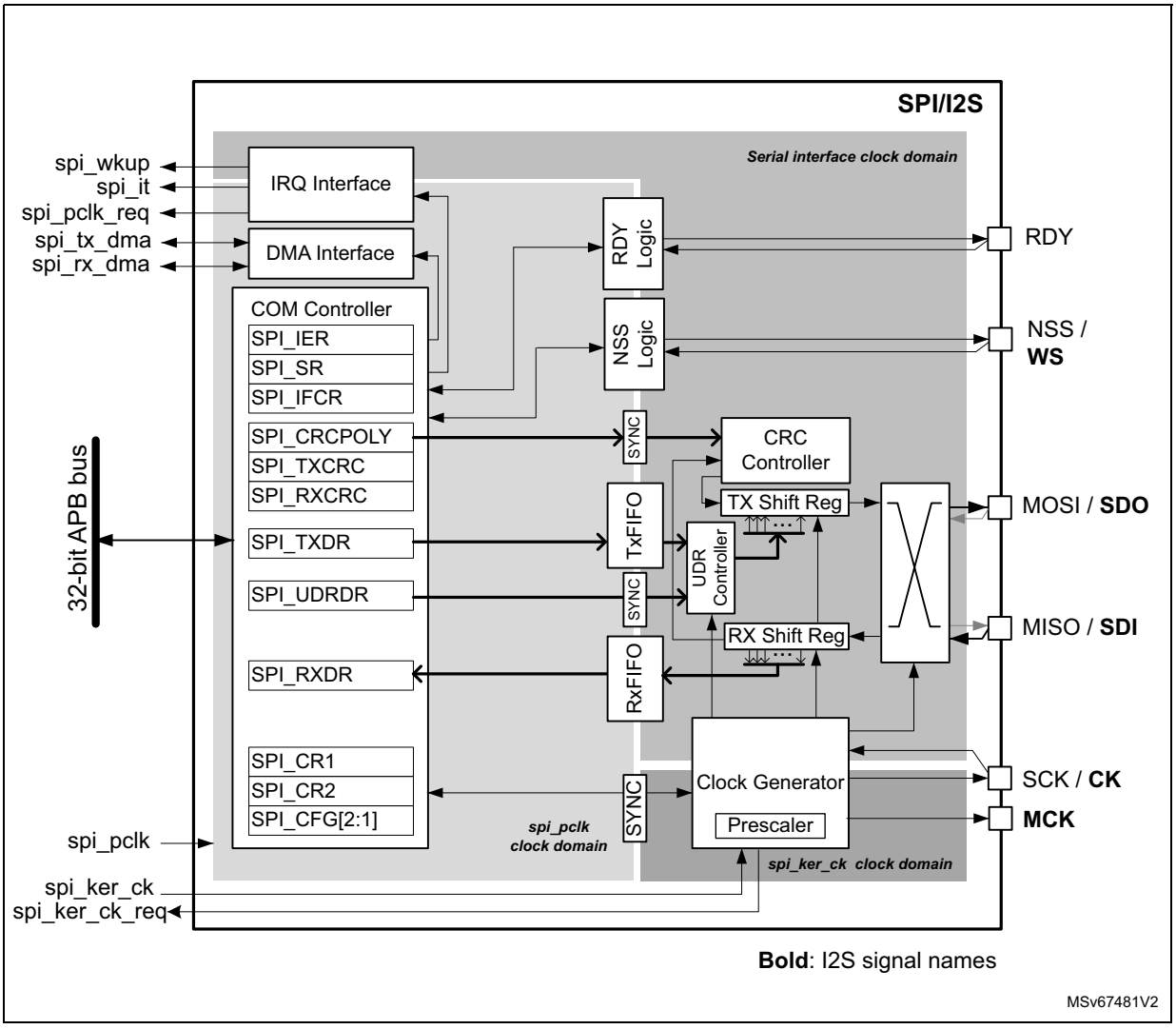
Note: For detailed information about instance capabilities to exit from Stop and Standby modes, refer to Table 661: SPI wake-up and interrupt requests.

67.4 SPI functional description

67.4.1 SPI block diagram

The SPI enables synchronous, serial communications between the MCU and external devices. The application software can manage the communication by polling the status flag or using a dedicated SPI interrupt. The main SPI elements and their interactions are shown in *Figure 880*.

Figure 880. SPI/I2S block diagram



The simplified scheme of *Figure 880* shows three fully independent clock domains:

- The **spi_pclk** clock domain
- The **spi_ker_ck** kernel clock domain
- The serial interface clock domain

All the control and status signals between these domains are strictly synchronized. There is no specific constraint concerning the frequency ratio between these clock signals. The user

has to consider a ratio compatible with the data flow speed to avoid data underrun or overrun events.

The **spi_pclk** clock signal feeds the peripheral bus interface. It must be active when accesses to the SPI registers are required.

The SPI working in slave mode handles a data flow using the serial interface clock derived from the external SCK signal provided by the external master SPI device. That is why the SPI slave is able to receive and send data even when the **spi_pclk** and **spi_ker_ck** clock signals are inactive. As a consequence, a specific slave logic working within the serial interface clock domain needs some additional traffic to be set up correctly (for example when underrun or overrun is evaluated, see *Section 67.5.2* for details). This cannot be done when the bus becomes idle. In some specific cases, the slave even requires the clock generator working (see *Section 67.5.1*).

When the SPI works as a master, the RCC must provide the **spi_ker_ck** kernel clock to the peripheral during communication to feed the serial interface clock via the clock generator where it can be divided by prescaler or bypassed. The signal is then provided to the slaves via the SCK pin and internally to the serial interface domain of the master.

67.4.2 SPI pins and internal signals

Up to five I/O pins are dedicated to SPI communication with external devices.

- **MISO:** master in / slave out data. In the general case, this pin is used to transmit data in slave mode and receive data in master mode.
- **MOSI:** master out / slave in data. In the general case, this pin is used to transmit data in master mode and receive data in slave mode.
- **SCK:** serial clock output pin for SPI masters and input pin for SPI slaves.
- **NSS:** slave select pin. Depending on the SPI and NSS settings, this pin can be used to either:
  - Select an individual slave device for communication
  - Synchronize the data frame, or
  - Detect a conflict between multiple masters

  See *Section 67.4.7* for details.
- **RDY:** optional status pin signaling slave FIFO occupancies and so the slave availability to continue the transfer without any risk of data flow corruption. It can be checked by the master to control the temporal suspension of the ongoing communication.

All these pins (except RDY) are shared in the I2S mode. This mode features an additional I2S specific MCK signal. For more details about I2S signals, see *Section 67.9.2*.

The SPI bus enables the communication between one master device and one or more slave devices. The bus consists of at least two wires: one for the clock signal and the other for synchronous data transfer. Other signals are optional and can be added depending on the data exchange between SPI nodes and their communication control management.

Refer to *Table 658* and *Table 659* for the list of SPI input / output pins and internal signals.

Table 658. SPI/I2S input/output pins(1)

Pin name	I/O type	Description
MISO/SDI(2)	Input/output	Master data input / slave data output
MOSI/SDO(2)	Input/output	Master data output / slave data input
SCK/CK	Input/output	Master clock output / slave clock input
NSS/WS	Input/output	Master output / slave selection input
RDY	Input/output	SPI master input / slave FIFOs status occupancy output
MCK	Output	I2S master frequency output

1. Refer to the section *Section 67.9.2: Pin sharing with SPI function* for details.
2. Functionality of MOSI/SDO and MISO/SDI pins can be swapped. Their directions may vary in SPI bidirectional half-duplex mode.

Description of SPI input/output signals

Table 659. SPI internal input/output signals

Signal name	Signal type	Description
spi_pclk	Input	SPI clock signal feeds the peripheral bus interface
spi_ker_ck	Input	SPI kernel clock
spi_ker_ck_req	Output	SPI kernel clock request
spi_pclk_req	Output	SPI clock request
spi_wkup	Output	SPI provides a wake-up interrupt
spi_it	Output	SPI global interrupt
spi_tx_dma	Input/output	SPI transmit DMA request
spi_rx_dma	Input/output	SPI receive DMA request

67.4.3 SPI communication general aspects

The SPI allows the MCU to communicate using different configurations, depending on the device targeted and the application requirements. These configurations use two or three wires (with software slave select management) or three or four wires (with hardware slave select management). The communication is always initiated and controlled by the master. The master provides a clock signal on the SCK line and selects or synchronizes slaves for communication by NSS line when it is managed by hardware.

The data between the master and the slave flow synchronously on the MOSI and/or MISO lines.

67.4.4 Communications between one master and one slave

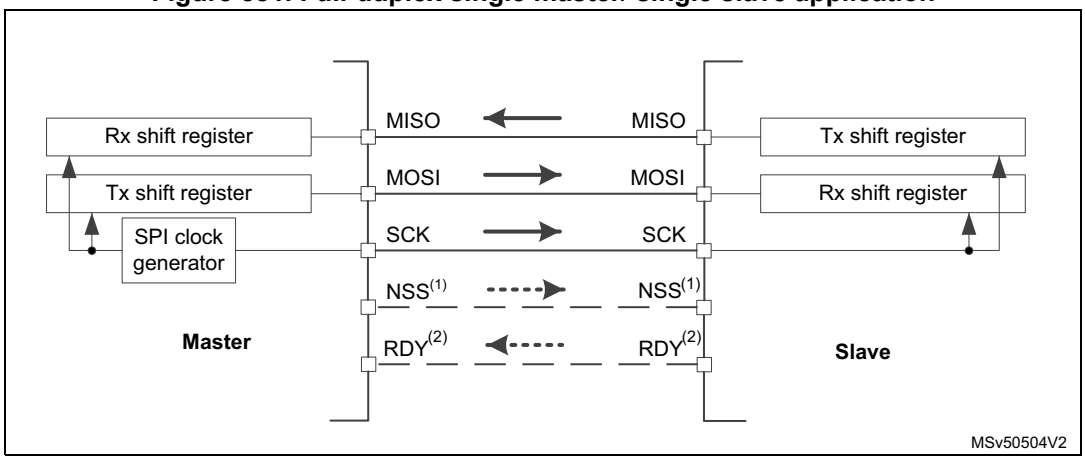
The communication flow can use one of three possible modes: the full-duplex (three wires) mode, half-duplex (two wires) mode, or the simplex (two wires) mode. The NSS signal is optional in single master-slave configuration and is often not connected between the two communication nodes. Nevertheless, the NSS signal can be helpful in this configuration to synchronize the data flow and it is used by default for some specific SPI modes (for example the TI mode).

The next optional RDY signal can help to ensure the correct management of all the transferred data at slave side.

Full-duplex communication

By default, the SPI is configured for full-duplex communication (bits COMM[1:0] = 00 in the SPI_CFG2 register). In this configuration, the shift registers of the master and slave are linked using two unidirectional lines between the MOSI and the MISO pins. During the SPI communication, the data are shifted synchronously on the SCK clock edges provided by the master. The master transmits the data to be sent to the slave via the MOSI line and receives data from the slave via the MISO line simultaneously. When the data frame transfer is complete (all the bits are shifted) the information between the master and slave is exchanged.

Figure 881. Full-duplex single master/ single slave application

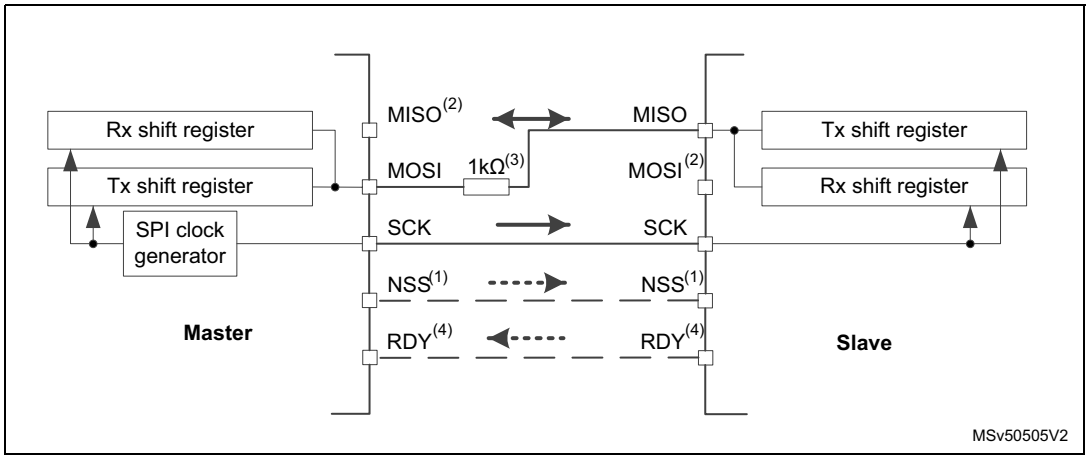


1. The NSS pin interconnection is optional. The slave can be configured to be permanently selected to operate in a single master-slave pair (see *Section 67.4.7*).
2. The RDY signal provided by the slave can be read by the master optionally.

Half-duplex communication

The SPI can communicate in half-duplex mode by setting COMM[1:0] = 11 in the SPI_CFG2 register. In this configuration, one single cross-connection line is used to link the shift registers of the master and slave together. During this communication, the data are synchronously shifted between the shift registers on the SCK clock edge in the transfer direction selected reciprocally by both master and slave with the HDDIR bit in their SPI_CR1 registers. Note that the SPI must be disabled when changing the direction of the communication. In this configuration, the MISO pin at master and the MOSI pin at slave are free for other application uses and act as GPIOs.

Figure 882. Half-duplex single master/ single slave application



1. The NSS pin interconnection is optional. The slave can be configured to be permanently selected to operate in a single master-slave pair (see *Section 67.4.7*).
2. In this configuration, the MISO pin at master and MOSI pin at slave can be used as GPIOs
3. A critical situation can happen when the communication direction is not changed synchronously between two nodes working in bidirectional mode. The new transmitter accesses the common data line while the former transmitter still keeps an opposite value on the line (the value depends on the SPI configuration and communicated data). The nodes can conflict temporarily with opposite output levels on the line until the former transmitter changes its data direction setting. It is suggested to insert a serial resistance between MISO and MOSI pins in this mode to protect the conflicting outputs and limit the current flow between them.
4. The RDY signal provided by the slave can be read by the master optionally.

Simplex communications

The SPI can communicate in simplex mode by setting the SPI in transmit-only or in receive-only using the COMM[1:0] field in the SPI_CFG2 register. In this configuration, only one line is used for the transfer between the shift registers of the master and slave. The remaining MISO or MOSI pin pair is not used for communication and can be used as standard GPIOs.

- **Transmit-only mode: COMM[1:0] = 01**

  The master in transmit-only mode generates the clock as long as there are data available in the TxFIFO and the master transfer is ongoing.

  The slave in transmit-only mode sends data as long as it receives a clock on the SCK pin and the NSS pin (or software managed internal signal) is active (see *Section 67.4.7*).

- **Receive-only mode: COMM[1:0] = 10**

  In master mode, the MOSI output is disabled and can be used as a GPIO. The clock signal is generated continuously as long as the SPI is enabled and the CSTART bit in the SPI_CR1 register is set. The clock is stopped either by software explicitly requesting this by setting the CSUSP bit in the SPI_CR1 register or automatically when the RxFIFO is full, when the MASRX bit in the SPI_CR1 is set.

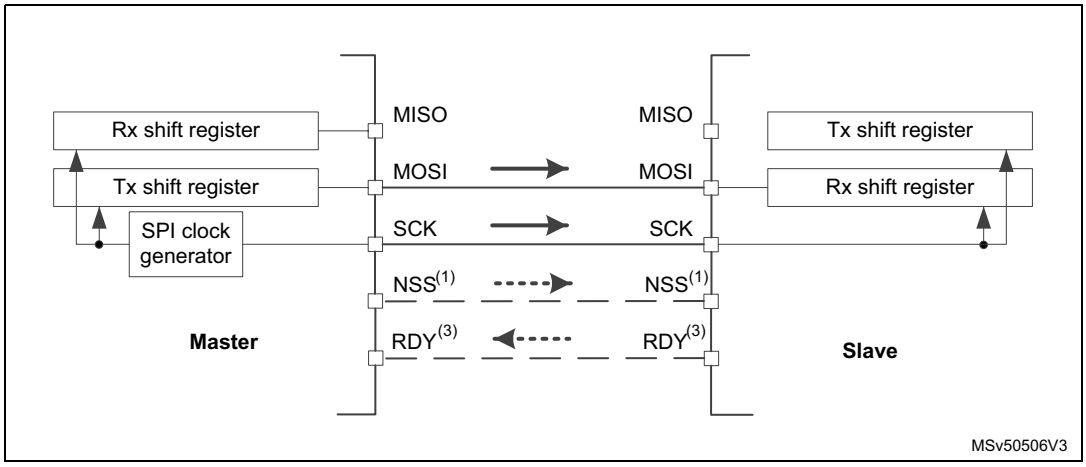
  In slave configuration, the MISO output is disabled and the pin can be used as a GPIO. The slave continues to receive data from the MOSI pin while its slave select signal is active (see *Section 67.4.7*).

Note: In whatever master and slave modes, the data pin dedicated for transmission can be replaced by the data pin dedicated for reception and vice versa by changing the IOSWP bit value in the SPI_CFG2 register (this bit can only be modified when the SPI is disabled).

Any simplex communication can be replaced by a variant of the half-duplex communication with a constant setting of the transfer direction (bidirectional mode is enabled, while the

HDDIR bit is never changed) or by full-duplex control when unused data line and corresponding data flow is ignored.

Figure 883. Simplex single master / single slave application (master in transmit-only / slave in receive-only mode)



1. The NSS pin interconnection is optional. The slave can be configured to be permanently selected to operate in a single master-slave pair (see *Section 67.4.7*).
2. In this configuration, both the MISO pins can be used as GPIOs.
3. The RDY signal provided by the slave can be read by the master optionally.

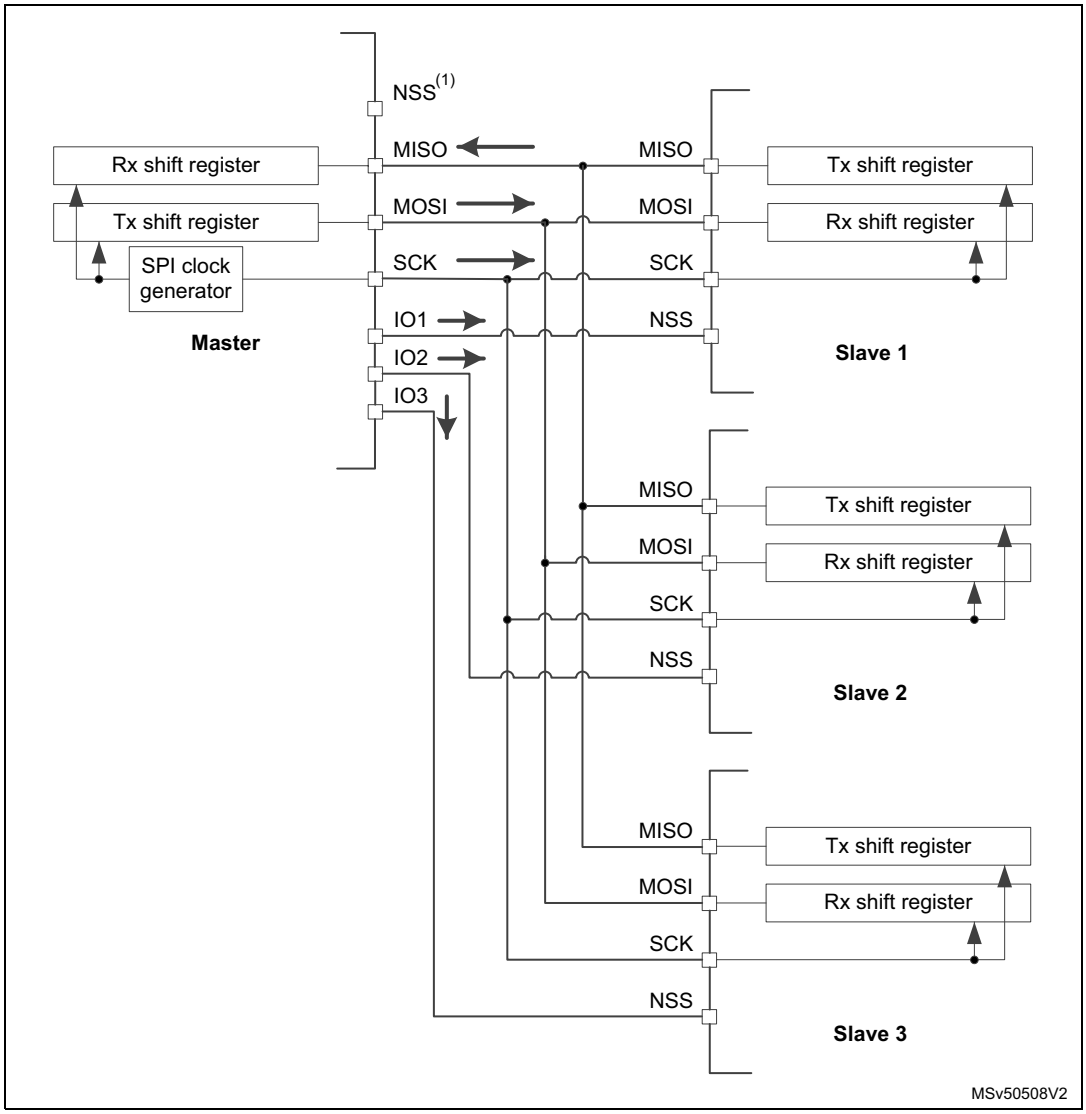
67.4.5 Standard multislave communication

In a configuration with two or more independent slaves, the master uses a star topology with dedicated GPIO pins to manage the chip select lines for each slave separately (see *Figure 884.*).

The master must select one of the slaves individually by pulling low the GPIO connected to the slave NSS input (only one slave can control data on a common MISO line at a given time).

When this is done, a communication between the master and the selected slave is established. In addition to the simplicity, the advantage of this topology is that a specific SPI configuration can be applied for each slave as all the communication sessions are performed separately just within a single master-slave pair. Optionally, when there is no need to read any information from slaves, the master can transmit the same information to the multiple slaves.

Figure 884. Master and three independent slaves connected in star topology



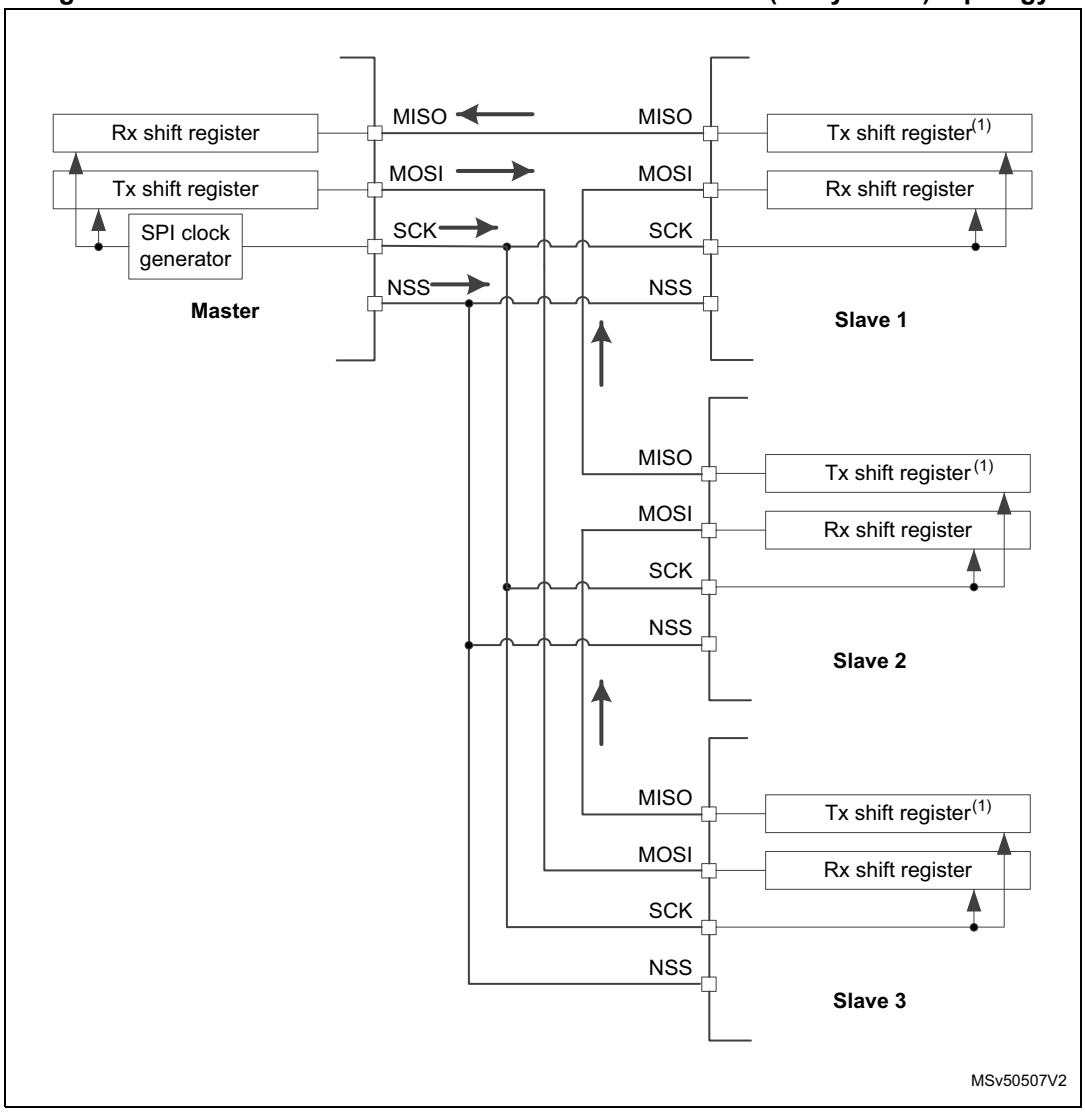
1. The master single NSS pin hardware output functionality cannot support this topology (to be replaced by a set of GPIOs under software control). The NSS pin is free for other application uses (such as GPIO or other alternate functions). Refer to *Section 67.4.7* for details.
2. If the application cannot ensure that no more than a single NSS active signal is provided by the master at a given time, it is better to configure the MISO pins in an open-drain configuration with an external pull-up on the MISO line to prevent conflicts between the interconnected outputs of the slaves. Else, a push-pull configuration can be applied without an extra resistor (see I/O alternate function input/output (GPIO) section).
3. The RDY signals can be read by the master from the slaves optionally.

The master can handle the SPI communication with all the slaves at the same time when a circular topology is applied (see *Figure 885*). All the slaves behave like simple shift registers applied in serial chain under control of common slave select (NSS) and clock (SCK) signals. All the information is shifted simultaneously around the circle while returning back to the master. Sessions have fixed the length where the number of data frames transferred by the master is equal to the number of slaves.

Then when a first data frame is transferred in the chain, the master just sends the information dedicated for the last slave node in the chain, via the first slave node input, while the first information received by the master comes from the last node output at this time.

Correspondingly, the last transferred data finishing the session is dedicated for the first slave node while its first outgoing data just reaches the master input, after its circling around the chain passing through all the other slaves during the session.
The data format configuration and clock setting must be the same for all the nodes in the chain in this topology. As the receive and transmit shift registers are separated internally, a trick with intentional underrun must be applied to the TxFIFO slaves when the information is transferred between the receiver and the transmitter by hardware.
In this case, the transmission underrun feature is configured in a mode repeating the last received data frame (UDRCFG=1). A session can start optionally with a single data pattern written into the TxFIFO by each slave (usually slave status information is applied) before the session starts. In this case, the underrun happens in fact after this first data frame is transferred. To be able to clear the internal underrun condition immediately and restart the session by the TxFIFO content again, the user must disable and enable the SPI between the sessions and must fill the TxFIFO by a new single data pattern (to overcome the propagating delay of the clearing raised in case the underrun is cleared in a standard way by the UDRC bit).

Figure 885. Master and three slaves connected in circular (daisy chain) topology



1. The underrun feature is used by the slaves in this configuration when the slaves are able to transmit data received previously into the Rx shift register once their TxFIFOs become empty.
2. The RDY signals can be read by the master optionally, either separately or configured as open drain outputs (while RDIOP = 0) and connected with a pull-up resistor as a common chain ready status overdriven by the slowest device.

67.4.6 Multimaster communication

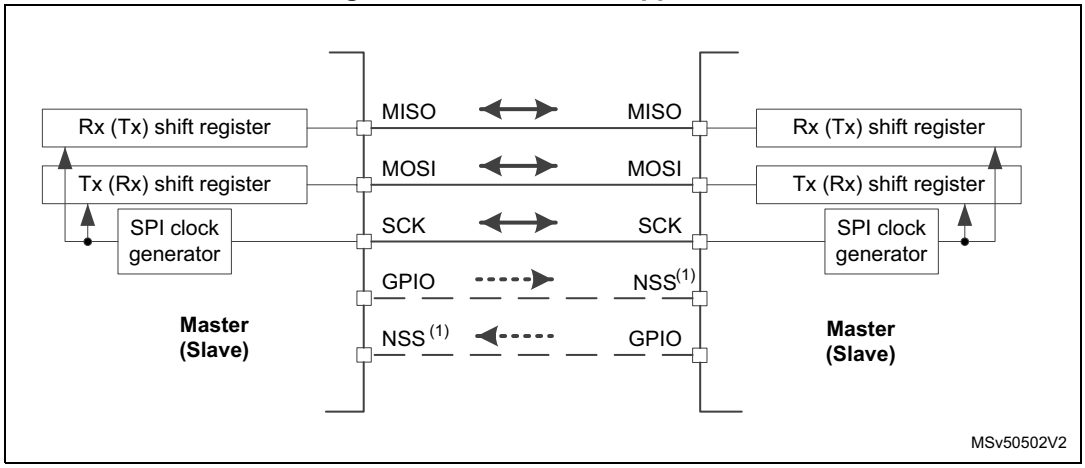
Unless the SPI bus is not designed primarily for a multimaster capability, the user can use a built-in feature that detects a potential conflict between two nodes trying to master the bus at the same time. For this detection, the NSS pin is configured in hardware input mode. The connection of more than two SPI nodes working in this mode is impossible, as only one node can apply its output on a common data line at a given time.

When the nodes are not active, both stay in slave mode by default. Once a node wants to overtake control on the bus, it switches itself into master mode and applies active level on the slave select input of the other node via the dedicated GPIO pin. After the session is

complete, the active slave select signal is released and the node mastering the bus temporarily returns back to passive slave mode waiting for the next session to start.

If both nodes raise their mastering request at the same time, a bus conflict event appears (see mode fault MODF event). The user can apply some simple arbitration process (for example postpone the next attempt by different predefined timeouts applied to both nodes).

Figure 886. Multimaster application



1. The NSS pin is configured at hardware input mode at both nodes. Its active level enables the MISO line output control since the passive node is configured as a slave.
2. The RDY signal is not used in this communication.

67.4.7 Slave select (NSS pin) management

In slave mode, the NSS works as a standard ‘chip select’ input and lets the slave communicate with the master. In master mode, the NSS can be used either as an output or an input. As an input it can prevent a multimaster bus collision, and as an output it can drive a slave select signal of a single slave. The NSS signal can be managed internally (software management of the NSS input) or externally when both the NSS input and output are associated with the NSS pin (hardware slave select management). The user can configure which level of this input/output external signal (present on the NSS pin) is considered as active by the SSIOP bit setting. NSS level is considered as active if it is equal to SSIOP.

The hardware or software slave select management can be set using the SSM bit in the SPI_CFG2 register:

- **Software slave select management (SSM = 1)**: in this configuration, slave select information is driven internally by the SSI bit value in the SPI_CR1 register. The external NSS pin is free for other application uses (such as GPIO or other alternate functions).
- **Hardware slave select management (SSM = 0)**: in this case, there are two possible configurations. The configuration used depends on the NSS output configuration (SSOE bit in the SPI_CFG2 register).
  - **NSS output enable (SSOE = 1)**: this configuration is only used when the MCU is set as master. The NSS pin is managed by the hardware. The functionality is tied to CSTART and EOT control. As a consequence, the master must apply the proper TSIZE > 0 setting to control the NSS output correctly. Even if SPI AF is not applied at the NSS pin (it can be used as a standard GPIO then), keep anyway SSOE = 1 to ensure the default NSS input level and prevent any mode fault

evaluation at the input of the master NSS internal logic applicable at a multimaster topology exclusively.

a) When SSOM = 0 and SP = 000, the NSS signal is driven to the active level as soon as the master transfer starts (CSTART = 1) and it is kept active until its EOT flag is set or the transmission is suspended.

b) When SP = 001, a pulse is generated as defined by the TI mode.

c) When SSOM = 1, SP = 000 and MIDI > 1 the NSS is pulsed inactive between data frames, and kept inactive for a number of SPI clock periods defined by the MIDI value decremented by one (from 1 to 14).

d) NSS input is forced to nonactive state internally at master to prevent any mode fault.

– **NSS output disable (SSM = 0, SSOE = 0)**:

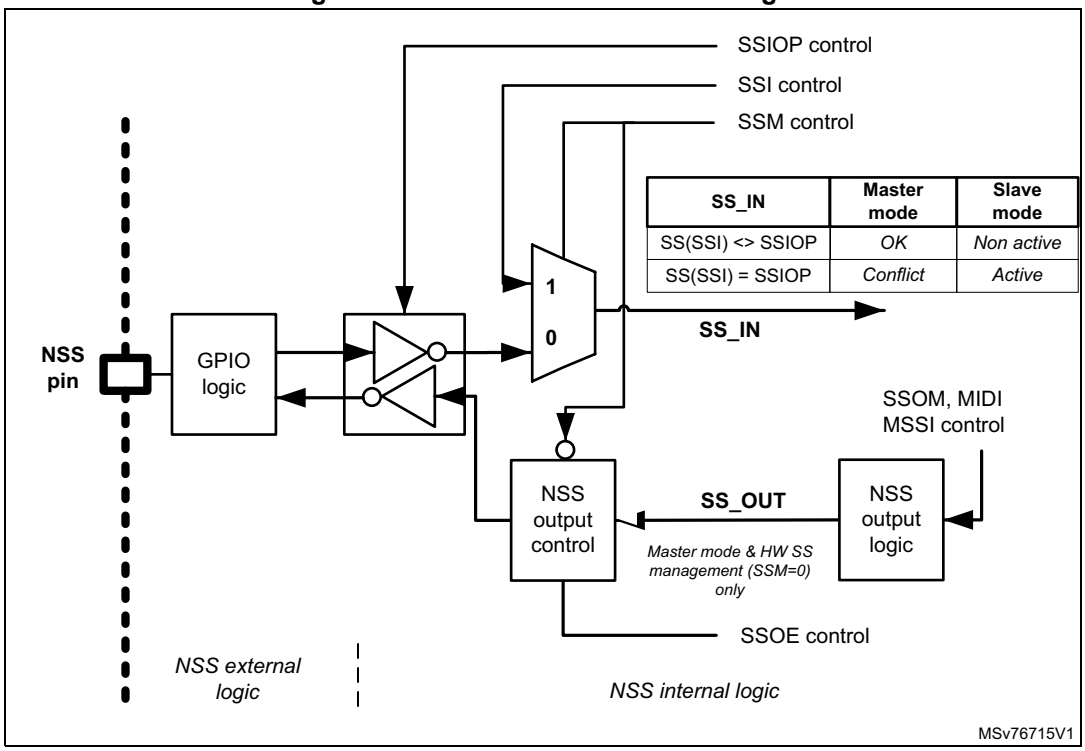
a) If the microcontroller is acting as the master on the bus, this configuration allows multimaster capability. If the NSS pin is pulled into an active level in this mode, the SPI enters master mode fault state and the SPI device is automatically reconfigured in slave mode (MASTER = 0).

b) In slave mode, the NSS pin works as a standard ‘chip select’ input and the slave is selected while the NSS line is at its active level.

Note: The purpose of automatic switching into slave mode when a mode fault occurs is to avoid the possible conflicts on data and clock lines. As the SPE is automatically reset in this condition, both Rx and Tx FIFOs are flushed and current data is lost.

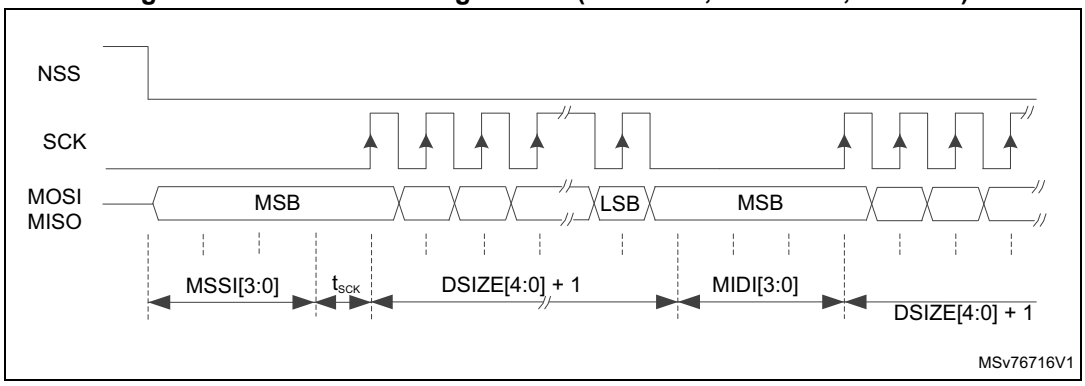
When the SPI slave is enabled in the hardware slave select management mode, all the transfers are ignored even in case of the NSS is found at active level. They are ignored until the slave detects a start of the NSS signal (transition from nonactive to active level) just synchronizing the slave with the master. This is because the hardware management mode cannot be used when the external NSS pin is fixed. There is no such protection in the slave select software management. Then the SSI bit must be changed when there is no traffic on the bus and the SCK signal is in idle state level between transfers exclusively in this case.

Figure 887. Scheme of NSS control logic



When the hardware output NSS control is applied (SSM = 0, SSOE = 1), by configuration of the MIDI[3:0] and MSSI[3:0] bitfields, the user can control the timing of the NSS signal between data frames and can insert an extra delay at the beginning of every transfer (to separate the NSS and clock starts). This can be useful when the slave needs to slow down the flow to obtain sufficient room for correct data handling (see *Figure 888*).

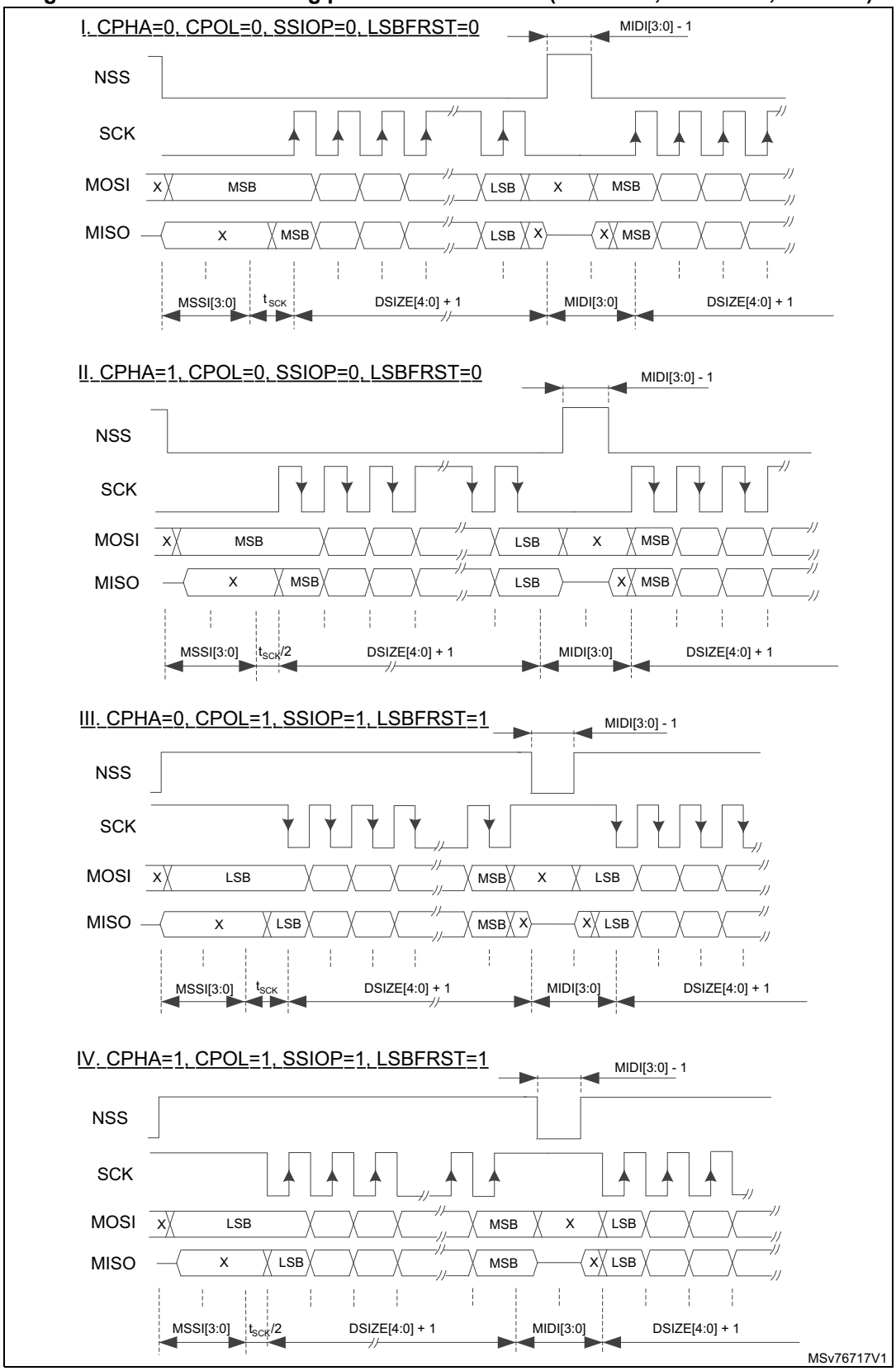
Figure 888. Data flow timing control (SSOE = 1, SSOM = 0, SSM = 0)



1. MSSI[3:0] = 0011, MIDI[3:0] = 0011 (SCK flow is continuous when MIDI[3:0] = 0).
2. CPHA = 0, CPOL = 0, SSIOP = 0, LSBFRST = 0.

Additionally, setting the SSOM bit invokes a specific mode, which interleaves pulses between data frames if there is a sufficient space to provide them (MIDI[3:0] must be set greater than one SPI period). Some configuration examples are shown in *Figure 889*.

Figure 889. NSS interleaving pulses between data (SSOE = 1, SSOM = 1, SSM = 0)



1. MSSI[3:0] = 0010, MIDI[3:0] = 0010.
2. NSS interleaves between data when MIDI[3:0] > 1 wide of the interleaving pulse is always one SCK period less than the gap provided between the frames (defined by the MIDI parameter). If MIDI is set, the frames are separated by a single SCK period but no interleaving pulse appears on NSS.

67.4.8 Ready pin (RDY) management

The status of the slave capability to handle data can be checked on the RDY pin. By default, a low level indicates that the slave is not ready for transfer. The reason can be that the slave TxFIFO is empty, RxFIFO full or the SPI is disabled. An active level of the signal can be selected by the RDIOP bit. If the master continues or starts to communicate with the slave when it indicates a not ready status, it is highly probable that the transfer fails.

The logic to control the RDY output is rather complex, tied closely with the TSIZE and DSIZE settings. The RDY reaction is more pessimistic and sensitive to TxFIFO becoming nearly empty and/or RxFIFO nearly full during a frame transfer. This pessimistic logic is suppressed at the end of a transfer only when RDY stays active, despite TxFIFO becomes fully empty and/or RxFIFO becomes fully occupied. The target is to prevent any data corruption and inform the master in time that it is necessary to suspend the transfer temporarily until the next transferred data can be processed safely again. When the RDY signal input is enabled at master side, the master suspends the communication once the slave indicates not ready status. This prevents the master to complete the transfer of an ongoing frame, which just empties the slave TxFIFO or full fills its RxFIFO until a next data is written and/or read there (despite the frame still can be completed without any constraint). It can make a problem if the TSIZE = 0 configuration is applied at slave because slave then never evaluates the end of the transfer (which suppresses the not ready status just when the last data is sent). Then the user has to release the RxFIFO and/or write additional (even dummy) data to TxFIFO by software at slave side to release the not RDY signal, unblock ST master and so enable it to continue at the communication suspended at middle of a frame occasionally.

When RDY is not used by the master, it must be disabled (RDIOM = 0). Then an internal logic of the master simulates the slave status always ready. In this case, the RDIOP bit setting has no meaning.

Due to synchronization between clock domains and evaluation of the RDY logic on both master and slave sides, the RDY pin feature is not reliable and cannot be used when the size of data frames is configured shorter than 8 bits.

67.4.9 Communication formats

During SPI communication, receive and transmit operations are performed simultaneously. The serial clock (SCK) synchronizes the shifting and sampling of the information on the data lines. The communication format depends on the clock phase, the clock polarity, and the data frame format. To be able to communicate together, the master and slave devices must follow the same communication format and be synchronized correctly.

Clock phase and polarity controls

Four possible timing relationships can be chosen by software, using the CPOL and CPHA bits in the SPI_CFG2 register. The CPOL (clock polarity) bit controls the idle state value of the clock when no data is being transferred. This bit affects both master and slave modes. If CPOL is reset, the SCK pin has a low-level idle state. If CPOL is set, the SCK pin has a high-level idle state.

If the CPHA bit is set, the second edge on the SCK pin captures the first data bit transferred (falling edge if the CPOL bit is reset, rising edge if the CPOL bit is set). Data are latched on each occurrence of this clock transition type. If the CPHA bit is reset, the first edge on the SCK pin captures the first data bit transferred (falling edge if the CPOL bit is set, rising edge if the CPOL bit is reset). Data are latched on each occurrence of this clock transition type.

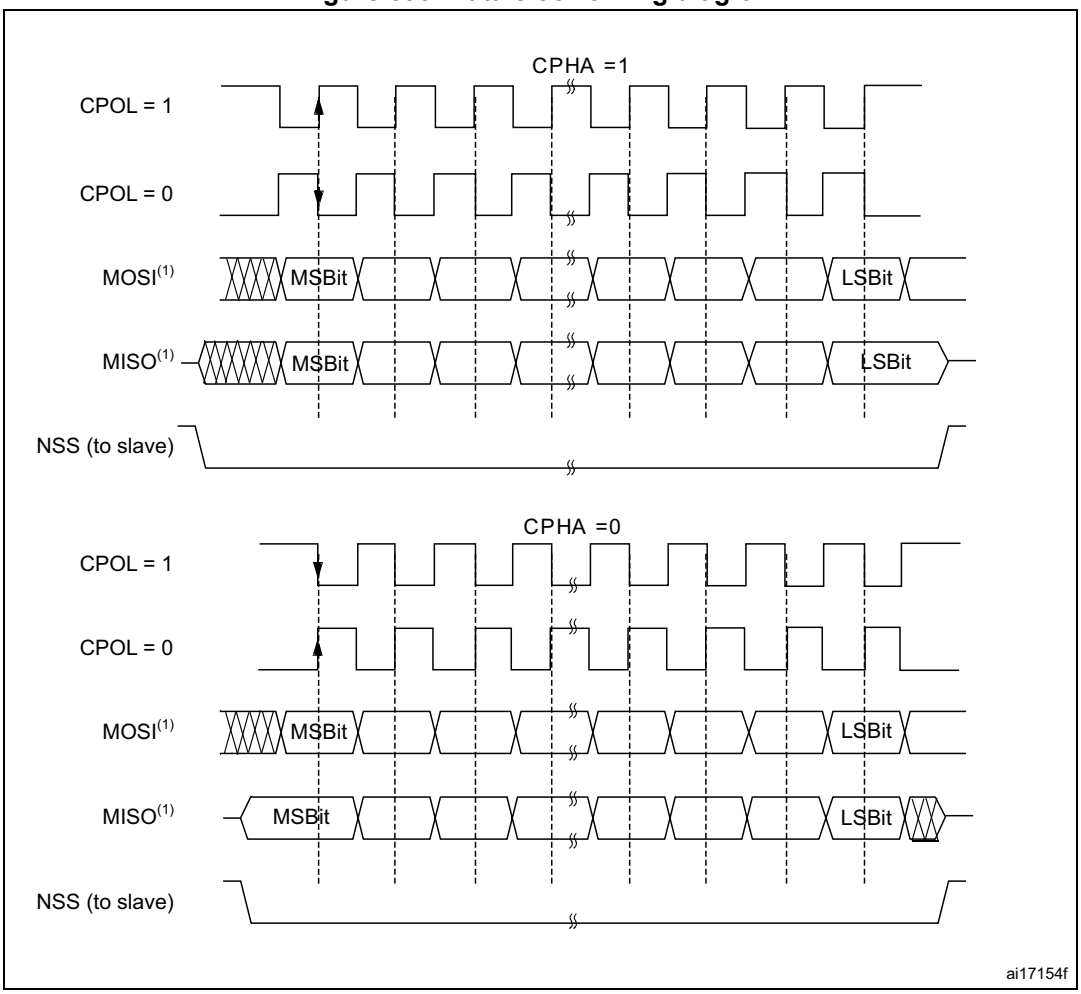
The combination of the CPOL (clock polarity) and CPHA (clock phase) bits selects the data capture clock edges (dotted lines in *Figure 890*).

Figure 890 shows an SPI full-duplex transfer with the four combinations of the CPHA and CPOL bits.

Note: Prior to changing the CPOL/CPHA bits the SPI must be disabled by resetting the SPE bit.

The idle state of SCK must correspond to the polarity selected in the SPI_CFG2 register (by pulling the SCK pin up if CPOL = 1 or pulling it down if CPOL = 0).

Figure 890. Data clock timing diagram



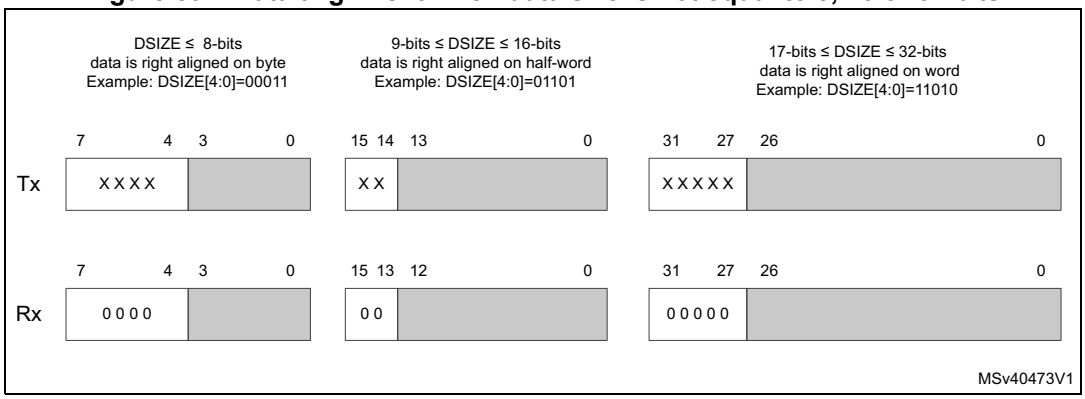
1. The order of data bits depends on the LSBFRST bit setting.

Data frame format

The SPI shift register can be set up to shift out MSB-first or LSB-first, depending on the value of the LSBFRST bit of the SPI_CFG2 register.

The data frame size is configured through the DSIZE bitfield of the SPI_CFG1 register to a range that depends on the SPI instance (see *Section 67.3: SPI implementation*). The setting applies to both transmission and reception. The data bits in the SPI_TXDR/SPI_RXDR registers are right-aligned with 8, 16, or 32 bits, depending on the data size (see *Figure 891*). These registers can consequently be accessed by bytes, half-words, or words. FIFO accesses smaller than a single data are forbidden. When the FIFO occupancy flag is raised, a FIFO access greater than the FIFO threshold can lead to spurious data being read or written data being lost. When the access to/from the SPI_TXDR/SPI_RXDR registers is a multiple of the configured data size, data packing is applied automatically, while the lowest significant bits/bytes are communicated first. For more details, see *Section 67.4.12: SPI data transmission and reception procedures*.

Figure 891. Data alignment when data size is not equal to 8, 16 or 32 bits



67.4.10 Configuring the SPI

The configuration procedure is almost the same for the master and the slave. For specific mode setups, follow the dedicated chapters. When a standard communication must be initialized, perform these steps:

1. Write the proper GPIO registers: configure GPIO alternate functions at MOSI, MISO, SCK, NSS, and RDY pins if applied.
2. Write into the SPI_CFG1 and SPI_CFG2 registers and set up the proper values of all ‘not reserved’ bits and bitfields, prior to enabling the SPI, with the following exceptions:
   a) The SSOM, MASRX, SSOE, RDIOM, MBR[2:0], BPASS, MIDI[3:0], MSSI[3:0] bits are taken into account in master mode only, the MSSI[3:0] bits take effect when the SSOE bit is set, the RDIOP bit takes no effect when the RDIOM bit is not set in master mode. When the slave is configured in TI mode, the MBR[2:0] setting is also considered.
   b) UDRCFG is taken into account in slave mode only.
   c) CRCSIZE[4:0] bitfield must be configured if CRCEN is set.
   d) CPOL, CPHA, LSBFRST, SSOM, SSOE, SSIOP, SSM, RDIOP, RDIOM, MSSI, and MIDI are not required in TI mode.

e) Once the AFCNTR bit is set in the SPI_CFG2 register, all the SPI outputs start to be propagated onto the associated GPIO pins regardless of the peripheral enable. So, any later configuration changes of the SPI_CFG1 and SPI_CFG2 registers can affect the level of signals on these pins.

3. Write to the SPI_CR2 register to select the length of the transfer, if it is not known TSIZE must be programmed to zero.
4. Write to the SPI_CRCPOLY and into the TCRCINI, RCRCINI, and CRC33_17 bits of the SPI_CR1 register to configure the CRC polynomial and CRC calculation if needed.
5. Configure DMA streams dedicated for the SPI Tx and Rx in DMA registers if the DMA streams are used (see *Section 67.4.14: Communication using DMA (direct memory addressing)*).
6. Configure SSI, HDDIR, and MASRX in the SPI_CR1 register if required.
7. Program the IOLOCK bit in the SPI_CR1 register if the configuration protection is required (for safety).

67.4.11 Enabling the SPI

It is recommended to configure and enable the SPI slave before the master sends the clock. But there is no impact if the configuration and enabling procedure is done while traffic is ongoing on the bus, assuming that the NSS signal is managed by hardware at slave or kept inactive by the slave software when the software management of the NSS signal is applied (see *Section 67.4.7*). To prevent any risk of any data underrun, all the data to be sent have to be written to the slave transmitter data register before the master starts its clocking. The SCK signal must be settled to the idle state level corresponding to the selected polarity, before the SPI slave is selected by NSS, else the following transfer may be desynchronized.

When the SPI slave is enabled at the hardware slave select management mode, all the transfers are ignored even in case of the NSS is found at active level. They are ignored until the slave detects a start of the NSS signal (its transition from nonactive to active level) just synchronizing the slave with the master. That is why the hardware management mode cannot be used when the external NSS pin is fixed. There is no such protection at the slave select software management. In this case, the SSI bit must be changed when there is no traffic on the bus and the SCK signal is at idle state level between transfers exclusively in this case.

The master in full duplex (or in any transmit-only mode) starts to communicate when the SPI is enabled, the CSTART bit is set, and the TxFIFO is not empty, or with the next write to TxFIFO.

In any master receive-only mode, the master starts to communicate and the clock starts running after the SPI is enabled and the CSTART bit is set.

For handling DMA, see *Section 67.4.14*.

67.4.12 SPI data transmission and reception procedures

The SPI can transfer data at a very high communication speed. Even though the data is FIFO buffered, handling a continuous data flow by servicing data frames individually leads to an enormous CPU or DMA load, in particular when the data size is small. This potentially leads to a significant limitation of the overall system performance, as well as communication errors such as data overrun or underrun that may be raised when the system latencies in servicing requests become comparable to single data frame transfer time.

The SPI offers advanced hardware control features to prevent these issues from occurring:

- Decreasing the number of events requiring service by collecting the data frames into larger **data packets**

  The number of data frames per packet (FIFO threshold) can be configured. The complete data packet generates a FIFO occupancy event, which is then handled within a single, compact, service.
- Decreasing the number of CPU execution cycles required for the service, by applying the appropriate **data access**:
  - The widest access to the SPI data registers must be applied. This allows cumulative data to be handled by a single register read or write operation (data packing).
  - The number of read/write accesses necessary to complete the data packet must be aligned with the data frame size and the FIFO threshold.
- **Concurrent read and write services** to handle full-duplex data flow based on common FIFO occupancy events.
- Embedded **hardware data counters** to define the exact number of data involved in transfers.

Note: Using these features is optional. Nothing prevents the application from handling the data flow frame by frame.

The following sections give more details and describe specific cases for using these advanced hardware control features to handle data transfers.

Data packet control

The data frame size (number of bits in the frame) is defined through the DSIZE bitfield of the SPI_CFG1 register. The number of data frames per packet can be configured by selecting the FIFO threshold through the FTHLV bitfield of the SPI_CFG1 register. If the threshold value is set to zero, each completed data frame raises the FIFO occupancy event. The data packet occupancy must not exceed half of the FIFO size, which depends on the SPI instance of the product. The FIFO capacity (multiple of 8 bits) is consumed according to the following rules, depending on the data frame size:

- Data frames from 4 to 8 bits occupy one byte of the FIFO.
- Data frames from 9 to 16 bits occupy two bytes of the FIFO.
- Data frames from 17 to 24 bits occupy three bytes of the FIFO.
- Data frames from 25 to 32 bits occupy four bytes of the FIFO.

For example, if the FIFO capacity is 16 bytes, it accommodates up to five 24-bit frames. In this case, the data packet configuration must not exceed two data frames.

The SPI features two separate FIFOs to handle the reception and transmission data flow, the RxFIFO, and the TxFIFO. Their content can be handled by monitoring the occupancy

flags RXP, TXP, and DXP of the SPI_SR register according to the SPI mode (duplex or simplex):

- If RXP is set, at least one complete data packet can be read from the RxFIFO.
- If TXP is set, at least one complete data packet can be written to the TxFIFO.
- If DXP is set, at least one complete data packet can be read from the RxFIFO and written to the TxFIFO at full-duplex mode.

These flags can be polled by software, or they can trigger an interrupt and/or a DMA request (if enabled through the RXPIE, TXPIE, and DXPIE bits of the SPI_EIR register or the RXDMAEN and TXDMAEN bits of the SPI_CFG1 register).

Once an occupancy flag is set, the software or the DMA must read and/or write a complete data packet before checking the flag again to verify if the next packet can be handled. This cycle can be repeated until the corresponding flag is read at zero.
Both RxFIFO and TxFIFO contents are flushed and cannot be accessed when the SPI is disabled (SPE cleared in the SPI_CR1 register).

Data packing versus data register access control

The content of the RxFIFO and TxFIFO can be accessed by reading/writing from/to the SPI data registers SPI_RXDR and SPI_TXDR, respectively. A read access from the SPI_RXDR register returns the oldest values stored in the RxFIFO that have not been read yet. A write access to the SPI_TXDR register stores the data written at the end of the send queue in the TxFIFO.

These data registers can be accessed by 8-, 16-, or 32-bit read and write CPU instructions, forced by the register address casting applied by the software. Data packing is performed automatically by hardware if the data access applied by the software is a multiple of the data size. It allows handling more than one data in parallel in a single data register access. Then the SPI operates using the lowest significant byte or half-word first. FIFO data accesses of less than the configured data size are forbidden. To avoid spurious data being read or written data being lost, a complete data packet (configured through the FIFO threshold) must be serviced when the corresponding FIFO occupancy flag is set. If the data pattern is not byte-, half-word, or word-aligned, only the valid data bits are stored right-aligned to the FIFO and transferred on the bus. Unused bits are discarded on the transmitter side and padded with zeros on the receiver side.

For example, if the data frame size fits into one byte, the data packing is used automatically when a 16-bit or 32-bit read/write access is performed by software from/to the SPI_RXDR/SPI_TXDR register. In this case, two respectively four data are handled by a single 16-bit respectively 32-bit access. Additionally, if such data frames are grouped into packets of four via the FIFO threshold setting, the packet to be serviced is signalized by raising the corresponding FIFO occupancy flag (TXP, RXP, or DXP). If the instance FIFO threshold is sufficient, a packet containing eight 8-bit data frames can be handled by two consecutive 32-bit accesses upon the same single threshold event. The most efficient data handling is achieved when the data packet size (defined by DSIZE and FTHLV bitfields) is aligned with the read/write access from/to the data registers.

Concurrent read and write services

In full-duplex mode, both TxFIFO and RxFIFO packet occupancies can be monitored through a common FIFO flag (DXP). When the DXP flag is set, the application performs the specified number of writes to SPI_TXDR to upload the content of one entire data packet for transmission, followed by the same number of reads from SPI_RXDR to download the

content of one received data packet. Once one data packet is uploaded and one packet is downloaded, the application software or the DMA checks again the DXP flag and repeats the data packet read and write operation sequence until DXP is read as zero.

The drawback of services based on DXP exclusively is that servicing the TxFIFO is delayed on purpose due to the SPI nature since the TXP events precedes the RXP ones. To allow continuous SCK clock flow on the master side and prevent underrun on the slave side, it is recommended to prefill a few data ahead to the TxFIFO at the transfer start, and wait until the transfer is complete to read the last received data.

Hardware data counter

If a hardware data counter is used (TSIZE bitfield of the SPI_CR2 register set to a nonzero value), the application software does not need to calculate the remaining number of data to be handled. The end of transfer automatically controls the CRC, as well as the hardware slave select management, when used. The user application does not need to ensure that the overall number of data is a multiple of the packet size. If the last data packet is incomplete, it is serviced in the same way as any full packet. The unused part of this last packet is not handled by the peripheral. Only the valid data are written into the TxFIFO and/or read from the RxFIFO, and the redundant writes and reads are discarded. When the hardware counter reaches zero, the EOT flag is raised, and the RXP flag is not set for the last data packet. If the last packet is not aligned with the packet size, the TXP and EOT occupancy events are not related to the configured packet size but to the number of remaining data calculated by hardware.

If TSIZE is kept at zero (for example due to an unpredictable number of data), only the number of transferred data corresponding to the FIFO thresholds is supported. If some data are not aligned with the configured packet size, they may remain pending and available in the RxFIFO. In this case, the FTHLV level is not reached and the RXP flag is not set. Then the number of remaining received data frames in the RxFIFO is indicated by the RXWNE and RXPLVL bitfields of the SPI_SR register. Nevertheless, the application software can still read the complete data packet from the RxFIFO and the redundant data are read as zero. To prevent such an unaligned data reception, the user must configure the FIFO threshold to a single data (FTHLV = 0). In this case, each data frame is serviced by its own RXP occupancy event.

Data transfer handling

Data are transferred using MOSI and MISO lines, depending on the SPI communication mode and associated configuration. The mode is configured through the COMM[1:0] bitfield of the SPI_CFG2 register. The HDDIR bit of the SPI_CR1 register controls the data flow direction in half-duplex mode. The communication flow is handled by the master via the NSS and SCK signals. The slave selected for the communication is fully subordinated to the master communication activity, no matter if it handles the data flow on time or not. The slave can only temporarily suspend the master communication by using the RDY signal. The active levels of RDY and NSS signals can be changed, or MISO and MOSI functionality swapped (via the RDIOP, SSIOP and IOSWP bits of the SPI_CFG2 register).

The SPI master transmitter can operate in full-duplex, simplex, or half-duplex mode. In full-duplex mode, data are received synchronously. The master starts the data transfer once the CSTART bit of the SPI_CR1 register is set, provided the SPI is enabled and the TxFIFO

content is not empty. The master then provides the serial clock signal continuously on the SCK pin until:

- The total number of required data programmed in TSIZE is transferred, or
- The transfer is suspended.

An automatic temporary suspension of the master transmission occurs when:

- The slave does not assert the RDY signal, if it is used, or
- The TxFIFO becomes empty, or
- In full-duplex mode, the RxFIFO becomes full while the automatic suspension is enabled (MASRX set in the SPI_CR1 register).

When an automatic suspension occurs, the master stops providing the clock, and the transfer proceeds depending on the cause of the suspension, when:

- The slave asserts RDY.
- The master software or the DMA writes additional data to TxFIFO.
- In full-duplex mode, the master software or the DMA releases the RxFIFO to enable it to accommodate new data.

The SPI master can receive data only when it operates in simplex receiver or half-duplex receiver mode. In these modes, the master starts the data transfer when the SPI is enabled, and the transfer is released by setting the CSTART bit. The serial clock signal is then provided continuously on the SCK pin by the master until:

- The total number of required data programmed in TSIZE is received, or
- The transfer is suspended by the master.

An automatic temporary suspension of the reception occurs when:

- The slave does not assert the RDY signal, if it is used, or
- The RxFIFO becomes full while the automatic suspension is enabled (MASRX bit set in the SPI_CR1 register).

When an automatic suspension occurs, the master stops providing the clock and the transfer proceeds depending on the cause of the suspension, when:

- The slave asserts RDY.
- The master software or the DMA releases the RxFIFO to enable it in order to accommodate new data.

A preferable way to terminate a transfer is to program the TSIZE bitfield to generate an EOT event. Hardware NSS signal or CRC handshake can then be controlled. To restart the internal state machine properly, it is recommended to disable the SPI and enable it again when the transfer is complete, even if the SPI configuration is not changed.

A transfer can be suspended at any time by setting the CSUSP bit of the SPI_CR1 register, which clears the CSTART bit. This software suspension control ensures the completion of any ongoing data frame. To restart the internal state machine properly, the SPI must be disabled and enabled again before the next transfer starts.

Oppositely, a temporary automatic suspension controlled by hardware typically results in a frozen and incomplete data frame transfer. This depends on baud rate setting, but usually, due to internal synchronization delays, the SCK signal stops when a few bits from the next data frame are already transferred on the bus. Once the suspension is released, the data frame transmission completes by transferring the remaining bits. That is why this automatic suspension is not quite reliable when the data frame size is less than eight bits. When shorter data frames are used, to prevent any data loss and assure proper RDY and/or

MASRX operation, the user can interleave and so extend the transfer time by inserting additional dummy clock cycles so that the period for a single data frame is higher or equal to eight SCK duration. This is done by setting the MIDI[3:0] bitfield of the SPI_CFG2 register.

Caution: If the SPE bit is cleared in master mode while the transfer is ongoing without any suspension, the clock is stopped even if the current frame transmission is not complete, and the content of the FIFOs is flushed and lost.

67.4.13 Disabling the SPI

To disable the SPI, it is mandatory to follow the disable procedures described in this paragraph.

In the master mode, it is important to do this before the system enters a low-power mode when the peripheral clock is stopped, otherwise, ongoing transfers may be corrupted.

In slave mode, the SPI communication can continue when the **spi_pclk** and **spi_ker_ck** clocks are stopped, without interruption, until any end of communication or data service request condition is reached. The **spi_pclk** can generally be stopped by setting the system into Stop mode. Refer to the RCC section for further information.

The master in full-duplex or transmit-only mode can finish any transfers when it stops providing data for transmission. In this case, the clock stops after the last data transfer. TXC flag can be polled (or interrupt enabled with EOTIE = 1) in order to wait for the last data frame to be sent.

When the master is in any receive-only mode, to stop the peripheral, the SPI communication must first be suspended, by setting the CSUSP bit.

The data received but not read remain stored in RxFIFO when the SPI is suspended.

After such a software suspension, SPI must always be disabled to restart the internal state machine properly.

When SPI is disabled, RxFIFO is flushed. To prevent losing unread data, the user must ensure that RxFIFO is empty when disabling the SPI, by reading all remaining data (as indicated by the RXP, RXWNE, and RXPLVL fields in the SPI_SR register).

The standard disable procedure is based on polling EOT and/or TXC status to check if a transmission session is (fully) completed. This check can be done in specific cases, too, when it is necessary to identify the end of ongoing transfers, for example:

- When the master handles the NSS signal by a GPIO not related to SPI (for example at case of multislave star topology) and it has to provide a proper end-of-NSS pulse for the slave, or
- When transfers from DMA or FIFO are completed while the last data frame or CRC frame transfer is still ongoing in the peripheral bus.

When TSIZE > 0, EOT and TXC signals are equal so polling of EOT is reliable at whatever SPI communication mode to check the end of the bus activity. When TSIZE = 0, the user has to check TXC, SUSP, or FIFO occupancy flags according to the applied SPI mode and the way of the data flow termination.

The correct disable procedure in master mode, except when receive-only mode is used, is:

1. Wait until TXC = 1 and/or EOT = 1 (no more data to transmit and last data frame sent). When CRC is used, it is sent automatically after the last data in the block is processed.

TXC/EOT is set when the CRC frame is complete in this case. When a transmission is suspended the software has to wait until the CSTART bit is cleared.

2. Read all RxFIFO data (until RXWNE = 0 and RXPLVL = 00).
3. Disable the SPI (SPE = 0).

The correct disable procedure for master receive-only modes is:

1. Wait on EOT or break the receive flow by suspending SPI (CSUSP = 1).
2. Wait until SUSP = 1 (the last data frame is processed) if the receive flow is suspended.
3. Read all RxFIFO data (until RXWNE = 0 and RXPLVL = 00).
4. Disable the SPI (SPE = 0).

In slave mode, any ongoing data are lost when disabling the SPI.

Controlling the I/Os

As soon as the SPI is disabled, the associated and enabled AF outputs can still be driven by the device depending on the AFCNTR bit of the SPI_CFG2 register. When active output control is applied (AFCNTR = 1) and SPI has just been disabled (SPE = 0), the enabled outputs associated with SPI control signals (like NSS and SCK at master and RDY at slave) can immediately toggle to inactive level (according to SSIOP and CPOL settings at master and RDIOP at slave respectively). Instead, the data line output (MOSI at master and MISO at slave) can immediately change its level depending on the actual TxFIFO content, with the effect of potentially making invalid and no more guaranteed the value of the latest transferred bit on the bus. If necessary, the user has to take care about proper data hold time at the data line and avoid any eventual fast SPI disable just after the last data transfer is complete.

Note: Despite stability of the latest bit is guaranteed by design during the sampling edge of the clock, some devices can require even extension of this data bit stability interval during the sampling. It can be done, for example by inserting a small software delay between EOT event occurrence and SPI disable action.

67.4.14 Communication using DMA (direct memory addressing)

To operate at its maximum speed and to facilitate the data register read/write process required to avoid overrun, the SPI features a DMA capability, which implements a simple request/acknowledge protocol.

A DMA access is requested when the TXDMAEN or RXDMAEN enable bits of the SPI_CFG1 register are set. Separate requests must be issued to the Tx and Rx buffers to fulfill the service of the defined packet.

- In transmission, a series of DMA requests is triggered each time TXP is set. The DMA then performs a series of writes to the SPI_TXDR register.
- In reception, a series of DMA requests is triggered each time RXP is set. The DMA then performs a series of reads from the SPI_RXDR register. When EOT is set at the end of the transfer and the last data packet is incomplete, then DMA request is activated automatically according to RXWNE and RXPLVL[1:0] setting to read the rest of data.

If the SPI is programmed in receive-only mode, UDR is never set.

If the SPI is programmed in a transmit mode, TXP and UDR can be eventually set at slave side, because transmit data may not be available. In this case, some data are sent on the TX line according with the UDR management selection.

When the SPI is used at a simplex mode, the user must enable the adequate DMA channel only while keeping the complementary unused channel disabled.

If the SPI is programmed in transmit-only mode, RXP and OVR are never set.

If the SPI is programmed in full-duplex mode, RXP and OVR are eventually set, because received data are not read.

In transmission mode, when the DMA or the user has written all the data to be transmitted (the TXTF flag is set in the SPI_SR register), the EOT (or TXC at case TSIZE = 0) flag can be monitored to ensure that the SPI communication is complete. This is required to avoid corrupting the last transmission before disabling the SPI or before disabling the **spi_pclk** in master mode. The software must first wait until EOT = 1 and/or TXC = 1.

When starting communication using DMA, to prevent DMA channel management raising error events, these steps must be followed in order:

1. Enable the DMA Rx buffer in the RXDMAEN bit of the SPI_CFG1 register, if DMA Rx is used.
2. Enable DMA requests for Tx and Rx in DMA registers, if the DMA is used.
3. Enable the DMA Tx buffer in the TXDMAEN bit of the SPI_CFG1 register, if DMA Tx is used.
4. Enable the SPI by setting the SPE bit.

To close communication, it is mandatory to follow these steps in order:

1. Disable DMA requests for Tx and Rx in the DMA registers, if the DMA issued.
2. Disable the SPI by following the SPI disable procedure.
3. Disable DMA Tx and Rx buffers by clearing the TXDMAEN and RXDMAEN bits of the SPI_CFG1 register, if DMA Tx and/or DMA Rx are used.

Data packing with DMA

If the transfers are managed by DMA (TXDMAEN and RXDMAEN set in the SPI_CFG1 register) the packing mode is enabled/disabled automatically depending on the PSIZE value configured for SPI TX and the SPI RX DMA channel.

The packing mode is enabled if the DMA channel PSIZE value is a multiple of the data size. Then the DMA automatically manages the sequences of write and read operations to/from the SPI data registers, based on FIFO occupancy flags, and depending on the FIFO threshold and data size configurations.

The DMA completes the transfer automatically according to the TSIZE field setting, whatever the data packing mode used, and even if the number of data to transfer is not a multiple of the DMA data size (16 bits or 32 bits) while the frame size is smaller.

Alternatively, the last data frames can be written by software, in the single/unpacked mode.

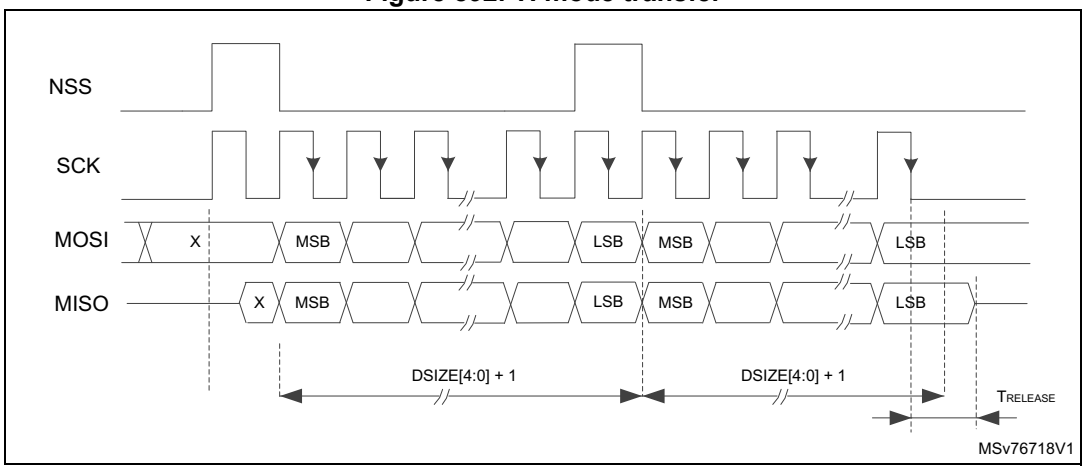
Configuring any DMA data access to less than the configured data size is forbidden. One complete data frame must be always accessed at minimum.

67.5 SPI specific modes and control

67.5.1 TI mode

With a specific SP[2:0] bitfield setting of the SPI_CFG2 register, the SPI can be configured compliant with the TI protocol. The SCK and NSS signals polarity, phase and flow, as well as the bit order are fixed, so the setting of CPOL, CPHA, LSBFRST, SSOM, SSOE, SSIOP, SSM, RDIOP, RDIOM, MSSI, and MIDI is not required when the SPI is in TI mode configuration. The NSS signal synchronizes the protocol by pulses over the LSB data bit as it is shown in *Figure 892*.

Figure 892. TI mode transfer



In slave mode, the clock generator is used to define the time when the slave output at MISO pin becomes high-Z when the current transfer finishes. The master baud rate setting (MBR[2:0] of SPI_CFG1) is applied and any baud rate can be used to determine this moment with optimal flexibility. The delay for the MISO signal to become high-Z (TRELEASE) depends on internal resynchronization, too, which takes the next additional 2-4 periods of the clock signal feeding the generator. It is given by the following formula:

$$\frac{Tbaud}{2} + 2 \times Tspi_ker_ck \leq Trelease \leq \frac{Tbaud}{2} + 4 \times Tspi_ker_ck$$

If the slave detects a misplaced NSS pulse during a data transfer the TIFRE flag is set.

67.5.2 SPI error flags

An SPI interrupt is generated if one of the following error flags is set and the interrupt is enabled by setting the corresponding interrupt enable bit.

Overrun flag (OVR)

An overrun condition occurs when data are received by a master or slave and the RxFIFO has not enough space to store these received data. This can happen if the software or the DMA did not have enough time to read the previously received data (stored in the RxFIFO).

When an overrun condition occurs, the OVR flag is set and the newly received value does not overwrite the previous one in the RxFIFO. The newly received value is discarded and all

data transmitted subsequently are lost. The OVR flag triggers an interrupt if the OVRIE bit is set. Clearing the OVR bit is done by setting the OVRC bit of the SPI_IFCR register. Clearing the RxFIFO content by performing software reads before the OVR bit is cleared reduces the risk of any immediate repetition of its next overrun. It is suggested to release the RxFIFO space as much as possible, this means to read out all the available data packets based on RXP flag indication.

In master mode, the user can prevent the RxFIFO overrun by automatic communication suspend (MASRX bit).

Underrun flag (UDR)

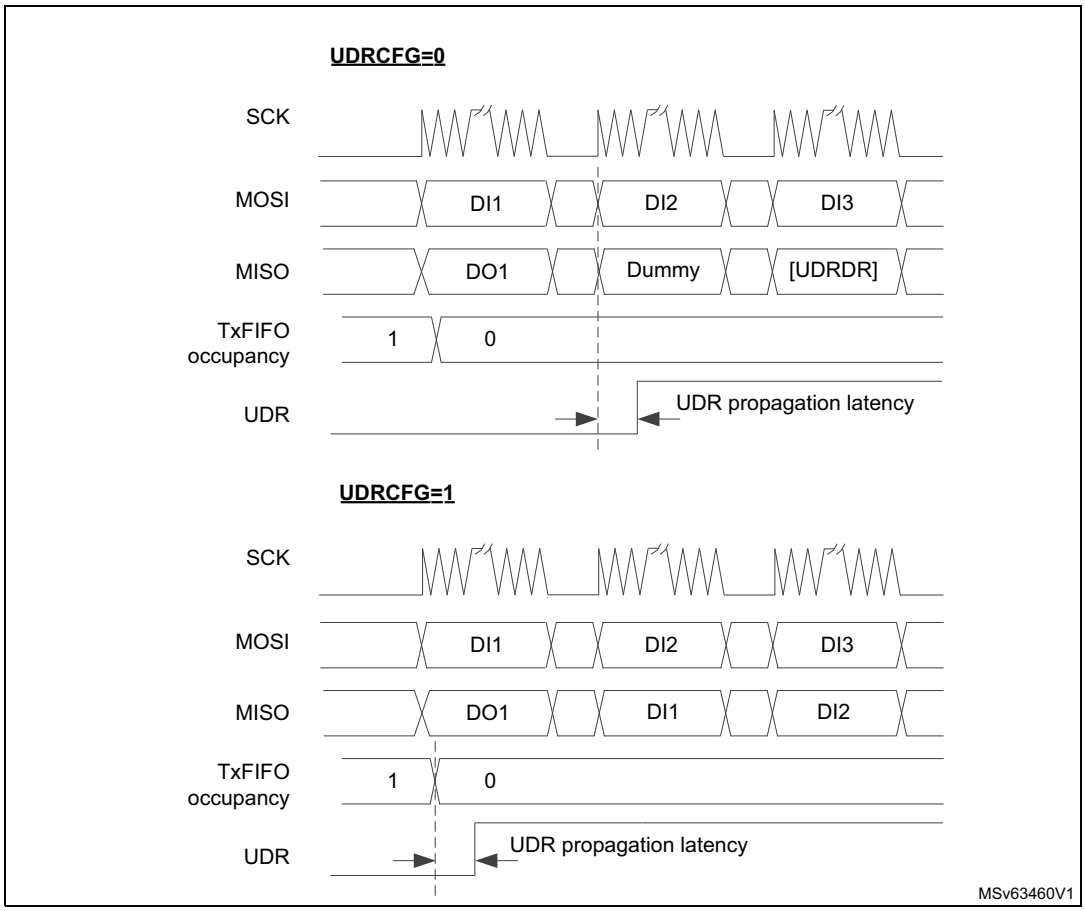
In a slave-transmitting mode, the underrun condition is captured internally by hardware if no data is available for transmission in the slave TxFIFO commonly. The UDR flag setting is then propagated into the status register by hardware (see note below). UDR triggers an interrupt if the UDRIE bit is set.

Underrun detection logic and system behavior depend on the UDRCFG bit. When an underrun is detected by the slave, it can provide out either a constant pattern stored by the user at the UDRDR register or the data received previously from the master. When the first configuration (UDRCFG = 0) is applied, the underrun condition is evaluated whenever the master starts to communicate a new data frame while TxFIFO is empty. Then single additional dummy (accidental) data is always inserted between the last valid data and the constant pattern defined at the UDRDR register (see *Figure 893*). The second configuration (UDRCFG=1) can be used in circular topography structures (see *Figure 885*). Assuming that TxFIFO is not empty when the master starts the communication, the underrun condition is evaluated just once the FIFO becomes empty during the next data flow. Valid data from TxFIFO is then upended by the last received data immediately.

The standard transmission is reenabled once the software clears the UDR flag and this clearing is propagated into SPI logic by hardware. Writing some data to the TxFIFO before the UVR bit is cleared reduces the risk of any immediate repetition of its next underrun.

The data transferred by the slave is unpredictable especially when the transfer starts or continues while TxFIFO is empty and the underrun condition is either not yet captured or just cleared. Typically, this is the case when SPI is just enabled or when a transfer with a defined size just starts. First bits can be corrupted in this case, as well, when the slave software writes the first data into the empty TxFIFO too close prior to starting the data transfer (propagation of the data into TxFIFO takes a few APB clock cycles).

Figure 893. Optional configurations of the slave behavior when an underrun condition is detected



Note: The hardware propagation of an UDR event needs additional traffic on the bus. It always takes a few extra SPI clock cycles after the event happens (both underrun captured by hardware and cleared by software). If clearing of the UDR flag by software is applied close to the end of the data frame transfer or when the SCK line is at idle in between the frames, the next extra underrun pattern is sent initially by the slave before the valid data from TxFIFO becomes transferred again. The user can prevent this by SPI disable/enable action between sessions to restart the underrun logic and so initiate the next session by the valid data.

Mode fault (MODF)

Mode fault occurs when the master device has its internal NSS signal (NSS pin in slave select hardware mode, or SSI bit in slave select software mode) pulled low. This automatically affects the SPI interface in the following ways:

- The MODF bit is set and the SPI interrupt is triggered if the MODFIE bit is set.
- The SPE bit is forced to zero until the MODF bit is set. This disables the SPI and blocks all the peripheral outputs except the MODF interrupt request if enabled.
- The MASTER bit is cleared, thus forcing the device into slave mode.

MODF is cleared by writing 1 to the MODFC bit of the SPI_IFCR register.

To avoid any multiple slave conflicts in a system comprising several MCUs, the NSS pin must be pulled to its nonactive level before reenabling the SPI, by setting the SPE bit.

As a security, the hardware does not allow the SPE bit to be set while the MODF bit is set. In a slave device, the MODF bit cannot be set except as the result of a previous multimaster conflict.

A correct software procedure when a master overtakes the bus in multimaster systems must be the following one:

1. Switch into master mode while SSOE = 0 (potential conflict can appear when another master occupies the bus. In this case, MODF is raised, which prevents any next node switching into master mode).
2. Put GPIO pin dedicated for another master slave select control into active level.
3. Perform a data transfer.
4. Put GPIO pin dedicated for another master slave select control into nonactive level.
5. Switch back to slave mode.

CRC error (CRCE)

This flag is used to verify the validity of the value received when the CRCEN bit of the SPI_CFG1 register is set. The CRCE flag of the SPI_SR register is set if the value received in the shift register does not match the receiver SPI_RXCRC value, after the last data is received (as defined by TSIZE). The CRCE flag triggers an interrupt if the CRCEIE bit is set. Clearing the bit CRCE is done by a writing 1 to the CRCEC bit of the SPI_IFCR register.

TI mode frame format error (TIFRE)

A TI mode frame format error is detected when an NSS pulse occurs during an ongoing communication when the SPI is operating in slave mode and configured to conform to the TI mode protocol. When this error occurs, the TIFRE flag is set in the SPI_SR register. The SPI is not disabled when an error occurs, the NSS pulse is ignored, and the SPI waits for the next NSS pulse before starting a new transfer. The data may be corrupted since the error detection may result in the loss of a few data frames.

The TIFRE flag is cleared by writing 1 to the TIFREC bit of the SPI_IFCR register. If the TIFREIE bit is set, an interrupt is generated on the NSS error detection. As data consistency is no longer guaranteed, communication must be reinitiated by software between master and slave.

67.5.3 CRC computation

Two separate CRC calculators are implemented to check the reliability of transmitted and received data. The CRC polynomial configuration depends on the instance. Refer to *Section 67.3: SPI implementation* for more information.

The length of the polynomial is defined by the most significant bit of the value stored in the SPI_CRCPOLY register. It must be greater than the data frame size (in bits) defined in the DSIZE[4:0] bitfield of the SPI_CFG1 register. To obtain a full-size polynomial, the polynomial length must exceed the maximum data size of the peripheral instance, and the CRC33_17 bit of the SPI_CR1 register must be set to select the most significant bit of the polynomial string.

For example, to select the standard CRC16-CCITT (XMODEM) polynomial x^16 + x^12 + x^5 + 1:

- For a 32-bit instance: write 0x11021 to the SPI_CRCPOLY register and clear the CRC33_17 bit.
- For a 16-bit instance: to obtain the full size, write 0x1021, and set the CRC33_17 bit.

The CRCSIZE field of the SPI_CFG1 register then defines how many most significant bits from the CRC calculation registers are transferred and compared as CRC frame. It is defined independently from the data frame length, but it must be either equal or an integer multiple of the data frame size. Its size cannot exceed the maximum data size of the instance.

To fully benefit from the CRC calculation capability, the polynomial length setting must correspond to the CRC frame size, else the bits unused at the calculation are transferred and expected all zero at the end of the CRC frame if its size is set greater than the polynomial length.

CRC principle

The CRC calculation is enabled by setting the CRCEN bit of the SPI_CFG1 register before the SPI is enabled (SPE = 1). The CRC value is then calculated using the CRC polynomial defined by the CRCPOLY register and CRC33_17 bit. When SPI is enabled, the CRC polynomial can be changed but only in the case when there is no traffic on the bus.

The CRC computation is done, bit by bit, on the sampling clock edge defined by the CPHA and CPOL bits of the SPI_CR1 register. The calculated CRC value is checked automatically at the end of the data block defined by the SPI_CR2 register exclusively.

When a mismatch is detected between the CRC calculated internally on the received data and the CRC received from the transmitter, a CRCE flag is set to indicate a data corruption error. The right procedure for handling the CRC depends on the SPI configuration and the chosen transfer management.

CRC transfer management

Communication starts and continues normally until the last data frame has been sent or received in the SPI_DR register.

The length of the transfer must be defined by TSIZE. When the desired number of data is transferred, the TXCRC is transmitted and the data received on the line are compared to the RXCRC value.

Whatever the CRCSIZE configuration, TSIZE cannot be set to 0xFFFF for a full-featured instance, and to 0x3FF for a limited-featured instance, if CRC is enabled.

In transmission, the CRC computation is frozen during the CRC transfer and the TXCRC is transmitted, in a frame of length equal to the CRCSIZE field value.

In reception, the RXCRC is also frozen when the desired number of data is transferred. Information to be compared with the RXCRC register content is then received in a frame of length equal to the CRCSIZE value.

Once the CRC frame is complete, an automatic check is performed comparing the received CRC value and the value calculated in the SPI_RXCRC register. The software has to check the CRCE flag of the SPI_SR register to determine if the data transfers were corrupted or not. Software clears the CRCE flag by writing 1 to the CRCEC.

The user takes no care about any flushing redundant CRC information, it is done automatically.

Resetting the SPI_TXCRC and SPI_RXCRC values

The SPI_TXCRC and SPI_RXCRC values are initialized automatically when new data is sampled after a CRC phase. This allows the use of DMA circular mode to transfer data without any interruption (several data blocks covered by intermediate CRC checking phases). Initialization patterns for receiver and transmitter can be configured either to zero or to all ones in dependency on setting bits TCRCINI and RCRCINI of the SPI_CR1 register.

The CRC values are reset when the SPI is disabled.

67.6 SPI in low-power modes

Table 660. Effect of low-power modes on the SPI

Mode	Description
Sleep	No effect. SPI interrupts cause the device to exit Sleep mode.
Stop(1)	The content of the SPI registers is kept.
Standby	The SPI instance is not functional in this mode. It is powered down, and must be reinitialized after exiting Standby mode.

1. Refer to *Section 67.3: SPI implementation* for information about wake-up from Stop mode support per instance as well as Standby mode availability. If an instance is not functional in a Stop mode, it must be disabled before entering this Stop mode.

67.7 SPI interrupts

Table 661 gives an overview of the SPI events capable of generating interrupts if enabled. Some of them feature wake-up from low-power mode capability, additionally. Most of them can be enabled and disabled independently while using specific interrupt enable control bits. The flags associated with the events are cleared by specific methods. Refer to the description of SPI registers for more details about the event flags. When SPI is disabled, all the pending interrupt requests are blocked to prevent their propagation into the interrupt services, except the MODF interrupt request.

Table 661. SPI wake-up and interrupt requests

Interrupt vector	Interrupt event	Event flag	Enable Control bit	Event clear method	Exit from Stop and Standby modes capability[1][2]
SPI	TxFIFO ready to be loaded (space available for one data packet - FIFO threshold)	TXP	TXPIE	TXP cleared by hardware when TxFIFO contains less than FTHLV empty locations	Yes
	Data received in RxFIFO (one data packet available - FIFO threshold)	RXP	RXPIE	RXP cleared by hardware when RxFIFO contains less than FTHLV samples	Yes
	Both TXP and RXP active	DXP	DXPIE	When TXP or RXP are cleared	Yes
	Transmission Transfer Filled	TXTF	TXTFIE	Writing TXTFC to 1	No
	Underrun	UDR	UDRIE	Writing UDRC to 1	Yes
	Overrun	OVR	OVRIE	Writing OVRC to 1	Yes
	CRC Error	CRCE	CRCEIE	Writing CRCEC to 1	Yes
	TI Frame Format Error	TIFRE	TIFREIE	Writing TIFREC to 1	No
	Mode Fault	MODF	MODFIE	Writing MODFC to 1	No
	End Of Transfer (full transfer sequence completed - based on TSIZE value)	EOT	EOTIE	Writing EOTC to 1	Yes
	Master mode suspended	SUSP		Writing SUSPC to 1	Yes
	TxFIFO transmission complete (TxFIFO empty)	TXC[3]		TXC cleared by hardware when a transmission activity starts on the bus	No

1. All the interrupt events can wake up the system from Sleep mode at each instance. For detailed information about instances capabilities to exit from concrete Stop and Standby mode refer to the *Functionalities depending on the working mode* table.
2. Refer to *Section 67.3: SPI implementation* for information about Standby mode availability.
3. The TXC flag behavior depends on the TSIZE setting. When TSIZE>0, the flag fully follows the EOT one including its clearing by EOTC.

67.8 I2S main features

- Full duplex communication
- Simplex communication (only transmitter or receiver)
- Master or slave operations
- 8-bit programmable linear prescaler
- Data length can be 16, 24 or 32 bits[a]
- Channel length can be 16 or 32 in master, any value in slave
- Programmable clock polarity
- Error flags signaling for improved reliability: Underrun, overrun, and frame errors
- Embedded Rx and TxFIFOs
- Supported I^2S protocols:
  - I^2S Philips standard
  - MSB-justified standard (left-justified)
  - LSB-justified standard (right-justified)
  - PCM standard (with short and long frame synchronization)
- Data ordering programmable (LSb or MSb first)
- DMA capability for transmission and reception
- Master clock can be output to drive an external audio component:
  - F_{MCK} = 256 x F_{WS} for all I2S modes
  - F_{MCK} = 128 x F_{WS} for all PCM modes

Note: F_{MCK} is the master clock frequency and F_{WS} is the audio sampling frequency.

67.9 I2S functional description

67.9.1 I2S general description

The block diagram shown on *Figure 880* also applies for I2S mode.

The SPI/I2S block can work in I2S/PCM mode, when the bit I2SMOD is set. A dedicated register (SPI_I2SCFGR) is available for configuring the dedicated I2S parameters, which include the clock generator, and the serial link interface.

The I2S/PCM function uses the clock generator to produce the communication clock when the SPI/I2S is set in master mode. This clock generator is also the source of the master clock output (MCK).

Resources such as RxFIFO, TxFIFO, DMA, and parts of interrupt signaling are shared with the SPI function. The low-power mode function is also available in I2S mode (refer to *Section 67.6: SPI in low-power modes* and *Section 67.10: I2S interrupts*).

a. Not always available, refer to *Section 67.3: SPI implementation* in order to check if 24 and 32-bit data widths are supported.

67.9.2 Pin sharing with SPI function

The I2S shares four common pins with the SPI:

- SDO: serial data output (mapped on the MOSI pin) to transmit the audio samples in master, and receive the audio sample in slave mode. Refer to *Section : Serial data line swapping on page 3470*.
- SDI: serial data input (mapped on the MISO pin) to receive the audio samples in master, and transmit the audio sample in slave mode. Refer to *Section : Serial data line swapping on page 3470*.
- WS: word select (mapped on the NSS pin). This is the frame synchronization. It is configured as output in master mode, and as input in slave mode.
- CK: serial clock (mapped on the SCK pin). This is the serial bit clock. It is configured as output in master mode, and as input in slave mode.

An additional pin can be used when a master clock output is needed for some external audio devices:

- MCK: master clock (mapped separately) is used when the I2S is configured in master mode.

67.9.3 Bitfields usable in I2S/PCM mode

When the I2S/PCM mode is selected (I2SMOD = '1'), some bitfields are no longer relevant, and must be forced to a specific value to guarantee the behavior of the I2S/PCM function. *Table 662* shows the list of bits and fields available in the I2S/PCM mode, and indicates which must be forced to a specific value.

Table 662. Bitfields usable in PCM/I2S mode

Register name	Bitfields usable in PCM/I2S Mode	Constraints on other bitfields
SPI/I2S control register 1 (SPI_CR1)	IOLOCK, CSUSP, CSTART, SPE	Other fields set to their reset values
SPI/I2S control register 2 (SPI_CR2)	-	Set to reset value
SPI/I2S configuration register 1 (SPI_CFG1)	TXDMAEN, RXDMAEN, FTHLV	Other fields set to their reset values
SPI/I2S configuration register 2 (SPI_CFG2)	AFCNTR, LSBFRST, IOSWP	Other fields set to their reset values
SPI/I2S interrupt enable register (SPI_IER)	TIFREIE, OVRIE, UDRIE, TXPIE, RXPIE	
SPI/I2S status register (SPI_SR)	SUSP, TIFRE, OVR, UDR, TXP, RXP	Other flags not relevant
SPI/I2S interrupt/status flags clear register (SPI_IFCR)	SUSPC, TIFREC, OVRC, UDRC	Other fields set to their reset values
SPI/I2S receive data register (SPI_RXDR)	The complete register	-

Table 662. Bitfields usable in PCM/I2S mode (continued)

Register name	Bitfields usable in PCM/I2S Mode	Constraints on other bitfields
SPI/I2S polynomial register (SPI_CRCPOLY)	-	Set to reset value
SPI/I2S transmitter CRC register (SPI_TXCRC)	-	
SPI/I2S receiver CRC register (SPI_RXCRC)	-	
SPI/I2S underrun data register (SPI_UDRDR)	-	
SPI/I2S configuration register (SPI_I2SCFGR)	The complete register	-

67.9.4 Slave and master modes

The SPI/I2S block supports master and slave modes for both I2S and PCM protocols. In master mode, both CK, WS and MCK signals are set to output.

In slave mode, both CK and WS signals are set to input. The signal MCK cannot be used in slave mode.

To improve the robustness of the SPI/I2S block in slave mode, the peripheral resynchronizes each reception and transmission on WS signal. This means that:

- In I2S Philips standard, the shift-in or shift-out of each data is triggered one bit clock after each transition of WS.
- In I2S MSB-justified standard, the shift-in or shift-out of each data is triggered as soon as a transition of WS is detected.
- In PCM short standard, the shift-in or shift-out of each data is triggered one bit clock after the active edge of WS.
- In PCM long standard, the shift-in or shift-out of each data is triggered as soon as the active edge of WS is detected

Note: This resynchronization mechanism is not available for the I2S LSB-justified standard.

Note as well that there is no need to provide a kernel clock when the SPI/I2S is configured in slave mode.

67.9.5 Supported audio protocols

The I2S/PCM interface supports four audio standards, configurable using the I2SSTD[1:0] and PCMSYNC bits of the SPI_I2SCFGR register.

In the I2S protocol, the audio data are time-multiplexed on two channels: the left channel and the right channel. The WS signal is used to indicate which channel must be considered as the left, and which one is the right.

In I2S master mode, four frame formats are supported:

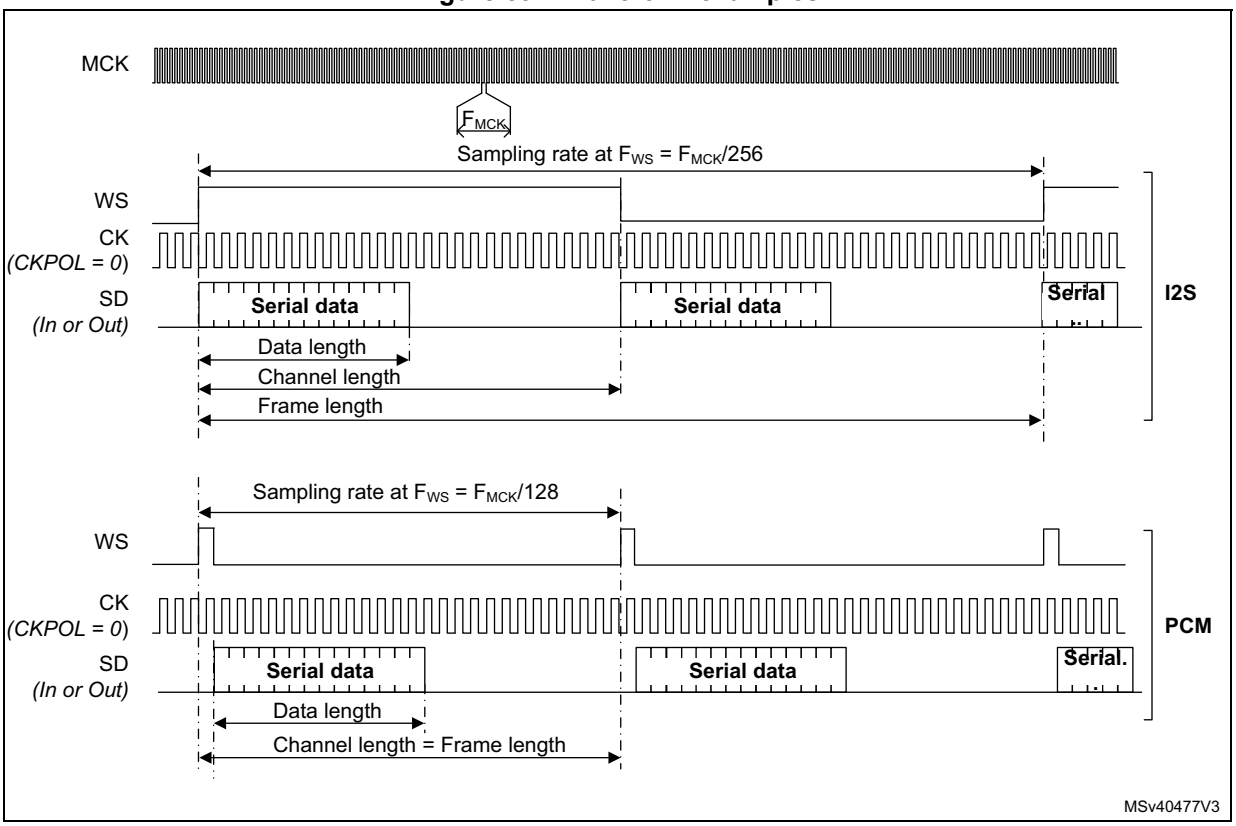
- 16-bit data packed in a 16-bit channel
- 16-bit data packed in a 32-bit channel
- 24-bit data packed in a 32-bit channel[(a)]
- 32-bit data packed in a 32-bit channel[(a)]

In PCM master mode, three frame formats are supported:

- 16-bit data packed in a 16-bit channel
- 16-bit data packed in a 32-bit channel
- 24-bit data packed in a 32-bit channel[(a)]

The figure hereafter shows the main definition used in this section: data length, channel length and frame length.

Figure 894. Waveform examples



For simplicity sake, in the next figures, SDI represents the serial data input and SDO the serial data output. Refer to *Section : Serial data line swapping* for details about the direction control of serial data lines.

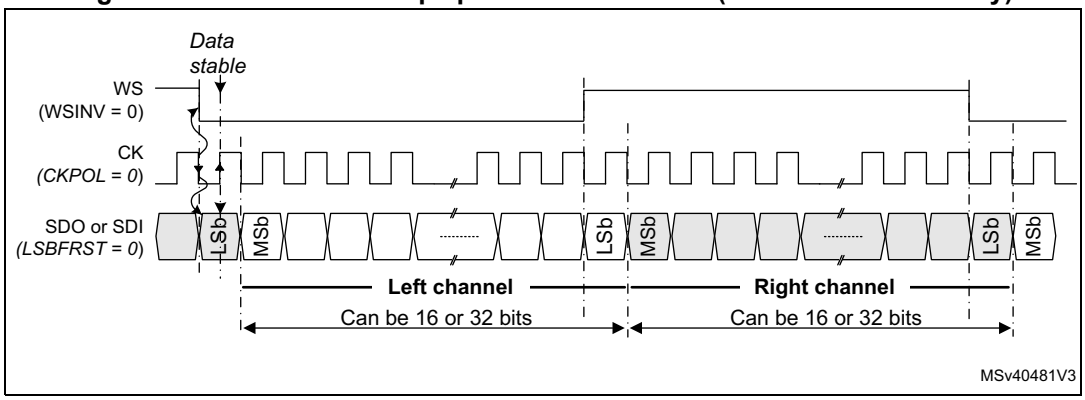
I²S Philips standard

The I2S Philips standard is selected by setting I2SSTD to 0b00. This standard is supported in master and slave mode.

In this standard, the WS signal toggles one CK clock cycle before the first bit (MSb in I2S Philips standard) is available. A falling edge transition of WS indicates that the next data transferred is the left channel, and a rising edge transition indicates that the next data transferred is the right channel.

a. Not always available, refer to *Section 67.3: SPI implementation* to check if 24 and 32-bit data widths are supported.

Figure 895. Master I2S Philips protocol waveforms (16/32-bit full accuracy)



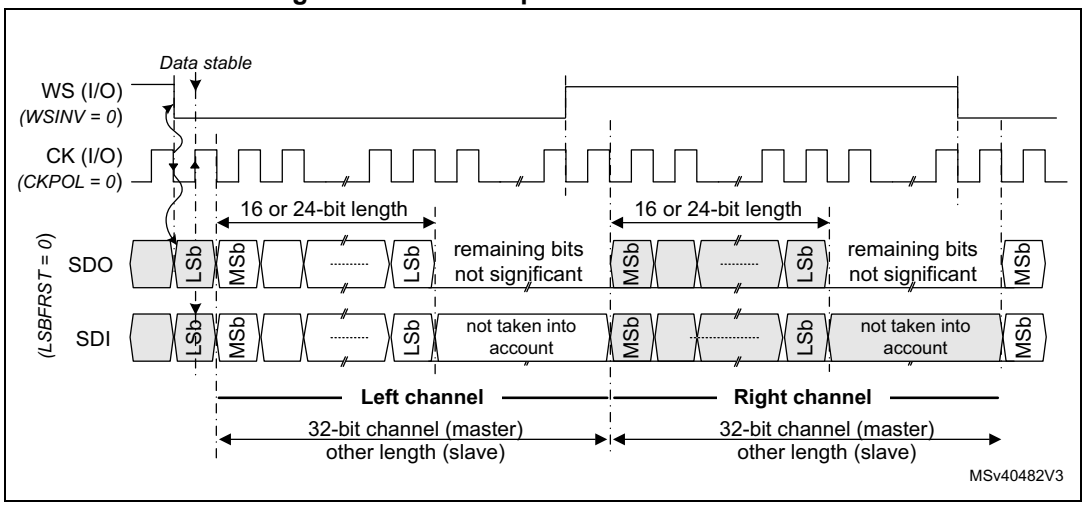
1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 67.3: SPI implementation* for the supported data sizes.

CKPOL is cleared to match the I2S Philips protocol. See *Selection of the CK sampling edge* for information concerning the handling of the WS signal.

Figure 895 shows an example of waveform generated by the SPI/I2S in the case where the channel length is equal to the data length. More precisely, this is true when CHLEN = 0 and DATLEN = 0b00 or when CHLEN = 1 and DATLEN = 0b10.

See *Control of the WS inversion* for information concerning the handling of the WS signal.

Figure 896. I2S Philips standard waveforms



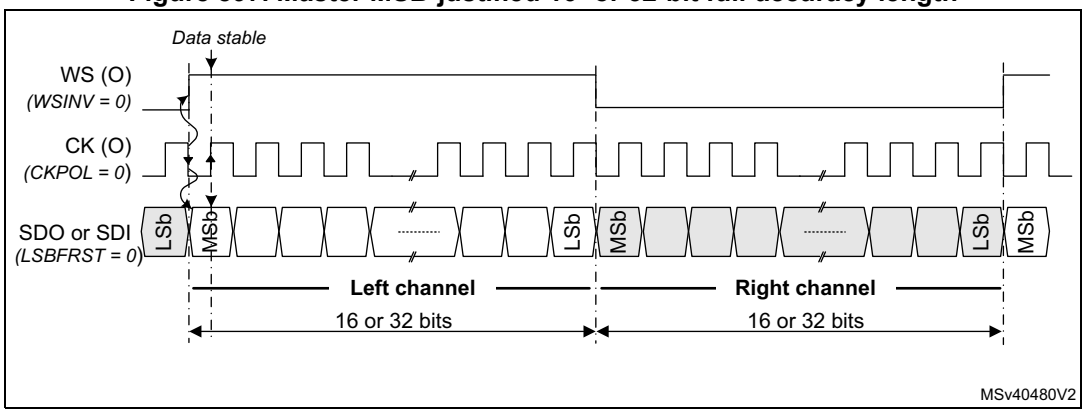
1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 67.3: SPI implementation* for the supported data sizes.

In the case where the channel length is bigger than the data length, the remaining bits are not significant when the SPI/I2S is configured in transmit mode. This is applicable for both master and slave mode.

MSB-justified standard

For this standard, the WS signal toggles when the first data bit is provided. The data transferred represents the left channel if WS is high and the right channel if WS is low.

Figure 897. Master MSB-justified 16- or 32-bit full-accuracy length

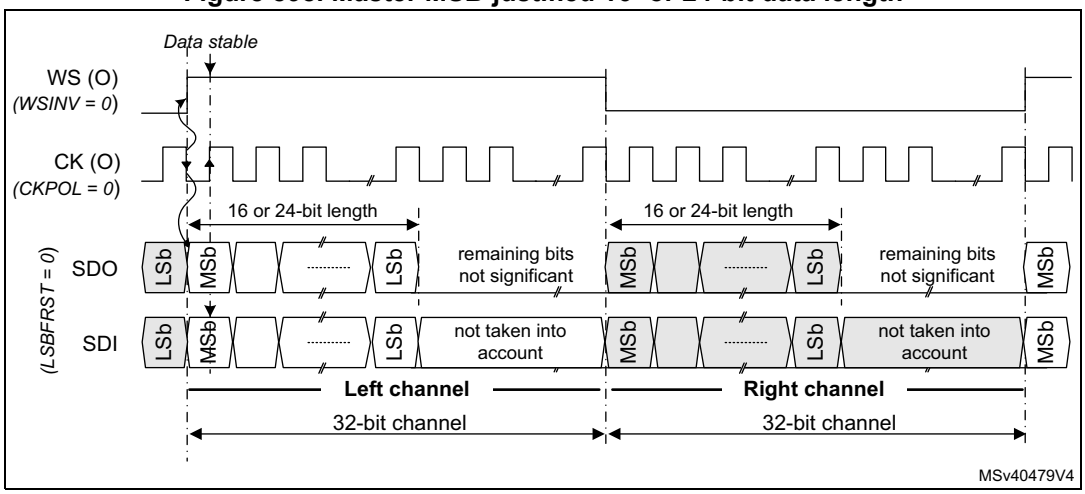


1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 67.3: SPI implementation* to check the supported data size.

CKPOL is cleared to match the I2S MSB-justified protocol. See *Selection of the CK sampling edge* for information concerning the handling of the WS signal.

See *Control of the WS inversion* for information concerning the handling of the WS signal.

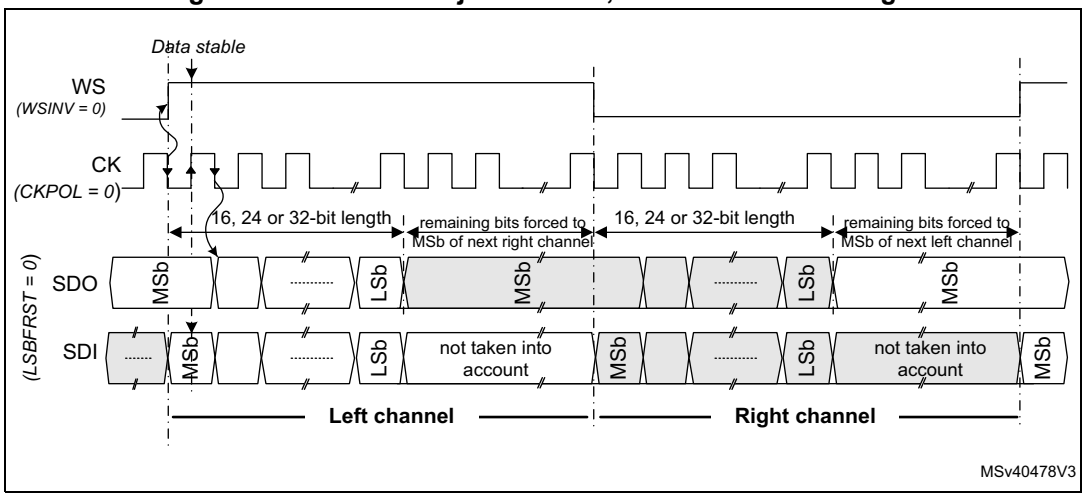
Figure 898. Master MSB-justified 16- or 24-bit data length



1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 67.3: SPI implementation* to check the supported data size.

In the case where the channel length is bigger than the data length, the remaining bits are not significant when the SPI/I2S is configured in master transmit mode. In slave transmit mode, the remaining bits are forced to the value of the first bit of the next data to be generated to avoid timing issues (see *Figure 899*).

Figure 899. Slave MSB-justified 16-, 24- or 32-bit data length



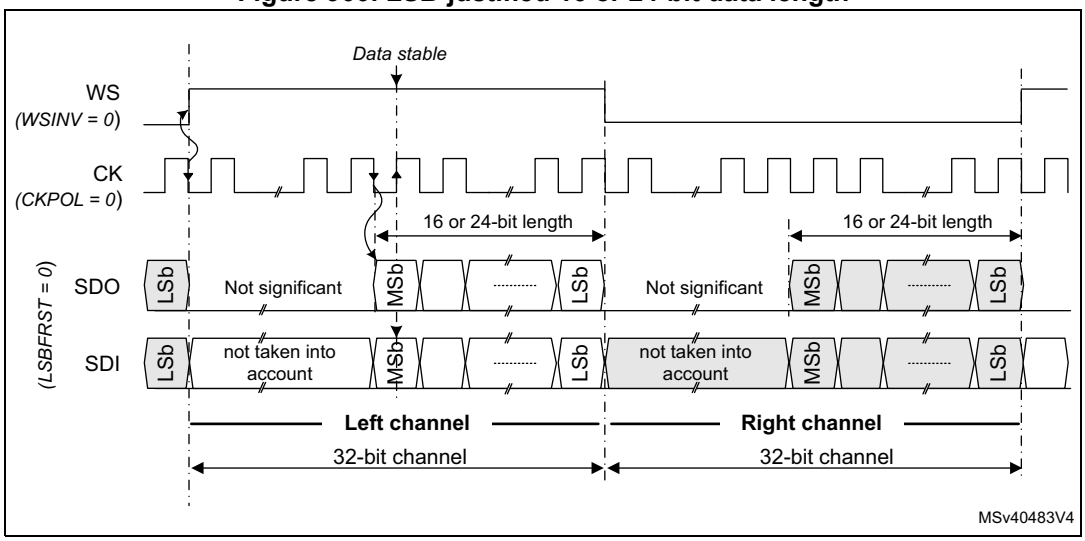
1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 67.3: SPI implementation* to check the supported data size.

LSB-justified standard

This standard is similar to the MSB-justified standard in master mode (no difference for the 16- and 32-bit full-accuracy frame formats). The LSB-justified 16- or 32-bit full-accuracy format gives similar waveforms to MSB-justified mode (see *Figure 897*) because the channel and data have the same length.

Note: In the LSB-justified format, only 16- and 32-bit channel lengths are supported in master and slave mode. This is because it is not possible to transfer properly the data if the channel length is not known by the transmitter and receiver side.

Figure 900. LSB-justified 16 or 24-bit data length



1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 67.3: SPI implementation* to check the supported data size.

CKPOL is cleared in order to match the I2S LSB-justified protocol. See *Selection of the CK sampling edge* for information concerning the handling of the WS signal.

See *Control of the WS inversion* for information concerning the handling of the WS signal.

PCM standard

For the PCM standard, there is no need to use channel-side information. The two PCM modes (short and long frame) are available and can be selected using the PCMSYNC bit of SPI_I2SCFGR register.

In PCM long frame:

- The assertion time of the WS signal is fixed to 13 cycles of CK in master mode,
- The first data bit is received or transmitted as soon as the WS signal is asserted.

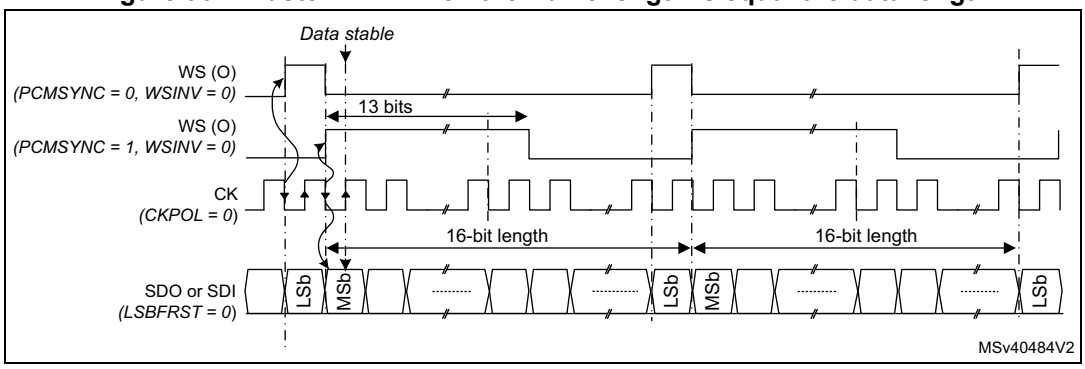
In PCM short frame:

- The assertion time of the WS signal is fixed to one cycle of CK in master mode,
- The first data bit is received or transmitted one cycle of CK after the WS assertion.

For both PCM modes:

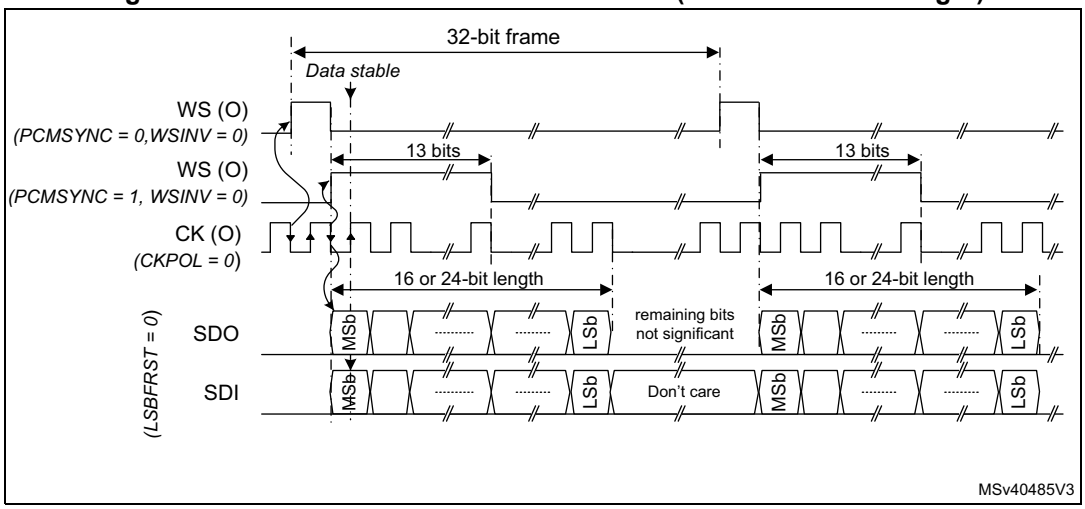
- The first data bit is MSb or LSb depending on the LSBFIRST bit value.
- The CK sampling edge can be selected thanks to CKPOL bit.
- The WS signal can be inverted thanks to the WSINV bit. See *Control of the WS inversion* for information concerning the handling of WS signal.

Figure 901. Master PCM when the frame length is equal the data length



A data size of 16 or 24 bits can be used when the channel length is set to 32 bits.

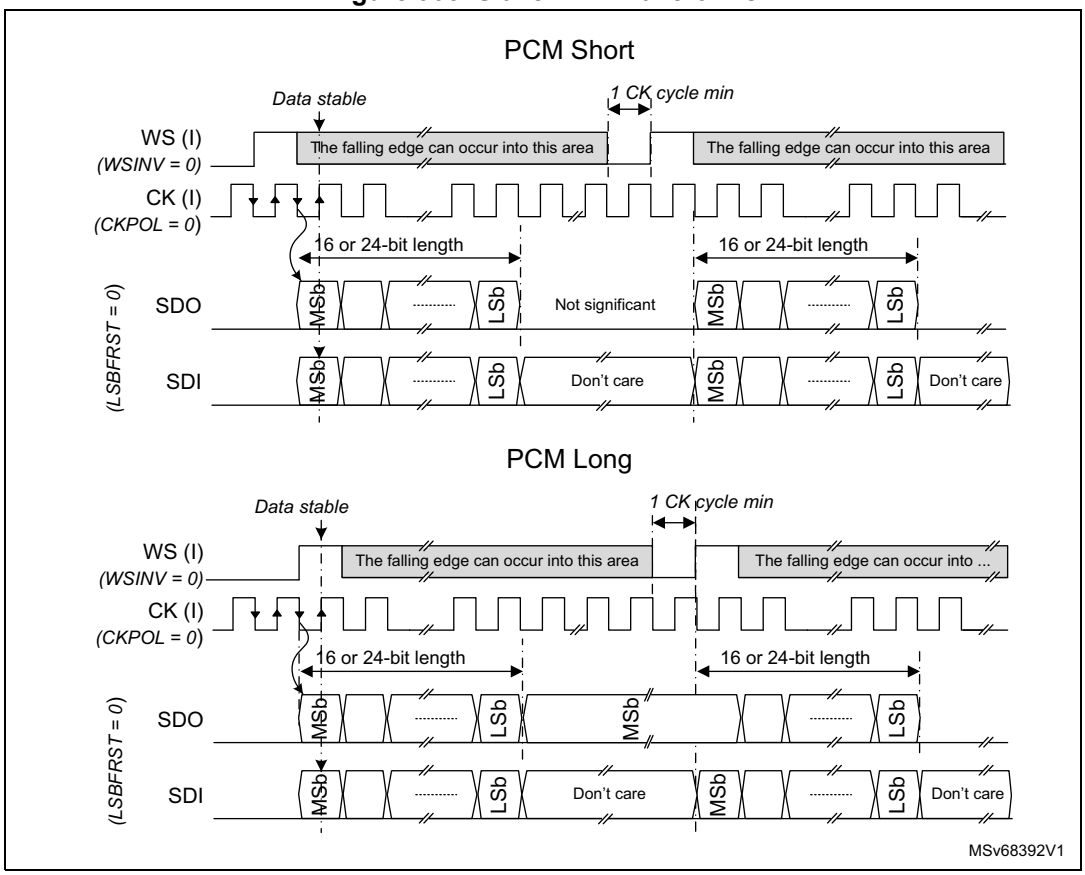
Figure 902. Master PCM standard waveforms (16 or 24-bit data length)



1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 67.3: SPI implementation* to check the supported data size.

If the PCM protocol is used in slave mode, frame lengths can be different from 16 or 32 bits. As shown in *Figure 903*, in slave mode various pulse widths of WS can be accepted as the start of frame is detected by a rising edge of WS. The only constraint is that the WS must go back to its inactive state for at least one CK cycle.

Figure 903. Slave PCM waveforms



1. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 67.3: SPI implementation* to check the supported data size.

Note: In the case where the channel length is bigger than the data length, in slave PCM long, the transmission of the remaining bits is forced to the value of the first bit of the next data to be generated if the TXFIFO contains the next data to be transmitted.

In slave mode, CHLEN must be always programmed properly (whatever FIXCH value). For example, if CHLEN is cleared (16-bit length), the data transfer is truncated to 16 bits even if DATLEN is different from 0. To avoid this situation, CHLEN must be set, if DATLEN = 1 or 2.

67.9.6 Additional serial interface flexibility

Variable frame length in slave

In slave mode, channel lengths different from 16 or 32 bits can be accepted, as long as the channel length is bigger than the data length. This is true for all protocols except for the I2S LSB-justified protocol.

Data ordering

For all data formats and communication standards, it is possible to select the data ordering (MSb or LSb first) thanks to the bit LSBFRST located into *SPI/I2S configuration register 2 (SPI_CFG2)*.

Selection of the CK sampling edge

The CKPOL bit located into *SPI/I2S configuration register (SPI_I2SCFGR)* allows the user to choose the sampling edge polarity of the CK for slave and master modes, for all protocols.

- When CKPOL = 0, serial data SDO and WS (when master) are changed on the falling edge of CK and the serial data SDI and WS (when slave) are read on the rising edge.
- When CKPOL = 1, serial data SDO and WS (when master) are changed on the rising edge of CK and the serial data SDI and WS (when slave) are read on the falling edge.

Control of the WS inversion

It is possible to invert the default WS signal polarity for master and slave modes, for all protocols, by setting WSINV. By default the WS polarity is the following:

- In I2S Philips standard, WS is LOW for left channel, and HIGH for right channel
- In MSB/LSB-justified mode, WS is HIGH for left channel, and LOW for right channel
- In PCM mode, the start of frame is indicated by a rising edge of WS.

When WSINV is set, the WS polarity is inverted, then:

- In I2S Philips standard, WS is HIGH for left channel, and LOW for right channel
- In MSB/LSB-justified mode, WS is LOW for left channel, and HIGH for right channel
- In PCM mode, the start of frame is indicated by a falling edge of WS.

WSINV is located in the *SPI/I2S configuration register (SPI_I2SCFGR)*.

Control of the I/Os

The SPI/I2S block allows the settling of the WS and CK signals to their inactive state before enabling the SPI/I2S thanks to the AFCNTR bit of *SPI/I2S configuration register 2 (SPI_CFG2)*.

This can be done by programming CKPOL and WSINV using the following sequence:

Assuming that AFCNTR is initially cleared:

- Set I2SMOD = 1, (to inform the hardware that the CK and WS polarity is controlled via CKPOL and WSINV).
- Set bits CKPOL and WSINV to the wanted value.
- Set AFCNTR = 1.
  Then the inactive level of CK and WS I/Os is set according to CKPOL and WSINV values, even if the SPI/I2S is not yet enabled.
- Then performs the activation sequence of the I2S/PCM

Table 663 shows the level of WS and CK signals, when the AFCNTR bit is set, and before the SPI/I2S block is enabled (that is inactive level). Note that the level of WS also depends on the protocol selected.

Table 663. WS and CK level before SPI/I2S is enabled when AFCNTR = 1

WSINV	I2SSTD		WS level before SPI/I2S is enabled
0	I2S Std (00)	→	High
	Others	→	Low
1	I2S Std (00)	→	Low
	Others	→	High

CKPOL		CK level before SPI/I2S is enabled
0	→	Low
1	→	High

Note: The bit AFCNTR must not be set when the SPI is in slave mode.

Serial data line swapping

The direction of SDI and SDO depends on the IOSWP bit of *SPI/I2S configuration register 2 (SPI_CFG2)*, and on the slave/master mode. *Table 664* gives details on this feature.

Table 664. Serial data line swapping

Direction	IOSWP	Master mode		Slave mode	
		Input line	Output line	Input line	Output line
RX	0	SDI	-	SDO	-
	1	SDO	-	SDI	-
TX	0	-	SDO	-	SDI
	1	-	SDI	-	SDO
Full-duplex	0	SDI	SDO	SDO	SDI
	1	SDO	SDI	SDI	SDO

67.9.7 Startup sequence

When the bit SPE is cleared, the user is not allowed to read and write from/to the SPI_RXDR and SPI_TXDR registers, but the access to other registers is allowed.

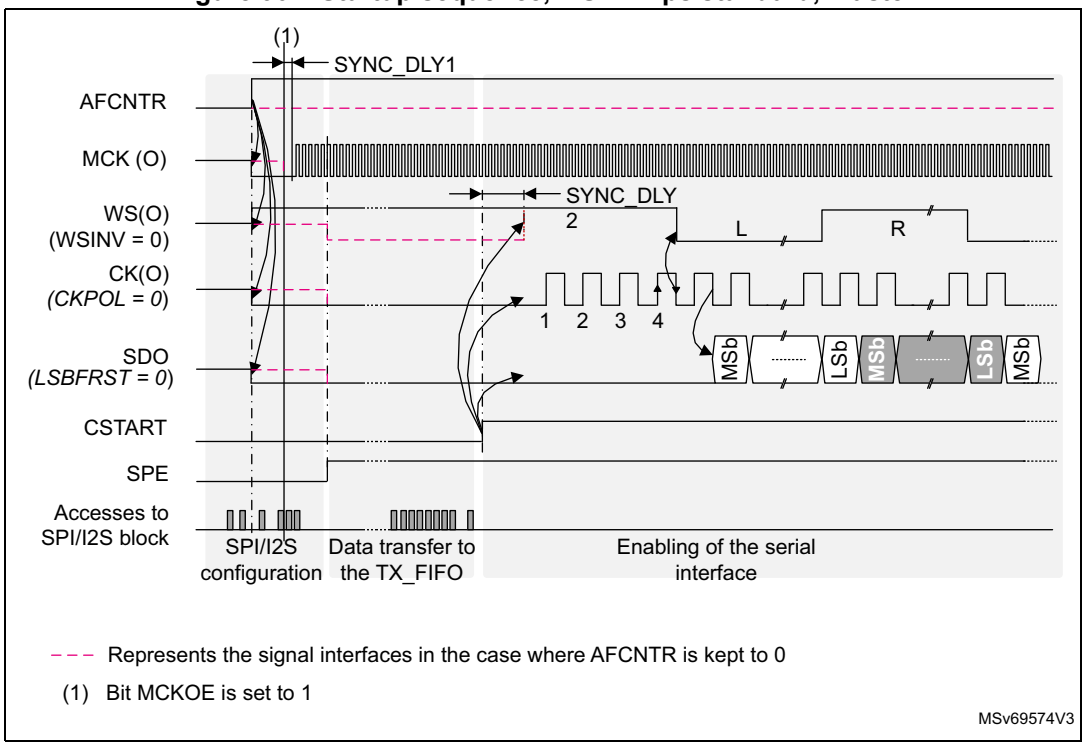
When the application wants to use the SPI/I2S block the user has to proceed as follows:

1. Ensure that the SPE is cleared, otherwise write SPE to 0.
2. Program all the configuration and control registers according to the wanted configuration. Refer to *Section 67.9.16* for detailed programming examples.
3. Clear all the status flags by setting the USPC, TIFREC, OVRC, and UDRC bits of the SPI_IFCR register. Note that if the flag SUSP is not cleared (via the SUSPC bit) the CSTART control bit has no effect.
4. Set the SPE bit to activate the SPI/I2S block. When this bit is set, the serial interface is still disabled, but the DMA and interrupt services are working, allowing for example, the data transfer into the TxFIFO.
5. Set bit CSTART to activate the serial interface.

As shown in *Figure 904*, in I2S Philips standard master TX, the generation of the WS and CK signals starts after a resynchronization delay (SYNC_DLY2) when CSTART goes to 1 and the TxFIFO is not empty. Note that the CK bit clock is activated four rising edges before

the falling edge of WS in order to ensure that the external slave device can detect properly WS transition. Other standards behave similarly.

Figure 904. Startup sequence, I2S Philips standard, master



2. When MCKOE is set, the MCK is generated after a synchronization delay (SYNC_DLY1). For more information, see MCK generation in *Section 67.9.9: Clock generator*.
3. Note that the level of WS and CK signals are controlled by the SPI/I2S block during the configuration phase as soon as the AFCNTR bit is set.

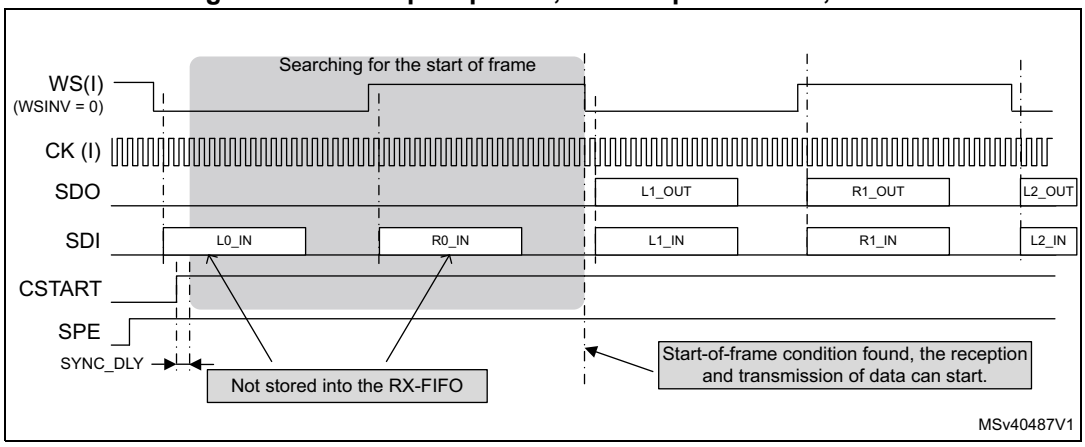
Note: Due to clock domain resynchronization, the CSTART bit is taken into account by the hardware after about three periods of CK clock (SYNC_DLY2).

In slave mode, once the bit CSTART is set, the data transfer starts when the start-of-frame condition is met:

- For the I2S Philips standard, the start-of-frame condition is a falling edge of the WS signal. The transmission/reception starts one bit clock later.
  If WSINV = 1, then the start-of-frame condition is a rising edge.
- For other protocols, the start-of-frame condition is a rising edge of the WS signal. The transmission/reception starts at the rising edge of WS for MSB-aligned protocol. The transmission/reception starts one bit clock later for PCM protocol.
  If WSINV = 1, then the start-of-frame condition is a falling edge.

Figure 905 shows an example of startup sequence in I2S Philips standard, slave mode.

Figure 905. Startup sequence, I2S Philips standard, slave



Note: Due to clock domain resynchronization, the CSTART bit is taken into account by the hardware after two periods of CK clock (SYNC_DLY).

67.9.8 Stop sequence

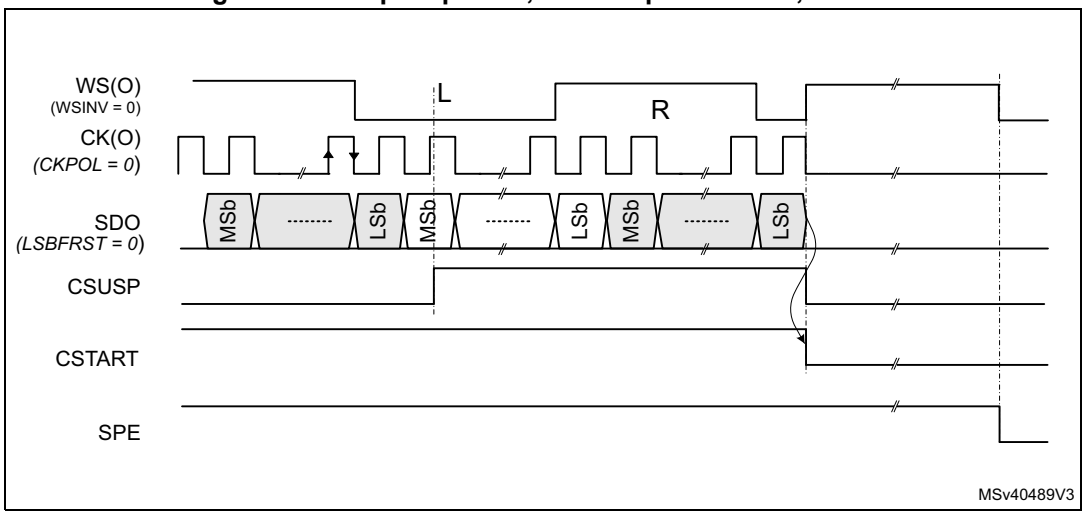
The application can stop the I2S/PCM transfers by clearing the SPE bit. In that case the communication is stopped immediately, without waiting for the end of the current frame.

In master mode it is also possible to stop the I2S/PCM transfers at the end of the current frame. For that purpose, the user has to set the bit CSUSP, and polls the CSTART bit until it goes to 0. The CSTART bit goes to 0 when the current stereo (if an I2S mode was selected) or mono sample are completely shifted in or out. Then the SPE bit can be cleared.

Figure 906 shows an example of stop sequence in the case of master mode. The CSUSP bit is set during the transmission of the left sample, the transfer continue until the last bit of the right sample is transferred. Then CSTART and CSUSP go back to 0, CK and WS signals go back to their inactive state, and the user can clear the SPE bit.

In slave mode, CSUSP must be kept at 0.

Figure 906. Stop sequence, I2S Philips standard, master

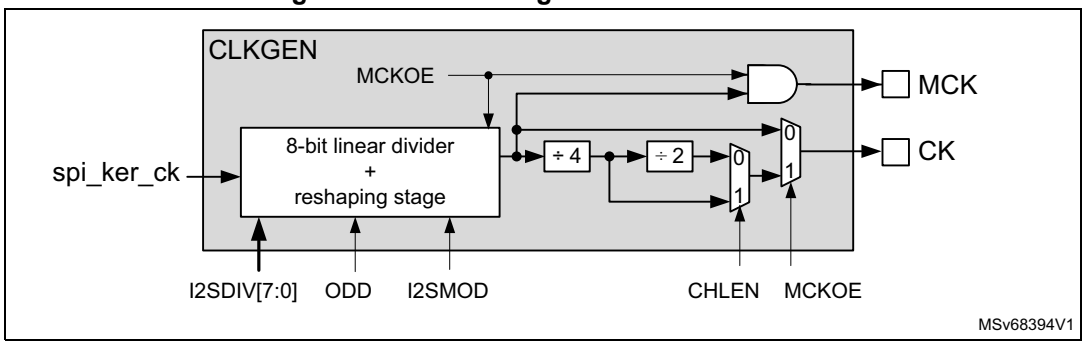


Note: In slave mode, the stop sequence is only controlled by the SPE bit.

67.9.9 Clock generator

When the I2S or PCM is configured in master mode, the user needs to program the clock generator in order to produce the frame synchronization (WS), the bit clock (CK) and the master clock (MCK) at the desired frequency.
If the I2S or PCM is used in slave mode, there is no need to configure the clock generator.

Figure 907. I²S clock generator architecture



The frequency generated on MCK, CK and WS depends mainly on I2SDIV, ODD, CHLEN, and MCKOE. The bit MCKOE indicates if a master clock need to be generated or not. The master clock has a frequency 256 or 128 times higher than the frame synchronization. This master clock is often required to provide a reference clock to external audio codecs.

Note: In master mode, there are no specific constraints on the ratio between the bus clock rate (F_{pclk}) and the bit clock (F_{CK}). The bus clock frequency must be high enough in order to support the data throughput.

When the master clock is generated (MCKOE = 1), the frequency of the frame synchronization is given by the following formula in I2S mode:

$$F_{WS} = \frac{F_{i2s_clk}}{256 \times \{(2 \times I2SDIV) + ODD\}}$$

And by this formula in PCM mode:

$$F_{WS} = \frac{F_{i2s_clk}}{128 \times \{(2 \times I2SDIV) + ODD\}}$$

In addition, the frequency of the MCK (F_{MCK}) is given by the formula:

$$F_{MCK} = \frac{F_{i2s_clk}}{\{(2 \times I2SDIV) + ODD\}}$$

When the master clock is disabled (MCKOE = 0), the frequency of the frame synchronization is given by the following formula in I2S mode:

$$F_{WS} = \frac{F_{i2s_clk}}{32 \times (CHLEN + 1) \times \{(2 \times I2SDIV) + ODD\}}$$

And by this formula in PCM mode:

$$F_{WS} = \frac{F_{i2s_clk}}{16 \times (CHLEN + 1) \times \{(2 \times I2SDIV) + ODD\}}$$

Where F_{WS} is the frequency of the frame synchronization, and F_{i2s_clk} is the frequency of the kernel clock provided to the SPI/I2S block.

Note: CHLEN and ODD can be either 0 or 1.

When I2SDIV = 0, then {(2 x I2SDIV) + ODD} is forced to 1.

Note: MCK duty cycle is 50 %, even for odd division ratios, when the kernel clock duty cycle is 50 %. When the division ratio is set (I2SDIV = 0), the MCK duty cycle is the same as the duty cycle of the kernel clock.

Table 665 provides examples of clock generator programming for I2S modes.

MCK generation

The master clock MCK is generated when the following conditions are met:

- I2SMOD must be equal to 1,
- I2SCFG must select a master mode,
- MCKOE must be set,

Table 665. CLKGEN programming examples for usual I2S frequencies

i2s_clk (MHz)	Channel length (bits)	I2SDIV	ODD	MCK	Sampling rate: F_{WS} (kHz)
12.288	16	12	0	No	16
12.288	32	6	0		16
12.288	16	6	0		32
12.288	32	3	0		32
49.152	16	16	0		48
49.152	32	8	0		48
49.152	16	8	0		96
49.152	32	4	0		96
49.152	16	4	0		192
49.152	32	2	0		192

Table 665. CLKGEN programming examples for usual I2S frequencies (continued)

i2s_clk (MHz)	Channel length (bits)	I2SDIV	ODD	MCK	Sampling rate: F_{WS} (kHz)
4.096	16 or 32	0	-	Yes	16
24.576	16 or 32	3	0		
49.152	16 or 32	3	0		32
12.288	16 or 32	0	-		48
49.152	16 or 32	2	0		
61.44	16 or 32	2	1		
98.304	16 or 32	2	0		96
196.608	16 or 32	2	0		192

67.9.10 Internal FIFOs

The I2S interface can use a dedicated FIFO for the RX and the TX path. The samples to transmit can be written into the TxFIFO via the SPI_TXDR register. The reading of RxFIFO is performed via the SPI_RXDR register.

Data alignment and ordering

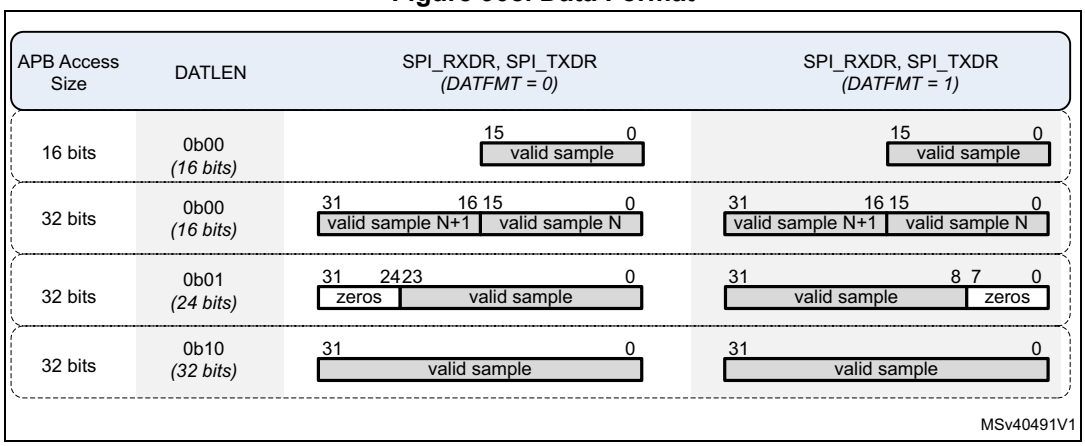
It is possible to select the data alignment into the SPI_RXDR and SPI_TXDR registers thanks to the DATFMT bit.

Note as well that the format of the data located into the SPI_RXDR or SPI_TXDR depends as well on the way those registers are accessed via the APB bus.

Figure 908 shows the allowed settings between APB access sizes, DATFMT, and DATLEN.

Note: Caution must be taken when the APB access size is 32 bits, and DATLEN = 0. For read operation the RxFIFO must contain at least two data, otherwise the read data are invalid. In the same way, for write operation, the TxFIFO must have at least two empty locations, otherwise a data can be lost.

Figure 908. Data Format



1. In I2S mode, the sample N represents the left sample, and the sample N+1 is the right sample.
2. Data width of 24 and 32 bits are not always supported (DATLEN = 01 or 10). Refer to *Section 67.3: SPI implementation* to check the supported data size.

It is possible to generate an interrupt or a DMA request according to a programmable FIFO threshold levels. The FIFO threshold is common to RX and TxFIFOs can be adjusted via FTHLV.

In I2S mode, the left and right audio samples are interleaved into the FIFOs. It means that for transmit operations, the user has to start to fill-up the TxFIFO with a left sample, followed by a right sample, and so on. For receive mode, the first data read from the RxFIFO is supposed to represent a left channel, the next one is a right channel, and so on.

Note that the read and write pointers of the FIFOs are reset when the bit SPE is cleared.

Refer to *Section 67.9.11* and *Section 67.9.15* for additional information.

FIFO size optimization

The basic element of the FIFO is the byte. This allows an optimization of the FIFO locations. For example when the data size is fixed to 24 bits, each audio sample takes three basic FIFO elements.

For example, a FIFO with 16 basic elements can have a depth of:

- 8 samples, if the DATLEN = 0 (16 bits),
- 5 samples, if the DATLEN = 1 (24 bits)[a],
- 4 samples, if the DATLEN = 2 (32 bits)[a].

67.9.11 FIFO status flags

Two status flags are provided for the application to fully monitor the state of the I2S interface. Both flags can generate an interrupt request. The receive interrupt is generated if RXPIE bit is enabled, the transmit interrupt is generated if TXPIE bit is enabled. Those bits are located into the SPI_IER register.

TxFIFO threshold reached (TXP)

When set, this flag indicates that the TxFIFO contains at least FTHLV empty locations. Thus, FTHLV new data to be transmitted can be written to SPI_TXDR. The TXP flag is reset when the number of empty locations is lower than FTHLV. Note that TXP = 1, when the I2S is disabled (SPE bit is reset).

RxFIFO threshold reached (RXP)

When set, this flag indicates that there is at least FTHLV valid data into the RxFIFO, thus the user can read those data via SPI_RXDR. It is reset when the RxFIFO contains less than FTHLV data.

See *Section 67.10* for additional information on the interrupt function in I2S mode.

67.9.12 Handling of underrun situation

In transmit mode, an underrun situation is detected when a new data needs to be loaded into the shift register while the TxFIFO is already empty. In such a situation the UDR flag is set, and at least one audio frame contains unexpected data.

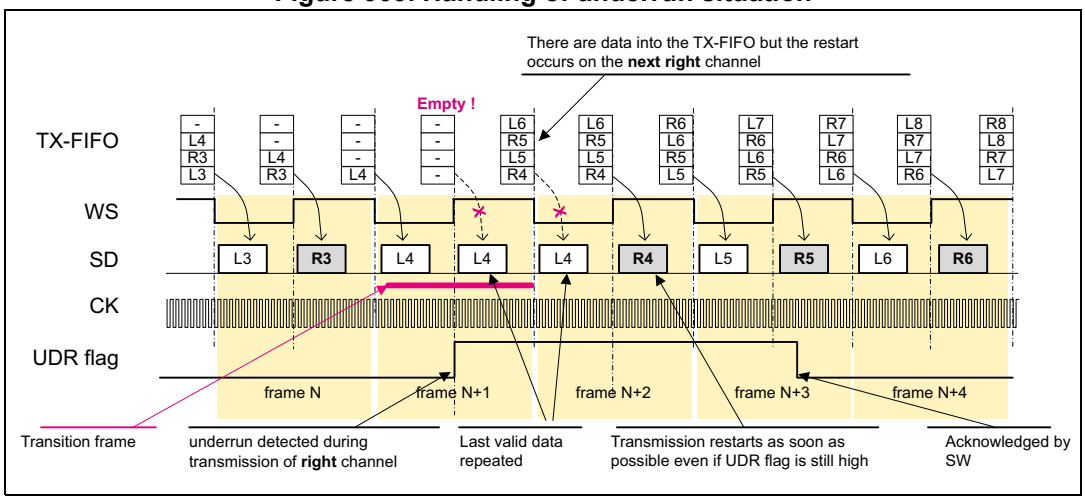
a. Not always available, refer to *Section 67.3: SPI implementation* in order to check if 24 and 32-bit data widths are supported.

For I2S modes, there is a hardware mechanism to prevent misalignment situations (left and right channel swapped). When an underrun situation is detected, the last valid data present in the shift register is repeated if the real channel length matches the length selected by the CHLEN bit. In the other cases, an undefined data is repeated. Typically, if the block is programmed in slave TX mode, and the external master audio device is using channel lengths other than 16 or 32 bits, the repeated data value is undefined in case of underrun.

The following figure shows the case where an underrun occurs and the peripheral replays the last valid data on left and right channels as long as conditions of restart are not met. The transmission restarts:

- When there is enough data into the TxFIFO, and
- When the restart condition is met:
  - If the underrun occurs when a right channel data needs to be transmitted, the transmission restarts when a right channel needs to be transmitted, or
  - If the underrun occurs when a left channel data needs to be transmitted, the transmission restarts when a left channel needs to be transmitted.

Figure 909. Handling of underrun situation



When the block is configured in one of the PCM modes, the transmission restarts at the start of the next frame, when there is enough data in the TxFIFO.

The UDR flag can trigger an interrupt if the UDRIE bit of the SPI_IER register is set. The UDR bit is cleared by setting the UDRC bit of the SPI_IFCR register.

Note: An underrun situation can occur in master or slave mode. In master mode, when an underrun occurs, the WS, CK, and MCK signals are not gated.

Due to resynchronization, any change of the UDR flag is taken into account by the hardware after at least two periods of CK clock.

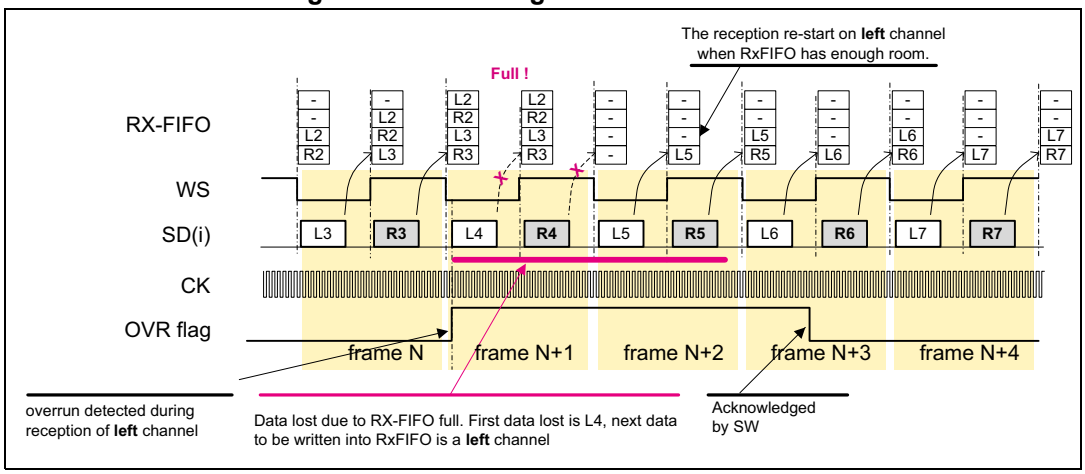
67.9.13 Handling of overrun situation

In receive mode, an overrun situation is detected when the shift register needs to store a new data into the RxFIFO, while the RxFIFO is full. In such a situation the OVR flag is set, and the incoming data is lost.

In I2S mode, there is a hardware mechanism in to prevent misalignment situations (left and right channel swapped). As shown in the following figure, when an overrun occurs, the peripheral stops writing data into the RxFIFO as long as restart conditions are not met.

The reception restarts when there is enough room into the RxFIFO. The block starts by writing next the right channel into the RxFIFO if the overrun occurs when a right channel data is received or by writing the next left channel if the overrun occurs when a left channel data is received.

Figure 910. Handling of overrun situation



When the block is configured in PCM mode, after an overrun error, the block stops writing data in the RxFIFO as long as conditions of restart are not met. When there is enough room in the RxFIFO, the next received data are written in the RxFIFO.

An interrupt can be generated if the OVRIE bit of the SPI_IER register is set. The OVR bit is cleared by setting the OVRC bit of the SPI_IFCR register.

Note: An overrun situation can occur in master or slave mode. In master mode when an overrun occurs, the WS, CK, and MCK signals are not gated.

67.9.14 Frame error detection

When configured in slave mode, the SPI/I2S block detects two kinds of frame errors:

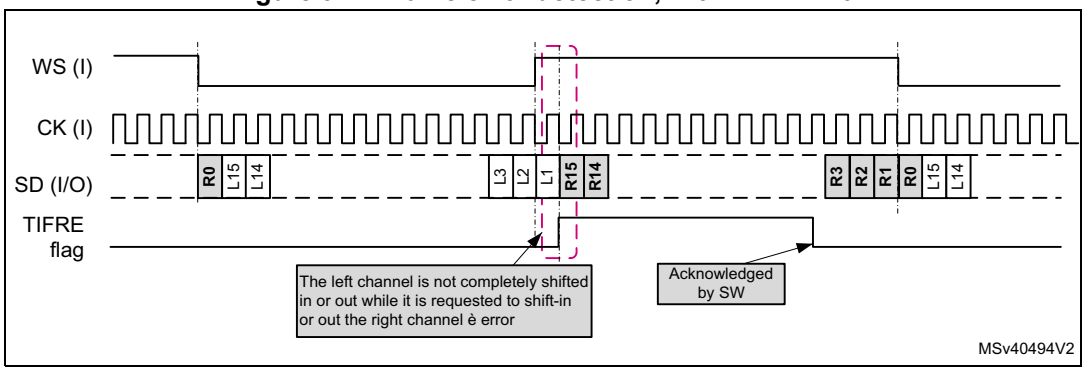
- A frame synchronization is received while the shift-in or shift-out of the previous data is not complete (early frame error). This mode is selected with FIXCH = 0.
- A frame synchronization occurs at an unexpected position. This mode is selected with FIXCH = 1.

In slave mode, if the frame length provided by the external master device is different from 32 or 64 bits, the user has to clear FIXCH. As the SPI/I2S synchronize each transfer with the WS there is no misalignment risk, but in a noisy environment, if a glitch occurs in the CK signal, a sample may be affected and the application is not aware of this.

If the frame length provided by the external master device is equal to 32 or 64 bits, then the user can set FIXCH and adjust accordingly CHLEN. As the SPI/I2S synchronize each transfer with the WS there is still no misalignment risk, and if the amount of bit clock between each channel boundary is different from CHLEN, the frame error flag (TIFRE) is set.

Figure 911 shows an example of frame error detection. The SPI/I2S block is in slave mode and the number of bit clock periods for the left channel are not enough to shift-in or shift-out the data. The figure shows that the ongoing transfer is interrupted and the next one is started in order to remain aligned to the WS signal.

Figure 911. Frame error detection, with FIXCH = 0

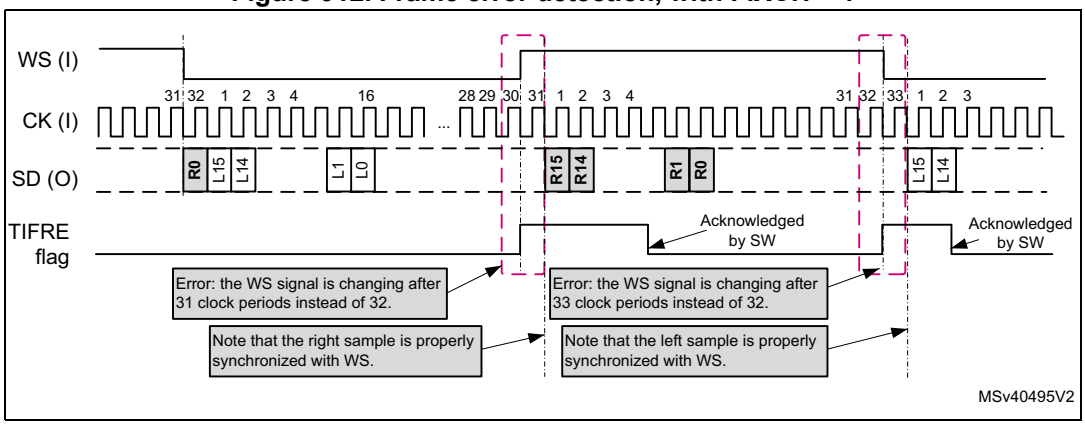


An interrupt can be generated if the TIFREIE bit is set. The frame error flag (TIFRE) is cleared by writing the TIFREC bit of the SPI_IFCR register to 1.

It is possible to extend the coverage of the frame error flag by setting the bit FIXCH. When this bit is set, then the SPI/I2S is expecting fixed channel lengths in slave mode. This means that the expected channel length can be 16 or 32 bits, according to CHLEN. As shown in *Figure 912*, in this mode, the SPI/I2S block is able to detect if the WS signal is changing at the expected moment (too early or too late).

Note: *Figure 911* and *Figure 912* show the mechanism for the slave transmit mode, but this is also true for slave receive and slave full-duplex.

Figure 912. Frame error detection, with FIXCH = 1



The frame error detection can be generally due to noisy environment disturbing the good reception of WS or CK signals.

Note: *The SPI/I2S is not able to recover properly if an overrun and an early frame occur within the same frame. In this case the user has to disable and re-enable the SPI/I2S.*

67.9.15 DMA interface

The I2S/PCM mode shares the same DMA requests lines than the SPI function. There is a separated DMA channel for TX and RX paths. Each DMA channel can be enabled via RXDMAEN and TXDMAEN bits of the SPI_CFG1 register.

In receive mode, the DMA interface is working as follows:

1. The hardware evaluates the RxFIFO level,
2. If the RxFIFO contains at least FTHLV samples, then FTHLV DMA requests are generated,
   – When the FTHLV DMA requests are completed, the hardware loops to step 1
3. If the RxFIFO contains less than FTHLV samples, no DMA request is generated, and the hardware loop to step 1

In transmit mode, the DMA interface is working as follows:

1. The hardware evaluates the TxFIFO level,
2. If the TxFIFO contains at least FTHLV empty locations, then FTHLV DMA requests are generated,
   – When the FTHLV DMA requests are completed, the hardware loops to step 1
3. If the TxFIFO contains less than FTHLV empty locations, no DMA request is generated, and the hardware loop to step 1

67.9.16 Programing examples

Master I2S Philips standard, full-duplex

This example shows how to program the interface for supporting the I2S Philips standard protocol in master full-duplex mode, with a sampling rate of 48 kHz, using the master clock. The assumption has been taken that the SPI/I2S is receiving a kernel clock (i2s_clk) of 61.44 MHz from the clock controller of the circuit. In the example above we took the assumption that the external audio codec needs to be programmed, for example via an I2C interface before starting the transfer. In addition, it is supposed that this external audio codec needs the MCK to accept I2C commands.

Procedure

1. Via the RCC block, enable the bus interface and the kernel clocks, assert and release the reset signal if needed.
2. Program the AFMUX in order to select the wanted I/Os. In the current example MCK, CK, WS, SDO, SDI are needed.
3. Program the clock generator to provide the MCK clock, and to have a frame synchronization rate at exactly 48 kHz. Set I2SDIV to 2, ODD to 1, and MCKOE to 1.
4. Program the serial interface protocol: CKPOL = 0, WSINV = 0, LSBFRST = 0, CHLEN = 1 (32 bits per channel) DATLEN = 1 (24 bits), I2SSTD = 0 (I2S Philips standard), I2SCFG = 5 (master full-duplex), I2SMOD = 1, for I2S/PCM mode.
5. Adjust the FIFO threshold by setting the wanted value into FTHLV. For example, if a threshold of two audio samples is required, FTHLV = 1.
6. If the application wishes to perform data transfer via DMA, set the bits TXDMAEN and RXDMAEN.
7. If needed, the application can then configure the external audio codec via an I2C interface.
8. Clear all interrupt enable fields located in the SPI_IER register.
9. Clear all status flags, by setting SUSPC, TIFREC, OVRC, UDRC of the SPI_IFCR register.
10. When the application wants to start the data transfer, set the SPE bit. Then:
    - If the data transfer uses DMA:
    a) Program the DMA peripheral: two channels, one for RX and one for TX.
    b) Initialize the memory buffer with valid audio samples for TX path.
    c) Enable the DMA channels.
    d) Enable interrupt events such as UDRIE and OVRIE if needed in order to detect transfer errors.
    e) Enable the DMA channel. If TXDMAEN was set, the TxFIFO is filled up.
    - If the data transfer is done via interrupt:

      Enable the interrupt events UDRIE, OVRIE, TXPIE, and RXPIE (by writing 1). An interrupt request is immediately activated allowing the interrupt handler to fill-up the TxFIFO.
11. Finally, the SPI/I2S serial interface can be enabled by setting the bit CSTART. CSTART bit is located into the SPI_CR1 register.

Stop procedure in master mode

1. Set the bit CSUSP, in order to stop ongoing transfers
2. Check the value of the CSTART bit until it goes to 0
3. Clear the SUSP flag by setting SUSPC
4. Stop DMA peripheral, bus clock...
5. Clear bit SPE in order to disable the SPI/I2S block

Slave I2S Philips standard, receive

This example shows how to program the interface for supporting the I2S Philips standard protocol in slave receiver mode, with a sampling rate of 48 kHz. Note that in slave mode the

SPI/I2S block cannot control the sample rate of the received samples. In this example we took the assumption that the external master device is delivering an I2S frame structure with a channel length of 24 bits. So, we cannot use the capability offered for frame error detection when FIXCH is set.

Procedure

1. Via the RCC block, enable the bus interface and the kernel clocks, assert and release the reset signal if needed,
2. Program the AFMUX in order to select the wanted I/Os. In the current example CK, WS, SDI,
3. Program the serial interface protocol: CKPOL = 0, WSINV = 0, LSBFRST = 0, FIXCH = 0 (because channel length is different from 16 and 32 bits), DATLEN = 0 (16 bits), I2SSTD = 0 (Philips protocol), I2SCFG = 1 (slave RX), I2SMOD = 1, for I2S mode.
4. Adjust the FIFO threshold by setting the wanted value into FTHLV. For example if a threshold of two audio samples is required, FTHLV = 1.
5. Clear all status flag registers.
6. Enable the flags that generate an interrupt such as OVRIE and TIFRE.
7. If the data transfer uses DMA:
   – Program the DMA peripheral: one RX channel
   – Enable the DMA channel,
   – In the SPI/I2S block, enable the DMA by setting the RXDMAEN bit.
8. If the data transfer is done via interrupt, then the user has to enable the interrupt by setting the RXPIE bit.
9. Set SPE.
10. Finally, the user can set the bit CSTART in order to enable the serial interface. The SPI/I2S starts to store data into the RxFIFO on the next occurrence of left data transmitted by the external master device.

Stop procedure in slave mode

1. Clear bit SPE in order to disable the SPI/I2S block
2. Stop DMA peripheral, bus clock...

67.10 I2S interrupts

In PCM/I2S mode an interrupt (**spi_it**) or a wake-up event signal (**spi_wkup**) can be generated according to the events described in the *Table 666*.

Interrupt events can be enabled and disabled separately.

Table 666. I2S interrupt requests

Interrupt vector	Interrupt event	Event flag	Event/Interrupt clearing method	Exit Sleep mode	Exit Stop modes	Exit Standby mode
SPI	TxFIFO threshold reached	TXP	When the TxFIFO contains less than FTHLV empty locations	Yes	Yes	No
	RxFIFO threshold reached	RXP	When the RxFIFO contains less than FTHLV samples			
	Overrun error	OVR	Write OVRC to 1			
	Underrun error	UDR	Write UDRC to 1			
	Frame error flag	TIFRE	Write TIFREC to 1		No	

67.11 SPI/I2S registers

67.11.1 SPI/I2S control register 1 (SPI_CR1)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOLOCK
															rs
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TCRCINI	RCRCINI	CRC33_17	SSI	HDDIR	CSUSP	CSTART	MASRX	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPE
rw	rw	rw	rw	rw	w	rs	rw								rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **IOLOCK**: locking the AF configuration of associated I/Os

This bit can be changed by software only when SPI is disabled (SPE = 0). It is cleared by hardware if a MODF event occurs

When this bit is set, the SPI_CFG2 register content cannot be modified. This bit is write-protected when SPI is enabled (SPE = 1).

0: AF configuration is not locked

1: AF configuration is locked

Bit 15 **TCRCINI**: CRC calculation initialization pattern control for transmitter

0: all zero pattern is applied

1: all ones pattern is applied

Bit 14 **RCRCINI**: CRC calculation initialization pattern control for receiver

0: All zero pattern is applied

1: All ones pattern is applied

Bit 13 **CRC33_17**: Full size (33-bit or 17-bit) CRC polynomial configuration

0: Full size (33-bit or 17-bit) CRC polynomial is not used

1: Full size (33-bit or 17-bit) CRC polynomial is used

Bit 12 **SSI:** Internal slave select signal input level

This bit has an effect only when the SSM bit is set. The value of this bit is forced onto the peripheral NSS input internally and the I/O value of the NSS pin is ignored.

Bit 11 **HDDIR:** Rx/Tx direction at half-duplex mode

In half-duplex configuration the HDDIR bit establishes the Rx/Tx direction of the data transfer. This bit is ignored in Full-Duplex or any Simplex configuration.

0: SPI is receiver

1: SPI is transmitter

Bit 10 **CSUSP:** master suspend request

This bit reads as zero.

In master mode, when this bit is set by software, the CSTART bit is reset at the end of the current frame and communication is suspended. The user has to check SUSP flag to check end of the frame transfer.

The master mode communication must be suspended (using this bit or keeping TXDR empty) before going to Low-power mode.

After software suspension, SUSP flag must be cleared and SPI disabled and re-enabled before the next transfer starts.

Bit 9 **CSTART:** master transfer start

This bit can be set by software if SPI is enabled only to start an SPI or I2S/PCM communication. In SPI mode, it is cleared by hardware when end of transfer (EOT) flag is set or when a transfer suspend request is accepted. In I2S/PCM mode, it is also cleared by hardware as described in the *Section 67.9.8: Stop sequence*.

In SPI mode, the bit is taken into account at master mode only. If transmission is enabled, communication starts or continues only if any data is available in the transmission FIFO.

0: master transfer is at idle

1: master transfer is ongoing or temporary suspended by automatic suspend

Bit 8 **MASRX:** master automatic suspension in Receive mode

This bit is set and cleared by software to control continuous SPI transfer in master receiver mode and automatic management in order to avoid overrun condition.

When SPI communication is suspended by hardware automatically, it may happen that few bits of next frame are already clocked out due to internal synchronization delay.

This is why, the automatic suspension is not quite reliable when size of data drops below 8 bits. In this case, a safe suspension can be achieved by combination with delay inserted between data frames applied when MIDI parameter keeps a non zero value; sum of data size and the interleaved SPI cycles must always produce interval at length of 8 SPI clock periods at minimum. After software clearing of the SUSP bit, the communication resumes and continues by subsequent bits transfer without any next constraint. Before the SUSP bit is cleared, the user must release the RxFIFO space as much as possible by reading out all the data packets available at RxFIFO based on the RXP flag indication to prevent any subsequent suspension.

0: SPI flow/clock generation is continuous, regardless of overrun condition. (data are lost)

1: SPI flow is suspended temporary on RxFIFO full condition, before reaching overrun condition. The SUSP flag is set when the SPI communication is suspended.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **SPE:** serial peripheral enable

This bit is set by and cleared by software.

When SPE = 1, SPI data transfer is enabled, the SPI_CFG1 and SPI_CFG2 configuration registers, CRCPOLY, UDRDR, IOLOCK bit in the SPI_CR1 register are write protected. They can be changed only when SPE = 0.

When SPE = 0 any SPI operation is stopped and disabled, all the pending requests of the events with enabled interrupt are blocked except the MODF interrupt request (but their pending still propagates the request of the spi_plck clock), the NSS output is deactivated at master, the RDY signal keeps not ready status at slave, the internal state machine is reseted, all the FIFOs content is flushed, CRC calculation initialized, receive data register is read zero. SPE is cleared and cannot be set when MODF error flag is active.

0: Serial peripheral disabled.

1: Serial peripheral enabled

67.11.2 SPI/I2S control register 2 (SPI_CR2)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSIZE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TSIZE[15:0]**: number of data at current transfer

When these bits are changed by software, the SPI must be disabled.

Endless transfer is initialized when CSTART is set while zero value is stored in TSIZE. TSIZE cannot be set to 0xFFFF for a full-featured instance and 0x3FF for a limited-featured instance, when CRC is enabled.

Note: TSIZE[15:10] bits are reserved for limited-featured instances, and must be kept at reset value.

67.11.3 SPI/I2S configuration register 1 (SPI_CFG1)

Address offset: 0x008

Reset value: 0x0007 0007

The content of this register is write protected when SPI is enabled.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BPASS	MBR[2:0]			Res.	Res.	Res.	Res.	Res.	CRCEN	Res.	CRCSIZE[4:0]				
rw	rw	rw	rw						rw		rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXDMAEN	RXDMAEN	Res.	Res.	Res.	Res.	UDRCFG	FTHLV[3:0]				DSIZE[4:0]				
rw	rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **BPASS:** bypass of the prescaler at master baud rate clock generator
0: bypass is disabled
1: bypass is enabled

Bits 30:28 **MBR[2:0]:** master baud rate prescaler setting
000: SPI master clock/2
001: SPI master clock/4
010: SPI master clock/8
011: SPI master clock/16
100: SPI master clock/32
101: SPI master clock/64
110: SPI master clock/128
111: SPI master clock/256
Note: MBR setting is considered at slave working at TI mode, too (see Section 67.5.1: TI mode).

Bits 27:23 Reserved, must be kept at reset value.

Bit 22 **CRCEN:** hardware CRC computation enable
0: CRC calculation disabled
1: CRC calculation enabled

Bit 21 Reserved, must be kept at reset value.

Bits 20:16 **CRCSIZE[4:0]:** length of CRC frame to be transferred and compared
Most significant bits are taken into account from polynomial calculation when CRC result is transferred or compared. The length of the polynomial is not affected by this setting.
The value must be equal or a multiple of the data size configured through the DSIZE bitfield. Its maximum size corresponds to the instance maximum data size.
00000: not used
00001: not used
00010: not used
00011: 4 bits
00100: 5 bits
00101: 6 bits
00110: 7 bits
00111: 8 bits
.....
11101: 30 bits
11110: 31 bits
11111: 32 bits
Note: The most significant bit at CRCSIZE bit field is reserved for the peripheral instances where data size is limited to 16 bits.
The CRCSIZE[2:0] bits are fixed to 1 for the peripheral instances with a limited set of features.

Bit 15 **TXDMAEN:** Tx DMA stream enable
0: Tx DMA disabled
1: Tx DMA enabled

Bit 14 **RXDMAEN:** Rx DMA stream enable
0: Rx-DMA disabled
1: Rx-DMA enabled

Bits 13:10 Reserved, must be kept at reset value.

Bit 9 **UDRCFG:** behavior of slave transmitter at underrun condition

For more details see *Figure 893: Optional configurations of the slave behavior when an underrun condition is detected*.

0: slave sends a constant pattern defined by the user in the SPI_UDRDR register

1: Slave repeats the last received data from master. When slave is configured at transmit only mode (COMM[1:0] = 01), all zeros pattern is repeated.

Bits 8:5 **FTHLV[3:0]:** FIFO threshold level

Defines number of data frames in a single data packet. It is recommended that the size of the packet does not exceed 1/2 of FIFO space.

SPI interface is more efficient if configured packet sizes are aligned with data register access parallelism:

–If SPI data register is accessed as a 16-bit register and DSIZE ≤ 8 bits, better to select FTHLV = 2, 4, 6.

–If SPI data register is accessed as a 32-bit register and DSIZE > 8 bits, better to select FTHLV = 2, 4, 6, while if DSIZE ≤ 8 bits, better to select FTHLV = 4, 8, 12.

0000: 1-data
0001: 2-data
0010: 3-data
0011: 4-data
0100: 5-data
0101: 6-data
0110: 7-data
0111: 8-data
1000: 9-data
1001: 10-data
1010: 11-data
1011: 12-data
1100: 13-data
1101: 14-data
1110: 15-data
1111: 16-data

Note: FTHLV[3:2] bits are reserved for instances with a limited set of features.

Bits 4:0 **DSIZE[4:0]:** number of bits in a single SPI data frame

00000: not used
00001: not used
00010: not used
00011: 4 bits
00100: 5 bits
00101: 6 bits
00110: 7 bits
00111: 8 bits
.....
11101: 30 bits
11110: 31 bits
11111: 32 bits

Note: The most significant bit DSIZE[4] is reserved for instances which data size is limited to 16 bits.

DSIZE[2:0] bits are fixed to 1 for instances with a limited set of features.

67.11.4 SPI/I2S configuration register 2 (SPI_CFG2)

Address offset: 0x00C

Reset value: 0x0000 0000

The content of this register is write-protected when SPI is enabled or the IOLOCK bit is set in the SPI_CR1 register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AFCNTR	SSOM	SSOE	SSIOP	Res.	SSM	CPOL	CPHA	LSBFRST	MASTER	SP[2:0]			COMM[1:0]		Res.
rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IOSWP	RDIOP	RDIOM	Res.	Res.	Res.	Res.	Res.	MIDI[3:0]				MSSI[3:0]			
rw	rw	rw						rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **AFCNTR:** alternate function GPIOs control

This bit is taken into account when SPE = 0 only.

When SPI must be disabled temporary for a specific configuration reason (for example CRC reset, CPHA or HDDIR change) setting this bit prevents any glitches on the associated outputs configured at alternate function mode by keeping them forced at state corresponding the current SPI configuration.

0: The peripheral takes no control of GPIOs while it is disabled

1: The peripheral keeps always control of all associated GPIOs

Note: This bit can be also used in PCM and I2S modes.

The bit AFCNTR must not be set, when the block is in slave mode.

Bit 30 **SSOM**: NSS output management in master mode

This bit is taken into account in master mode when SSOE is enabled. It allows the NSS output to be configured between two consecutive data transfers.

0: NSS is kept at active level until data transfer is complete, it becomes inactive with EOT flag

1: SPI data frames are interleaved with NSS nonactive pulses when MIDI[3:0] > 1

Bit 29 **SSOE:** NSS output enable

This bit is taken into account in master mode only

0: NSS output is disabled and the SPI can work in multimaster configuration

1: NSS output is enabled. The SPI cannot work in a multimaster environment. It forces the NSS pin at inactive level after the transfer is complete or SPI is disabled with respect to SSOM, MIDI, MSSI, SSIOP bits setting

Bit 28 **SSIOP**: NSS input/output polarity

0: low level is active for NSS signal

1: high level is active for NSS signal

Bit 27 Reserved, must be kept at reset value.

Bit 26 **SSM:** Software management of internal slave select signal input

0: Input value of the internal slave select signal is determined by the external NSS hardware pin

1: Input value of the internal slave select signal is determined by the SSI bit controlled by software

Note: When master uses hardware NSS output (SSM = 0 and SSOE = 1), the internal slave select signal input is forced to not active state internally to prevent master mode fault error.

Bit 25 **CPOL:** clock polarity
0: SCK signal is at 0 when idle
1: SCK signal is at 1 when idle

Bit 24 **CPHA:** clock phase
0: the first clock transition is the first data capture edge
1: the second clock transition is the first data capture edge

Bit 23 **LSBFRST:** data frame format
0: MSB transmitted first
1: LSB transmitted first
Note: This bit can be also used in PCM and I2S modes.

Bit 22 **MASTER:** SPI master
0: SPI slave
1: SPI master

Bits 21:19 **SP[2:0]:** serial protocol
000: SPI Motorola
001: SPI TI
others: reserved, must not be used

Bits 18:17 **COMM[1:0]:** SPI Communication Mode
00: full-duplex
01: simplex transmitter
10: simplex receiver
11: half-duplex

Bit 16 Reserved, must be kept at reset value.

Bit 15 **IOSWP:** swap functionality of MISO and MOSI pins
When this bit is set, the function of MISO and MOSI pins alternate functions are inverted. Original MISO pin becomes MOSI and original MOSI pin becomes MISO.
0: no swap
1: MOSI and MISO are swapped
Note: This bit can be also used in PCM and I2S modes to swap SDO and SDI pins.

Bit 14 **RDIOP:** RDY signal input/output polarity
0: high level of the signal means the slave is ready for communication
1: low level of the signal means the slave is ready for communication

Bit 13 **RDIOM:** RDY signal input/output management
0: RDY signal is defined internally fixed as permanently active (RDIOP setting has no effect)
1: RDY signal is overtaken from alternate function input (at master case) or output (at slave case) of the dedicated pin (RDIOP setting takes effect)
Note: When DSIZE in the SPI_CFG1 register is configured shorter than 8 bits, the RDIOM bit must be kept at zero.

Bits 12:8 Reserved, must be kept at reset value.

Bits 7:4 **MIDI[3:0]**: Master Inter-Data Idleness

Specifies minimum time delay (expressed in SPI clock cycles periods) inserted between two consecutive data frames in master mode.

0000: no delay

0001: 1 clock cycle period delay

...

1111: 15 clock cycle periods delay

Note: This feature is not supported in TI mode.

Bits 3:0 **MSSI[3:0]**: Master NSS Idleness

Specifies an extra delay, expressed in number of SPI clock cycle periods, inserted additionally between active edge of NSS opening a session and the beginning of the first data frame of the session in master mode when SSOE is enabled.

0000: no extra delay

0001: 1 clock cycle period delay added

...

1111: 15 clock cycle periods delay added

Note: This feature is not supported in TI mode.
To include the delay, the SPI must be disabled and re-enabled between sessions.

67.11.5 SPI/I2S interrupt enable register (SPI_IER)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	MODFIE	TIFREIE	CRCEIE	OVRIE	UDRIE	TXTFIE	EOTIE	DXPIE	TXPIE	RXPIE
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **MODFIE**: mode Fault interrupt enable

0: MODF interrupt disabled

1: MODF interrupt enabled

Bit 8 **TIFREIE**: TIFRE interrupt enable

0: TIFRE interrupt disabled

1: TIFRE interrupt enabled

Bit 7 **CRCEIE:** CRC error interrupt enable

0: CRC interrupt disabled

1: CRC interrupt enabled

Bit 6 **OVRIE:** OVR interrupt enable

0: OVR interrupt disabled

1: OVR interrupt enabled

Bit 5 **UDRIE:** UDR interrupt enable
0: UDR interrupt disabled
1: UDR interrupt enabled

Bit 4 **TXTFIE:** TXTF interrupt enable
0: TXTF interrupt disabled
1: TXTF interrupt enabled

Bit 3 **EOTIE:** EOT, SUSP and TXC interrupt enable
0: EOT/SUSP/TXC interrupt disabled
1: EOT/SUSP/TXC interrupt enabled

Bit 2 **DXPIE**: DXP interrupt enabled
DXPIE is set by software and cleared by TXTF flag set event.
0: DXP interrupt disabled
1: DXP interrupt enabled

Bit 1 **TXPIE:** TXP interrupt enable
TXPIE is set by software and cleared by TXTF flag set event.
0: TXP interrupt disabled
1: TXP interrupt enabled

Bit 0 **RXPIE:** RXP interrupt enable
0: RXP interrupt disabled
1: RXP interrupt enabled

67.11.6 SPI/I2S status register (SPI_SR)

Address offset: 0x014

Reset value: 0x0000 1002

All the flags of this register are not cleared automatically when the SPI is reenabled. They require specific clearing access exclusively via the flag clearing register as noted in the bits descriptions below.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CTSIZE[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXWNE	RXPLVL[1:0]		TXC	SUSP	Res.	MODF	TIFRE	CRCE	OVR	UDR	TXTF	EOT	DXP	TXP	RXP
r	r	r	r	r		r	r	r	r	r	r	r	r	r	r

Bits 31:16 **CTSIZE[15:0]**: number of data frames remaining in current TSIZE session
The value is not quite reliable when traffic is ongoing on bus .
Note: CTSIZE[15:0] bits are not available in instances with limited set of features.

Bit 15 **RXWNE:** RxFIFO word not empty
0: less than four bytes of RxFIFO space is occupied by data
1: at least four bytes of RxFIFO space is occupied by data
Note: This bit value does not depend on DSIZE setting and keeps together with RXPLVL[1:0] information about RxFIFO occupancy by residual data.

Bits 14:13 **RXPLVL[1:0]:** RxFIFO packing level

When RXWNE = 0 and data size is set up to 16 bits, the value gives number of remaining data frames persisting at RxFIFO.

When data size is greater than 16 bits, these bits are always read as 00. As a result, the single data frame received in the FIFO cannot be detected neither by RWNE nor by RXPLVL bits if the data size is set from 17 to 24 bits. The user must then apply other methods to detect the number of data received, such as monitor the EOT event when TSIZE > 0 or RXP events when FTHLV = 0.

00: no next frame is available at RxFIFO

01: 1 frame is available

10: 2 frames are available*

11: 3 frames are available*

Note: (*): Possible value when data size is set up to 8 bits only.

Bit 12 **TXC:** TxFIFO transmission complete

The flag behavior depends on TSIZE setting.

When TSIZE = 0, the TXC is changed by hardware exclusively and it raises each time the TxFIFO becomes empty and there is no activity on the bus.

If TSIZE ≠ 0 there is no specific reason to monitor TXC as it just copies the EOT flag value including its software clearing. The TXC generates an interrupt when EOTIE is set.

This flag is set when SPI is reset or disabled.

0: current data transfer is still ongoing, data is available in TxFIFO or last frame transmission is on going.

1: last TxFIFO frame transmission complete

Bit 11 **SUSP:** suspension status

In master mode, SUSP is set by hardware either as soon as the current frame is complete after CSUSP request is done or at master automatic suspend receive mode (the MASRX bit is set in the SPI_CR1 register) on RxFIFO full condition.

SUSP generates an interrupt when EOTIE is set.

This bit must be cleared prior to disabling the SPI. This is done by setting the SUSPC bit of SPI_IFCR exclusively.

0: SPI not suspended (master mode active or other mode).

1: master mode is suspended (current frame completed).

Bit 10 Reserved, must be kept at reset value.

Bit 9 **MODF**: mode fault

When MODF is set, SPE and IOLOCK bits of the SPI_CR1 register are reset and setting SPE again is blocked until MODF is cleared.

This bit is cleared by writing 1 to the MODFC bit of the SPI_IFCR exclusively.

0: no mode fault

1: mode fault detected.

Bit 8 **TIFRE**: TI frame format error

This bit is cleared by writing 1 to the TIFREC bit of the SPI_IFCR exclusively.

0: no TI Frame Error

1: TI frame error detected

Bit 7 **CRCE:** CRC error

This bit is cleared when SPI is re-enabled or by writing 1 to the CRCEC bit of the SPI_IFCR optionally.

0: no CRC error

1: CRC error detected

Bit 6 **OVR:** overrun

This bit is cleared when SPI is re-enabled or by writing 1 to the OVRC bit of the SPI_IFCR optionally.

0: no overrun

1: overrun detected

Bit 5 **UDR:** underrun

This bit is cleared when SPI is re-enabled or by writing 1 to the UDRC bit of the SPI_IFCR optionally.

0: no underrun

1: underrun detected

Note: In SPI mode, the UDR flag applies to slave mode only. In I2S/PCM mode, (when available) this flag applies to master and slave mode.

Bit 4 **TXTF:** transmission transfer filled

TXTF is set by hardware as soon as all of the data packets in a transfer have been submitted for transmission by application software or DMA, that is when TSIZE number of data have been pushed into the TxFIFO.

This bit is cleared by software write 1 to the TXTFC bit of the SPI_IFCR exclusively.

The TXTF flag triggers an interrupt if the TXTFIE bit is set.

TXTF setting clears the TXPIE and DXPIE masks so to off-load application software from calculating when to disable TXP and DXP interrupts.

0: upload of TxFIFO is ongoing or not started

1: TxFIFO upload is finished

Bit 3 **EOT:** end of transfer

EOT is set by hardware as soon as a full transfer is complete, that is when SPI is re-enabled or when TSIZE number of data have been transmitted and/or received on the SPI. EOT is cleared when SPI is re-enabled or by writing 1 to the EOTC bit of the SPI_IFCR register optionally.

EOT flag triggers an interrupt if the EOTIE bit is set.

If DXP flag is used until TXTF flag is set and DXPIE is cleared, EOT can be used to download the last packets contained into RxFIFO in one-shot.

In master, EOT event terminates the data transfer and handles NSS output optionally. When CRC is applied, the EOT event is extended over the CRC frame transfer.

To restart the internal state machine properly, SPI is strongly suggested to be disabled and re-enabled before next transfer starts despite its setting is not changed.

0: transfer is ongoing or not started

1: transfer complete

Bit 2 **DXP**: duplex packet

DXP flag is set whenever both TXP and RXP flags are set regardless SPI mode.

0: TxFIFO is Full and/or RxFIFO is Empty

1: both TxFIFO has space for write and RxFIFO contains for read a single packet at least

Bit 1 **TXP:** Tx-packet space available

TXP flag can be changed only by hardware. Its value depends on the physical size of the FIFO and its threshold (FTHLV[3:0]), data frame size (DSIZE[4:0] in SPI mode and respective DATLEN[1:0] in I2S/PCM mode), and actual communication flow. If the data packet is stored by performing consecutive write operations to SPI_TXDR, TXP flag must be checked again once a complete data packet is stored at TxFIFO. TXP is set despite SPI TxFIFO becomes inaccessible when SPI is reset or disabled.

0: not enough free space at TxFIFO to host next data packet

1: enough free space at TxFIFO to host at least one data packet

Bit 0 **RXP:** Rx-packet available

The flag is changed by hardware. It monitors the total number of data currently available at RxFIFO if SPI is enabled. RXP value depends on the FIFO threshold (FTHLV[3:0]), data frame size (DSIZE[4:0] in SPI mode and DATLEN[1:0] in I2S/PCM mode), and actual communication flow. If the data packet is read by performing consecutive read operations from SPI_RXDR, RXP flag must be checked again once a complete data packet is read out from RxFIFO.

0: RxFIFO is empty or an incomplete data packet is received

1: RxFIFO contains at least one data packet

67.11.7 SPI/I2S interrupt/status flags clear register (SPI_IFCR)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	SUSPC	Res.	MODFC	TIFREC	CRCEC	OVRC	UDRC	TXTFC	EOTC	Res.	Res.	Res.
				w		w	w	w	w	w	w	w			

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **SUSPC:** Suspend flag clear

Writing a 1 into this bit clears SUSP flag of the SPI_SR register

Bit 10 Reserved, must be kept at reset value.

Bit 9 **MODFC**: mode fault flag clear

Writing a 1 into this bit clears MODF flag of the SPI_SR register

Bit 8 **TIFREC**: TI frame format error flag clear

Writing a 1 into this bit clears TIFRE flag of the SPI_SR register

Bit 7 **CRCEC:** CRC error flag clear

Writing a 1 into this bit clears CRCE flag of the SPI_SR register

Bit 6 **OVRC:** overrun flag clear

Writing a 1 into this bit clears OVR flag of the SPI_SR register

Bit 5 **UDRC:** underrun flag clear

Writing a 1 into this bit clears UDR flag of the SPI_SR register

Bit 4 **TXTFC:** transmission transfer filled flag clear

Writing a 1 into this bit clears TXTF flag of the SPI_SR register

Bit 3 **EOTC:** end of transfer flag clear

Writing a 1 into this bit clears EOT flag of the SPI_SR register

Bits 2:0 Reserved, must be kept at reset value.

67.11.8 SPI/I2S transmit data register (SPI_TXDR)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXDR[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXDR[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **TXDR[31:0]:** transmit data register

The register serves as an interface with TxFIFO. A write to it accesses TxFIFO.

Note: In SPI mode, data is always right-aligned. Alignment of data at I2S mode depends on DATLEN and DATFMT setting. Unused bits are ignored when writing to the register, and read as zero when the register is read.

Note: DR can be accessed byte-wise (8-bit access): in this case only one data-byte is written by single access.

half-word-wise (16 bit access) in this case 2 data-bytes or 1 half-word-data can be written by single access.

word-wise (32 bit access). In this case 4 data-bytes or 2 half-word-data or word-data can be written by single access.

Write access of this register less than the configured data size is forbidden.

67.11.9 SPI/I2S receive data register (SPI_RXDR)

Address offset: 0x030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXDR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXDR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXDR[31:0]:** receive data register

The register serves as an interface with RxFIFO. When it is read, RxFIFO is accessed.

Note: In SPI mode, data is always right-aligned. Alignment of data at I2S mode depends on DATLEN and DATFMT setting. Unused bits are read as zero when the register is read. Writing to the register is ignored.

Note: DR can be accessed byte-wise (8-bit access): in this case only one data-byte is read by single access

half-word-wise (16 bit access) in this case 2 data-bytes or 1 half-word data can be read by single access

word-wise (32 bit access). In this case 4 data-bytes or 2 half-word data or word-data can be read by single access.

Read access of this register less than the configured data size is forbidden.

67.11.10 SPI/I2S polynomial register (SPI_CRCPOLY)

Address offset: 0x040

Reset value: 0x0000 0107

The content of this register is write protected when SPI is enabled.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CRCPOLY[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRCPOLY[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CRCPOLY[31:0]:** CRC polynomial register

This register contains the polynomial for the CRC calculation.

The default 9-bit polynomial setting 0x107 corresponds to default 8-bit setting of DSIZE. It is compatible with setting 0x07 used in other ST products with fixed length of the polynomial string, where the most significant bit of the string is always kept hidden.

Length of the polynomial is given by the most significant bit of the value stored in this register. It must be set greater than DSIZE. CRC33_17 bit must be set additionally with CRCPOLY register when DSIZE is configured to maximum data size and CRC is enabled (to keep polynomial length grater than data size).

Note: CRCPOLY[31:16] bits are reserved for instances with data size limited to 16 bits. There is no constrain when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

67.11.11 SPI/I2S transmitter CRC register (SPI_TXCRC)

Address offset: 0x044

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXCRC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXCRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXCRC[31:0]:** CRC register for transmitter

When CRC calculation is enabled, the TXCRC[31:0] bits contain the computed CRC value of the subsequently transmitted bytes. CRC calculation is initialized when the CRCEN bit of the SPI_CR1 register is set or when a data block is transferred completely. The CRC is calculated serially using the polynomial programmed in the SPI_CRCPOLY register.

The number of bits considered at calculation depends on the SPI_CRCPOLY register and CRCSIZE bits settings in the SPI_CFG1 register.

Note: A read to this register when the communication is ongoing may return an incorrect value.

Note: This bitfield is not used in I2S mode.

Note: TXCRC[31-16] bits are reserved for instances with data size limited to 16 bits. There is no constrain when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

Note: The configuration of the CRCSIZE bitfield is not taken into account when the content of this register is read by software. No masking is applied for unused bits in this case.

67.11.12 SPI/I2S receiver CRC register (SPI_RXCRC)

Address offset: 0x048

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXCRC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXCRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXCRC[31:0]:** CRC register for receiver

When CRC calculation is enabled, the RXCRC[31:0] bits contain the computed CRC value of the subsequently received bytes. CRC calculation is initialized when the CRCEN bit of the SPI_CR1 register is set or when a data block is transferred completely. The CRC is calculated serially using the polynomial programmed in the SPI_CRCPOLY register.

The number of bits considered at calculation depends on the SPI_CRCPOLY register and CRCSIZE bits settings in the SPI_CFG1 register.

Note: A read to this register when the communication is ongoing may return an incorrect value.

Note: This bitfield is not used in I2S mode.

Note: RXCRC[31-16] bits are reserved for the peripheral instances with data size limited to 16 bits. There is no constrain when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

Note: The configuration of CRCSIZE bit field is not taken into account when the content of this register is read by software. No masking is applied for unused bits in this case.

67.11.13 SPI/I2S underrun data register (SPI_UDRDR)

Address offset: 0x04C

Reset value: 0x0000 0000

The content of this register is write protected when SPI is enabled.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UDRDR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UDRDR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **UDRDR[31:0]:** data at slave underrun condition

The register is taken into account in slave mode and at underrun condition only. The number of bits considered depends on the DSIZE bit setting in the SPI_CFG1 register. Underrun condition handling depends on the UDRCFG bit setting in the SPI_CFG1 register.

Note: UDRDR[31-16] bits are reserved for the peripheral instances with data size limited to 16 bits. There is no constraint when 32-bit access is applied at these addresses. Reserved bits 31-16 are always read zero while any write to them is ignored.

67.11.14 SPI/I2S configuration register (SPI_I2SCFGR)

Address offset: 0x050

Reset value: 0x0000 0000

This register must be configured when the I2S is disabled (SPE = 0). The content of this register is not taken into account in SPI mode except for the I2SMOD bit, which needs to be kept at 0.

This register is reserved for instances not supporting I2S mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	MCKOE	ODD	I2SDIV[7:0]							
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DATFMT	WSINV	FIXCH	CKPOL	CHLEN	DATLEN[1:0]		PCMSYNC	Res.	I2SSTD[1:0]		I2SCFG[2:0]			I2SMOD
	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **MCKOE**: master clock output enable

0: Master clock output is disabled

1: Master clock output is enabled

Bit 24 **ODD**: odd factor for the prescaler

Refer to *Section 67.9.9: Clock generator* for details

0: Real divider value is = I2SDIV *2

1: Real divider value is = (I2SDIV * 2) + 1

Bits 23:16 **I2SDIV[7:0]**: I^2S linear prescaler
Refer to *Section 67.9.9: Clock generator* for details

Bit 15 Reserved, must be kept at reset value.

Bit 14 **DATFMT**: data format
0: The data inside the SPI_RXDR or SPI_TXDR register are right aligned
1: The data inside the SPI_RXDR or SPI_TXDR register are left aligned.

Bit 13 **WSINV**: word select inversion
This bit is used to invert the default polarity of WS signal.
0: In I2S Philips standard, the left channel transfer starts one CK cycle after the WS falling edge, and the right channel one CK cycle after the WS rising edge.
In MSB- or LSB-justified mode, the left channel is transferred when WS is HIGH, and the right channel when WS is LOW.
In PCM short mode the data transfer starts at the falling edge of WS, while it starts at the rising edge of WS in PCM long mode.
1: In I2S Philips standard, the left channel transfer starts one CK cycle after the WS rising edge, and the right channel one CK cycle after the WS falling edge.
In MSB- or LSB-justified mode, the left channel is transfered when WS is LOW, and right channel when WS is HIGH.
In PCM short mode the data transfer starts at the rising edge of WS, while it starts at the falling edge of WS in PCM long mode.

Bit 12 **FIXCH**: fixed channel length in slave
0: the channel length in slave mode is different from 16 or 32 bits (CHLEN must be set)
1: the channel length in slave mode is supposed to be 16 or 32 bits (according to CHLEN)

Bit 11 **CKPOL**: serial audio clock polarity
0: the signals generated by the SPI/I2S (that is SDO and WS) are changed on the falling edge of CK and the signals received by the SPI/I2S (that is SDI and WS) are read of the rising edge of CK.
1: the signals generated by the SPI/I2S (that is SDO and WS) are changed on the rising edge of CK and the signals received by the SPI/I2S (that is SDI and WS) are read of the falling edge of CK.

Bit 10 **CHLEN**: channel length (number of bits per audio channel)
0: 16-bit wide
1: 32-bit wide

Bits 9:8 **DATLEN[1:0]**: data length to be transferred.
00: 16-bit data length
01: 24-bit data length
10: 32-bit data length
11: Not allowed
Note: Data width of 24 and 32 bits are not always supported, (DATLEN = 01 or 10), refer to Section 67.3: SPI implementation to check the supported data size.

Bit 7 **PCMSYNC**: PCM frame synchronization
0: short frame synchronization
1: long frame synchronization

Bit 6 Reserved, must be kept at reset value.

Bits 5:4 **I2SSTD[1:0]**: I²S standard selection
For more details on I²S standards, refer to *Section 67.9.5: Supported audio protocols*
00: I²S Philips standard.
01: MSB-justified standard (left justified)
10: LSB-justified standard (right justified)
11: PCM standard

Bits 3:1 **I2SCFG[2:0]**: I2S configuration mode
000: slave - transmit
001: slave - receive
010: master - transmit
011: master - receive
100: slave - Full Duplex
101: master - Full Duplex
others: not used

Bit 0 **I2SMOD**: I2S mode selection
0: SPI mode is selected
1: I2S/PCM mode is selected

67.11.15 SPI/I2S register map

Table 667. SPI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	SPI_CR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IOLOCK	TCRCINI	RCRCINI	CRC33_17	SSI	HDDIR	CSUSP	CSTART	MASRX	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPE
	Reset value																0	0	0	0	0	0	0	0	0								0
0x004	SPI_CR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSIZE[15:0](1)															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	SPI_CFG1	BPASS	MBR[2:0]			Res.	Res.	Res.	Res.	Res.	CRCEN	Res.	CRCSIZE[4:0](1)					TXDMAEN	RXDMAEN	Res.	Res.	Res.	Res.	UDRCFG	FTHLV[3:0](1)				DSIZE[4:0](1)				
	Reset value	0	0	0	0						0		0	0	1	1	1	0	0					0	0	0	0	0	0	0	1	1	1
0x00C	SPI_CFG2	AFCNTR	SSOM	SSOE	SSIOP	Res.	SSM	CPOL	CPHA	LSBFRST	MASTER	SP[2:0]			COMM[1:0]		Res.	IOSWP	RDIOP	RDIOM	Res.	Res.	Res.	Res.	Res.	MIDI[3:0]				MSSI[3:0]			
	Reset value	0	0	0	0		0	0	0	0	0	0	0	0	0	0		0	0	0						0	0	0	0	0	0	0	0
0x010	SPI_IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODFIE	TIFREIE	CRCEIE	OVRIE	UDRIE	TXTFIE	EOTIE	DXPIE	TXPIE	RXPIE
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x014	SPI_SR	CTSIZE[15:0](1)																RXWNE	RXPLVL[1:0]		TXC	SUSP	Res.	MODF	TIFRE	CRCE	OVR	UDR	TXTF	EOT	DPXP	TXP	RXP
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0		0	0	0	0	0	0	0	0	1	0
0x018	SPI_IFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUSPC	Res.	MODFC	TIFREC	CRCEC	OVRC	UDRC	TXTFC	EOTC	Res.	Res.	Res.
	Reset value																					0		0	0	0	0	0	0	0			
0x01C	Reserved	Res.																															
0x020	SPI_TXDR	TXDR[31:16]																TXDR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 667. SPI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x024-0x02C	Reserved	Res.																															
0x030	SPI_RXDR	RXDR[31:16]																RXDR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034 - 0x03C	Reserved	Res.																															
0x040	SPI_CRCPOLY	CRCPOLY[31:16](2)																CRCPOLY[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	1
0x044	SPI_TXCRC	TXCRC[31:16](2)																TXCRC[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x048	SPI_RXCRC	RXCRC[31:16](2)																RXCRC[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04C	SPI_UDRDR	UDRDR[31:16](2)																UDRDR[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x050	SPI_I2SCFGR	Res.	Res.	Res.	Res.	Res.	Res.	MCKOE	ODD	I2SDIV[7:0]								Res.	Res.	WSINV	FIXCH	CKPOL	CHLEN	DATLEN[1:0]		PCMSYNC	Res.	I2SSTD[1:0]		I2SCFG[2:0]			I2SMOD
	Reset value							0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0	0		0	0	0	0	0	0

1. The configuration of this bitfield depends on the features of the SPI instance. For more details, refer to *Section 67.3: SPI implementation*.
2. The bits 31-16 are reserved for the peripheral instances with data size limited to 16 bits. There is no constraint when the 32-bit access is applied at these addresses. The bits 31-16, when reserved, are always read to zero while any write to them is ignored.

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

68 Serial audio interface (SAI)

68.1 SAI introduction

The SAI interface (serial audio interface) offers a wide set of audio protocols due to its flexibility and wide range of configurations. Many audio protocols can be addressed thanks to SAI “free protocol mode”. The free protocol mode allows the application to define the audio frame shape, number of slots, slot size, data size, and so on. For example, the SAI supports I2S standards, LSB- or MSB-justified, PCM/DSP, TDM, and AC’97 protocols. SPDIF output is offered when the audio block is configured as a transmitter.

To bring this level of flexibility and reconfigurability, the SAI contains two independent audio subblocks. Each block has it own clock generator and I/O line controller.

The SAI works in master or slave configuration. The audio subblocks are either receiver or transmitter and work synchronously or not (with respect to the other one).

The SAI can be connected with other SAIs to work synchronously.

68.2 SAI main features

- Two independent audio subblocks, which can be transmitters or receivers with their respective FIFO
- 8-word integrated FIFOs for each audio subblock
- Synchronous or asynchronous mode between the audio subblocks
- Possible synchronization between multiple SAIs
- Master or slave configuration independent for both audio subblocks
- Clock generator for each audio block to target independent audio frequency sampling when both audio subblocks are configured in master mode
- Data size configurable: 8-, 10-, 16-, 20-, 24-, 32-bit
- Audio protocol: I2S, LSB- or MSB-justified, PCM/DSP, TDM, AC’97
- PDM interface, supporting up to 4 microphone pairs
- SPDIF output available if required
- Up to 16 slots available with configurable size
- Number of bits by frame can be configurable
- Frame synchronization active level configurable (offset, bit length, level)
- First active bit position in the slot is configurable
- LSB first or MSB first for data transfer
- Mute mode
- Stereo/Mono audio frame capability
- Communication clock strobing edge configurable (SCK)
- Error flags with associated interrupts if enabled respectively
  - Overrun and underrun detection
  - Anticipated frame synchronization signal detection in slave mode
  - Late frame synchronization signal detection in slave mode
  - Codec not ready for the AC’97 mode in reception

- Interrupt sources when enabled:
  - Errors
  - FIFO requests
- 2-channel DMA interface

68.3 SAI implementation

Table 668. SAI features(1)

SAI features	SAI1	SAI2
I2S, LSB- or MSB-justified, PCM/DSP, TDM, AC'97	X	X
FIFO size	8 words	8 words
SPDIF	X	X
PDM	X(2)	-

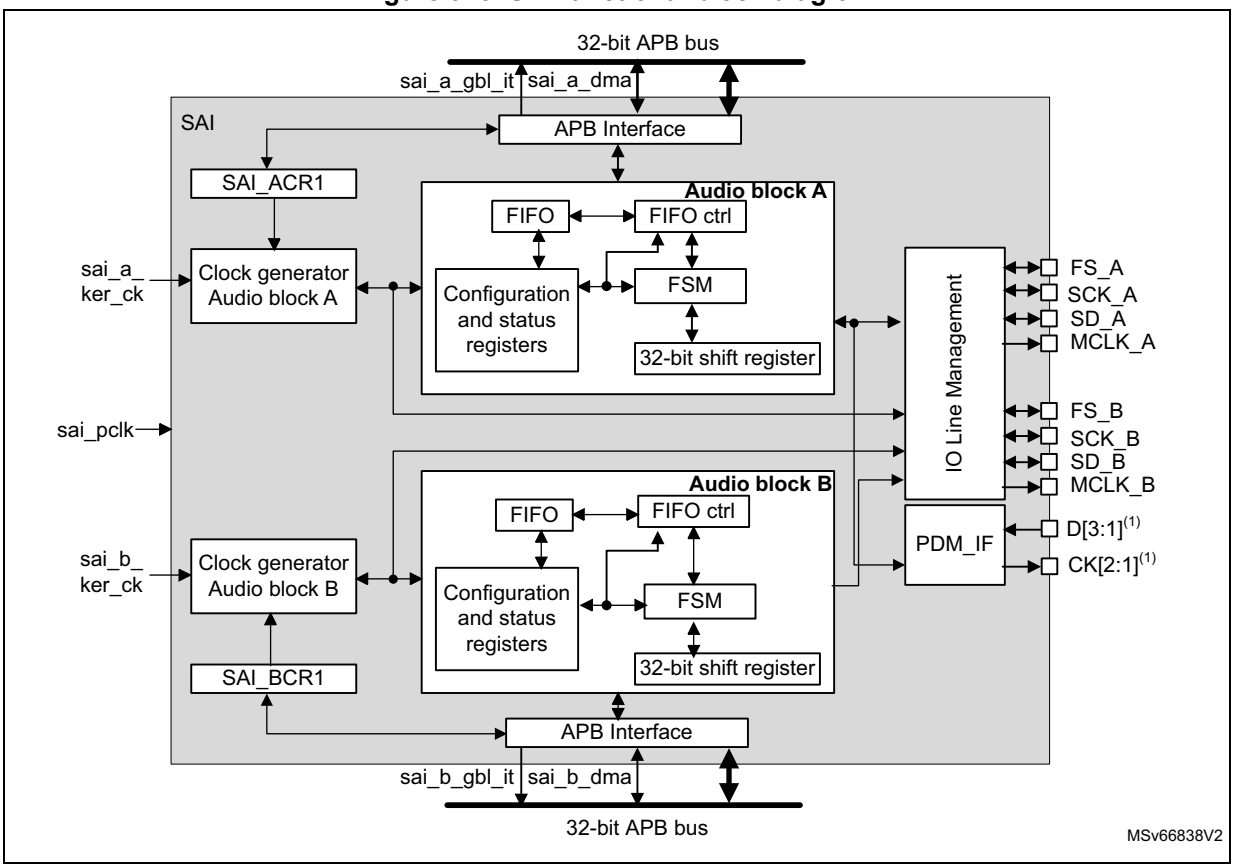
1. 'X' = supported, '-' = not supported.
2. Only signals D[3:1], and CK[2:1] are available.

68.4 SAI functional description

68.4.1 SAI block diagram

Figure 913 shows the SAI block diagram while *Table 669* and *Table 670* list SAI internal and external signals.

Figure 913. SAI functional block diagram



1. These signals might not be available for all SAI instances. Refer to *Section 68.3: SAI implementation* for details.

The SAI is mainly composed of two audio subblocks with their own clock generator. Each audio block integrates a 32-bit shift register controlled by their own functional state machine. Data are stored or read from the dedicated FIFO. FIFO may be accessed by the CPU, or by DMA to leave the CPU free during the communication. Each audio block is independent. They can be synchronous with each other.

An I/O line controller manages a set of 4 dedicated pins (SD, SCK, FS, MCLK) for a given audio block in the SAI. Some of these pins can be shared if the two subblocks are declared as synchronous to leave some free to be used as general purpose I/Os. The MCLK pin can be output, or not, depending on the application, the decoder requirement and whether the audio block is configured as the master.

If one SAI is configured to operate synchronously with another one, even more I/Os can be freed (except for pins SD_x).

The functional state machine can be configured to address a wide range of audio protocols. Some registers are present to set up the desired protocols (audio frame waveform generator).

The audio subblock can be a transmitter or receiver, in master or slave mode. The master mode means the SCK_x bit clock and the frame synchronization signal are generated from the SAI, whereas in slave mode, they come from another external or internal master. There is a particular case for which the FS signal direction is not directly linked to the master or slave mode definition. In AC'97 protocol, it is an SAI output even if the SAI (link controller) is set up to consume the SCK clock (and so to be in Slave mode).

Note: For ease of reading of this section, the notation SAI_x refers to SAI_A or SAI_B, where 'x' represents the SAI A or B subblock.

68.4.2 SAI pins and internal signals

Table 669. SAI internal input/output signals

Internal signal name	Signal type	Description
sai_a_gbl_it/ sai_b_gbl_it	Output	Audio block A and B global interrupts
sai_a_dma, sai_b_dma	Input/output	Audio block A and B DMA acknowledges and requests
sai_sync_out_sck, sai_sync_out_fs	Output	Internal clock and frame synchronization output signals exchanged with other SAI blocks
sai_sync_in_sck, sai_sync_in_fs	Input	Internal clock and frame synchronization input signals exchanged with other SAI blocks
sai_a_ker_ck/ sai_b_ker_ck	Input	Audio block A/B kernel clock
sai_pclk	Input	APB clock

Table 670. SAI input/output pins

Pin name	Pin type	Comments
SAI_SCK_A	Input/output	Audio block A bit clock
SAI_MCLK_A	Output	Audio block A master clock
SAI_FS_A	Input/output	Frame synchronization line for audio block A
SAI_SD_A	Input/output	Data line for block A
SAI_SCK_B	Input/output	Audio block B bit clock
SAI_MCLK_B	Output	Audio block B master clock
SAI_FS_B	Input/output	Frame synchronization line for audio block B
SAI_SD_B	Input/output	Data line for block B
SAI_CK[2:1]	Output	PDM bitstream clock(1)
SAI_D[3:1]	Input	PDM bitstream data(1)

1. These signals might not be available in all SAI instances. Refer to *Section 68.3: SAI implementation* for details.

68.4.3 Main SAI modes

Each audio subblock of the SAI can be configured to be master or slave via the MODE bits in the SAI_xCR1 register of the selected audio block.

Master mode

In master mode, the SAI delivers the timing signals to the external connected device:

- The bit clock and the frame synchronization are output on pin SCK_x and FS_x, respectively.
- If needed, the SAI can also generate a master clock on the MCLK_x pin.

Both SCK_x, FS_x and MCLK_x are configured as outputs.

Slave mode

The SAI expects to receive timing signals from an external device.

- If the SAI subblock is configured in asynchronous mode, then the SCK_x and FS_x pins are configured as inputs.
- If the SAI subblock is configured to operate synchronously with another SAI interface or with the second audio subblock, the corresponding SCK_x and FS_x pins are left free to be used as general purpose I/Os.

In slave mode, the MCLK_x pin is not used and can be assigned to another function.

It is recommended to enable the slave device before enabling the master.

Configuring and enabling SAI modes

Each audio block can use a different audio protocol. When PRTCFG[1:0] of the SAI_xCR1 register is set to 0, the free protocol mode is selected and each SAI subblock can emulate I2S standards, LSB- or MSB-justified, PCM/DSP, TDM, or AC’97 protocols.

Each audio subblock can be independently defined as a transmitter or receiver through the MODE bit in the SAI_xCR1 register of the corresponding audio block. As a result, the SAI_SD_x pin is respectively configured as an output or an input.

Two master audio blocks in the same SAI can be configured with two different MCLK and SCK clock frequencies. In this case, they have to be configured in asynchronous mode.

Each of the audio blocks in the SAI is enabled by the SAIEN bit in the SAI_xCR1 register. As soon as this bit is active, the transmitter or the receiver is sensitive to the activity on the clock, data, and synchronization lines in slave mode.

In master Tx mode, enabling the audio block immediately generates the bit clock for the external slaves even if there is no data in the FIFO. However, FS signal generation is conditioned by the presence of data in the FIFO. After the FIFO receives the first data to transmit, this data is output to external slaves. If there is no data to transmit in the FIFO, 0 values are then sent in the audio frame with an underrun flag generation.

In slave mode, the audio frame starts when the audio block is enabled and when a start of frame is detected.

In Slave Tx mode, no underrun event is possible on the first frame after the audio block is enabled, because the mandatory operating sequence in this case is:

1. Write into the SAI_xDR (by software or by DMA).
2. Wait until the FIFO threshold *(*FLH) flag is different from 0b000 (FIFO empty).
3. Enable the audio block in slave transmitter mode.

68.4.4 SAI synchronization mode

There are two levels of synchronization, either at audio subblock level or at SAI level.

Internal synchronization

An audio subblock can be configured to operate synchronously with the second audio subblock in the same SAI. In this case, the bit clock and the frame synchronization signals are shared to reduce the number of external pins used for the communication. The audio block configured in synchronous mode sees its own SCK_x, FS_x, and MCLK_x pins released back as GPIOs while the audio block configured in asynchronous mode is the one for which FS_x and SCK_x ad MCLK_x I/O pins are relevant (if the audio block is considered as master).

Typically, the audio block in synchronous mode can be used to configure the SAI in full duplex mode. One of the two audio blocks can be configured as a master and the other as slave, or both as slaves with one asynchronous block (corresponding SYNCEN[1:0] bits set to 00 in SAI_xCR1) and one synchronous block (corresponding SYNCEN[1:0] bits set to 01 in the SAI_xCR1 register).

Note: Due to internal resynchronization stages, PCLK APB frequency must be higher than twice the bit rate clock frequency.

External synchronization

The audio subblocks can also be configured to operate synchronously with another SAI. This can be done as follows:

1. The SAI, which is configured as the source from which the other SAI is synchronized, has to define which of its audio subblocks is supposed to provide the FS and SCK signals to other SAI. This is done by programming SYNCOUT[1:0] bits.
2. The SAI which receives the synchronization signals, has to select which SAI provides the synchronization by setting the proper value on SYNCIN[1:0] bits. For each of the two SAI audio subblocks, the user must then specify if it operates synchronously with the other SAI via the SYNCEN bit.

Note: The SYNCIN[1:0] and SYNCOUT[1:0] bits are located in the SAI_GCR register, and the SYNCEN bits in the SAI_xCR1 register.

If both audio subblocks in a given SAI need to be synchronized with another SAI, it is possible to choose one of the following configurations:

- Configure each audio block to be synchronous with another SAI block through the SYNCEN[1:0] bits.
- Configure one audio block to be synchronous with another SAI through the SYNCEN[1:0] bits. The other audio block is then configured as synchronous with the second SAI audio block through SYNCEN[1:0] bits.

The following table shows how to select the proper synchronization signal depending on the SAI block used. For example SAI2 can select the synchronization from SAI1 by setting SAI2 SYNCIN to 0. If SAI1 wants to select the synchronization coming from SAI2, SAI1 SYNCIN must be set to 1. Positions noted as ‘Reserved’ must not be used.

Table 671. External synchronization selection

Block instance	SYNCIN = 1	SYNCIN = 0
SAI1	SAI2 sync.	Reserved
SAI2	Reserved	SAI1 sync.

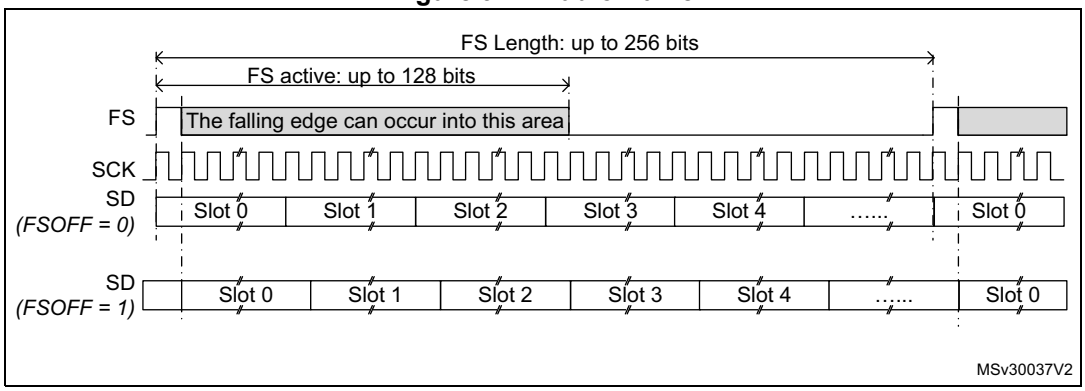
68.4.5 Audio data size

The audio frame can target different data sizes by configuring bit DS[2:0] in the SAI_xCR1 register. The data sizes may be 8, 10, 16, 20, 24, or 32 bits. During the transfer, either the MSB or the LSB of the data is sent first, depending on the configuration of the LSBFIRST bit in the SAI_xCR1 register.

68.4.6 Frame synchronization

The FS signal acts as the frame synchronization signal in the audio frame (start of frame). The shape of this signal is completely configurable to target the different audio protocols with their own specificities concerning this frame synchronization behavior. This reconfigurability is done using the SAI_xFRCR register. *Figure 914* illustrates this flexibility.

Figure 914. Audio frame



In AC'97 mode or in SPDIF mode (bit PRTCFG[1:0] = 10 or PRTCFG[1:0] = 01 in the SAI_xCR1 register), the frame synchronization shape is forced to match the AC'97 protocol. The SAI_xFRCR register value is ignored.

Each audio block is independent and consequently each one requires a specific configuration.

Frame length

- Master mode

  The audio frame length can be configured to up to 256-bit clock cycles, by configuring the FRL[7:0] field in the SAI_xFRCR register.

  If the frame length is greater than the number of declared slots for the frame, the remaining bits to transmit are extended to 0 or the SD line is released to high-Z

depending on the state of bit TRIS in the SAI_xCR2 register (refer to *FS signal role*). In reception mode, the remaining bit is ignored.

If bit NODIV is cleared, (FRL+1) must be equal to a power of 2, from 8 to 256, to ensure that an audio frame contains an integer number of MCLK pulses per bit clock cycle.

If bit NODIV is set, the (FRL+1) field can take any value from 8 to 256. Refer to *Section 68.4.8: SAI clock generator*"

- Slave mode

  The audio frame length is mainly used to specify to the slave the number of bit clock cycles per audio frame sent by the external master. It is used mainly to detect from the master any anticipated or late occurrence of the frame synchronization signal during an ongoing audio frame. In this case, an error is generated. For more details, refer to *Section 68.4.14: Error flags*.

  In slave mode, there are no constraints on the FRL[7:0] configuration in the SAI_xFRCR register.

The number of bits in the frame is equal to FRL[7:0] + 1.

The minimum number of bits to transfer in an audio frame is 8.

Frame synchronization polarity

The FSPOL bit in the SAI_xFRCR register sets the active polarity of the FS pin from which a frame is started. The start of the frame is edge sensitive.

In slave mode, the audio block waits for a valid frame to start transmitting or receiving. The start of the frame is synchronized to this signal. It is effective only if the start of the frame is not detected during an ongoing communication and assimilated to an anticipated start of frame (refer to *Section 68.4.14: Error flags*).

In master mode, the frame synchronization is sent continuously each time an audio frame is complete until the SAIEN bit in the SAI_xCR1 register is cleared. If no data are present in the FIFO at the end of the previous audio frame, an underrun condition is managed as described in *Section 68.4.14: Error flags*), but the audio communication flow is not interrupted.

Frame synchronization active level length

The FSALL[6:0] bits of the SAI_xFRCR register enable the configuration of the length of the active level of the frame synchronization signal. The length can be set from 1- to 128-bit clock cycles.

As an example, the active length can be half of the frame length in I2S, LSB or MSB-justified modes, or one-bit wide for PCM/DSP or TDM.

Frame synchronization offset

Depending on the audio protocol targeted in the application, the frame synchronization signal can be asserted when transmitting the last bit or the first bit of the audio frame (this is the case in I2S standard protocol and in MSB-justified protocol, respectively). The FSOFF bit in the SAI_xFRCR register enables the possibility to choose between two configurations.

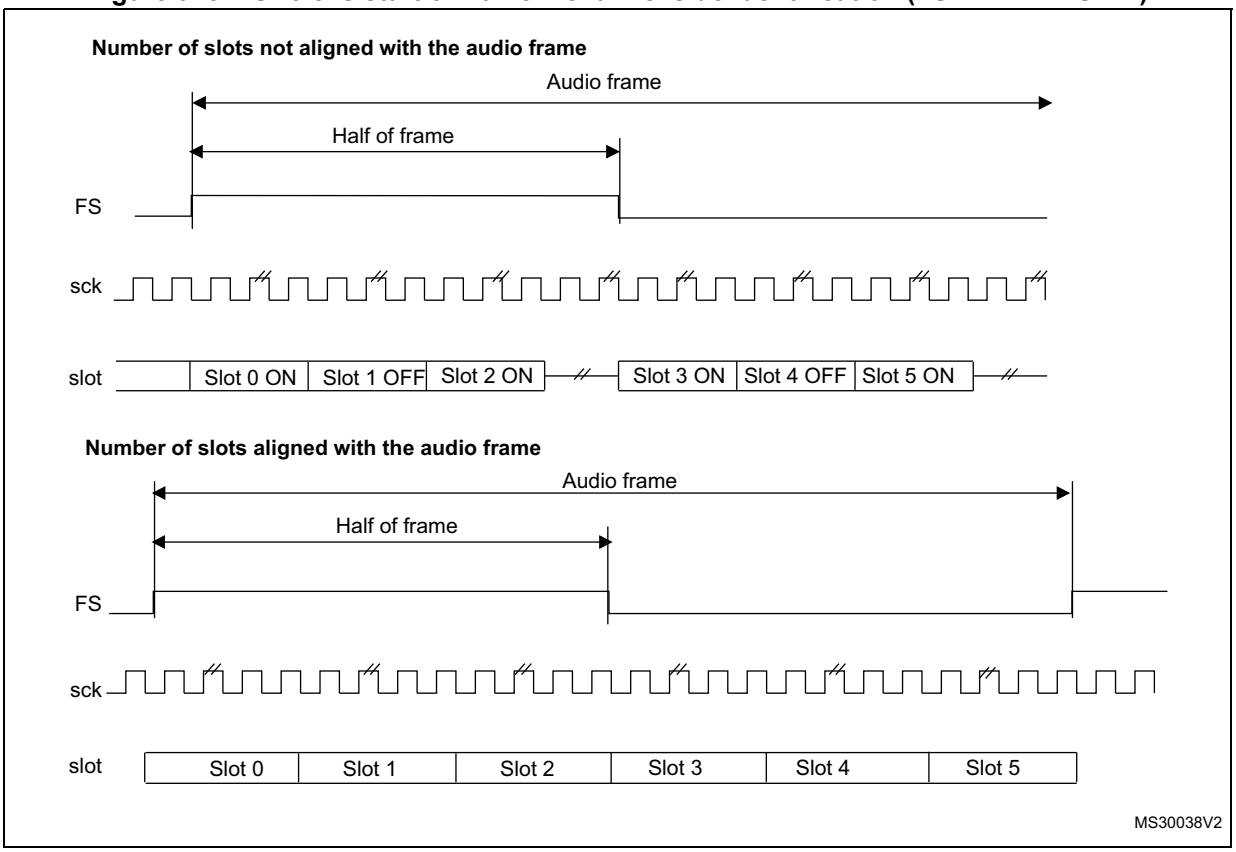
FS signal role

The FS signal can have a different meaning depending on the FS function. FSDEF bit in the SAI_xFRCR register selects which meaning it has:

- 0: start of frame, like, for instance, the PCM/DSP, TDM, AC'97, audio protocols,
- 1: start of frame and channel side identification within the audio frame like for the I2S or the MSB- or LSB-justified protocols.

When the FS signal is considered as a start of frame and channel side identification within the frame, the number of declared slots must be considered to be half the number for the left channel and half the number for the right channel. If the number of bit clock cycles on half audio frame is greater than the number of slots dedicated to a channel side, and TRIS = 0, 0 is sent for transmission for the remaining bit clock cycles in the SAI_xCR2 register. Otherwise, if TRIS = 1, the SD line is released to high-Z. In reception mode, the remaining bit clock cycles are not considered until the channel side changes.

Figure 915. FS role is start of frame + channel side identification (FSDEF = TRIS = 1)

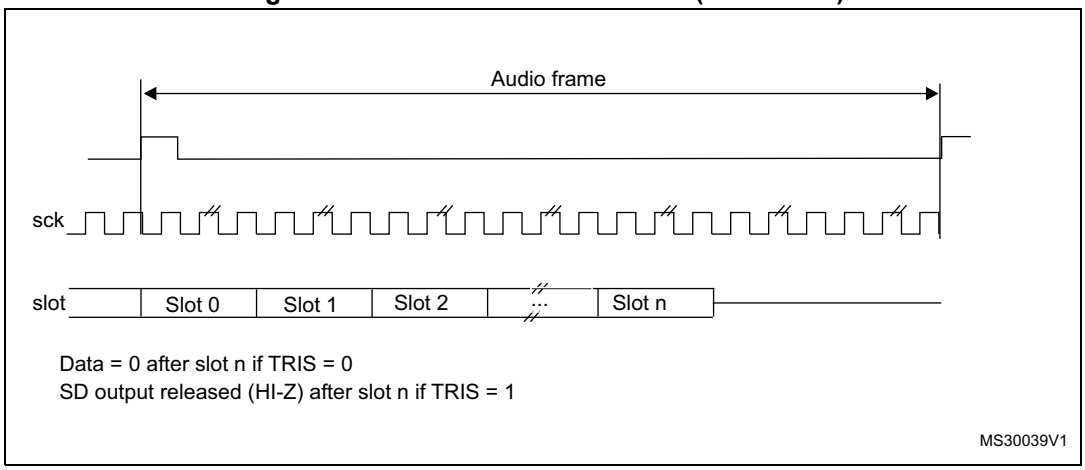


1. The frame length must be even.

If the FSDEF bit in SAI_xFRCR is kept clear, so FS signal is equivalent to a start of frame, and if the number of slots defined in NBSLOT[3:0] in SAI_xSLOTR multiplied by the number of bits by slot configured in SLOTSZ[1:0] in SAI_xSLOTR is less than the frame size (bit FRL[7:0] in the SAI_xFRCR register), then:

- If TRIS = 0 in the SAI_xCR2 register, the remaining bit after the last slot is forced to 0 until the end of frame in case of transmitter,
- If TRIS = 1, the line is released to high-Z during the transfer of these remaining bits. In reception mode, these bits are discarded.

Figure 916. FS role is start of frame (FSDEF = 0)



The FS signal is not used when the audio block in transmitter mode is configured to get the SPDIF output on the SD line. The corresponding FS I/O is released and left free for other purposes.

68.4.7 Slot configuration

The slot is the basic element in the audio frame. The number of slots in the audio frame is equal to NBSLOT[3:0] + 1.

The maximum number of slots per audio frame is fixed at 16.

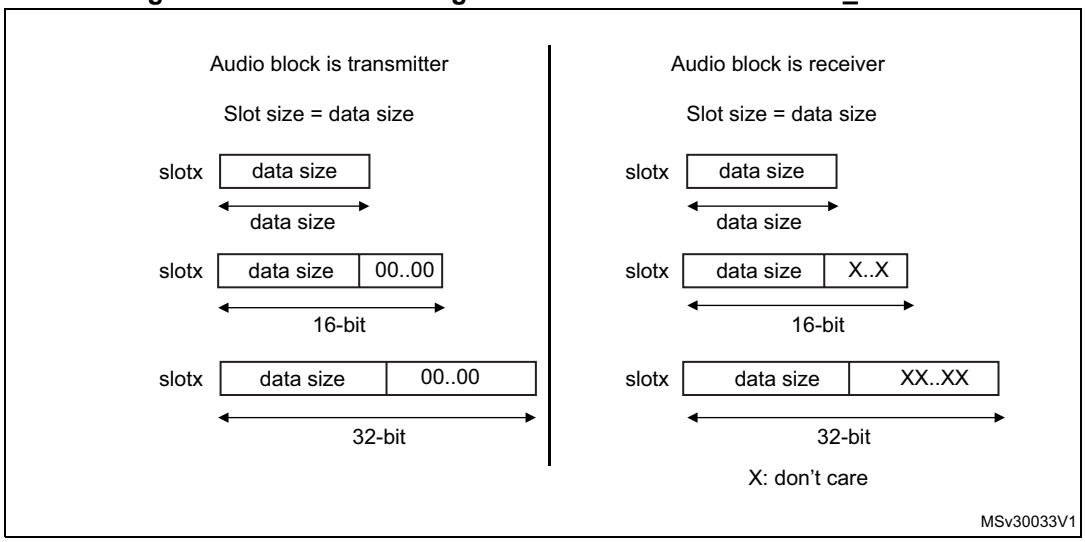
For AC'97 protocol or SPDIF (when bit PRTCFG[1:0] = 10 or PRTCFG[1:0] = 01), the number of slots is automatically set to target the protocol specification, and the value of NBSLOT[3:0] is ignored.

Each slot can be defined as a valid slot, or not, by setting SLOTEN[15:0] bits of the SAI_xSLOTR register.

When an invalid slot is transferred, the SD data line is either forced to 0 or released to high-Z depending on the TRIS bit configuration (refer to *Output data line management on an inactive slot*) in transmitter mode. In receiver mode, the received value from the end of this slot is ignored. Consequently, there is no FIFO access and so no request to read or write the FIFO linked to this inactive slot status.

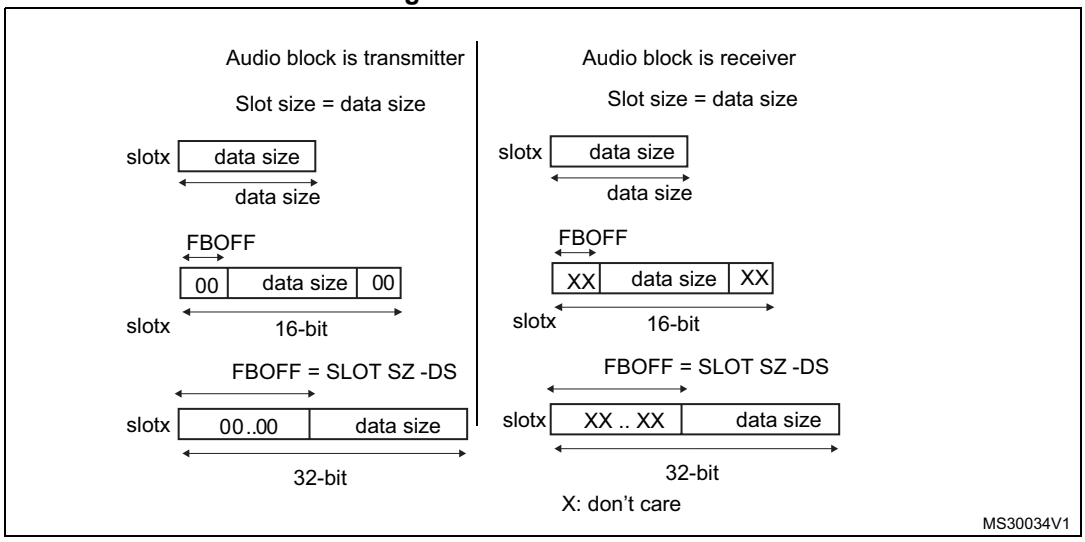
The slot size is also configurable as shown in *Figure 917.* The size of the slots is selected by configuring the SLOTSZ[1:0] bits in the SAI_xSLOTR register. The size is applied identically for each slot in an audio frame.

Figure 917. Slot size configuration with FBOFF = 0 in SAI_xSLOTR



It is possible to choose the position of the first data bit to transfer within the slots. This offset is configured by FBOFF[4:0] bits in the SAI_xSLOTR register. 0 values are injected in transmitter mode from the beginning of the slot until this offset position is reached. In reception, the bit in the offset phase is ignored. This feature targets the LSB justified protocol (if the offset is equal to the slot size minus the data size).

Figure 918. First bit offset



It is mandatory to respect the following conditions to avoid bad SAI behavior:

- FBOFF ≤ (SLOTSZ - DS)
- DS ≤ SLOTSZ
- NBSLOT x SLOTSZ ≤ FRL (frame length)

The number of slots must be even when bit FSDEF in the SAI_xFRCR register is set.

In AC'97 and SPDIF protocol (bit PRTCFG[1:0] = 10 or PRTCFG[1:0] = 01), the slot size is automatically set as defined in *Section 68.4.11: AC'97 link controller*.

68.4.8 SAI clock generator

Each audio block has its own clock generator. The clock generator builds the master clock (MCLK_x) and bit clock (SCK_x) signals from the sai_x_ker_ck. The sai_x_ker_ck clock is delivered by the clock controller of the product (RCC).

Generation of the master clock (MCLK_x)

The clock generator provides the master clock (MCLK_x) when the audio block is defined as Master or Slave. The master clock is generated as soon as the MCKEN bit is set to 1 even if the SAIEN bit for the corresponding block is set to 0. This feature can be useful if the MCLK_x clock is used as system clock for an external audio device, since it enables the generation of the MCLK_x before activating the audio stream.

To generate a master clock on MCLK_x output before transferring the audio samples, the user application has to follow the sequence below:

1. Check that SAIEN = 0.
2. Program the MCKDIV[5:0] divider to the required value.
3. Set the MCKEN bit to 1.
4. Later, the application can configure other parts of the SAI, and sets the SAIEN bit to 1 to start the transfer of audio samples.

To avoid disturbances on the clock generated on MCLK_x output, the following operations are not recommended:

- Changing MCKDIV when MCKEN = 1
- Setting MCKEN to 0 if the SAIEN = 1

The SAI makes sure that there are no spurs on MCLK_x output when the MCLK_x is switched ON and OFF via the MCKEN bit (with SAIEN = 0).

Table 672 shows MCLK_x activation conditions.

Table 672. MCLK_x activation conditions

MCLKEN	NODIV	SAIEN for block x	MCLK_x
0	X	0	Disabled
1			Enabled
0	1	1	Disabled
1			Enabled
X	0		Enabled

Note: MCLK_x can also be generated in AC'97 mode, when MCLKEN is set to 1.

Generation of the bit clock (SCK_x)

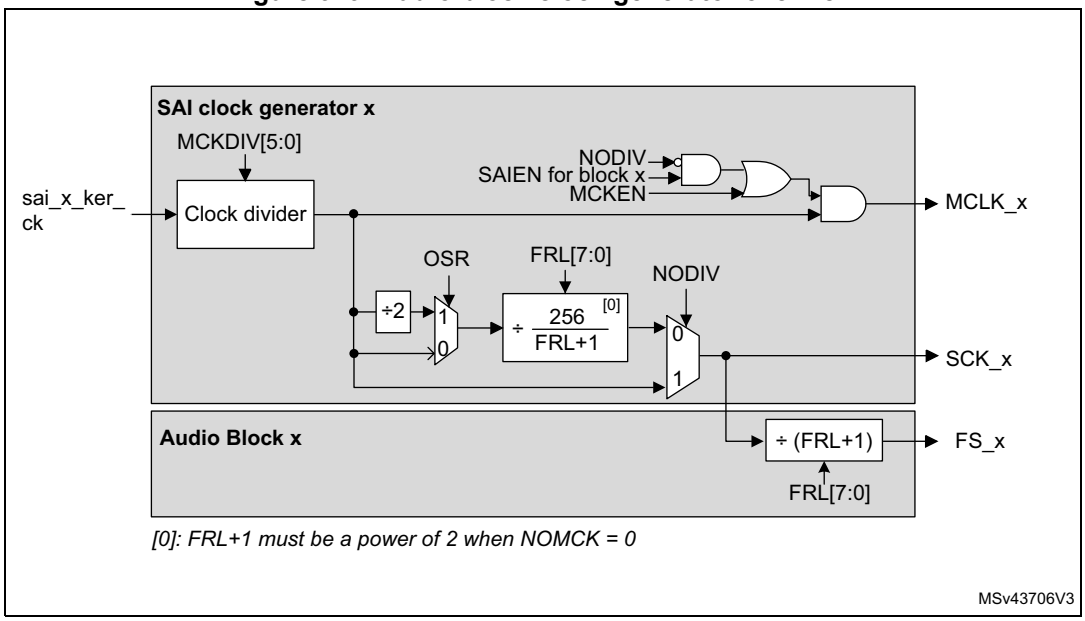
The clock generator provides the bit clock (SCK_x) when the audio block is defined as Master. The frame synchronization (FS_x) is also derived from the signals provided by the clock generator.

In Slave mode, the value of NODIV and OSR fields are ignored, and the SCK_x clock is not generated.

The bit clock strobing edge of SCK_x can be configured through the CKSTR fields, which is functional both in master and slave mode.

Figure 919 illustrates the architecture of the audio block clock generator.

Figure 919. Audio block clock generator overview



The NODIV bit must be used to force the ratio between the master clock (MCLK_x) and the frame synchronization (FS_x) frequency to 256 or 512.

- If NODIV is set to 0, the frequency ratio between the frame synchronization and the master clock is fixed to 512 or 256, according to OSR value, but the frame length must be a power of 2. More details are given below.
- If NODIV is set to 1, the application can adjust the frequency of the bit clock (SCK_x) via MCKDIV. In addition, there is no restriction on the frame length value as long as the frame length is bigger or equal to 8 (that is, FRL[7:0] > 6). The frame synchronization frequency depends on MCKDIV and frame length (FRL[7:0]). In that case, the frequency of the MCLK_x is equal to the SCK_x.

The NODIV, MCKEN, SAIEN, OVR, CKSTR, and MCKDIV[5:0] bits belong to the SAI_xCR1 register, while FRL[7:0] belongs to SAI_xFRCR.

Clock generator programming when NODIV = 0

In that case, the MCLK_x frequency is:

- $F_{MCLK_x} = 256 \times F_{FS_x}$ if OSR = 0
- $F_{MCLK_x} = 512 \times F_{FS_x}$ if OSR = 1

When MCKDIV is different from 0, the MCLK_x frequency is given by the formula below:

$$F_{MCLK_x} = \frac{F_{sai_x_ker_ck}}{MCKDIV}$$

The frame synchronization frequency is given by:

$$F_{FS_x} = \frac{F_{sai_x_ker_ck}}{MCKDIV \times (OSR + 1) \times 256}$$

The bit clock frequency (SCK_x) is given by the following formula:

$$F_{SCK_x} = \frac{F_{sai_x_ker_ck} \times (FRL + 1)}{MCKDIV \times (OSR + 1) \times 256}$$

Note: When NODIV is equal to 0, (FRL+1) must be a power of two. In addition, (FRL+1) must range between 8 and 256. (FRL +1) represents the number of bit clock in the audio frame.

When the MCKDIV division ratio is odd, the MCLK duty cycle is not 50%. The bit clock signal (SCK_x) can also have a duty cycle different from 50% if MCKDIV is odd, if OSR is equal to 0, and if (FRL+1) = 2^8.

It is recommended, to program MCKDIV to an even value or to large values (higher than 10).

Note that MCKDIV = 0 gives the same result as MCKDIV = 1.

Clock generator programming when NODIV = 1

When MCKDIV is different from 0, the frequency of the bit clock (SCK_x) is given in the formula below:

$$F_{SCK_x} = F_{MCLK_x} = \frac{F_{sai_x_ker_ck}}{MCKDIV}$$

The frequency of the frame synchronization (FS_x) in given by the following formula:

$$F_{FS_x} = \frac{F_{sai_x_ker_ck}}{(FRL + 1) \times MCKDIV}$$

Note: When NODIV is set to 1, (FRL+1) can take any values from 8 to 256.

MCKDIV = 0 gives the same result as MCKDIV = 1.

Clock generator programming examples

Table 673 gives programming examples for 48, 96, and 192 kHz.

Table 673. Clock generator programming examples

Input sai_x_ker_ck clock frequency	MCLK	F_{MCLK}/ F_{FS}	FRL (1)	OSR	NODIV	MCKEN	MCKDIV[5:0]	Audio sampling frequency (F_{FS})
98.304 MHz	Y	512	2^N-1	1	0	1	0 or 1	192 kHz
		512	2^N-1	1	0	1	2	96 kHz
		512	2^N-1	1	0	1	4	48 kHz
		256	2^N-1	0	0	1	2	192 kHz
		256	2^N-1	0	0	1	4	96 kHz
		256	2^N-1	0	0	1	8	48 kHz
	N	-	63	-	1	0	8	192 kHz
		-	63	-	1	0	16	96 kHz
		-	63	-	1	0	32	48 kHz

1. N is an integer value between 3 and 8.

68.4.9 Internal FIFOs

Each audio block in the SAI has its own FIFO. Depending on if the block is defined to be a transmitter or a receiver, the FIFO can be written or read, respectively. Thus, there is only one FIFO request linked to the FREQ bit in the SAI_xSR register.

An interrupt is generated if the FREQIE bit is enabled in the SAI_xIM register. This depends on:

- The FIFO threshold setting (FLVL bits in SAI_xCR2).
- Communication direction (transmitter or receiver). Refer to *Interrupt generation in transmitter mode* and *Interrupt generation in reception mode*.

Interrupt generation in transmitter mode

The interrupt generation depends on the FIFO configuration in transmitter mode:

- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO empty (FTH[2:0] set to 0b000), an interrupt is generated (FREQ bit set by hardware to 1 in the SAI_xSR register) if no data are available in the SAI_xDR register (FLVL[2:0] bits in SAI_xSR are less than 0b001). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when the FIFO is no longer empty (FLVL[2:0] bits in SAI_xSR are different from 0b000), that is, one or more data are stored in the FIFO.
- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO quarter full (FTH[2:0] set to 0b001), an interrupt is generated (FREQ bit set by hardware to 1 in the SAI_xSR register) if less than a quarter of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are less than 0b010). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when at least a quarter of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b010).

- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO half full (FTH[2:0] set to 0b010), an interrupt is generated (FREQ bit set by hardware to 1 in the SAI_xSR register) if less than half of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are less than 0b011). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when at least half of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b011).
- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO three quarter (FTH[2:0] set to 0b011), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if less than three quarters of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are less than 0b100). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when at least three quarters of the FIFO contains data (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b100).
- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO full (FTH[2:0] set to 0b100), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if the FIFO is not full (FLVL[2:0] bits in SAI_xSR are less than 0b101). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when the FIFO is full (FLVL[2:0] bits in SAI_xSR are equal to 0b101).

Interrupt generation in reception mode

The interrupt generation depends on the FIFO configuration in reception mode:

- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO empty (FTH[2:0] set to 0b000), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if at least one data is available in the SAI_xDR register (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b001). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when the FIFO becomes empty (FLVL[2:0] bits in the SAI_xSR are equal to 0b000), that is, no data are stored in FIFO.
- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO quarter full (FTH[2:0] set to 0b001), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if at least one quarter of the FIFO data locations are available (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b010). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when less than a quarter of the FIFO data locations become available (FLVL[2:0] bits in SAI_xSR are less than 0b010).
- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO half full (FTH[2:0] set to 0b010), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if at least half of the FIFO data locations are available (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b011). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when less than half of the FIFO data locations become available (FLVL[2:0] bits in SAI_xSR are less than 0b011).
- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO three quarter full (FTH[2:0] set to 0b011), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if at least three quarters of the FIFO data locations are available (FLVL[2:0] bits in SAI_xSR are higher than or equal to 0b100). This interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when the FIFO has less than three quarters of the FIFO data locations available (FLVL[2:0] bits in SAI_xSR is less than 0b100).
- When the FIFO threshold bits in the SAI_xCR2 register are configured as FIFO full (FTH[2:0] set to 0b100), an interrupt is generated (FREQ bit is set by hardware to 1 in the SAI_xSR register) if the FIFO is full (FLVL[2:0] bits in SAI_xSR are equal to 0b101). This

interrupt (FREQ bit in the SAI_xSR register) is cleared by hardware when the FIFO is not full (FLVL[2:0] bits in SAI_xSR are less than 0b101).

Like interrupt generation, the SAI can use the DMA if the DMAEN bit in the SAI_xCR1 register is set. The FREQ bit assertion mechanism is the same as the interrupt generation mechanism described above for FREQIE.

Each FIFO is an 8-word FIFO. Each read or write operation from/to the FIFO targets one word FIFO location whatever the access size. Each FIFO word contains one audio slot. FIFO pointers are incremented by one word after each access to the SAI_xDR register.

Data must be right-aligned when written in the SAI_xDR register.

Data received are right-aligned in the SAI_xDR register.

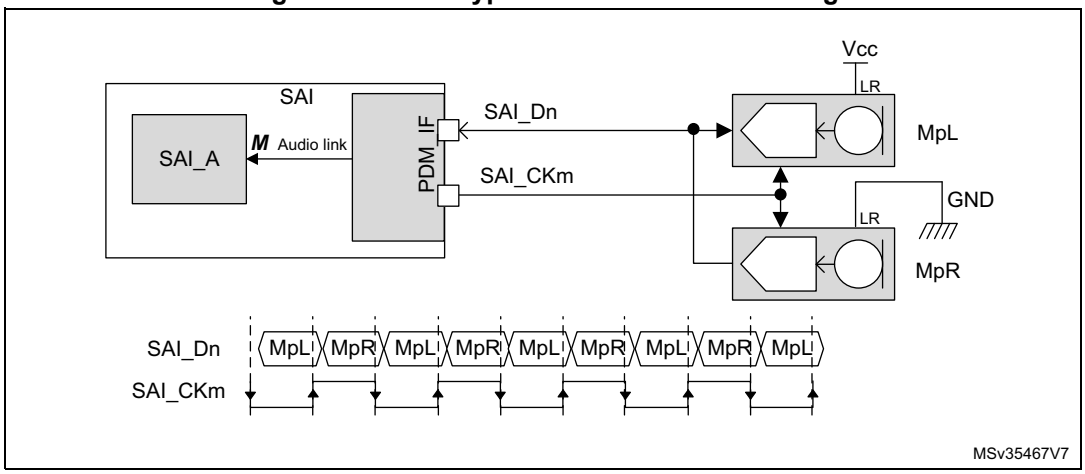
The FIFO pointers can be reinitialized when the SAI is disabled by configuring the FFLUSH bit in the SAI_xCR2 register. If FFLUSH is set when the SAI is enabled, the data present in the FIFO are lost automatically.

68.4.10 PDM interface

The PDM (pulse density modulation) interface is provided in order to support digital microphones. Up to 4 digital microphone pairs can be connected in parallel. Depending on product implementation, less microphones can be supported (refer to *Section 68.3: SAI implementation*).

Figure 920 shows a typical connection of a digital microphone pair via a PDM interface. Both microphones share the same bitstream clock and data line. Thanks to a configuration pin (LR), a microphone can provide valid data on SAI_CK[m] rising edge while the other provides valid data on SAI_CK[m] falling edge (m being the number of clock lines).

Figure 920. PDM typical connection and timing



1. **n** refers to the number of data lines and **p** to the number of microphone pairs.

The PDM function is intended to be used in conjunction with SAI_A subblock configured in TDM master mode. It cannot be used with SAI_B subblock. The PDM interface uses the timing signals provided by the serial audio interface of SAI_A and adapts them to generate a bitstream clock (SAI_CK[m]).

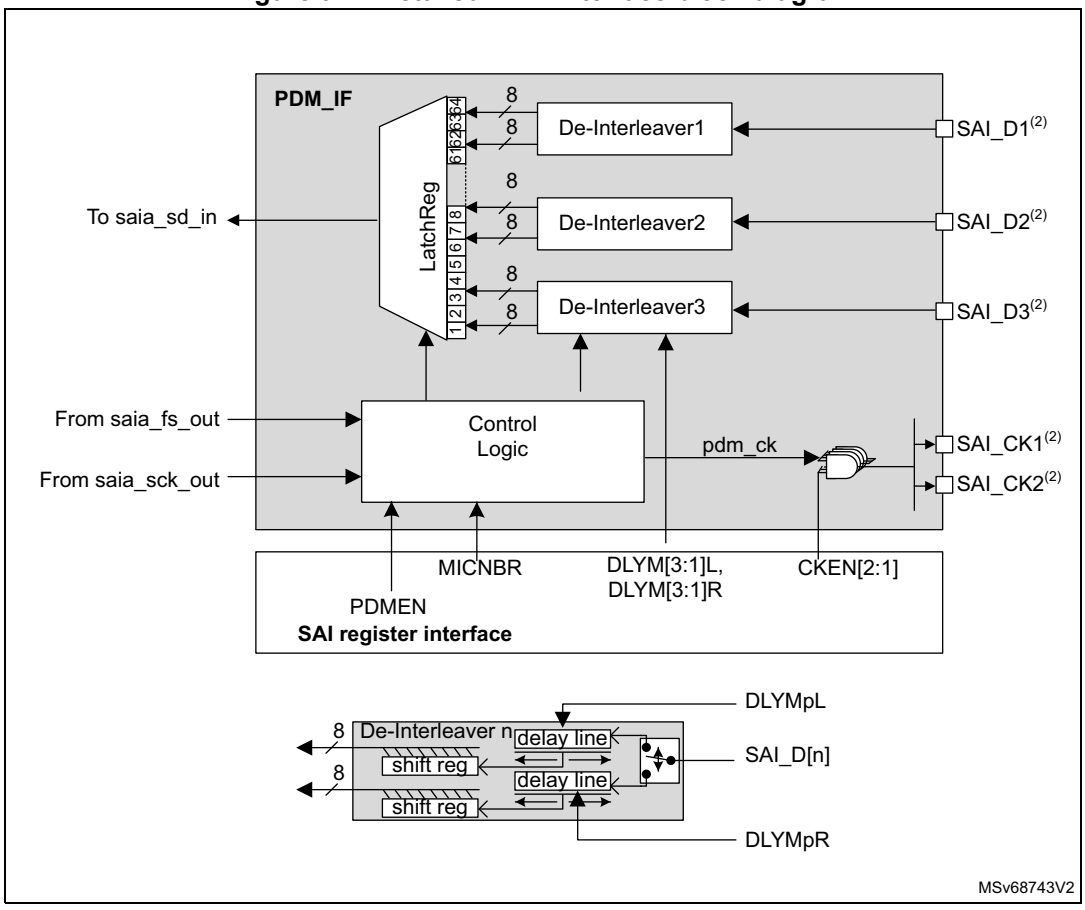
The processing performed into the PDM interface is the following:

1. The PDM interface builds the bitstream clock from the bit clock received from the serial audio interface of SAI_A.
2. The bitstream data received from the microphones (SAI_D[n]) are de-interleaved and go through a 7-bit delay line to fine-tune the delay of each microphone with the accuracy of the bitstream clock.
3. The shift registers translate each serial bitstream into bytes.
4. The last operation consists in shifting-out the resulting bytes to SAI_A through the data line of the serial audio interface.

Figure 921 below shows the block diagram of the PDM interface, with a detailed view of a de-interleaver.

Note: The PDM interface does not embed the decimation filter required to build-up the PCM audio samples from the bitstream. It is up to the application software to perform this operation.

Figure 921. Detailed PDM interface block diagram



1. **n** refers to the number of data lines and **p** to the number of microphone pairs.
2. These signals might not be available in all SAI instances. Refer to *Section 68.3: SAI implementation* for details.

The PDM interface can be enabled through the PDMEN bit in the SAI_PDMCR register. However, the PDM interface must be enabled prior to enabling the SAI_A block.

To reduce the memory footprint, the user can select the number of microphones the application needs. This can be done through the MICNBR[1:0] bits. It is possible to choose between 2, 4, 6, or 8 microphones. For example, if the application is using 3 microphones, the user has to select 4.

Enabling the PDM interface

To enable the PDM interface, follow the sequence below:

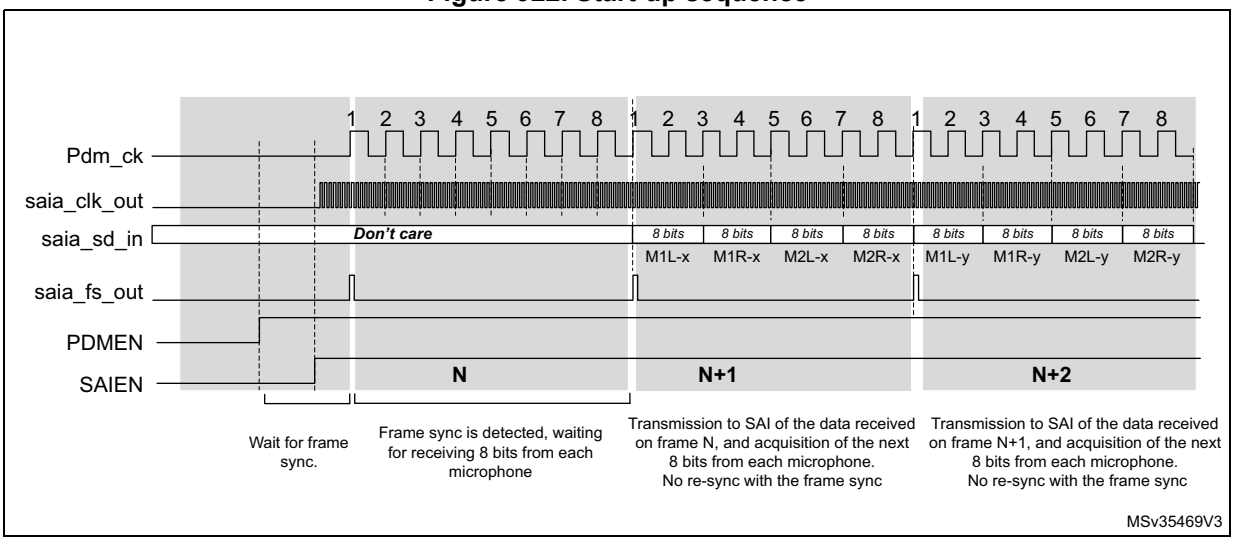
1. Configure SAI_A in TDM master mode (see *Table 674*).
2. Configure the PDM interface as follows:
   a) Define the number of digital microphones via MICNBR.
   b) Enable the bitstream clock needed in the application by setting the corresponding bits on CKEN to 1.
3. Enable the PDM interface, via the PDMEN bit.
4. Enable the SAI_A.

Note: Once the PDM interface and SAI_A are enabled, the first two frames received on SAI_ADR are invalid and must be dropped.

Startup sequence

Figure 922 shows the startup sequence: Once the PDM interface is enabled, it waits for the frame synchronization event prior to starting the acquisition of the microphone samples. After 8 SAI_CK clock periods, a data byte coming from each microphone is available, and transferred to the SAI, via the serial audio interface.

Figure 922. Start-up sequence



SAI_ADR data format

The arrangement of the data coming from the microphone into the SAI_ADR register depends on the following parameters:

- Number of microphones
- Slot width selected
- LSBFIRST bit

The slot width defines the number of significant bits into each word available into the SAI_ADR.

When a slot width of 32 bits is selected, each data available into the SAI_ADR contains 32 useful bits. This reduces the number of words stored into the memory. However, the counterpart is that the software has to perform some operations to de-interleave the data of each microphone.

On the other hand, when the slot width is set to 8 bits, each data available into the SAI_ADR contains 8 useful bits. This increases the number of words stored in the memory. However, it offers the advantage of avoiding extra processing since each word contains information from one microphone.

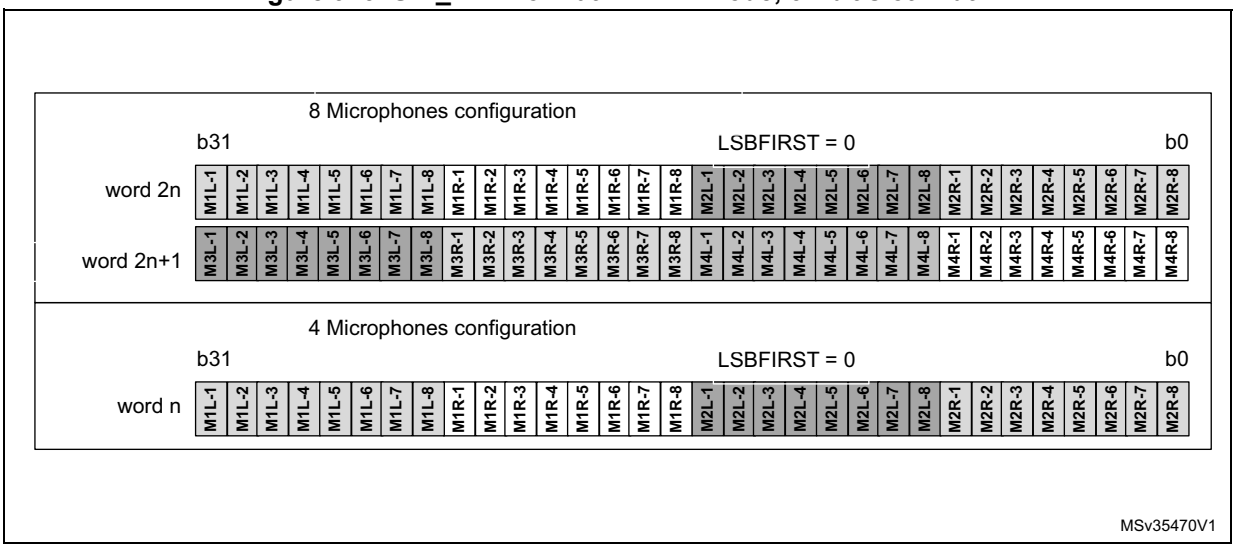
SAI_ADR data format example

- **32-bit slot width** (DS = 0b111 and SLOTSZ = 0). Refer to *Figure 923*.

  For an 8 microphone configuration, two consecutive words read from the SAI_ADR register contain a data byte from each microphone.

  For a 4 microphones configuration, each word read from the SAI_ADR register contains a data byte from each microphone.

Figure 923. SAI_ADR format in TDM mode, 32-bit slot width

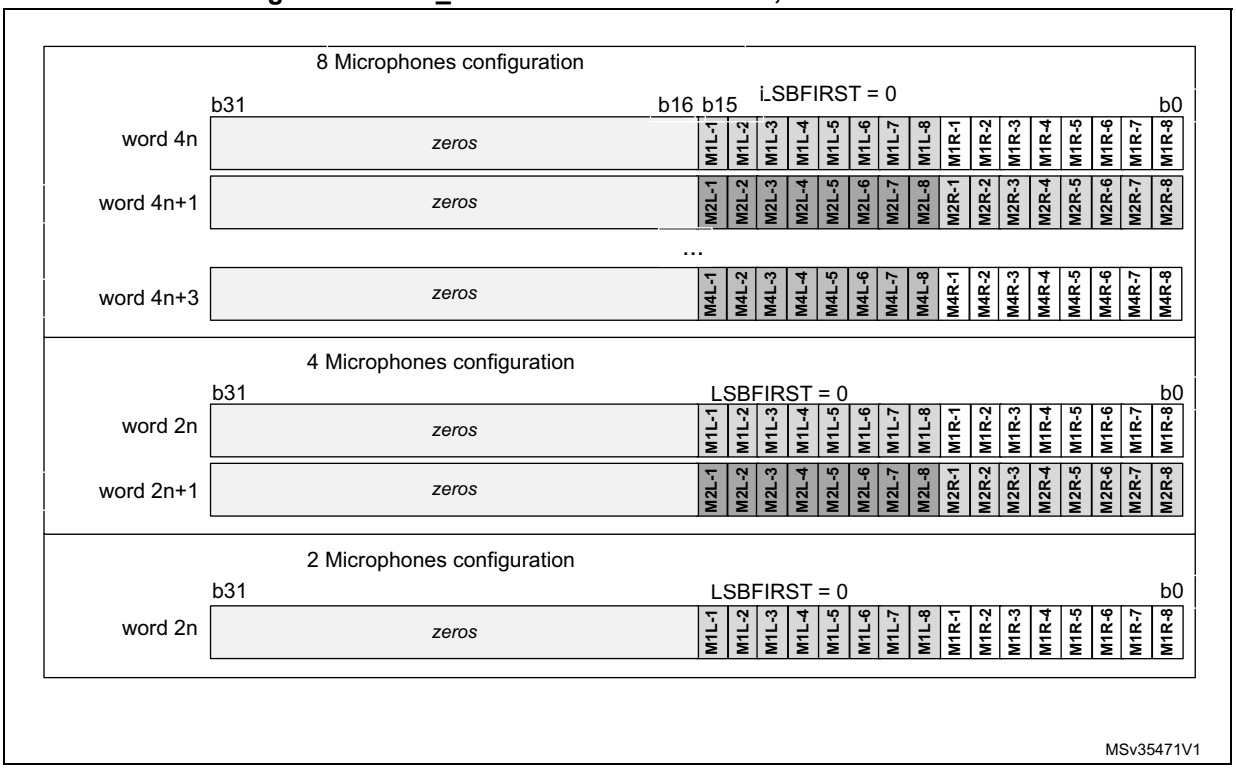


- **16-bit slot width** (DS = 0b100 and SLOTSZ = 0). Refer to *Figure 924*.

  For an 8-microphone configuration, four consecutive words read from the SAI_ADR register contain a data byte from each microphone. Note that the 16-bit data of SAI_ADR are right-aligned.

  For a 4- or 2-microphone configuration, the SAI behavior is similar to the 8-microphone configuration. Up to 2 words of 16 bits are required to acquire a byte from 4 microphones and a single word for 2 microphones.

Figure 924. SAI_ADR format in TDM mode, 16-bit slot width

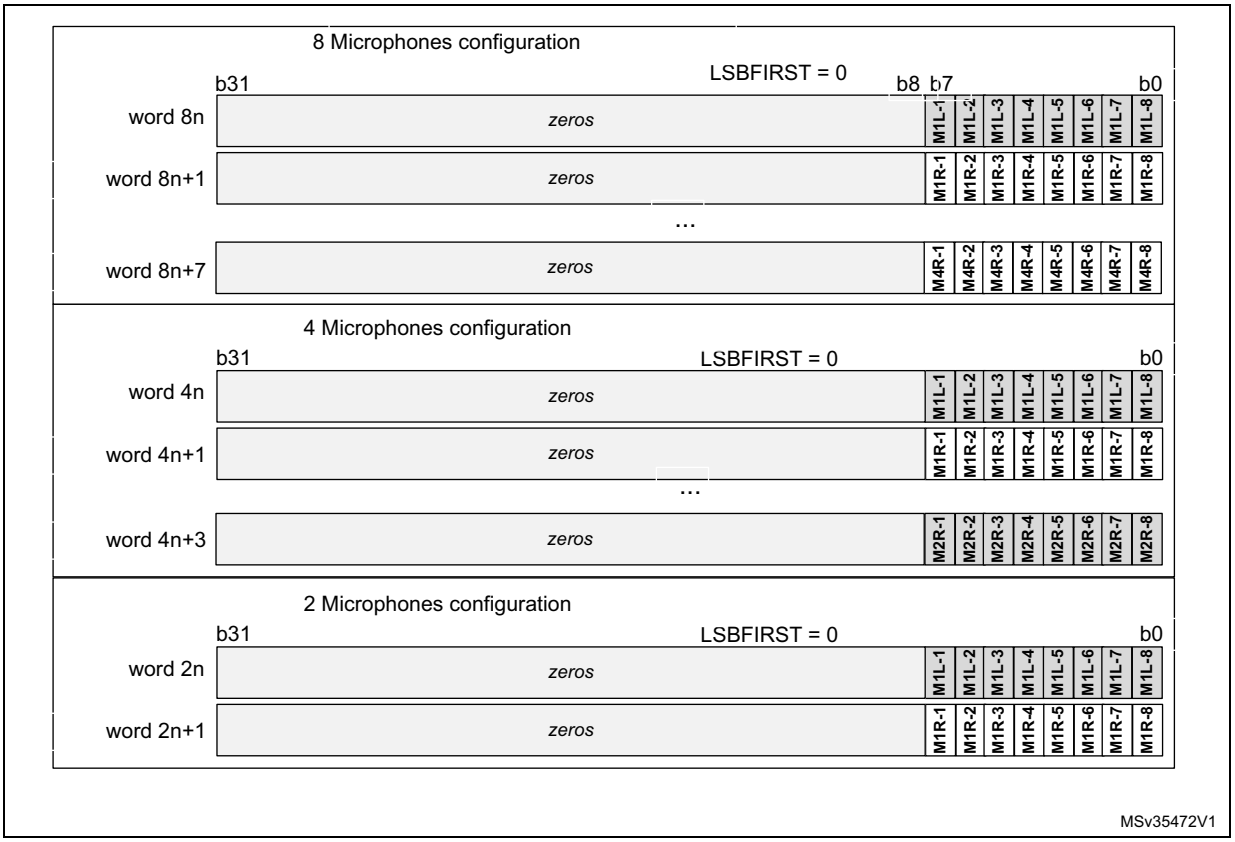


- **Using an 8-bit slot width** (DS = 0b010 and SLOTSZ = 0). Refer to *Figure 925*.

  For an 8-microphone configuration, eight consecutive words read from the SAI_ADR register contain a byte of data from each microphone. Note that the 8-bit data of SAI_ADR are right-aligned.

  For a 4- or 2-microphone configuration, the SAI behavior is similar to the 8-microphone configuration. Up to four words of eight bits are required to acquire a byte from four microphones and two words from two microphones.

Figure 925. SAI_ADR format in TDM mode, 8-bit slot width



TDM mode configuration for PDM interface

The SAI_A serial audio interface is internally connected to the PDM interface to get the microphone samples. The user application must configure the SAI_A serial audio interface as shown in *Table 674* to ensure a good connection with the PDM interface.

Table 674. SAI_A configuration for TDM mode

Bit Fields	Values	Comments
MODE	0b01	Mode must be MASTER receiver.
PRTCFG	0b00	Free protocol for TDM.
DS	X	To be adjusted according to the required data format, in accordance with the frame length and the number of slots (FRL and NBSLOT). See *Table 675*.
LSBFIRST	X	This parameter can be used according to the desired data format.
CKSTR	0	Signal transitions occur on the rising edge of the SCK_A bit clock. Signals are stable on the falling edge of the bit clock.
MONO	0	Stereo mode.
FRL	X	To be adjusted according to the number of microphones (MICNBR). See *Table 675*.
FSALL	0	Pulse width is one bit clock cycle.

Table 674. SAI_A configuration for TDM mode (continued)

Bit Fields	Values	Comments
FSDEF	0	FS signal is a start of frame.
FSPOL	1	FS is active high.
FSOFF	0	FS is asserted on the first bit of slot 0.
FBOFF	0	No offset on slot.
SLOTSZ	0	Slot size = data size.
NBSLOT	X	To be adjusted according to the required data format, in accordance with the slot size, and the frame length (FRL and DS). See *Table 675*.
SLOTEN	X	To be adjusted according to NBSLOT.
NODIV	1	No need to generate a master clock MCLK.
MCKDIV	X	Depends on the frequency provided to sai_a_ker_ck input. This parameter must be adjusted to generate the proper bitstream clock frequency. See *Table 675*.

Adjusting the bitstream clock rate

To program the serial audio interface properly, the user application must take into account the settings given in *Table 674*, and follow the sequence below:

1. Adjust the bit clock frequency (F_{SCK_A}) according to the required frequency for the PDM bitstream clock, using the following formula:

$$F_{SCK_A} = F_{PDM_CK} \times (MICNBR + 1) \times 2$$

   MICNBR can be 0, 1, 2 or 3 (0 = 2 microphones; see *Section 68.6.18*)

2. Set the frame length (FRL) using the following formula:

$$FRL = (16 \times (MICNBR + 1)) - 1$$

Table 675. TDM frame configuration examples(1)(2)

Microphone sampling rate	Nber of microphones	Wanted SAI_CKn frequency	bit clock (SCK_A) frequency	Frame sync. (FS_A) frequency	FRL	DS	NBSLOT	Comments
48 kHz	up to 8	3.072 MHz	24.576 MHz	384 kHz	63	0b111	1	2 slots of 32 bits per frame
		3.072 MHz	24.576 MHz	384 kHz	63	0b100	3	4 slots of 16 bits per frame
		3.072 MHz	24.576 MHz	384 kHz	63	0b010	7	8 slots of 8 bits per frame
	up to 6	3.072 MHz	18.432 MHz	384 kHz	47	0b110	1	2 slots of 24 bits per frame
		3.072 MHz	18.432 MHz	384 kHz	47	0b100	2	3 slots of 16 bits per frame
		3.072 MHz	18.432 MHz	384 kHz	47	0b010	5	6 slots of 8 bits per frame
	up to 4	3.072 MHz	12.288 MHz	384 kHz	31	0b111	0	1 slot of 32 bits per frame
		3.072 MHz	12.288 MHz	384 kHz	31	0b100	1	2 slots of 16 bits per frame
		3.072 MHz	12.288 MHz	384 kHz	31	0b010	3	4 slots of 8 bits per frame
	up to 2	3.072 MHz	6.144 MHz	384 kHz	15	0b100	0	1 slots of 16 bits per frame
		3.072 MHz	6.144 MHz	384 kHz	15	0b010	1	2 slots of 8 bits per frame
16 kHz	up to 8	1.024 MHz	8.192 MHz	128 kHz	63	0b111	1	2 slots of 32 bits per frame
		1.024 MHz	8.192 MHz	128 kHz	63	0b100	3	4 slots of 16 bits per frame
		1.024 MHz	8.192 MHz	128 kHz	63	0b010	7	8 slots of 8 bits per frame
	up to 6	1.024 MHz	6.144 MHz	128 kHz	47	0b110	1	2 slots of 24 bits per frame
		1.024 MHz	6.144 MHz	128 kHz	47	0b010	5	6 slots of 8 bits per frame
	up to 4	1.024 MHz	4.096 MHz	128 kHz	31	0b111	0	1 slot of 32 bits per frame
		1.024 MHz	4.096 MHz	128 kHz	31	0b100	1	2 slots of 16 bits per frame
		1.024 MHz	4.096 MHz	128 kHz	31	0b010	3	4 slots of 8 bits per frame
	up to 2	1.024 MHz	2.048 MHz	128 kHz	15	0b100	0	1 slot of 16 bits per frame
		1.024 MHz	2.048 MHz	128 kHz	15	0b010	1	2 slots of 8 bits per frame

1. Refer to *Table 674: SAI_A configuration for TDM mode* for additional information on TDM configuration. The sai_a_ker_ck clock frequency provided to the SAI must be a multiple of the SCK_A frequency, and MCKDIV must be programmed accordingly.
2. The above sai_a_ker_ck frequencies are given as examples only. Refer to section *Reset and clock controller (RCC) to check if they can be generated on the device.*
3. The table above gives allowed settings for a decimation ratio of 64.

Adjusting the delay lines

When the PDM interface is enabled, the application can adjust on-the-fly the delay cells of each microphone input via the SAI_PDMDLY register.

The new delay values become effective after two audio frames.

68.4.11 AC’97 link controller

The SAI is able to work as an AC’97 link controller. In this protocol:

- The slot number and the slot size are fixed.
- The frame synchronization signal is perfectly defined and has a fixed shape.

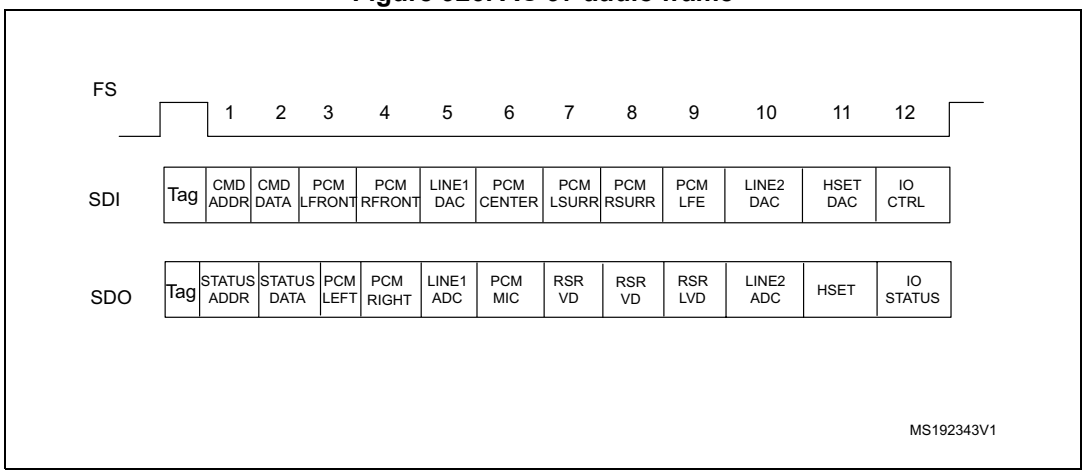
To select this protocol, set the PRTCFG[1:0] bits in the SAI_xCR1 register to 10. When AC’97 mode is selected, only data sizes of 16 or 20 bits can be used, otherwise the SAI behavior is not guaranteed.

- The NBSLOT[3:0] and SLOTSZ[1:0] bits are consequently ignored.
- The number of slots is fixed at 13 slots. The first one is 16 bits wide and all the others are 20 bits wide (data slots).
- The FBOFF[4:0] bits in the SAI_xSLOTR register are ignored.
- The SAI_xFRCR register is ignored.
- The MCLK is not used.

The FS signal from the block defined as asynchronous is configured automatically as an output, since the AC’97 controller link drives the FS signal whatever the master or slave configuration.

Figure 926 shows an AC’97 audio frame structure.

Figure 926. AC’97 audio frame



Note: In the AC’97 protocol, bit 2 of the tag is reserved (always 0), so bit 2 of the TAG is forced to 0 level whatever the value written in the SAI FIFO.

For more details about tag representation, refer to the AC’97 protocol standard.

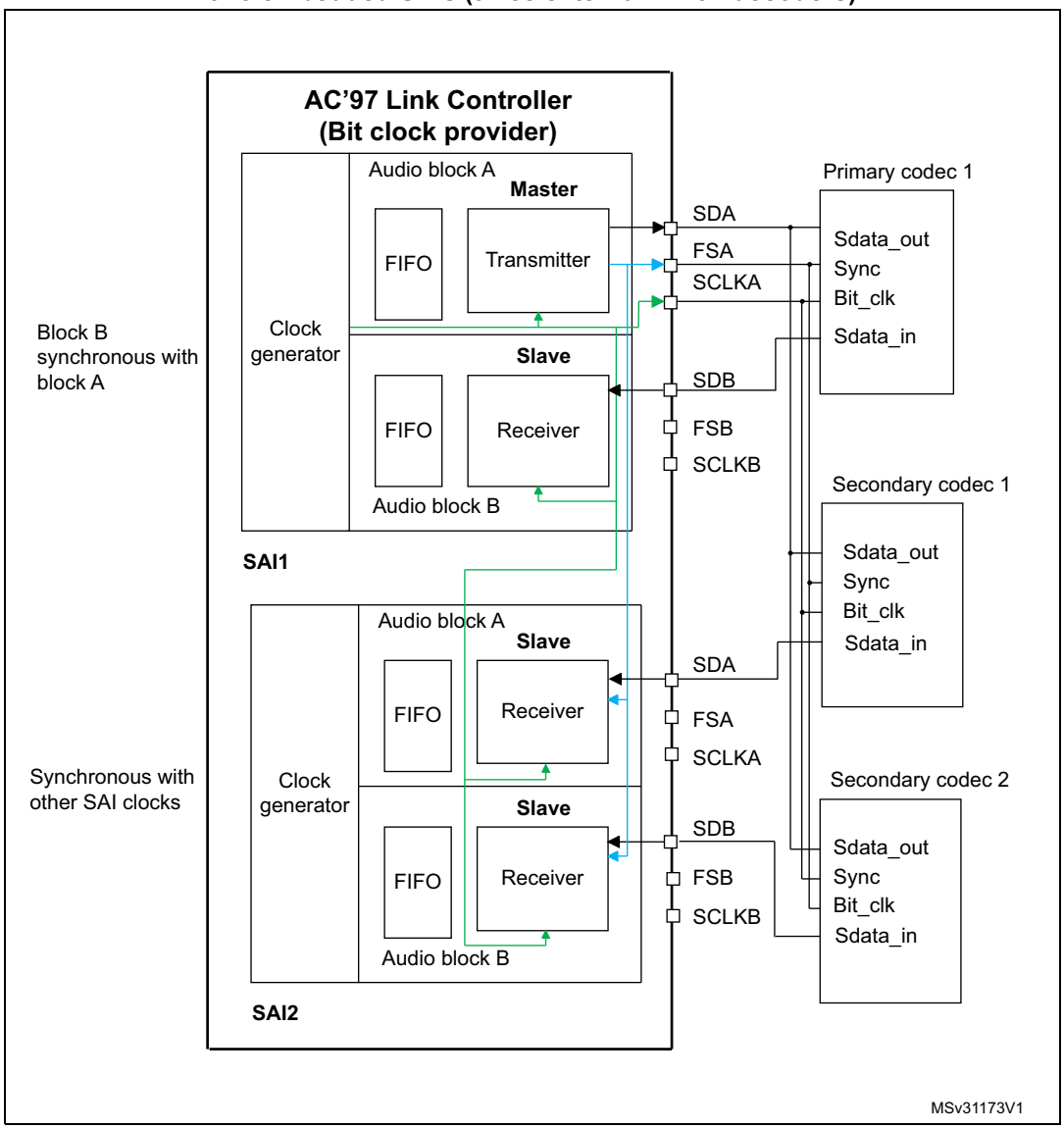
One SAI can be used to target an AC’97 point-to-point communication.

Using two SAIs (for devices featuring two embedded SAIs) enables the control of three external AC’97 decoders as illustrated in *Figure 927*.

In SAI1, the audio block A must be declared as asynchronous master transmitter, whereas the audio block B is defined to be slave receiver and internally synchronous to the audio block A.

The SAI2 is configured for audio block A and B both synchronous with the external SAI1 in slave receiver mode.

Figure 927. Example of typical AC'97 configuration on devices featuring at least two embedded SAIs (three external AC'97 decoders)



In receiver mode, the SAI acting as an AC'97 link controller requires no FIFO request and so no data storage in the FIFO when the codec-ready bit in slot 0 is decoded low. If bit CNRDYIE is enabled in the SAI_xIM register, flag CNRDY is set in the SAI_xSR register and an interrupt is generated. This flag is dedicated to the AC'97 protocol.

Clock generator programming in AC'97 mode

In AC'97 mode, the frame length is fixed at 256 bits, and its frequency must be set to 48 kHz. The formulas given in *Section 68.4.8: SAI clock generator* must be used with FRL = 255, to generate the proper frame rate (F_{FS_x}).

68.4.12 SPDIF output

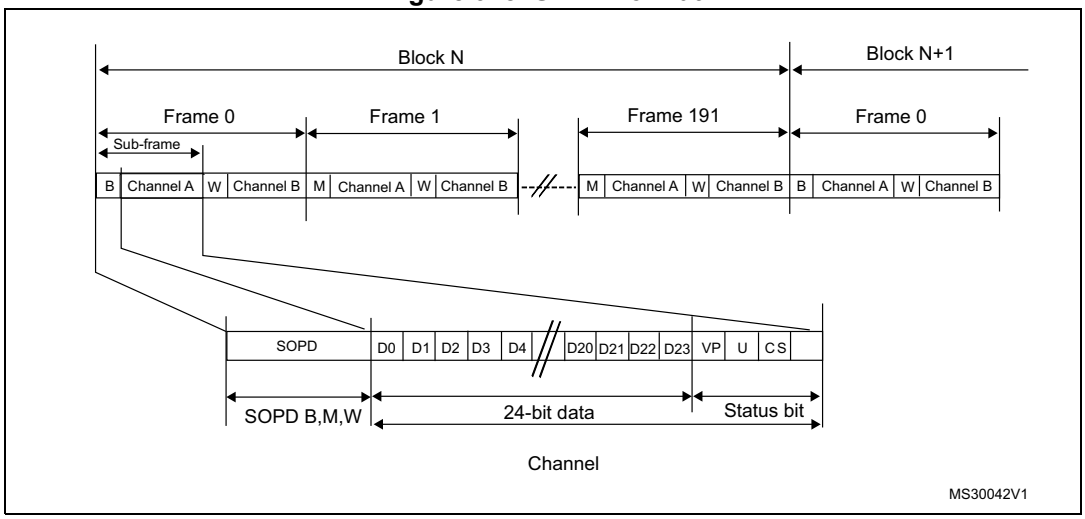
The SPDIF interface is available in transmitter mode only. It supports the audio IEC60958.

To select SPDIF mode, set the PRTCFG[1:0] bits to 01 in the SAI_xCR1 register.

For SPDIF protocol:

- Only the SD data line is enabled.
- The FS, SCK, and MCLK I/Os pins are left free.
- The MODE[1] bit is forced to 0 to select the master mode to enable the clock generator of the SAI and manage the data rate on the SD line.
- The data size is forced to 24 bits. The value set in the DS[2:0] bits in the SAI_xCR1 register is ignored.
- The clock generator must be configured to define the symbol-rate, knowing that the bit clock must be twice the symbol-rate. The data is coded in Manchester protocol.
- The SAI_xFRCR and SAI_xSLOTR registers are ignored. The SAI is configured internally to match the SPDIF protocol requirements as shown in *Figure 928*.

Figure 928. SPDIF format



An SPDIF block contains 192 frames. Each frame is composed of two 32-bit subframes, generally one for the left channel and one for the right channel. Each subframe is composed of a SOPD pattern (4-bit) to specify if the subframe is the start of a block (and so is identifying a channel A) or if it is identifying a channel A somewhere in the block, or if it is referring to channel B (see *Table 676*). The next 28 bits of channel information are composed of 24 data bits + 4 status bits.

Table 676. SOPD pattern

SOPD	Preamble coding		Description
	last bit is 0	last bit is 1	
B	11101000	00010111	Channel A data at the start of block
W	11100100	00011011	Channel B data somewhere in the block
M	11100010	00011101	Channel A data

The data stored in SAI_xDR has to be filled as follows:

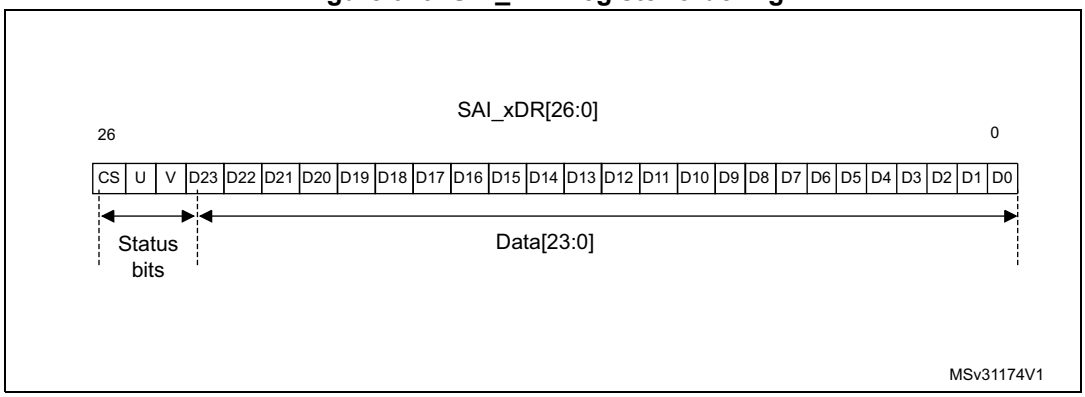
- SAI_xDR[26:24] contain the channel status, user, and validity bits.
- SAI_xDR[23:0] contain the 24-bit data for the considered channel.

If the data size is 20 bits, the data must be mapped on SAI_xDR[23:4].

If the data size is 16 bits, the data must be mapped on SAI_xDR[23:8].

SAI_xDR[23] always represents the MSB.

Figure 929. SAI_xDR register ordering



Note: The transfer is always performed with LSB first.

The SAI first sends the adequate preamble for each subframe in a block. The SAI_xDR is then sent on the SD line (Manchester coded). The SAI ends the subframe by transferring the parity bit calculated as described in *Table 677*.

Table 677. Parity bit calculation

SAI_xDR[26:0]	Parity bit P value transferred
odd number of 0	0
odd number of 1	1

The underrun is the only error flag available in the SAI_xSR register for SPDIF mode since the SAI can only operate in transmitter mode. As a result, the following sequence must be executed to recover from an underrun error detected via the underrun interrupt or the underrun status bit:

1. Disable the DMA stream (via the DMA peripheral) if the DMA is used.
2. Disable the SAI and check that the peripheral is physically disabled by polling the SAIEN bit in the SAI_xCR1 register.
3. Clear the COVRUNDR flag in the SAI_xCLRFR register.
4. Flush the FIFO by setting the FFLUSH bit in SAI_xCR2.

   The software needs to point to the address of the future data corresponding to the start of a new block (data for preamble B). If the DMA is used, the DMA source base address pointer must be updated accordingly.
5. Enable the DMA stream (DMA peripheral) again if the DMA is used to manage data transfers according to the new source base address.
6. Enable the SAI again by configuring the SAIEN bit in the SAI_xCR1 register.

Clock generator programming in SPDIF generator mode

For the SPDIF generator, the SAI provides a bit clock twice as fast as the symbol-rate. The table below shows examples of symbol rates with respect to the audio sampling rate.

Table 678. Audio sampling frequency versus symbol rates

Audio sampling frequencies (F_S)	Symbol-rate
44.1 kHz	2.8224 MHz
48 kHz	3.072 MHz
96 kHz	6.144 MHz
192 kHz	12.288 MHz

More generally, the relationship between the audio sampling frequency (F_S) and the bit clock rate (F_{SCK_X}) is given by the formula:

$$F_S = \frac{F_{SCK_x}}{128}$$

The bit clock rate is obtained as follows:

$$F_{SCK_x} = \frac{F_{sai_x_ker_ck}}{MCKDIV}$$

Note: The above formulas are valid only if NODIV is set to 1 in the SAI_ACR1 register.

68.4.13 Specific features

The SAI interface embeds specific features that can be useful depending on the audio protocol selected. These functions are accessible through specific bits of the SAI_xCR2 register.

Mute mode

The mute mode can be used when the audio subblock is a transmitter or a receiver.

Audio subblock in transmission mode

In transmitter mode, the mute mode can be selected at any time. The mute mode is active for entire audio frames. The MUTE bit in the SAI_xCR2 register enables the mute mode when it is configured during an ongoing frame.

The mute mode bit is strobed only at the end of the frame. If it is set at this time, the mute mode is active at the beginning of the new audio frame and for a complete frame, until the next end of frame. The bit is then strobed to determine if the next frame is still a mute frame.

If the number of slots set through the NBSLOT[3:0] bits in the SAI_xSLOTR register is lower than or equal to 2, it is possible to specify if the value sent in mute mode is 0 or if it is the last value of each slot. The selection is done via the MUTEVAL bit in the SAI_xCR2 register.

If the number of slots set in the NBSLOT[3:0] bits in the SAI_xSLOTR register is greater than 2, the MUTEVAL bit in the SAI_xCR2 register is meaningless as 0 values are sent on each bit on each slot.

The FIFO pointers are still incremented in mute mode. This means that data present in the FIFO and for which the mute mode is requested are discarded.

Audio subblock in reception mode

In reception mode, it is possible to detect a mute mode sent from the external transmitter when all the declared and valid slots of the audio frame receive 0 for a given consecutive number of audio frames (MUTECNT[5:0] bits in the SAI_xCR2 register).

When the number of MUTE frames is detected, the MUTEDET flag in the SAI_xSR register is set and an interrupt can be generated if the MUTEDETIE bit is set in SAI_xCR2.

The mute frame counter is cleared when the audio subblock is disabled or when a valid slot receives at least one data in an audio frame. The interrupt is generated just once, when the counter reaches the value specified in the MUTECNT[5:0] bits. The interrupt event is then reinitialized when the counter is cleared.

Note: The mute mode is not available for SPDIF audio blocks.

Mono/stereo mode

In transmitter mode, the mono mode can be addressed without any data preprocessing in memory, assuming the number of slots is equal to 2 (NBSLOT[3:0] = 0001 in SAI_xSLOTR). In this case, the access time to and from the FIFO is reduced by 2 since the data for slot 0 is duplicated into data slot 1.

To enable the mono mode:

1. Set the MONO bit to 1 in the SAI_xCR1 register.
2. Set NBSLOT to 1 and SLOTEN to 3 in SAI_xSLOTR.

In reception mode, the MONO bit can be set and is meaningful only if the number of slots is equal to 2, like in transmitter mode. When it is set, only slot 0 data are stored in the FIFO. The data belonging to slot 1 are discarded since, in this case, it is supposed to be the same as the previous slot. If the data flow in reception mode is a real stereo audio flow with a distinct and different left and right data, the MONO bit is meaningless. The conversion from the output stereo file to the equivalent mono file is done by software.

Companding mode

Telecommunication applications can require processing the data to be transmitted or received using a data companding algorithm.

Depending on the COMP[1:0] bits in the SAI_xCR2 register (used only when free protocol mode is selected), the application software can choose to process or not the data before sending it on the SD serial output line (compression) or to expand the data after the reception on the SD serial input line (expansion), as illustrated in *Figure 930*. The two companding modes supported are the µ-Law and the A-Law logs, which are a part of the CCITT G.711 recommendation.

The companding standard used in the United States and Japan is the µ-Law. It supports 14 bits of dynamic range (COMP[1:0] = 10 in the SAI_xCR2 register).

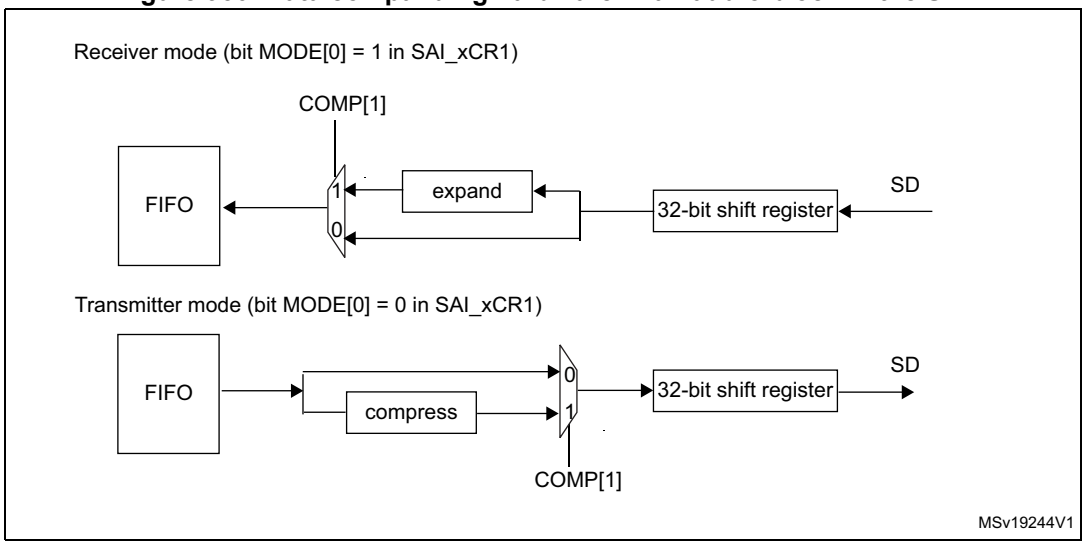
The European companding standard is A-Law and supports 13 bits of dynamic range (COMP[1:0] = 11 in the SAI_xCR2 register).

Both µ-Law and A-Law companding standard can be computed based on 1's complement or 2's complement representation, depending on the CPL bit setting in the SAI_xCR2 register.

In µ-Law and A-Law standards, data are coded as 8 bits with MSB alignment. Companded data are always 8 bits wide. For this reason, the DS[2:0] bits in the SAI_xCR1 register are forced to 010 when the SAI audio block is enabled (the SAIEN bit = 1 in the SAI_xCR1 register) and when one of these two companding modes is selected through the COMP[1:0] bits.

If no companding processing is required, the COMP[1:0] bits must be kept clear.

Figure 930. Data companding hardware in an audio block in the SAI



1. Not applicable when AC'97 or SPDIF are selected.

Expansion and compression mode are automatically selected through SAI_xCR2:

- If the SAI audio block is configured to be a transmitter, and if the COMP[1] bit is set in the SAI_xCR2 register, the compression mode is applied.
- If the SAI audio block is declared as a receiver, the expansion algorithm is applied.

Output data line management on an inactive slot

In transmitter mode, it is possible to choose the behavior of the SD line output when an inactive slot is sent on the data line (via the TRIS bit).

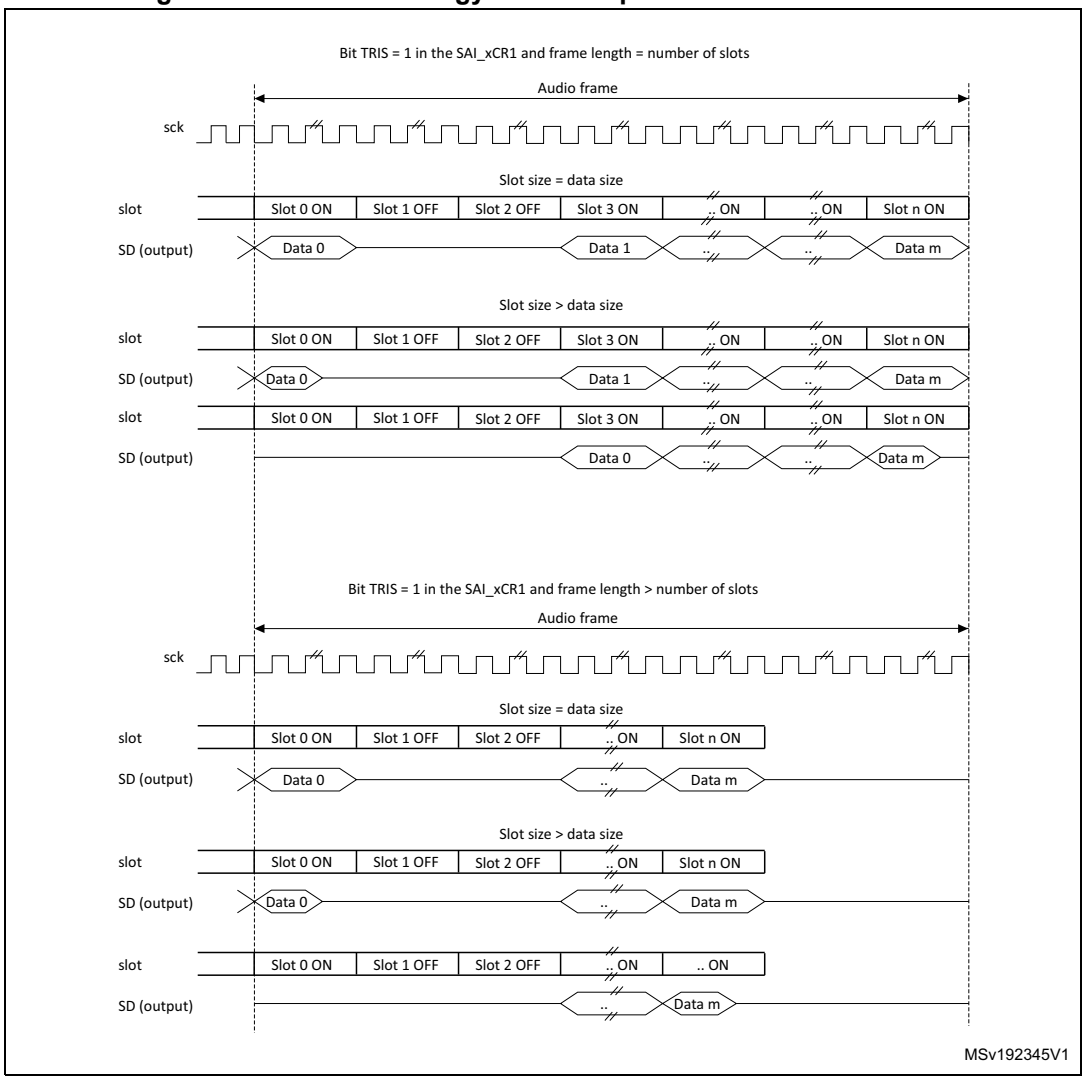
- Either the SAI forces 0 on the SD output line when an inactive slot is transmitted, or
- The line is released in high-Z state at the end of the last bit of data transferred, to release the line for other transmitters connected to this node.

It is important to note that the two transmitters cannot attempt to drive the same SD output pin simultaneously, which may result in a short circuit. To ensure a gap between transmissions, if the data is lower than 32-bit, the data can be extended to 32-bit by setting the bit SLOTSZ[1:0] = 10 in the SAI_xSLOTR register. The SD output pin is then tri-stated at the end of the LSB of the active slot (during the padding to 0 phase to extend the data to 32-bit) if the following slot is declared inactive.

In addition, if the number of slots multiplied by the slot size is lower than the frame length, the SD output line is tri-stated when the padding to 0 is done to complete the audio frame.

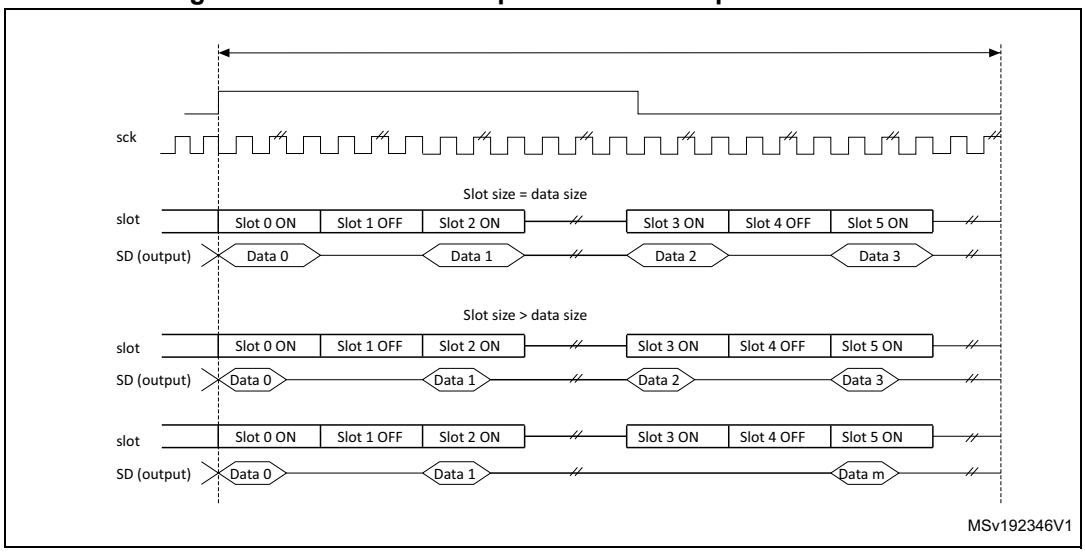
Figure 931 illustrates these behaviors.

Figure 931. Tristate strategy on SD output line on an inactive slot



When the selected audio protocol uses the FS signal as a start of frame and a channel side identification (bit FSDEF = 1 in the SAI_xFRCR register), the tristate mode is managed according to *Figure 932* (where the bit TRIS in the SAI_xCR1 register = 1, and FSDEF=1, and half frame length is higher than number of slots/2, and NBSLOT=6).

Figure 932. Tristate on output data line in a protocol like I2S



If the TRIS bit in the SAI_xCR2 register is cleared, all the high impedance states on the SD output line in *Figure 931* and *Figure 932* are replaced by a drive with a value of 0.

68.4.14 Error flags

The SAI implements the following error flags:

- FIFO overrun/underrun.
- Anticipated frame synchronization detection.
- Late frame synchronization detection.
- Codec not ready (AC'97 exclusively).
- Wrong clock configuration in master mode.

FIFO overrun/underrun (OVRUDR)

The FIFO overrun/underrun bit is called OVRUDR in the SAI_xSR register.

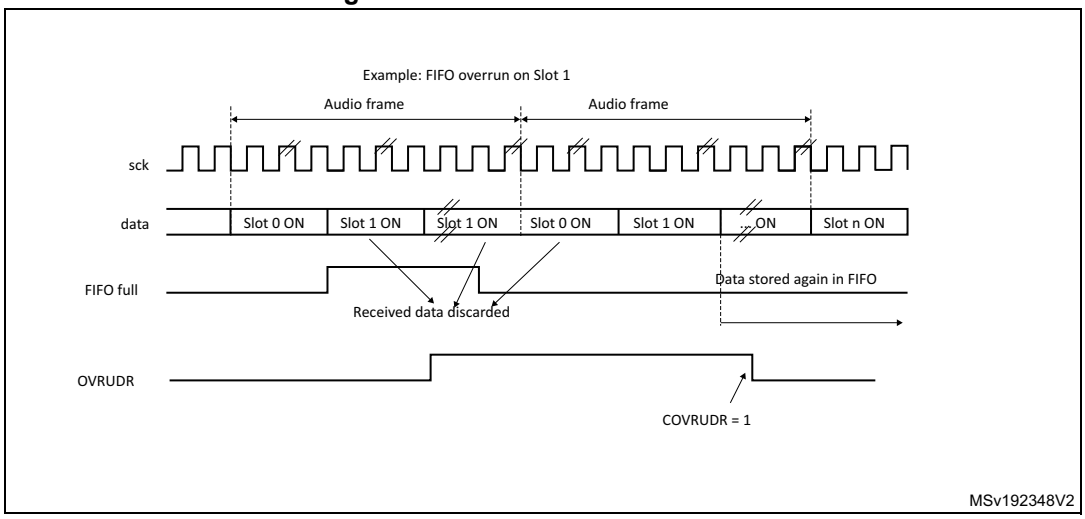
The overrun or underrun errors share the same bit since an audio block can be either receiver or transmitter and each audio block in a given SAI has its own SAI_xSR register.

Overrun

When the audio block is configured as receiver, an overrun condition may appear if data are received in an audio frame when the FIFO is full and not able to store the received data. In this case, the received data are lost, the OVRUDR flag in the SAI_xSR register is set, and an interrupt is generated if the OVRUDRIE bit is set in the SAI_xIM register. The slot number, from which the overrun occurs, is stored internally. No more data are stored into the FIFO until it becomes free to store new data. When the FIFO has at least one data free, the SAI audio block receiver stores new data (from a new audio frame) from the slot number that was stored internally when the overrun condition was detected. This avoids data slot dealignment in the destination memory (refer to *Figure 933*).

The OVRUDR flag is cleared when the COVRUDR bit is set in the SAI_xCLRFR register.

Figure 933. Overrun detection error



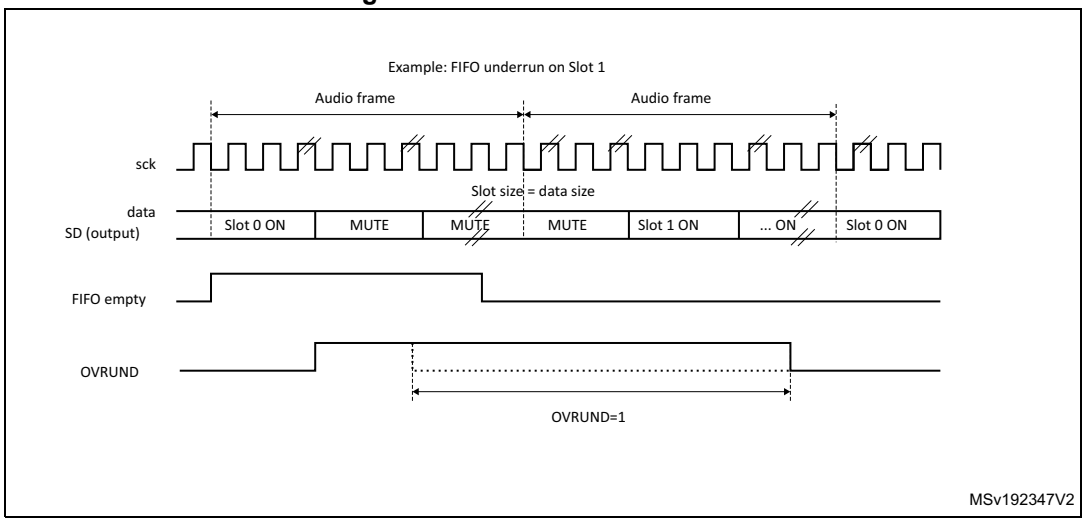
Underrun

An underrun may occur when the audio block in the SAI is a transmitter and the FIFO is empty when data need to be transmitted. If an underrun is detected, the slot number for which the event occurs is stored and the MUTE value (00) is sent until the FIFO is ready to transmit the data corresponding to the slot for which the underrun was detected (refer to *Figure 934*). This avoids desynchronization between the memory pointer and the slot in the audio frame.

The underrun event sets the OVRUDR flag in the SAI_xSR register and an interrupt is generated if the OVRUDRIE bit is set in the SAI_xIM register. To clear this flag, set the COVRUDR bit in the SAI_xCLRFR register.

The underrun event can occur when the audio subblock is configured as master or slave.

Figure 934. FIFO underrun event



Anticipated frame synchronization detection (AFSDET)

The AFSDET flag is used only in slave mode. It is never asserted in master mode. It indicates that a frame synchronization (FS) has been detected earlier than expected since the frame length, the frame polarity, and the frame offset are defined and known.

Anticipated frame detection sets the AFSDET flag in the SAI_xSR register.

This detection has no effect on the current audio frame, which is not sensitive to the anticipated FS. This means that “parasitic” events on signal FS are flagged without any perturbation of the current audio frame.

An interrupt is generated if the AFSDETIE bit is set in the SAI_xIM register. To clear the AFSDET flag, the CAFSDET bit must be set in the SAI_xCLRFR register.

To resynchronize with the master after an anticipated frame detection error, four steps are required:

1. Disable the SAI block by resetting the SAIEN bit in the SAI_xCR1 register. To make sure that the SAI is disabled, read back the SAIEN bit and check it is set to 0.
2. Flush the FIFO via the FFLUS bit in the SAI_xCR2 register.
3. Enable the SAI peripheral again (SAIEN bit set to 1).
4. The SAI block waits for the assertion on FS to restart the synchronization with master.

Note: The AFSDET flag is not asserted in AC’97 mode since the SAI audio block acts as a link controller and generates the FS signal even when declared as slave.It has no meaning in SPDIF mode since the FS signal is not used.

Late frame synchronization detection

The LFSDET flag in the SAI_xSR register can be set only when the SAI audio block operates as a slave. The frame length, the frame polarity, and the frame-offset configuration are known in register SAI_xFRCR.

If the external master does not send the FS signal at the expected time, thus generating the signal too late, the LFSDET flag is set and an interrupt is generated if the LFSDETIE bit is set in the SAI_xIM register.

The LFSDET flag is cleared when the CLFSDET bit is set in the SAI_xCLRFR register.

The late frame synchronization detection flag is set when the corresponding error is detected. The SAI needs to be resynchronized with the master (see sequence described in *Anticipated frame synchronization detection (AFSDET)*).

In a noisy environment, glitches on the SCK clock may be wrongly detected by the audio block state machine and shift the SAI data at a wrong frame position. This event can be detected by the SAI and reported as a late frame synchronization detection error.

There is no corruption if the external master is not managing the audio data frame transfer in continuous mode, which must not be the case in most applications. In this case, the LFSDET flag is set.

Note: The LFSDET flag is not asserted in AC’97 mode since the SAI audio block acts as a link controller and generates the FS signal even when declared as slave.It has no meaning in SPDIF mode since the signal FS is not used by the protocol.

Codec not ready (CNRDY AC'97)

The CNRDY flag in the SAI_xSR register is relevant only if the SAI audio block is configured to operate in AC'97 mode (PRTCFG[1:0] = 10 in the SAI_xCR1 register). If the CNRDYIE bit is set in the SAI_xIM register, an interrupt is generated when the CNRDY flag is set.

CNRDY is asserted when the codec is not ready to communicate during the reception of the TAG 0 (slot 0) of the AC'97 audio frame. In this case, no data are automatically stored into the FIFO since the codec is not ready, until the TAG 0 indicates that the codec is ready. All the active slots defined in the SAI_xSLOTR register are captured when the codec is ready.

To clear the CNRDY flag, the CCNRDY bit must be set in the SAI_xCLRFR register.

Wrong clock configuration in master mode (with NODIV = 0)

When the audio block operates as a master (MODE[1] = 0) and the NODIV bit is equal to 0, the WCKCFG flag is set as soon as the SAI is enabled if the following conditions are met:

- (FRL+1) is not a power of 2, and
- (FRL+1) is not between 8 and 256.

The MODE, NODIV, and SAIEN bits belong to the SAI_xCR1 register and FRL to the SAI_xFRCR register.

If the WCKCFGIE bit is set, an interrupt is generated when the WCKCFG flag is set in the SAI_xSR register. To clear this flag, set the CWCKCFG bit in the SAI_xCLRFR register.

When the WCKCFG bit is set, the audio block is automatically disabled, thus performing a hardware clear of the SAIEN bit.

68.4.15 Disabling the SAI

The SAI audio block can be disabled at any moment by clearing the SAIEN bit in the SAI_xCR1 register. All the already started frames are automatically completed before the SAI stops working. The SAIEN bit remains high until the SAI is completely switched off at the end of the current audio frame transfer.

If an audio block in the SAI operates synchronously with the other one, the one that is the master must be disabled first.

68.4.16 SAI DMA interface

To free the CPU and to optimize bus bandwidth, each SAI audio block has an independent DMA interface to read/write from/to the SAI_xDR register (to access the internal FIFO). There is one DMA channel per audio subblock supporting the basic DMA request/acknowledge protocol.

To configure the audio subblock for DMA transfer, set the DMAEN bit in the SAI_xCR1 register. The DMA request is managed directly by the FIFO controller depending on the FIFO threshold level (for more details refer to *Section 68.4.9: Internal FIFOs*). The DMA transfer direction is linked to the SAI audio subblock configuration:

- If the audio block operates as a transmitter, the audio block FIFO controller outputs a DMA request to load the FIFO with data written in the SAI_xDR register.
- If the audio block operates as a receiver, the DMA request is related to read operations from the SAI_xDR register.

Follow the sequence below to configure the SAI interface in DMA mode:

1. Configure the SAI and FIFO threshold levels to specify when the DMA request is launched.
2. Configure the SAI DMA channel.
3. Enable the DMA.
4. Enable the SAI interface.

68.5 SAI interrupts

The SAI supports 7 interrupt sources, as shown in *Table 679*.

Table 679. SAI interrupt sources

Interrupt acronym	Interrupt source	Interrupt group	Audio block mode	Interrupt enable	Interrupt clear
SAI	FREQ	FREQ	Master or slave Receiver or transmitter	FREQIE in SAI_xIM register	Depends on: – FIFO threshold setting (FLVL bits in SAI_xCR2) – Communication direction (transmitter or receiver) For more details refer to *Section 68.4.9: Internal FIFOs*
	OVRUDR	ERROR	Master or slave Receiver or transmitter	OVRUDRIE in SAI_xIM register	COVRUDR = 1 in SAI_xCLRFR register
	AFSDET	ERROR	Slave (not used in AC'97 mode and SPDIF mode)	AFSDETIE in SAI_xIM register	CAFSDET = 1 in SAI_xCLRFR register
	LFSDET	ERROR	Slave (not used in AC'97 mode and SPDIF mode)	LFSDETIE in SAI_xIM register	CLFSDET = 1 in SAI_xCLRFR register
	CNRDY	ERROR	Slave (only in AC'97 mode)	CNRDYIE in SAI_xIM register	CCNRDY = 1 in SAI_xCLRFR register
	MUTEDET	MUTE	Master or slave Receiver mode only	MUTEDETIE in SAI_xIM register	CMUTEDET = 1 in SAI_xCLRFR register
	WCKCFG	ERROR	Master with NODIV = 0 in SAI_xCR1 register	WCKCFGIE in SAI_xIM register	CWCKCFG = 1 in SAI_xCLRFR register

Follow the sequence below to enable an interrupt:

1. Disable SAI interrupt.
2. Configure SAI.
3. Configure SAI interrupt source.
4. Enable SAI.

68.6 SAI registers

The peripheral registers have to be accessed by words (32 bits).

68.6.1 SAI global configuration register (SAI_GCR)

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYNCOUT[1:0]		Res.	Res.	SYNCIN[1:0]	
										rw	rw			rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:4 **SYNCOUT[1:0]**: Synchronization outputs

These bits are set and cleared by software.

00: No synchronization output signals. SYNCOUT[1:0] must be configured as “No synchronization output signals” when audio block is configured as SPDIF

01: Block A used for further synchronization for others SAI

10: Block B used for further synchronization for others SAI

11: Reserved. These bits must be set when both audio block (A and B) are disabled.

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **SYNCIN[1:0]**: Synchronization inputs

These bits are set and cleared by software.

Refer to *Table 671: External synchronization selection* for information on how to program this field.

These bits must be set when both audio blocks (A and B) are disabled.

They are meaningful if one of the two audio blocks is defined to operate in synchronous mode with an external SAI (SYNCEN[1:0] = 10 in SAI_ACR1 or in SAI_BCR1 registers).

68.6.2 SAI configuration register 1 (SAI_ACR1)

Address offset: 0x04

Reset value: 0x0000 0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	MCKEN	OSR	MCKDIV[5:0]						NODIV	Res.	DMAEN	SAIEN
				rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	OUTDRIV	MONO	SYNCEN[1:0]		CKSTR	LSBFIRST	DS[2:0]			Res.	PRTCFG[1:0]		MODE[1:0]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **MCKEN:** Master clock generation enable
0: The master clock is not generated
1: The master clock is generated independently of SAIEN bit

Bit 26 **OSR:** Oversampling ratio for master clock
This bit is meaningful only when NODIV bit is set to 0.
0: Master clock frequency = F_{FS} x 256
1: Master clock frequency = F_{FS} x 512

Bits 25:20 **MCKDIV[5:0]**: Master clock divider
These bits are set and cleared by software.
000000: Divides by 1 the kernel clock input (sai_x_ker_ck).
Otherwise, The master clock frequency is calculated according to the formula given in *Section 68.4.8: SAI clock generator*.
These bits have no meaning when the audio block is slave.
They have to be configured when the audio block is disabled.

Bit 19 **NODIV:** No divider
This bit is set and cleared by software.
0: the ratio between the Master clock generator and frame synchronization is fixed to 256 or 512
1: the ratio between the Master clock generator and frame synchronization depends on FRL[7:0]

Bit 18 Reserved, must be kept at reset value.

Bit 17 **DMAEN**: DMA enable
This bit is set and cleared by software.
0: DMA disabled
1: DMA enabled
Note: Since the audio block defaults to operate as a transmitter after reset, the MODE[1:0] bits must be configured before setting DMAEN to avoid a DMA request in receiver mode.

Bit 16 **SAIEN**: Audio block enable
This bit is set by software.
To switch off the audio block, the application software must program this bit to 0 and poll the bit till it reads back 0, meaning that the block is completely disabled. Before setting this bit to 1, check that it is set to 0, otherwise the enable command is not taken into account.
This bit enables to control the state of the SAI audio block. If it is disabled when an audio frame transfer is ongoing, the ongoing transfer completes and the cell is fully disabled at the end of this audio frame transfer.
0: SAI audio block disabled
1: SAI audio block enabled.
Note: When the SAI block (A or B) is configured in master mode, the clock must be present on the SAI block input before setting SAIEN bit.

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **OUTDRIV**: Output drive
This bit is set and cleared by software.
0: Audio block output driven when SAIEN is set
1: Audio block output driven immediately after the setting of this bit.
Note: This bit has to be set before enabling the audio block and after the audio block configuration.

Bit 12 **MONO**: Mono mode

This bit is set and cleared by software. It is meaningful only when the number of slots is equal to 2. When the mono mode is selected, slot 0 data are duplicated on slot 1 when the audio block operates as a transmitter. In reception mode, the slot1 is discarded and only the data received from slot 0 are stored. Refer to *Section : Mono/stereo mode* for more details.

0: Stereo mode

1: Mono mode.

Bits 11:10 **SYNCEN[1:0]:** Synchronization enable

These bits are set and cleared by software. They must be configured when the audio subblock is disabled.

00: audio subblock in asynchronous mode.

01: audio subblock is synchronous with the other internal audio subblock. In this case, the audio subblock must be configured in slave mode

10: audio subblock is synchronous with an external SAI embedded peripheral. In this case the audio subblock must be configured in Slave mode.

11: Reserved

Note: The audio subblock must be configured as asynchronous when SPDIF mode is enabled.

Bit 9 **CKSTR:** Clock strobing edge

This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in SPDIF audio protocol.

0: Signals generated by the SAI change on SCK rising edge, while signals received by the SAI are sampled on the SCK falling edge.

1: Signals generated by the SAI change on SCK falling edge, while signals received by the SAI are sampled on the SCK rising edge.

Bit 8 **LSBFIRST:** Least significant bit first

This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in AC'97 audio protocol since AC'97 data are always transferred with the MSB first. This bit has no meaning in SPDIF audio protocol since in SPDIF data are always transferred with LSB first.

0: Data are transferred with MSB first

1: Data are transferred with LSB first

Bits 7:5 **DS[2:0]:** Data size

These bits are set and cleared by software. These bits are ignored when the SPDIF protocols are selected (bit PRTCFG[1:0]), because the frame and the data size are fixed in such case. When the companding mode is selected through COMP[1:0] bits, DS[1:0] are ignored since the data size is fixed to 8 bits by the algorithm.

These bits must be configured when the audio block is disabled.

000: Reserved

001: Reserved

010: 8 bits

011: 10 bits

100: 16 bits

101: 20 bits

110: 24 bits

111: 32 bits

Bit 4 Reserved, must be kept at reset value.

Bits 3:2 **PRTCFG[1:0]:** Protocol configuration

These bits are set and cleared by software. These bits have to be configured when the audio block is disabled.

00: Free protocol. Free protocol enables to use the powerful configuration of the audio block to address a specific audio protocol (such as I2S, LSB/MSB justified, TDM, PCM/DSP...) by setting most of the configuration register bits as well as frame configuration register.

01: SPDIF protocol

10: AC'97 protocol

11: Reserved

Bits 1:0 **MODE[1:0]:** SAIx audio block mode

These bits are set and cleared by software. They must be configured when SAIx audio block is disabled.

00: Master transmitter

01: Master receiver

10: Slave transmitter

11: Slave receiver

Note: When the audio block is configured in SPDIF mode, the master transmitter mode is forced (MODE[1:0] = 00).

68.6.3 SAI configuration register 2 (SAI_ACR2)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP[1:0]		CPL	MUTECNT[5:0]						MUTE VAL	MUTE	TRIS	F FLUSH	FTH[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:14 **COMP[1:0]**: Companding mode.

These bits are set and cleared by software. The µ-Law and the A-Law log are a part of the CCITT G.711 recommendation, the type of complement that is used depends on *CPL bit*.

The data expansion or data compression are determined by the state of bit MODE[0].

The data compression is applied if the audio block is configured as a transmitter.

The data expansion is automatically applied when the audio block is configured as a receiver.

Refer to *Section : Companding mode* for more details.

00: No companding algorithm

01: Reserved.

10: µ-Law algorithm

11: A-Law algorithm

Note: Companding mode is applicable only when Free protocol mode is selected.

Bit 13 **CPL**: Complement bit.

This bit is set and cleared by software.

It defines the type of complement to be used for companding mode

0: 1's complement representation.

1: 2's complement representation.

Note: This bit has effect only when the companding mode is µ-Law algorithm or A-Law algorithm.

Bits 12:7 **MUTECNT[5:0]**: Mute counter.

These bits are set and cleared by software. They are used only in reception mode.

The value set in these bits is compared to the number of consecutive mute frames detected in reception. When the number of mute frames is equal to this value, the flag MUTEDET is set and an interrupt is generated if bit MUTEDETIE is set.

Refer to *Section : Mute mode* for more details.

Bit 6 **MUTEVAL**: Mute value.

This bit is set and cleared by software. It is meaningful only when the audio block operates as a transmitter, the number of slots is lower or equal to 2 and the MUTE bit is set.

If more slots are declared, the bit value sent during the transmission in mute mode is equal to 0, whatever the value of MUTEVAL.

if the number of slot is lower or equal to 2 and MUTEVAL = 1, the MUTE value transmitted for each slot is the one sent during the previous frame.

Refer to *Section : Mute mode* for more details.

0: Bit value 0 is sent during the mute mode.

1: Last values are sent during the mute mode.

Note: This bit is meaningless and must not be used for SPDIF audio blocks.

Bit 5 **MUTE:** Mute.

This bit is set and cleared by software. It is meaningful only when the audio block operates as a transmitter. The MUTE value is linked to value of MUTEVAL if the number of slots is lower or equal to 2, or equal to 0 if it is greater than 2.

Refer to *Section : Mute mode* for more details.

0: No mute mode.

1: Mute mode enabled.

Note: This bit is meaningless and must not be used for SPDIF audio blocks.

Bit 4 **TRIS**: Tristate management on data line.

This bit is set and cleared by software. It is meaningful only if the audio block is configured as a transmitter. This bit is not used when the audio block is configured in SPDIF mode. It must be configured when SAI is disabled.

Refer to *Section : Output data line management on an inactive slot* for more details.

0: SD output line is still driven by the SAI when a slot is inactive.

1: SD output line is released (HI-Z) at the end of the last data bit of the last active slot if the next one is inactive.

Bit 3 **FFLUSH:** FIFO flush.

This bit is set by software. It is always read as 0. This bit must be configured when the SAI is disabled.

0: No FIFO flush.

1: FIFO flush. Programming this bit to 1 triggers the FIFO Flush. All the internal FIFO pointers (read and write) are cleared. In this case data still present in the FIFO are lost (no more transmission or received data lost). Before flushing, SAI DMA stream/interrupt must be disabled

Bits 2:0 **FTH[2:0]:** FIFO threshold.
This bit is set and cleared by software.
000: FIFO empty
001: ¼ FIFO
010: ½ FIFO
011: ¾ FIFO
100: FIFO full
101: Reserved
110: Reserved
111: Reserved

68.6.4 SAI frame configuration register (SAI_AFRCR)

Address offset: 0x0C

Reset value: 0x0000 0007

Note: This register has no meaning in AC'97 and SPDIF audio protocol.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSOFF	FSPOL	FSDEF
													rw	rw	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	FSALL[6:0]							FRL[7:0]							
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **FSOFF**: Frame synchronization offset.
This bit is set and cleared by software. It is meaningless and is not used in AC'97 or SPDIF audio block configuration. This bit must be configured when the audio block is disabled.
0: FS is asserted on the first bit of the slot 0.
1: FS is asserted one bit before the first bit of the slot 0.

Bit 17 **FSPOL**: Frame synchronization polarity.
This bit is set and cleared by software. It is used to configure the level of the start of frame on the FS signal. It is meaningless and is not used in AC'97 or SPDIF audio block configuration.
This bit must be configured when the audio block is disabled.
0: FS is active low (falling edge)
1: FS is active high (rising edge)

Bit 16 **FSDEF**: Frame synchronization definition.
This bit is set and cleared by software.
0: FS signal is a start frame signal
1: FS signal is a start of frame signal + channel side identification
When the bit is set, the number of slots defined in the SAI_xSLOTR register has to be even. It means that half of this number of slots are dedicated to the left channel and the other slots for the right channel (e.g: this bit has to be set for I2S or MSB/LSB-justified protocols...).
This bit is meaningless and is not used in AC'97 or SPDIF audio block configuration. It must be configured when the audio block is disabled.

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **FSALL[6:0]**: Frame synchronization active level length.

These bits are set and cleared by software. They specify the length in number of bit clock (SCK) + 1 (FSALL[6:0] + 1) of the active level of the FS signal in the audio frame

These bits are meaningless and are not used in AC'97 or SPDIF audio block configuration.

They must be configured when the audio block is disabled.

Bits 7:0 **FRL[7:0]**: Frame length.

These bits are set and cleared by software. They define the audio frame length expressed in number of SCK clock cycles: the number of bits in the frame is equal to FRL[7:0] + 1.

The minimum number of bits to transfer in an audio frame must be equal to 8, otherwise the audio block behaves in an unexpected way. This is the case when the data size is 8 bits and only one slot 0 is defined in NBSLOT[4:0] of SAI_xSLOTR register (NBSLOT[3:0] = 0000).

In master mode, if the master clock (available on MCLK_x pin) is used, the frame length must be aligned with a number equal to a power of 2, ranging from 8 to 256. When the master clock is not used (NODIV = 1), it is recommended to program the frame length to an value ranging from 8 to 256.

These bits are meaningless and are not used in AC'97 or SPDIF audio block configuration. They must be configured when the audio block is disabled.

68.6.5 SAI slot register (SAI_ASLOTR)

Address offset: 0x10

Reset value: 0x0000 0000

Note: This register has no meaning in AC'97 and SPDIF audio protocol.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SLOTEN[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	NBSLOT[3:0]				SLOTSZ[1:0]		Res.	FBOFF[4:0]				
				rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:16 **SLOTEN[15:0]**: Slot enable.
These bits are set and cleared by software.
Each SLOTEN bit corresponds to a slot position from 0 to 15 (maximum 16 slots).
0: Inactive slot.
1: Active slot.
The slot must be enabled when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **NBSLOT[3:0]**: Number of slots in an audio frame.
These bits are set and cleared by software.
The value set in this bitfield represents the number of slots in the audio frame (including the number of inactive slots). The maximum number of slots is 16.
The number of slots must be even if FSDEF bit in the SAI_xFRCR register is set.
The number of slots must be configured when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.
0000: Number of slots is 1.
0001: Number of slots is 2.
...
1111: Number of slots is 16.

Bits 7:6 **SLOTSZ[1:0]**: Slot size
This bits is set and cleared by software.
The slot size must be higher or equal to the data size. If this condition is not respected, the behavior of the SAI is undetermined.
Refer to *Output data line management on an inactive slot* for information on how to drive SD line.
These bits must be set when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.
00: The slot size is equivalent to the data size (specified in DS[3:0] in the SAI_xCR1 register).
01: 16-bit
10: 32-bit
11: Reserved

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **FBOFF[4:0]**: First bit offset
These bits are set and cleared by software.
The value set in this bitfield defines the position of the first data transfer bit in the slot. It represents an offset value. In transmission mode, the bits outside the data field are forced to 0. In reception mode, the extra received bits are discarded.
These bits must be set when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.

68.6.6 SAI interrupt mask register (SAI_AIM)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LFSDETIE	AFSDETIE	CNRDYIE	FREQIE	WCKCFGIE	MUTEDETIE	OVRUDRIE
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **LFSDETIE**: Late frame synchronization detection interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the LFSDET bit is set in the SAI_xSR register.
This bit is meaningless in AC’97, SPDIF mode or when the audio block operates as a master.

Bit 5 **AFSDETIE**: Anticipated frame synchronization detection interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the AFSDET bit in the SAI_xSR register is set.
This bit is meaningless in AC’97, SPDIF mode or when the audio block operates as a master.

Bit 4 **CNRDYIE**: Codec not ready interrupt enable (AC’97).
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When the interrupt is enabled, the audio block detects in the slot 0 (tag0) of the AC’97 frame if the Codec connected to this line is ready or not. If it is not ready, the CNRDY flag in the SAI_xSR register is set and an interrupt is generated.
This bit has a meaning only if the AC’97 mode is selected through PRTCFG[1:0] bits and the audio block is operates as a receiver.

Bit 3 **FREQIE**: FIFO request interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the FREQ bit in the SAI_xSR register is set.
Since the audio block defaults to operate as a transmitter after reset, the MODE bit must be configured before setting FREQIE to avoid a parasitic interrupt in receiver mode,

Bit 2 **WCKCFGIE**: Wrong clock configuration interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
This bit is taken into account only if the audio block is configured as a master (MODE[1] = 0) and NODIV = 0.
It generates an interrupt if the WCKCFG flag in the SAI_xSR register is set.
Note: This bit is used only in Free protocol mode *and is meaningless in other modes.*

Bit 1 **MUTEDETIE**: Mute detection interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the MUTEDET bit in the SAI_xSR register is set.
This bit has a meaning only if the audio block is configured in receiver mode.

Bit 0 **OVRUDRIE**: Overrun/underrun interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the OVRUDR bit in the SAI_xSR register is set.

68.6.7 SAI status register (SAI_ASR)

Address offset: 0x18

Reset value: 0x0000 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLVL[2:0]		
													r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LFSDET	AFSDET	CNRDY	FREQ	WCKCFG	MUTEDET	OVRUDR
									r	r	r	r	r	r	r

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **FLVL[2:0]**: FIFO level threshold.
This bit is read only. The FIFO level threshold flag is managed only by hardware and its setting depends on SAI block configuration (transmitter or receiver mode).
000: FIFO empty (transmitter and receiver modes)
001: FIFO ≤ ¼ but not empty (transmitter mode), FIFO < ¼ but not empty (receiver mode)
010: ¼ < FIFO ≤ ½ (transmitter mode), ¼ ≤ FIFO < ½ (receiver mode)
011: ½ < FIFO ≤ ¾ (transmitter mode), ½ ≤ FIFO < ¾ (receiver mode)
100: ¾ < FIFO but not full (transmitter mode), ¾ ≤ FIFO but not full (receiver mode)
101: FIFO full (transmitter and receiver modes)
Others: Reserved

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **LFSDET**: Late frame synchronization detection.
This bit is read only.
0: No error.
1: Frame synchronization signal is not present at the right time.
This flag can be set only if the audio block is configured in slave mode.
It is not used in AC’97 or SPDIF mode.
It can generate an interrupt if LFSDETIE bit is set in the SAI_xIM register.
This flag is cleared when the software sets bit CLFSDET in SAI_xCLRFR register

Bit 5 **AFSDET**: Anticipated frame synchronization detection.
This bit is read only.
0: No error.
1: Frame synchronization signal is detected earlier than expected.
This flag can be set only if the audio block is configured in slave mode.
It is not used in AC'97 or SPDIF mode.
It can generate an interrupt if AFSDETIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CAFSDET bit in SAI_xCLRFR register.

Bit 4 **CNRDY**: Codec not ready.
This bit is read only.
0: External AC'97 Codec is ready
1: External AC'97 Codec is not ready
This bit is used only when the AC'97 audio protocol is selected in the SAI_xCR1 register and configured in receiver mode.
It can generate an interrupt if CNRDYIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CCNRDY bit in SAI_xCLRFR register.

Bit 3 **FREQ**: FIFO request.
This bit is read only.
0: No FIFO request.
1: FIFO request to read or to write the SAI_xDR.
The request depends on the audio block configuration:
– If the block is configured in transmission mode, the FIFO request is related to a write request operation in the SAI_xDR.
– If the block configured in reception, the FIFO request related to a read request operation from the SAI_xDR.
This flag can generate an interrupt if FREQIE bit is set in SAI_xIM register.

Bit 2 **WCKCFG**: Wrong clock configuration flag.
This bit is read only.
0: Clock configuration is correct
1: Clock configuration does not respect the rule concerning the frame length specification defined in *Section 68.4.6: Frame synchronization* (configuration of FRL[7:0] bit in the SAI_xFRCR register)
This bit is used only when the audio block operates in master mode (MODE[1] = 0) and NODIV = 0.
It can generate an interrupt if WCKCFGIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CWCKCFG bit in SAI_xCLRFR register.

Bit 1 **MUTEDET**: Mute detection.
This bit is read only.
0: No MUTE detection on the SD input line
1: MUTE value detected on the SD input line (0 value) for a specified number of consecutive audio frame
This flag is set if consecutive 0 values are received in each slot of a given audio frame and for a consecutive number of audio frames (set in the MUTECNT bit in the SAI_xCR2 register).
It can generate an interrupt if MUTEDETIE bit is set in SAI_xIM register.
This flag is cleared when the software sets bit CMUTEDET in the SAI_xCLRFR register.

Bit 0 **OVRUDR**: Overrun / underrun.
This bit is read only.
0: No overrun/underrun error.
1: Overrun/underrun error detection.
The overrun and underrun conditions can occur only when the audio block is configured as a receiver and a transmitter, respectively.
It can generate an interrupt if OVRUDRIE bit is set in SAI_xIM register.
This flag is cleared when the software sets COVRUDR bit in SAI_xCLRFR register.

68.6.8 SAI clear flag register (SAI_ACLRFR)

Address offset: 0x1C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLFSDET	CAFSDET	CCNRDY	Res.	CWCKCFG	CMUTEDET	COVRUDR
									w	w	w		w	w	w

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CLFSDET**: Clear late frame synchronization detection flag.
This bit is write only.
Programming this bit to 1 clears the LFSDET flag in the SAI_xSR register.
This bit is not used in AC’97 or SPDIF mode
Reading this bit always returns the value 0.

Bit 5 **CAFSDET**: Clear anticipated frame synchronization detection flag.
This bit is write only.
Programming this bit to 1 clears the AFSDET flag in the SAI_xSR register.
It is not used in AC’97 or SPDIF mode.
Reading this bit always returns the value 0.

Bit 4 **CCNRDY**: Clear Codec not ready flag.
This bit is write only.
Programming this bit to 1 clears the CNRDY flag in the SAI_xSR register.
This bit is used only when the AC’97 audio protocol is selected in the SAI_xCR1 register.
Reading this bit always returns the value 0.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CWCKCFG**: Clear wrong clock configuration flag.
This bit is write only.
Programming this bit to 1 clears the WCKCFG flag in the SAI_xSR register.
This bit is used only when the audio block is set as master (MODE[1] = 0) and NODIV = 0 in the SAI_xCR1 register.
Reading this bit always returns the value 0.

Bit 1 **CMUTEDET**: Mute detection flag.
This bit is write only.
Programming this bit to 1 clears the MUTEDET flag in the SAI_xSR register.
Reading this bit always returns the value 0.

Bit 0 **COVRUDR**: Clear overrun / underrun.
This bit is write only.
Programming this bit to 1 clears the OVRUDR flag in the SAI_xSR register.
Reading this bit always returns the value 0.

68.6.9 SAI data register (SAI_ADR)

Address offset: 0x20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATA[31:0]**: Data
A write to this register loads the FIFO provided the FIFO is not full.
A read from this register empties the FIFO if the FIFO is not empty.

68.6.10 SAI configuration register 1 (SAI_BCR1)

Address offset: 0x24

Reset value: 0x0000 0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	MCKEN	OSR	MCKDIV[5:0]						NODIV	Res.	DMAEN	SAIEN
				rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	OUTDRIV	MONO	SYNCEN[1:0]		CKSTR	LSBFIRST	DS[2:0]			Res.	PRTCFG[1:0]		MODE[1:0]	
		rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **MCKEN:** Master clock generation enable
0: The master clock is not generated
1: The master clock is generated independently of SAIEN bit

Bit 26 **OSR:** Oversampling ratio for master clock
This bit is meaningful only when NODIV bit is set to 0.
0: Master clock frequency = F_{FS} x 256
1: Master clock frequency = F_{FS} x 512

Bits 25:20 **MCKDIV[5:0]**: Master clock divider
These bits are set and cleared by software.
000000: Divides by 1 the kernel clock input (sai_x_ker_ck).
Otherwise, the master clock frequency is calculated according to the formula given in *Section 68.4.8*.
These bits have no meaning when the audio block is slave.
They have to be configured when the audio block is disabled.

Bit 19 **NODIV:** No divider
This bit is set and cleared by software.
0: the ratio between the Master clock generator and frame synchronization is fixed to 256 or 512
1: the ratio between the Master clock generator and frame synchronization depends on FRL[7:0]

Bit 18 Reserved, must be kept at reset value.

Bit 17 **DMAEN**: DMA enable
This bit is set and cleared by software.
0: DMA disabled
1: DMA enabled
Note: Since the audio block defaults to operate as a transmitter after reset, the MODE[1:0] bits must be configured before setting DMAEN to avoid a DMA request in receiver mode.

Bit 16 **SAIEN**: Audio block enable
This bit is set by software.
To switch off the audio block, the application software must program this bit to 0 and poll the bit till it reads back 0, meaning that the block is completely disabled. Before setting this bit to 1, check that it is set to 0, otherwise the enable command is not taken into account.
This bit enables to control the state of the SAI audio block. If it is disabled when an audio frame transfer is ongoing, the ongoing transfer completes and the cell is fully disabled at the end of this audio frame transfer.
0: SAI audio block disabled
1: SAI audio block enabled.
Note: When the SAI block (A or B) is configured in master mode, the clock must be present on the SAI block input before setting SAIEN bit.

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **OUTDRIV**: Output drive
This bit is set and cleared by software.
0: Audio block output driven when SAIEN is set
1: Audio block output driven immediately after the setting of this bit.
Note: This bit has to be set before enabling the audio block and after the audio block configuration.

Bit 12 **MONO**: Mono mode
This bit is set and cleared by software. It is meaningful only when the number of slots is equal to 2. When the mono mode is selected, slot 0 data are duplicated on slot 1 when the audio block operates as a transmitter. In reception mode, the slot1 is discarded and only the data received from slot 0 are stored. Refer to *Section : Mono/stereo mode* for more details.
0: Stereo mode
1: Mono mode.

Bits 11:10 **SYNCEN[1:0]:** Synchronization enable

These bits are set and cleared by software. They must be configured when the audio subblock is disabled.

00: audio subblock in asynchronous mode.

01: audio subblock is synchronous with the other internal audio subblock. In this case, the audio subblock must be configured in slave mode

10: audio subblock is synchronous with an external SAI embedded peripheral. In this case the audio subblock must be configured in Slave mode.

11: Reserved

Note: The audio subblock must be configured as asynchronous when SPDIF mode is enabled.

Bit 9 **CKSTR:** Clock strobing edge

This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in SPDIF audio protocol.

0: Signals generated by the SAI change on SCK rising edge, while signals received by the SAI are sampled on the SCK falling edge.

1: Signals generated by the SAI change on SCK falling edge, while signals received by the SAI are sampled on the SCK rising edge.

Bit 8 **LSBFIRST:** Least significant bit first

This bit is set and cleared by software. It must be configured when the audio block is disabled. This bit has no meaning in AC'97 audio protocol since AC'97 data are always transferred with the MSB first. This bit has no meaning in SPDIF audio protocol since in SPDIF data are always transferred with LSB first.

0: Data are transferred with MSB first

1: Data are transferred with LSB first

Bits 7:5 **DS[2:0]:** Data size

These bits are set and cleared by software. These bits are ignored when the SPDIF protocols are selected (bit PRTCFG[1:0]), because the frame and the data size are fixed in such case. When the companding mode is selected through COMP[1:0] bits, DS[1:0] are ignored since the data size is fixed to 8 bits by the algorithm.

These bits must be configured when the audio block is disabled.

000: Reserved

001: Reserved

010: 8 bits

011: 10 bits

100: 16 bits

101: 20 bits

110: 24 bits

111: 32 bits

Bit 4 Reserved, must be kept at reset value.

Bits 3:2 **PRTCFG[1:0]:** Protocol configuration

These bits are set and cleared by software. These bits have to be configured when the audio block is disabled.

00: Free protocol. Free protocol enables to use the powerful config uration of the audio block to address a specific audio protocol (such as I2S, LSB/MSB justified, TDM, PCM/DSP...) by setting most of the configuration register bits as well as frame configuration register.

01: SPDIF protocol

10: AC'97 protocol

11: Reserved

Bits 1:0 **MODE[1:0]:** SAIx audio block mode

These bits are set and cleared by software. They must be configured when SAIx audio block is disabled.

00: Master transmitter

01: Master receiver

10: Slave transmitter

11: Slave receiver

Note: When the audio block is configured in SPDIF mode, the master transmitter mode is forced (MODE[1:0] = 00). In Master transmitter mode, the audio block starts generating the FS and the clocks immediately.

68.6.11 SAI configuration register 2 (SAI_BCR2)

Address offset: 0x28

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP[1:0]		CPL	MUTECNT[5:0]						MUTE VAL	MUTE	TRIS	F FLUSH	FTH[2:0]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	w	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:14 **COMP[1:0]**: Companding mode.

These bits are set and cleared by software. The µ-Law and the A-Law log are a part of the CCITT G.711 recommendation, the type of complement that is used depends on *CPL bit*.

The data expansion or data compression are determined by the state of bit MODE[0].

The data compression is applied if the audio block is configured as a transmitter.

The data expansion is automatically applied when the audio block is configured as a receiver.

Refer to *Section : Companding mode* for more details.

00: No companding algorithm

01: Reserved.

10: µ-Law algorithm

11: A-Law algorithm

Note: Companding mode is applicable only when Free protocol mode is selected.

Bit 13 **CPL**: Complement bit.

This bit is set and cleared by software.

It defines the type of complement to be used for companding mode

0: 1’s complement representation.

1: 2’s complement representation.

Note: This bit has effect only when the companding mode is µ-Law algorithm or A-Law algorithm.

Bits 12:7 **MUTECNT[5:0]**: Mute counter.

These bits are set and cleared by software. They are used only in reception mode.

The value set in these bits is compared to the number of consecutive mute frames detected in reception. When the number of mute frames is equal to this value, the flag MUTEDET is set and an interrupt is generated if bit MUTEDETIE is set.

Refer to *Section : Mute mode* for more details.

Bit 6 **MUTEVAL**: Mute value.

This bit is set and cleared by software. It is meaningful only when the audio block operates as a transmitter, the number of slots is lower or equal to 2 and the MUTE bit is set.

If more slots are declared, the bit value sent during the transmission in mute mode is equal to 0, whatever the value of MUTEVAL.

if the number of slot is lower or equal to 2 and MUTEVAL = 1, the MUTE value transmitted for each slot is the one sent during the previous frame.

Refer to *Section : Mute mode* for more details.

0: Bit value 0 is sent during the mute mode.

1: Last values are sent during the mute mode.

Note: This bit is meaningless and must not be used for SPDIF audio blocks.

Bit 5 **MUTE:** Mute.

This bit is set and cleared by software. It is meaningful only when the audio block operates as a transmitter. The MUTE value is linked to value of MUTEVAL if the number of slots is lower or equal to 2, or equal to 0 if it is greater than 2.

Refer to *Section : Mute mode* for more details.

0: No mute mode.

1: Mute mode enabled.

Note: This bit is meaningless and must not be used for SPDIF audio blocks.

Bit 4 **TRIS**: Tristate management on data line.

This bit is set and cleared by software. It is meaningful only if the audio block is configured as a transmitter. This bit is not used when the audio block is configured in SPDIF mode. It must be configured when SAI is disabled.

Refer to *Section : Output data line management on an inactive slot* for more details.

0: SD output line is still driven by the SAI when a slot is inactive.

1: SD output line is released (HI-Z) at the end of the last data bit of the last active slot if the next one is inactive.

Bit 3 **FFLUSH:** FIFO flush.

This bit is set by software. It is always read as 0. This bit must be configured when the SAI is disabled.

0: No FIFO flush.

1: FIFO flush. Programming this bit to 1 triggers the FIFO Flush. All the internal FIFO pointers (read and write) are cleared. In this case data still present in the FIFO are lost (no more transmission or received data lost). Before flushing, SAI DMA stream/interrupt must be disabled

Bits 2:0 **FTH[2:0]:** FIFO threshold.

This bit is set and cleared by software.

000: FIFO empty

001: ¼ FIFO

010: ½ FIFO

011: ¾ FIFO

100: FIFO full

101: Reserved

110: Reserved

111: Reserved

68.6.12 SAI frame configuration register (SAI_BFRCR)

Address offset: 0x2C

Reset value: 0x0000 0007

Note: This register has no meaning in AC'97 and SPDIF audio protocol

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSOFF	FSPOL	FSDEF
													rw	rw	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	FSALL[6:0]							FRL[7:0]							
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **FSOFF**: Frame synchronization offset.
This bit is set and cleared by software. It is meaningless and is not used in AC'97 or SPDIF audio block configuration. This bit must be configured when the audio block is disabled.
0: FS is asserted on the first bit of the slot 0.
1: FS is asserted one bit before the first bit of the slot 0.

Bit 17 **FSPOL**: Frame synchronization polarity.
This bit is set and cleared by software. It is used to configure the level of the start of frame on the FS signal. It is meaningless and is not used in AC'97 or SPDIF audio block configuration.
This bit must be configured when the audio block is disabled.
0: FS is active low (falling edge)
1: FS is active high (rising edge)

Bit 16 **FSDEF**: Frame synchronization definition.
This bit is set and cleared by software.
0: FS signal is a start frame signal
1: FS signal is a start of frame signal + channel side identification
When the bit is set, the number of slots defined in the SAI_xSLOTR register has to be even. It means that half of this number of slots is dedicated to the left channel and the other slots for the right channel (e.g: this bit has to be set for I2S or MSB/LSB-justified protocols...).
This bit is meaningless and is not used in AC'97 or SPDIF audio block configuration. It must be configured when the audio block is disabled.

Bit 15 Reserved, must be kept at reset value.

Bits 14:8 **FSALL[6:0]**: Frame synchronization active level length.
These bits are set and cleared by software. They specify the length in number of bit clock (SCK) + 1 (FSALL[6:0] + 1) of the active level of the FS signal in the audio frame
These bits are meaningless and are not used in AC'97 or SPDIF audio block configuration.
They must be configured when the audio block is disabled.

Bits 7:0 **FRL[7:0]**: Frame length.

These bits are set and cleared by software. They define the audio frame length expressed in number of SCK clock cycles: the number of bits in the frame is equal to FRL[7:0] + 1.

The minimum number of bits to transfer in an audio frame must be equal to 8, otherwise the audio block behaves in an unexpected way. This is the case when the data size is 8 bits and only one slot 0 is defined in NBSLOT[4:0] of SAI_xSLOTR register (NBSLOT[3:0] = 0000).

In master mode, if the master clock (available on MCLK_x pin) is used, the frame length must be aligned with a number equal to a power of 2, ranging from 8 to 256. When the master clock is not used (NODIV = 1), it is recommended to program the frame length to an value ranging from 8 to 256.

These bits are meaningless and are not used in AC'97 or SPDIF audio block configuration.

68.6.13 SAI slot register (SAI_BSLOTR)

Address offset: 0x30

Reset value: 0x0000 0000

Note: This register has no meaning in AC'97 and SPDIF audio protocol.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SLOTEN[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	NBSLOT[3:0]				SLOTSZ[1:0]		Res.	FBOFF[4:0]				
				rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bits 31:16 **SLOTEN[15:0]**: Slot enable.

These bits are set and cleared by software.

Each SLOTEN bit corresponds to a slot position from 0 to 15 (maximum 16 slots).

0: Inactive slot.

1: Active slot.

The slot must be enabled when the audio block is disabled.

They are ignored in AC'97 or SPDIF mode.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **NBSLOT[3:0]**: Number of slots in an audio frame.

These bits are set and cleared by software.

The value set in this bitfield represents the number of slots in the audio frame (including the number of inactive slots). The maximum number of slots is 16.

The number of slots must be even if FSDEF bit in the SAI_xFRCR register is set.

The number of slots must be configured when the audio block is disabled.

They are ignored in AC'97 or SPDIF mode.

0000: Number of slots is 1.

0001: Number of slots is 2.

...

1111: Number of slots is 16.

Bits 7:6 **SLOTSZ[1:0]**: Slot size
This bits is set and cleared by software.
The slot size must be higher or equal to the data size. If this condition is not respected, the behavior of the SAI is undetermined.
Refer to *Output data line management on an inactive slot* for information on how to drive SD line.
These bits must be set when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.
00: The slot size is equivalent to the data size (specified in DS[3:0] in the SAI_xCR1 register).
01: 16-bit
10: 32-bit
11: Reserved

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **FBOFF[4:0]**: First bit offset
These bits are set and cleared by software.
The value set in this bitfield defines the position of the first data transfer bit in the slot. It represents an offset value. In transmission mode, the bits outside the data field are forced to 0. In reception mode, the extra received bits are discarded.
These bits must be set when the audio block is disabled.
They are ignored in AC'97 or SPDIF mode.

68.6.14 SAI interrupt mask register (SAI_BIM)

Address offset: 0x34

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LFSDETIE	AFSDETIE	CNRDYIE	FREQIE	WCKCFGIE	MUTEDETIE	OVRUDRIE
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **LFSDETIE**: Late frame synchronization detection interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the LFSDET bit is set in the SAI_xSR register.
This bit is meaningless in AC'97, SPDIF mode or when the audio block operates as a master.

Bit 5 **AFSDETIE**: Anticipated frame synchronization detection interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the AFSDET bit in the SAI_xSR register is set.
This bit is meaningless in AC'97, SPDIF mode or when the audio block operates as a master.

Bit 4 **CNRDYIE**: Codec not ready interrupt enable (AC'97).
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When the interrupt is enabled, the audio block detects in the slot 0 (tag0) of the AC'97 frame if the Codec connected to this line is ready or not. If it is not ready, the CNRDY flag in the SAI_xSR register is set and an interrupt is generated.
This bit has a meaning only if the AC'97 mode is selected through PRTCFG[1:0] bits and the audio block is operates as a receiver.

Bit 3 **FREQIE**: FIFO request interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the FREQ bit in the SAI_xSR register is set.
Since the audio block defaults to operate as a transmitter after reset, the MODE bit must be configured before setting FREQIE to avoid a parasitic interrupt in receiver mode,

Bit 2 **WCKCFGIE**: Wrong clock configuration interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
This bit is taken into account only if the audio block is configured as a master (MODE[1] = 0) and NODIV = 0.
It generates an interrupt if the WCKCFG flag in the SAI_xSR register is set.
Note: This bit is used only in Free protocol mode and is meaningless in other modes.

Bit 1 **MUTEDETIE**: Mute detection interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the MUTEDET bit in the SAI_xSR register is set.
This bit has a meaning only if the audio block is configured in receiver mode.

Bit 0 **OVRUDRIE**: Overrun/underrun interrupt enable.
This bit is set and cleared by software.
0: Interrupt is disabled
1: Interrupt is enabled
When this bit is set, an interrupt is generated if the OVRUDR bit in the SAI_xSR register is set.

68.6.15 SAI status register (SAI_BSR)

Address offset: 0x38

Reset value: 0x0000 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLVL[2:0]		
													r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LFSDET	AFSDET	CNRDY	FREQ	WCKCFG	MUTEDET	OVRUDR
									r	r	r	r	r	r	r

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:16 **FLVL[2:0]**: FIFO level threshold.
This bit is read only. The FIFO level threshold flag is managed only by hardware and its setting depends on SAI block configuration (transmitter or receiver mode).
000: FIFO empty (transmitter and receiver modes)
001: FIFO ≤ ¼ but not empty (transmitter mode), FIFO < ¼ but not empty (receiver mode)
010: ¼ < FIFO ≤ ½ (transmitter mode), ¼ ≤ FIFO < ½ (receiver mode)
011: ½ < FIFO ≤ ¾ (transmitter mode), ½ ≤ FIFO < ¾ (receiver mode)
100: ¾ < FIFO but not full (transmitter mode), ¾ ≤ FIFO but not full (receiver mode)
101: FIFO full (transmitter and receiver modes)
Others: Reserved

Bits 15:7 Reserved, must be kept at reset value.

Bit 6 **LFSDET**: Late frame synchronization detection.
This bit is read only.
0: No error.
1: Frame synchronization signal is not present at the right time.
This flag can be set only if the audio block is configured in slave mode.
It is not used in AC'97 or SPDIF mode.
It can generate an interrupt if LFSDETIE bit is set in the SAI_xIM register.
This flag is cleared when the software sets bit CLFSDET in SAI_xCLRFR register

Bit 5 **AFSDET**: Anticipated frame synchronization detection.
This bit is read only.
0: No error.
1: Frame synchronization signal is detected earlier than expected.
This flag can be set only if the audio block is configured in slave mode.
It is not used in AC'97or SPDIF mode.
It can generate an interrupt if AFSDETIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CAFSDET bit in SAI_xCLRFR register.

Bit 4 **CNRDY**: Codec not ready.
This bit is read only.
0: External AC'97 Codec is ready
1: External AC'97 Codec is not ready
This bit is used only when the AC'97 audio protocol is selected in the SAI_xCR1 register and configured in receiver mode.
It can generate an interrupt if CNRDYIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CCNRDY bit in SAI_xCLRFR register.

Bit 3 **FREQ**: FIFO request.
This bit is read only.
0: No FIFO request.
1: FIFO request to read or to write the SAI_xDR.
The request depends on the audio block configuration:
– If the block is configured in transmission mode, the FIFO request is related to a write request operation in the SAI_xDR.
– If the block configured in reception, the FIFO request related to a read request operation from the SAI_xDR.
This flag can generate an interrupt if FREQIE bit is set in SAI_xIM register.

Bit 2 **WCKCFG**: Wrong clock configuration flag.
This bit is read only.
0: Clock configuration is correct
1: Clock configuration does not respect the rule concerning the frame length specification defined in *Section 68.4.6: Frame synchronization* (configuration of FRL[7:0] bit in the SAI_xFRCR register)
This bit is used only when the audio block operates in master mode (MODE[1] = 0) and NODIV = 0.
It can generate an interrupt if WCKCFGIE bit is set in SAI_xIM register.
This flag is cleared when the software sets CWCKCFG bit in SAI_xCLRFR register.

Bit 1 **MUTEDET**: Mute detection.
This bit is read only.
0: No MUTE detection on the SD input line
1: MUTE value detected on the SD input line (0 value) for a specified number of consecutive audio frame
This flag is set if consecutive 0 values are received in each slot of a given audio frame and for a consecutive number of audio frames (set in the MUTECNT bit in the SAI_xCR2 register).
It can generate an interrupt if MUTEDETIE bit is set in SAI_xIM register.
This flag is cleared when the software sets bit CMUTEDET in the SAI_xCLRFR register.

Bit 0 **OVRUDR**: Overrun / underrun.
This bit is read only.
0: No overrun/underrun error.
1: Overrun/underrun error detection.
The overrun and underrun conditions can occur only when the audio block is configured as a receiver and a transmitter, respectively.
It can generate an interrupt if OVRUDRIE bit is set in SAI_xIM register.
This flag is cleared when the software sets COVRUDR bit in SAI_xCLRFR register.

68.6.16 SAI clear flag register (SAI_BCLRFR)

Address offset: 0x3C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLFSDET	CAFSDET	CCNRDY	Res.	CWCKCFG	CMUTEDET	COVRUDR
									w	w	w		w	w	w

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **CLFSDET**: Clear late frame synchronization detection flag.
This bit is write only.
Programming this bit to 1 clears the LFSDET flag in the SAI_xSR register.
This bit is not used in AC'97or SPDIF mode
Reading this bit always returns the value 0.

Bit 5 **CAFSDET**: Clear anticipated frame synchronization detection flag.
This bit is write only.
Programming this bit to 1 clears the AFSDET flag in the SAI_xSR register.
It is not used in AC'97or SPDIF mode.
Reading this bit always returns the value 0.

Bit 4 **CCNRDY**: Clear Codec not ready flag.
This bit is write only.
Programming this bit to 1 clears the CNRDY flag in the SAI_xSR register.
This bit is used only when the AC'97 audio protocol is selected in the SAI_xCR1 register.
Reading this bit always returns the value 0.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CWCKCFG**: Clear wrong clock configuration flag.
This bit is write only.
Programming this bit to 1 clears the WCKCFG flag in the SAI_xSR register.
This bit is used only when the audio block is set as master (MODE[1] = 0) and NODIV = 0 in the SAI_xCR1 register.
Reading this bit always returns the value 0.

Bit 1 **CMUTEDET**: Mute detection flag.
This bit is write only.
Programming this bit to 1 clears the MUTEDET flag in the SAI_xSR register.
Reading this bit always returns the value 0.

Bit 0 **COVRUDR**: Clear overrun / underrun.
This bit is write only.
Programming this bit to 1 clears the OVRUDR flag in the SAI_xSR register.
Reading this bit always returns the value 0.

68.6.17 SAI data register (SAI_BDR)

Address offset: 0x40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATA[31:0]**: Data
A write to this register loads the FIFO provided the FIFO is not full.
A read from this register empties the FIFO if the FIFO is not empty.

68.6.18 SAI PDM control register (SAI_PDMCR)

Address offset: 0x44

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CKEN2	CKEN1	Res.	Res.	MICNBR[1:0]		Res.	Res.	Res.	PDMEN
						rw	rw			rw	rw				rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **CKEN2**: Clock enable of bitstream clock number 2

This bit is set and cleared by software.

0: SAI_CK2 clock disabled

1: SAI_CK2 clock enabled

Note: It is not recommended to configure this bit when PDMEN = 1.

SAI_CK2 might not be available for all SAI instances. Refer to Section 68.3: SAI implementation for details.

Bit 8 **CKEN1**: Clock enable of bitstream clock number 1

This bit is set and cleared by software.

0: SAI_CK1 clock disabled

1: SAI_CK1 clock enabled

Note: It is not recommended to configure this bit when PDMEN = 1.

SAI_CK1 might not be available for all SAI instances. Refer to Section 68.3: SAI implementation for details.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **MICNBR[1:0]**: Number of microphones

This bit is set and cleared by software.

00: Configuration with 2 microphones

01: Configuration with 4 microphones

10: Configuration with 6 microphones

11: Configuration with 8 microphones

*Note: It is not recommended to configure this field when PDMEN = 1.**

The complete set of data lines might not be available for all SAI instances. Refer to Section 68.3: SAI implementation for details.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **PDMEN**: PDM enable

This bit is set and cleared by software. This bit enables to control the state of the PDM interface block.

Make sure that the SAI in already operating in TDM master mode before enabling the PDM interface.

0: PDM interface disabled

1: PDM interface enabled

68.6.19 SAI PDM delay register (SAI_PDMDLY)

Address offset: 0x48

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	DLYM4R[2:0]			Res.	DLYM4L[2:0]			Res.	DLYM3R[2:0]			Res.	DLYM3L[2:0]		
	rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DLYM2R[2:0]			Res.	DLYM2L[2:0]			Res.	DLYM1R[2:0]			Res.	DLYM1L[2:0]		
	rw	rw	rw		rw	rw	rw		rw	rw	rw		rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **DLYM4R[2:0]**: Delay line for second microphone of **pair 4**

This bitfield is set and cleared by software.

000: No delay

001: Delay of 1 T_{SAI_CK} period

010: Delay of 2 T_{SAI_CK} periods

...

111: Delay of 7 T_{SAI_CK} periods

This bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D4 line is available.Refer to Section 68.3: SAI implementation to check if it is available.

Bit 27 Reserved, must be kept at reset value.

Bits 26:24 **DLYM4L[2:0]**: Delay line for first microphone of pair 4

This bitfield is set and cleared by software.

000: No delay

001: Delay of 1 T_{SAI_CK} period

010: Delay of 2 T_{SAI_CK} periods

...

111: Delay of 7 of T_{SAI_CK} periods

This bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D4 line is available.Refer to Section 68.3: SAI implementation to check if it is available.

Bit 23 Reserved, must be kept at reset value.

Bits 22:20 **DLYM3R[2:0]**: Delay line for second microphone of pair 3
This bitfield is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D3 line is available.Refer to Section 68.3: SAI implementation to check if it is available.

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **DLYM3L[2:0]**: Delay line for first microphone of pair 3
This bitfield is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D3 line is available.Refer to Section 68.3: SAI implementation to check if it is available.

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **DLYM2R[2:0]**: Delay line for second microphone of pair 2
This bitfield is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D2 line is available.Refer to Section 68.3: SAI implementation to check if it is available.

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **DLYM2L[2:0]**: Delay line for first microphone of pair 2
This bitfield is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D2 line is available.Refer to Section 68.3: SAI implementation to check if it is available.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **DLYM1R[2:0]**: Delay line adjust for second microphone of pair 1
This bitfield is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D1 line is available.Refer to Section 68.3: SAI implementation to check if it is available.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **DLYM1L[2:0]**: Delay line adjust for first microphone of pair 1
This bitfield is set and cleared by software.
000: No delay
001: Delay of 1 T_{SAI_CK} period
010: Delay of 2 T_{SAI_CK} periods
...
111: Delay of 7 T_{SAI_CK} periods

This bitfield can be changed on-the-fly.

Note: This bitfield can be used only if D1 line is available.Refer to Section 68.3: SAI implementation to check if it is available.

68.6.20 SAI register map

Table 680. SAI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	SAI_GCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYNCOUT[1:0]		Res.	Res.	SYNCIN[1:0]	
	Reset value																											0	0			0	0
0x04 or 0x24	SAI_xCR1	Res.	Res.	Res.	Res.	MCKEN	OSR	MCKDIV[5:0]						NODIV	Res.	DMAEN	SAIEN	Res.	Res.	OUTDRIV	MONO	SYNCEN[1:0]		CKSTR	LSBFIRST	DS[2:0]		Res.	PRTCFG[1:0]		MODE[1:0]		
	Reset value					0	0	0	0	0	0	0	0	0		0	0			0	0	0	0	0	0	0	1	0		0	0	0	0
0x08 or 0x28	SAI_xCR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMP[1:0]		CPL	MUTECN[5:0]						MUTE VAL	MUTE	TRIS	FFLUS	FTH[2:0]		
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C or 0x2C	SAI_xFRCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSOFF	FSPOL	FSDEF	Res.	FSALL[6:0]							FRL[7:0]							
	Reset value														0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

Table 680. SAI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x10 or 0x30	SAI_xSLOTR	SLOTEN[15:0]																Res.	Res.	Res.	Res.	NBSLOT[3:0]				SLOTSZ[1:0]		Res.	FBOFF[4:0]				
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0		0	0	0	0	0
0x14 or 0x34	SAI_xIM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LFSDETIE	AFSDETIE	CNRDYIE	FREQIE	WCKCFGIE	MUTEDETIE	OVRUDRIE
	Reset value																										0	0	0	0	0	0	0
0x18 or 0x38	SAI_xSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLVL[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LFSDET	AFSDET	CNRDY	FREQ	WCKCFG	MUTEDET	OVRUDR
	Reset value														0	0	0										0	0	0	1	0	0	0
0x1C or 0x3C	SAI_xCLRFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLFSDET	CAFSDET	CCNRDY	Res.	CWCKCFG	CMUTEDET	COVRUDR
	Reset value																										0	0	0		0	0	0
0x20 or 0x40	SAI_xDR	DATA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x44	SAI_PDMCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CKEN2	CKEN1	Res.	Res.	MICNBR[1:0]		Res.	Res.	Res.	PDMEN
	Reset value																							0	0			0	0				0
0x48	SAI_PDMDLY	Res.	DLYM4R[2:0]			Res.	DLYM4L[2:0]			Res.	DLYM3R[2:0]			Res.	DLYM3L[2:0]			Res.	DLYM2R[2:0]			Res.	DLYM2L[2:0]			Res.	DLYM1R[2:0]			Res.	DLYM1L[2:0]		
	Reset value		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0		0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

69 SPDIF receiver interface (SPDIFRX)

69.1 SPDIFRX interface introduction

The SPDIFRX interface handles S/PDIF audio protocol.

69.2 SPDIFRX main features

- Up to 4 inputs available
- Automatic symbol rate detection
- Maximum symbol rate: 12.288 MHz
- Stereo stream from 8 to 192 kHz[a] supported
- Supports audio IEC-60958 and IEC-61937, consumer applications
- Parity bit management
- Communication using DMA for audio samples
- Communication using DMA for control and user channel information
- Interrupt capabilities

69.3 SPDIFRX functional description

The SPDIFRX peripheral, is designed to receive an S/PDIF flow compliant with IEC-60958 and IEC-61937. These standards support simple stereo streams up to high sample rate, and compressed multi-channel surround sound, such as those defined by Dolby or DTS.

The receiver provides all the necessary features to detect the symbol rate, and decode the incoming data. It is possible to use a dedicated path for the user and channel information in order to ease the interface handling. *Figure 935* shows a simplified block diagram.

The SPDIFRX_DC block is responsible of the decoding of the S/PDIF stream received from SPDIFRX_IN[3:0] inputs. This block re-sample the incoming signal, decode the manchester stream, recognize frames, sub-frames and blocks elements. It delivers to the REG_IF part, decoded data, and associated status flags.

This peripheral can be fully controlled via the APB bus, and can handle two DMA channels:

- A DMA channel dedicated to the transfer of audio samples
- A DMA channel dedicated to the transfer of IEC60958 channel status and user information

Interrupt services are also available either as an alternative function to the DMA, or for signaling error or key status of the peripheral.

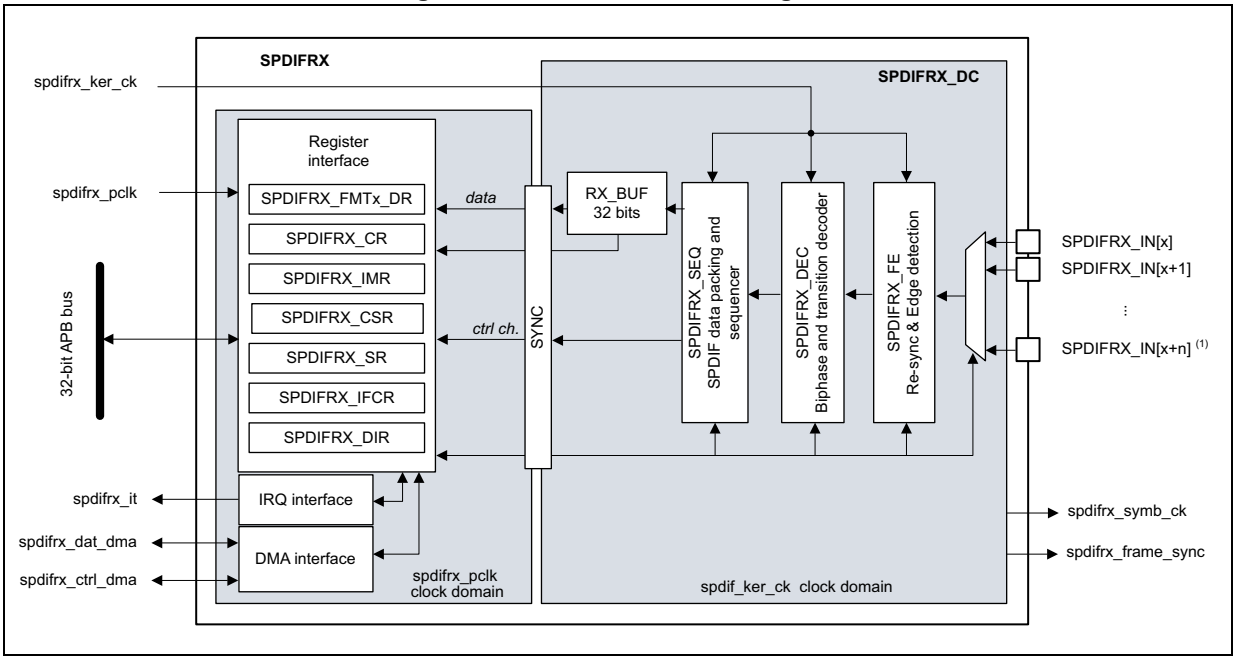
The SPDIFRX also offers a signal named spdifrx_frame_sync, which toggles every time that a sub-frame's preamble is detected. So the duty cycle is 50%, and the frequency equal to the frame rate.

This signal can be connected to timer events, in order to compute frequency drift.

a. Check the RCC capabilities in order to verify which sampling rates can be supported.

In addition the SPDIFRX also provides a signal named **spdifrx_symb_ck** toggling at the symbol rate.

Figure 935. SPDIFRX block diagram



1. 'n' is fixed to 3, and 'x' is set to 0.

69.3.1 SPDIFRX pins and internal signals

Table 681 lists the SPDIFRX internal input/output signals, *Table 682* the SPDIFRX pins (alternate functions).

Table 681. SPDIFRX internal input/output signals

Signal name	Signal type	Description
spdifrx_ker_ck	Digital input	SPDIFRX kernel clock
spdifrx_pclk	Digital input	SPDIFRX register interface clock
spdifrx_it	Digital output	SPDIFRX global interrupt
spdifrx_dat_dma	Digital input/output	SPDIFRX DMA request (and acknowledge) for data transfer
spdifrx_ctrl_dma	Digital input/output	SPDIFRX DMA request (and acknowledge) for channel status and user information transfer
spdifrx_frame_sync	Digital output	SPDIFRX frame rate synchronization signal
spdifrx_symb_ck	Digital output	SPDIFRX channel symbol clock

Table 682. SPDIFRX pins

Pin name	Pin type	Description
SPDIFRX_IN0	Digital input	Input 0 for S/PDIF signal
SPDIFRX_IN1	Digital input	Input 1 for S/PDIF signal

Table 682. SPDIFRX pins (continued)

Pin name	Pin type	Description
SPDIFRX_IN2	Digital input	Input 2 for S/PDIF signal
SPDIFRX_IN3	Digital input	Input 3 for S/PDIF signal

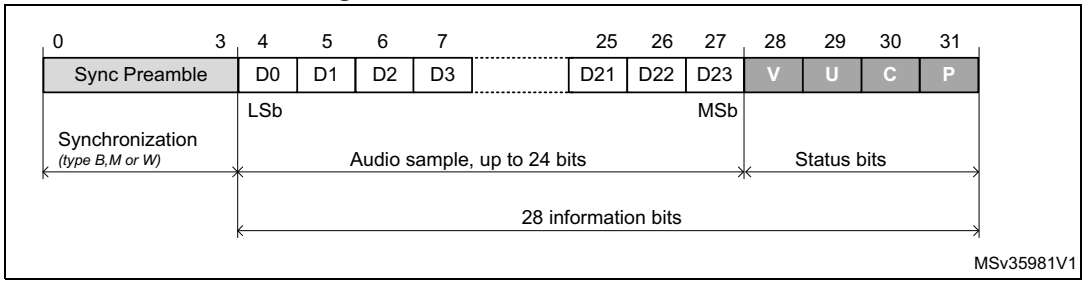
69.3.2 S/PDIF protocol (IEC-60958)

S/PDIF block

A S/PDIF frame is composed of two sub-frames (see *Figure 936*). Each sub-frame contains 32 bits (or time slots):

- Bits 0 to 3 carry one of the synchronization preambles
- Bits 4 to 27 carry the audio sample word in linear 2's complement representation. The most significant bit (MSB) is carried by bit 27. When a 20-bit coding range is used, bits 8 to 27 carry the audio sample word with the LSB in bit 8.
- Bit 28 (validity bit “V”) indicates if the data is valid (for converting it to analog for example)
- Bit 29 (user data bit “U”) carries the user data information like the number of tracks of a Compact Disk.
- Bit 30 (channel status bit “C”) carries the channel status information like sample rate and protection against copy.
- Bit 31 (parity bit “P”) carries a parity bit such that bits 4 to 31 inclusive carry an even number of ones and an even number of zeroes (even parity).

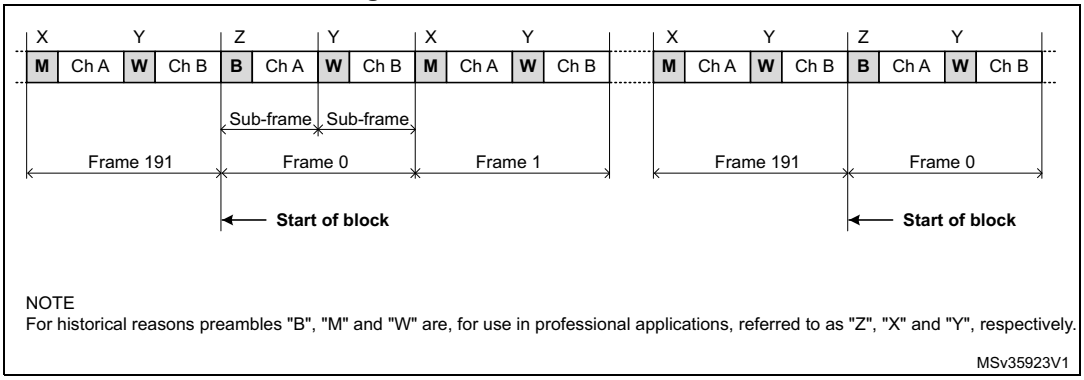
Figure 936. S/PDIF sub-frame format



For linear coded audio applications, the first sub-frame (left or “A” channel in stereophonic operation and primary channel in monophonic operation) normally starts with preamble “M”. However, the preamble changes to preamble “B” once every 192 frames to identify the start of the block structure used to organize the channel status and user information. The second sub-frame (right or “B” channel in stereophonic operation and secondary channel in monophonic operation) always starts with preamble “W”.

A S/PDIF block contains 192 pairs of sub-frames of 32 bits.

Figure 937. S/PDIF block format



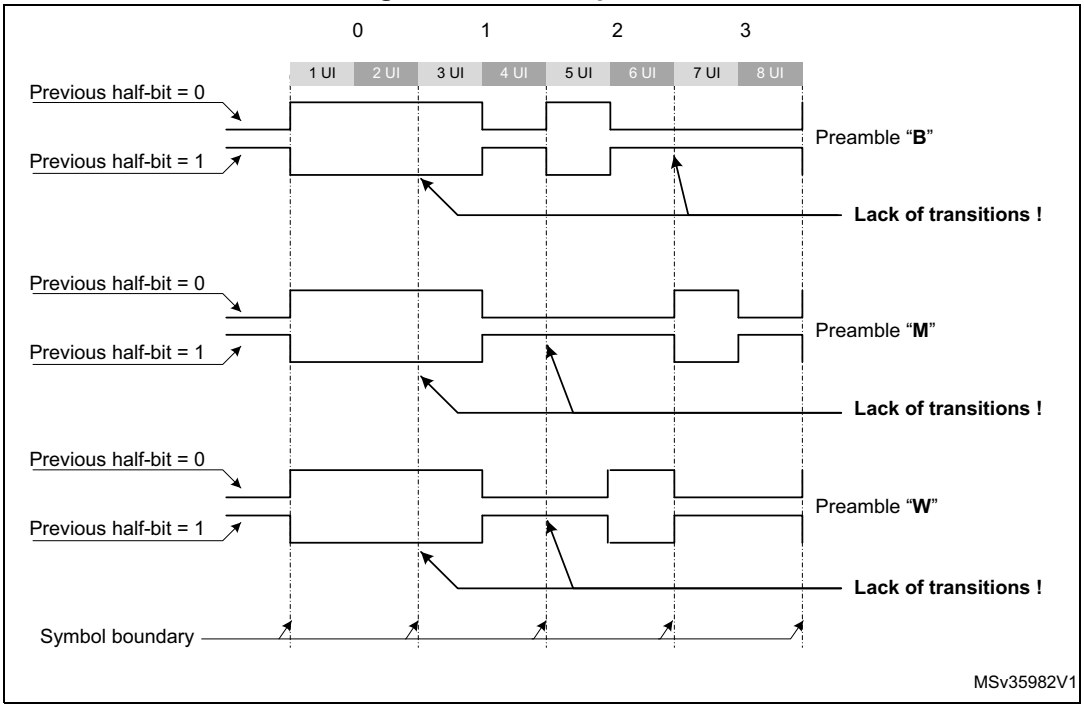
NOTE
For historical reasons preambles "B", "M" and "W" are, for use in professional applications, referred to as "Z", "X" and "Y", respectively.

MSv35923V1

Synchronization preambles

The preambles patterns are inverted or not according to the previous half-bit value. This previous half-bit value is the level of the line before enabling a transfer for the first “B” preamble of the first frame. For the others preambles, this previous half-bit value is the second half-bit of the parity bit of the previous sub-frame. The preambles patterns B, M and W are described in the *Figure 938*.

Figure 938. S/PDIF preambles



MSv35982V1

Coding of information bits

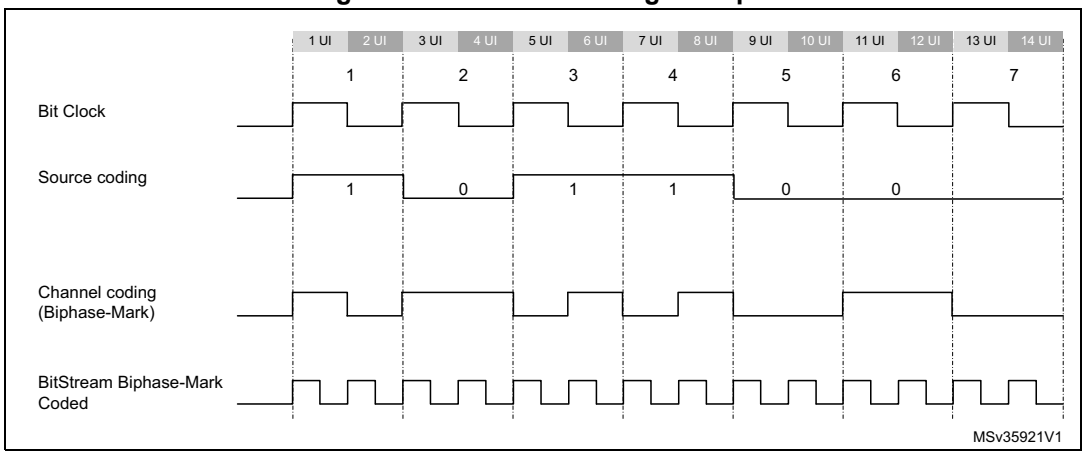
In order to minimize the DC component value on the transmission line, and to facilitate clock recovery from the data stream, bits 4 to 31 are encoded in biphase-mark.

Each bit to be transmitted is represented by a symbol comprising two consecutive binary states. The first state of a symbol is always different from the second state of the previous symbol. The second state of the symbol is identical to the first if the bit to be transmitted is

logical 0. However, it is different if the bit is logical 1. These states are named “UI” (unit interval) in the IEC-60958 specification.

The 24 data bits are transferred LSB first.

Figure 939. Channel coding example



69.3.3 SPDIFRX decoder (SPDIFRX_DC)

Main principle

The technique used by the SPDIFRX in order to decode the S/PDIF stream is based on the measurement of the time interval between two consecutive edges. Three kinds of time intervals may be found into an S/PDIF stream:

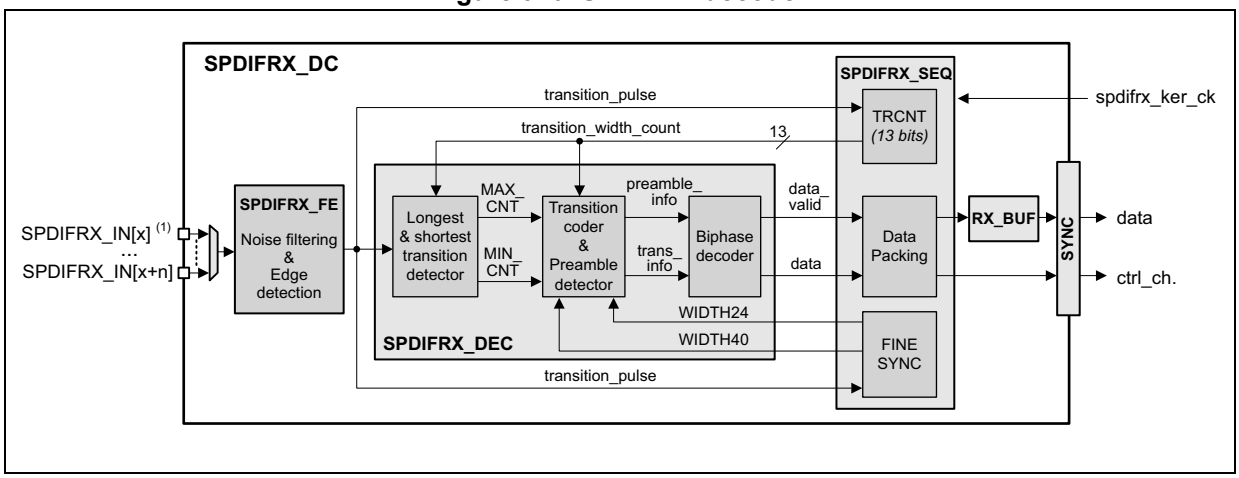
- The long time interval, having a duration of 3 x UI, noted TL. It appears only during preambles.
- The medium time interval, having a duration of 2 x UI, noted TM. It appears both in some preambles or into the information field.
- The short time interval, having a duration of 1 x UI, noted TS. It appears both in some preambles or into the information field.

The SPDIFRX_DC block is responsible of the decoding of the received S/PDIF stream. It takes care of the following functions:

- Resampling and filtering of the incoming signal
- Estimation of the time-intervals
- Estimation of the symbol rate and synchronization
- Decoding of the serial data, and check of integrity
- Detection of the block, and sub-frame preambles
- Continuous tracking of the symbol rate

Figure 940 gives a detailed view of the SPDIFRX decoder.

Figure 940. SPDIFRX decoder



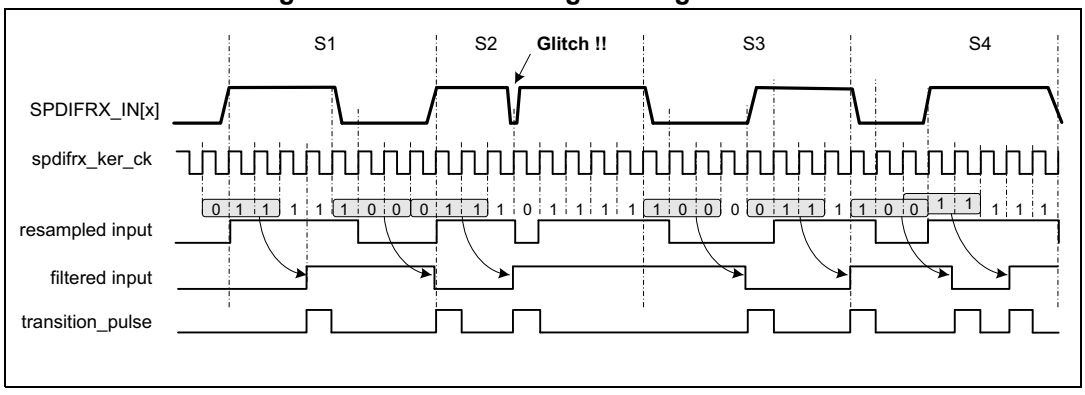
1. 'n' is fixed to 3, and 'x' is set to 0.

Noise filtering and rising/falling edge detection

The S/PDIF signal received on the selected SPDIFRX_IN is re-sampled using the spdifrx_ker_ck clock (acquisition clock). A simple filtering is applied in order cancel spurs. This is performed by the stage detecting the edge transitions. The edge transitions are detected as follow:

- A rising edge is detected when the sequence 0 followed by two 1 is sampled.
- A falling edge is detected when the sequence 1 followed by two 0 is sampled.
- After a rising edge, a falling edge sequence is expected.
- After a falling edge, a rising edge sequence is expected.

Figure 941. Noise filtering and edge detection



Longest and shortest transition detector

The **longest and shortest transition detector** block detects the maximum (MAX_CNT) and minimum (MIN_CNT) duration between two transitions. The TRCNT counter is used to measure the time interval duration. It is clocked by the spdifrx_ker_ck signal. On every transition pulse, the counter value is stored and the counter is reset to start counting again.

The maximum duration is normally found during the preamble period. This maximum duration is sent out as MAX_CNT. The minimum duration is sent out as MIN_CNT.

The search of the longest and shortest transition is stopped when the transition timer expires. The transition timer is like a watchdog timer that generates a trigger after 70 transitions of the incoming signal. Note that counting 70 transitions insures a delay a bit longer than a sub-frame.

Note that when the TRCNT overflows due to a too long time interval between two pulses, the SPDIFRX is stopped and the flag TERR of SPDIFRX_SR register is set to 1.

Transition coder and preamble detector

The **transition coder and preamble detector** block receives the MAX_CNT and MIN_CNT. It also receives the current transition width from the TRCNT counter (see *Figure 940*). This block encodes the current transition width by comparing the current transition width with two different thresholds, names TH_{HI} and TH_{LO}.

- If the current transition width is less than (TH_{LO} - 1), then the data received is half part of data bit ‘1’, and is coded as TS.
- If the current transition width is greater than (TH_{LO} - 1), and less than TH_{HI}, then the data received is data bit ‘0’, and is coded as TM.
- If the current transition width is greater than TH_{HI}, then the data received is the long pulse of preambles, and is coded as TL.
- Else an error code is generated (FERR flag is set).

The thresholds TH_{HI} and TH_{LO} are elaborated using two different methods.

If the peripheral is doing its initial synchronization (‘coarse synchronization’), then the thresholds are computed as follow:

- TH_{LO} = MAX_CNT / 2.
- TH_{HI} = MIN_CNT + MAX_CNT / 2.

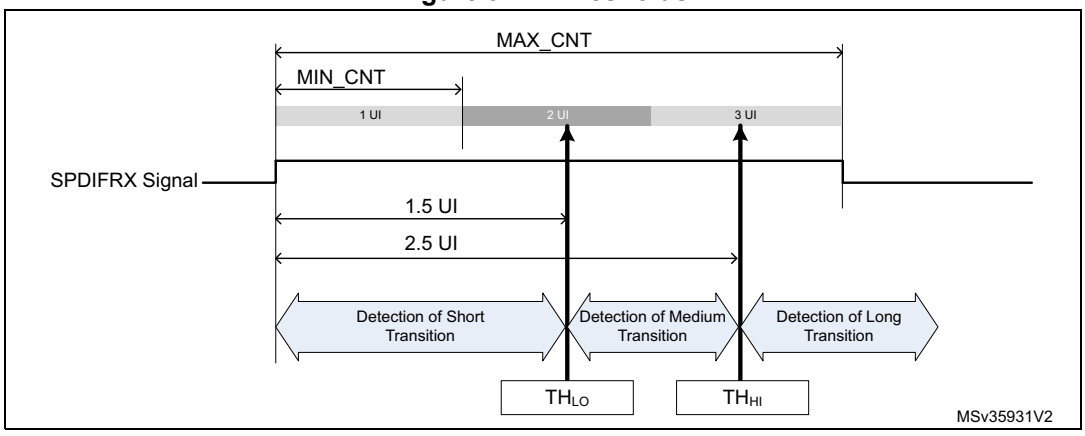
Once the ‘coarse synchronization’ is completed, then the SPDIFRX uses a more accurate reference in order to elaborate the thresholds. The SPDIFRX measures the length of 24 symbols (WIDTH24) for defining TH_{LO} and the length of 40 symbols (WIDTH40) for TH_{HI}. TH_{HI} and TH_{LO} are computed as follow:

- TH_{LO} = (WIDTH24) / 32
- TH_{HI} = (WIDTH40) / 32

This second synchronization phase is called the ‘fine synchronization’. Refer to *Figure 944* for additional information.

As shown in the figure hereafter, TH_{LO} is ideally equal to 1.5 UI, and to TH_{HI} 2.5 UI.

Figure 942. Thresholds



The preamble detector checks four consecutive transitions of a specific sequence to determine if they form the part of preamble. Let us say TRANS0, TRANS1, TRANS2 and TRANS3 represent four consecutive transitions encoded as mentioned above. *Table 683* shows the values of these four transitions to form a preamble. Absence of this pattern indicates that these transitions form part of the data in the sub frame and bi-phase decoder decode them.

Table 683. Transition sequence for preamble

Preamble type	Biphase data pattern	TRANS3	TRANS2	TRANS1	TRANS0
Preamble B	11101000	TL	TS	TS	TL
Preamble M	11100010	TL	TL	TS	TS
Preamble W	11100100	TL	TM	TS	TM

Bi-phase decoder

The Bi-phase decoder decodes the input bi-phase marked data stream using the transition information provided by the **transition coder and preamble detector** block. It first waits for the preamble detection information. After the preamble detection, it decodes the following transition information:

- If the incoming transition information is TM then it is decoded as a ‘0’.
- Two consecutive TS are decoded as a ‘1’.
- Any other transition sequence generates an error signal (FERR set to 1).

After decoding 28 data bits this way, this module looks for the following preamble data. If the new preamble is not what is expected, then this block generates an error signal (FERR set to 1). Refer to *Section 69.3.9*, for additional information on error flags.

Data packing

This block is responsible of the decoding of the IEC-60958 frames and blocks. It also handles the writing into the RX_BUF or into SPDIFRX_CSR register.

69.3.4 SPDIFRX tolerance to clock deviation

The SPDIFRX tolerance to clock deviation depends on the number of sample clock cycles in one bit slot. The fastest spdifrx_ker_ck is, the more robust the reception is. The ratio between spdifrx_ker_ck frequency and the symbol rate must be at least 11.

Two kinds of phenomenon (at least) can degrade the reception quality:

- The cycle-to-cycle jitter which reflects the difference of transition length between two consecutive transitions.
- The long term jitter which reflects a cumulative effect of the cycle-to-cycle jitter. It can be seen as a low-frequency symbol modulation.

69.3.5 SPDIFRX synchronization

The synchronization phase starts when setting SPDIFRXEN to 01 or 11. *Figure 943* shows the synchronization process.

If the bit WFA of SPDIFRX_CR register is set to 1, then the peripheral must first detect activity on the selected SPDIFRX_IN line before starting the synchronization process. The activity detection is performed by detecting four transitions on the selected SPDIFRX_IN. The peripheral remains in this state until transitions are not detected. This function can be particularly helpful because the SPDIFRX switches in COARSE SYNC mode only if activity is present on the selected SPDIFRX_IN input, avoiding synchronization errors. See *Section 69.4* for additional information.
The user can still set the SPDIFRX into STATE_IDLE by setting SPDIFRXEN to 0. If the WFA is set to 0, the peripheral starts the coarse synchronization without checking activity.

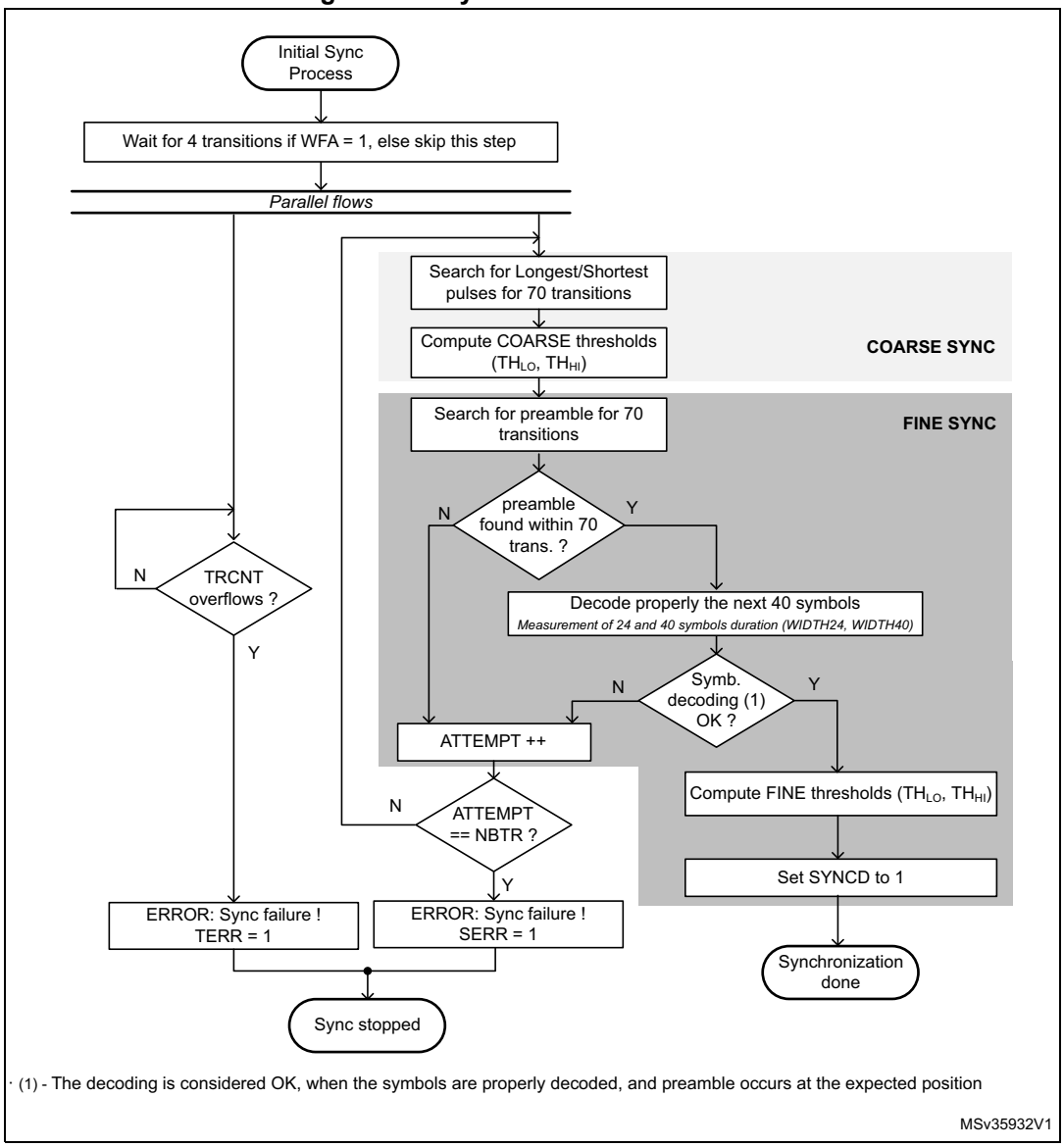
The next step consists on doing a first estimate of the thresholds (COARSE SYNC), in order to perform the fine synchronization (FINE SYNC). Due to disturbances of the SPDIFRX line, it can happen that the process is not executed first time right. For this purpose, the user can program the number of allowed re-tries (NBTR) before setting SERR error flag.
When the SPDIFRX is able to measure properly the duration of 24 and 40 consecutive symbols then the FINE SYNC is completed, the threshold values are updated, and the flag SYNCD is set to 1. Refer to *Transition coder and preamble detector* for additional information.

Two kinds of errors are detected:

- An overflow of the TRCNT, which generally means that there is no valid S/PDIF stream in the input line. This overflow is indicated by TERR flag.
- The number of retries reached the programmed value. This means that strong jitter is present on the S/PDIF signal. This error is indicated by SERR flag.

When the first FINE SYNC is completed, the reception of channel status (C) and user data (U) starts when the next “B” preamble is detected (see *Figure 947*).Then the user can read IEC-60958 C and U bits through SPDIFRX_CSR register. According to this information the user can then select the proper settings for DRFMT and RXSTEO. For example if the user detects that the current audio stream transports encoded data, then he can put RXSTEO to 0, and DRFMT to 10 prior to start data reception. Note that DRFMT and RXSTEO cannot be modified when SPDIFRXEN = 11. Writes to these fields are ignored if SPDIFRXEN is already 11, though these field can be changed with the same write instruction that causes SPDIFRXEN to become 11. Then the SPDIFRX waits for SPDIFRXEN = 11 and the “B” preamble before starting saving audio samples.

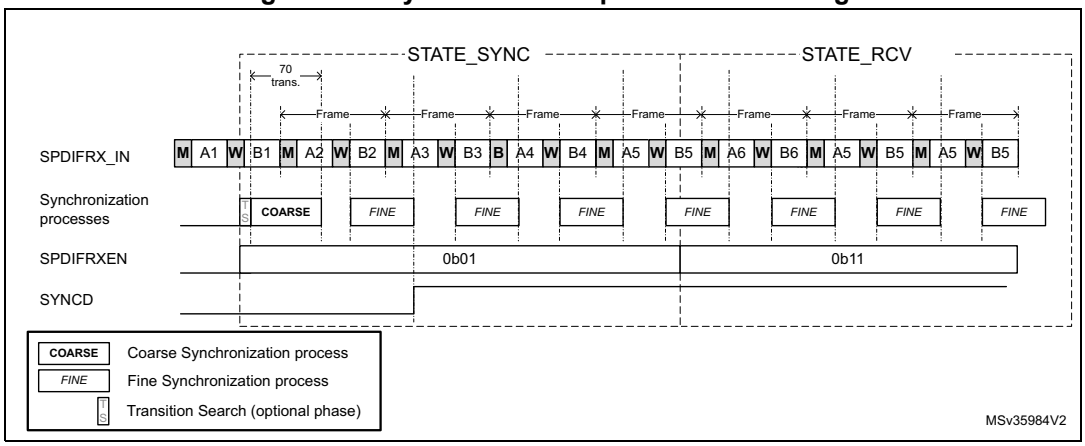
Figure 943. Synchronization flowchart



Refer to *Frame structure and synchronization error* for additional information concerning TRCNT overflow.

The FINE SYNC process is re-triggered every frame in order to update thresholds as shown in *Figure 944* in order to continuously track S/PDIF synchronization.

Figure 944. Synchronization process scheduling



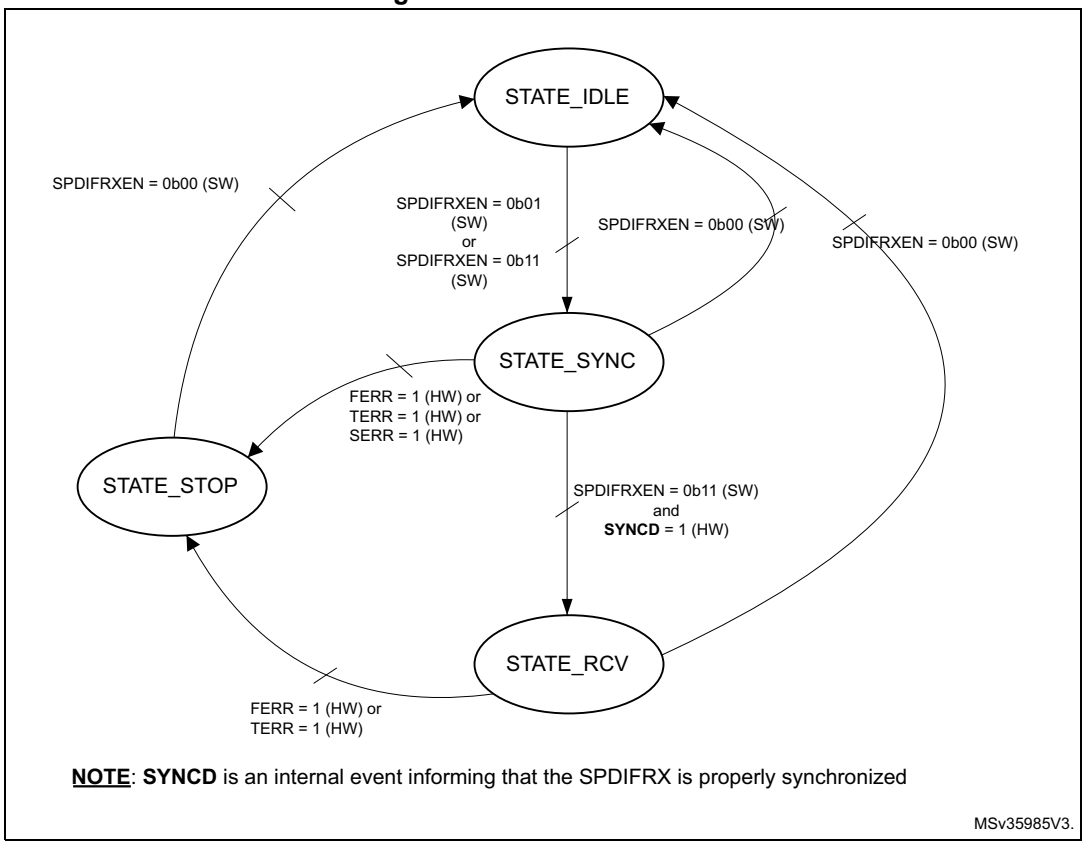
69.3.6 SPDIFRX handling

The software can control the state of the SPDIFRX through SPDIFRXEN field. The SPDIFRX can be into one of the following states:

- STATE_IDLE:
  The peripheral is disabled, the spdifrx_ker_ck domain is reset. The spdifrx_pclk domain is functional.
- STATE_SYNC:
  The peripheral is synchronized to the stream, thresholds are updated regularly, user and channel status can be read via interrupt of DMA. The audio samples are not provided to receive buffer.
- STATE_RCV:
  The peripheral is synchronized to the stream, thresholds are updated regularly, user, channel status and audio samples can be read via interrupt or DMA channels. When SPDIFRXEN goes to 11, the SPDIFRX waits for “B” preamble before starting saving audio samples.
- STOP_STATE:
  The peripheral is no longer synchronized, the reception of the user, channel status and audio samples are stopped. It is expected that the software re-starts the SPDIFRX.

Figure 945 shows the possible states of the SPDIFRX, and how to transition from one state to the other. The bits under software control are followed by the mention “(SW)”, the bits under SPDIFRX control are followed by the mention “(HW)”.

Figure 945. SPDIFRX states



When SPDIFRX is in STATE_IDLE:

- The software can transition to STATE_SYNC by setting SPDIFRXEN to 01 or 11

When SPDIFRX is in STATE_SYNC:

- If the synchronization fails or if the received data are not properly decoded with no chance of recovery without a re-synchronization (FERR or SERR or TERR = 1), the SPDIFRX goes to STATE_STOP, and waits for software acknowledge.
- When the synchronization phase is completed, if SPDIFRXEN = 01 the peripheral remains in this state.
- At any time the software can set SPDIFRXEN to 0, then SPDIFRX returns immediately to STATE_IDLE. If a DMA transfer is on-going, it is properly completed.
- The SPDIFRX goes to STATE_RCV if SPDIFRXEN = 11 and if the SYNCD = 1

When SPDIFRX is in STATE_RCV:

- If the received data are not properly decoded with no chance of recovery without a re-synchronization (FERR or SERR or TERR = 1), the SPDIFRX goes to STATE_STOP, and waits for software acknowledge.
- At any time the software can set SPDIFRXEN to 0, then SPDIFRX returns immediately to STATE_IDLE. If a DMA transfer is on-going, it is properly completed.

When SPDIFRX is in STATE_STOP:

- The SPDIFRX stops reception and synchronization, and waits for the software to set the bit SPDIFRXEN to 0, in order to clear the error flags.

When SPDIFRXEN is set to 0, the SPDIFRX is disabled, meaning that all the state machines are reset, and RX_BUF is flushed. Note as well that flags FERR, SERR and TERR are reset.

69.3.7 Data reception management

The SPDIFRX offers a double buffer for the audio sample reception. A 32-bit buffer located into the spdifrx_ker_ck clock domain (RX_BUF), and the SPDIFRX_FMTx_DR register. The valid data contained into the RX_BUF are immediately transferred into SPDIFRX_FMTx_DR if SPDIFRX_FMTx_DR is empty.

The valid data contained into the RX_BUF are transferred into SPDIFRX_FMTx_DR when the two following conditions are reached:

- The transition between the parity bit (P) and the next preamble is detected (this indicated that the word is completely received).
- The SPDIFRX_FMTx_DR is empty.

Having a 2-word buffer gives more flexibility for the latency constraint.

The maximum latency allowed is $T_{SAMPLE} - 2T_{PCLK} - 2T_{spdifrx_ker_ck}$

Where T_{SAMPLE} is the audio sampling rate of the received stereo audio samples, T_{PCLK} is the period of spdifrx_pclk clock, and $T_{spdifrx_ker_ck}$ is the period of spdifrx_ker_ck clock.

The SPDIFRX offers the possibility to use either DMA (spdifrx_dat_dma and spdifrx_ctrl_dma) or interrupts for transferring the audio samples into the memory. The recommended option is DMA, refer to *Section 69.3.12: DMA interface* for additional information.

The SPDIFRX offers several way on handling the received data. The user can either have a separate flow for control information and audio samples, or get them all together.

For each sub-frame, the data reception register SPDIFRX_FMTx_DR contains the 24 data bits, and optionally the V, U, C, PE status bits, and the PT (see *Mixing data and control flow*).

Note that PE bit stands for parity rrror bit, and is set to 1 when a parity error is detected in the decoded sub-frame.
The PT field carries the preamble type (B, M or W).

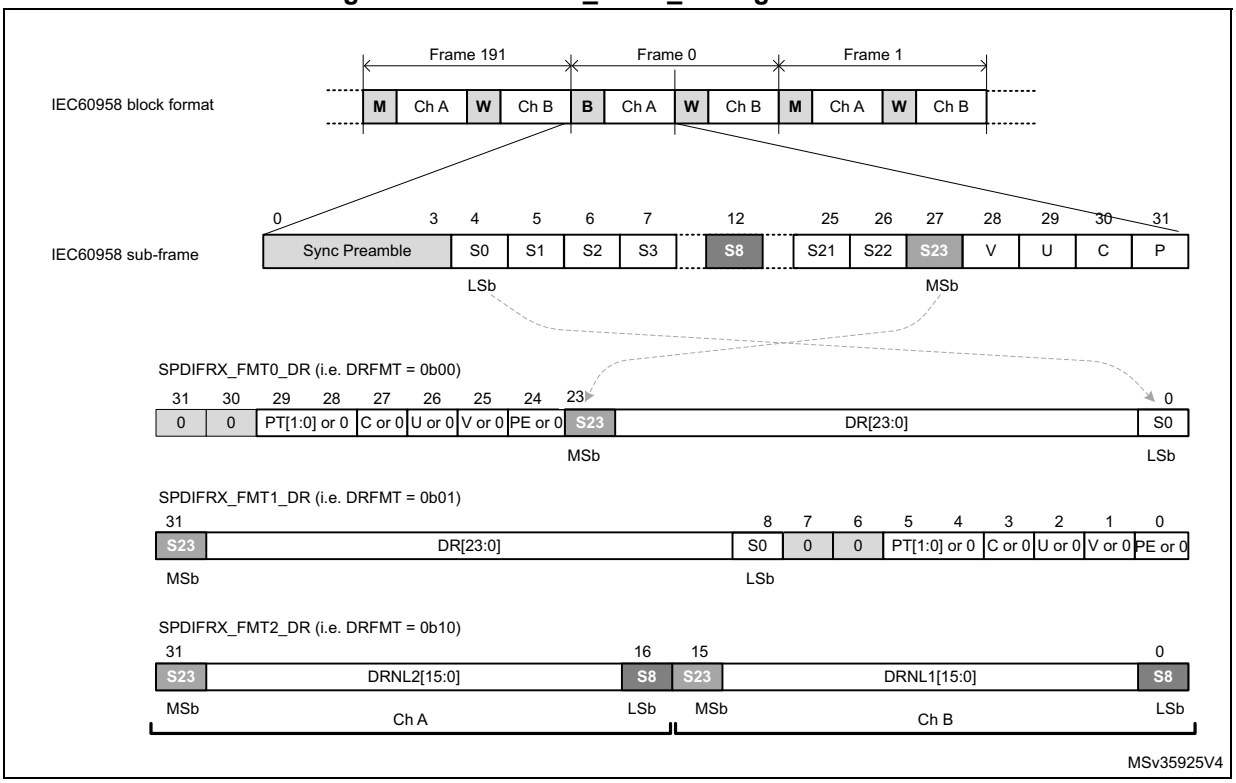
V, U and C are a direct copy of the value received from the S/PDIF interface.

The bit DRFMT allows the selection between 3 audio formats as shown in *Figure 946*.

This document describes 3 data registers: SPDIFRX_FMTx[2:0] (x = 2 to 0), but in reality there is only one physical data register, having 3 possible formats:

- When DRFMT = 0, the format of the data register is the one described by SPDIFRX_FMT0_DR
- When DRFMT = 1, the format of the data register is the one described by SPDIFRX_FMT1_DR
- When DRFMT = 2, the format of the data register is the one described by SPDIFRX_FMT2_DR"

Figure 946. SPDIFRX_FMTx_DR register format



Setting DRFMT to 00 or 01, offers the possibility to have the data either right or left aligned into the SPDIFRX_FMTx_DR register. The status information can be enabled or forced to zero according to the way the software wants to handle them.

The format given by DRFMT= 10 is interesting in non-linear mode, as only 16 bits per sub-frame are used. By using this format, the data of two consecutive sub-frames are stored into SPDIFRX_FMTx_DR, dividing by two the amount of memory footprint. Note that when RXSTEO = 1, there is no misalignment risks (i.e. data from ChA are always stored into SPDIFRX_FMTx_DR[31:16]). If RXSTEO = 0, then there is a misalignment risk is case of overrun situation. In that case SPDIFRX_FMTx_DR[31:16] always contain the oldest value and SPDIFRX_FMTx_DR[15:0] the more recent value (see *Figure 948*).

In this format the status information cannot be mixed with data, but the user can still get them through SPDIFRX_CSR register, and use a dedicated DMA channel or interrupt to transfer them to memory (see *Section 69.3.8*)

Mixing data and control flow

The user can choose to use this mode in order to get the full flexibility of the handling of the control flow. The user can select which field must be kept into the data register (SPDIFRX_FMTx_DR).

- When bit PMSK = 1, the parity error information is masked (set to 0), otherwise it is copied into SPDIFRX_FMTx_DR.
- When bit VMSK = 1, the validity information is masked (set to 0), otherwise it is copied into SPDIFRX_FMTx_DR.

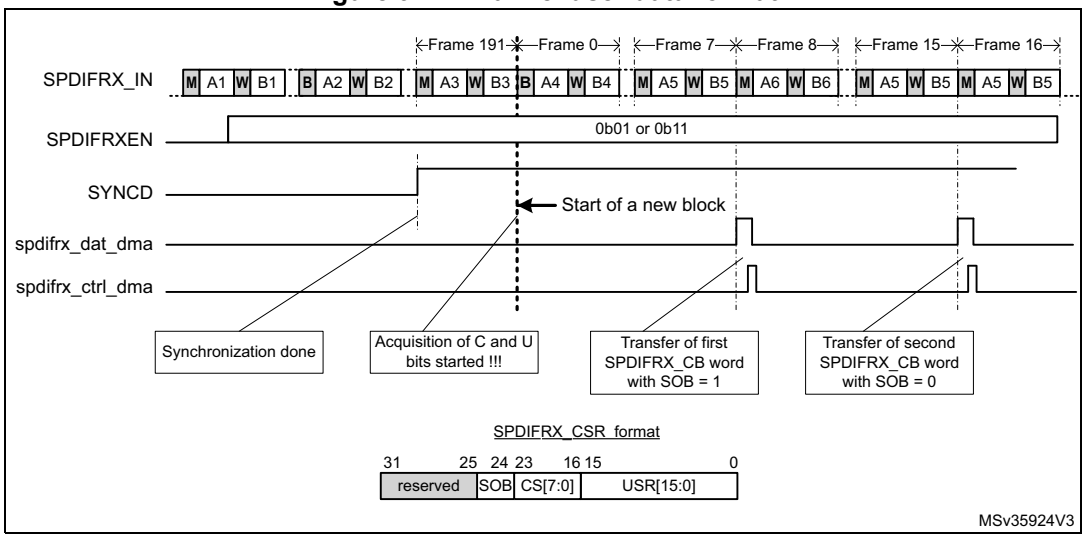
- When bit CUMSK = 1, the channel status, and used data information are masked (set to 0), otherwise they are copied into SPDIFRX_FMTx_DR.
- When bit PTMSK = 1, the preamble type is masked (set to 0), otherwise it is copied into SPDIFRX_FMTx_DR.

69.3.8 Dedicated control flow

The SPDIFRX offers the possibility to catch both user data and channel status information via a dedicated DMA channel. This feature allows the SPDIFRX to acquire continuously the channel status and user information. The acquisition starts at the beginning of a IEC 60958 block. Two fields are available to control this path: CBDMAEN and SPDIFRXEN. When SPDIFRXEN is set to 01 or 0x11, the acquisition is started, after completion of the synchronization phase. When 8 channel status and 16 user data bits are received, they are packed and stored into SPDIFRX_CSR register. A DMA request is triggered if the bit CBDMAEN is set to 1 (see *Figure 947*).

If CS[0] corresponds to the first bit of a new block, the bit SOB is set to 1. Refer to *Section 69.5.8*. A bit is available (CHSEL) in order to select if the user wants to select channel status information (C) from the channel A or B.

Figure 947. Channel/user data format



Note: Once the first start of block is detected (B preamble), the SPDIFRX is checking the preamble type every 8 frames.

Note: Overrun error on SPDIFRX_FMTx_DR register does not affect this path.

69.3.9 Reception errors

Frame structure and synchronization error

The SPDIFRX, detects errors, when one of the following condition occurs:

- The FERR bit is set to 1 on the following conditions:
  - For each of the 28 information bits, if one symbol transition sequence is not correct: for example if short pulses are not grouped by pairs.
  - If preambles occur to an unexpected place, or an expected preamble is not received.
- The SERR bit is set when the synchronization fails, because the number of re-tries exceeded the programmed value.
- The TERR bit is set when the counter used to estimate the width between two transitions overflows (TRCNT).
  The overflow occurs when no transition is detected during 8192 periods of spdifrx_ker_ck clock. It represents at most a time interval of 11.6 frames.

When one of those flags goes to 1, the traffic on selected SPDIFRX_IN is then ignored, an interrupt is generated if the IFEIE bit of the SPDIFRX_CR register is set.

The normal procedure when one of those errors occur is:

- Set SPDIFRXEN to 0 in order to clear the error flags
- Set SPDIFRXEN to 01 or 11 in order to restart the SPDIFRX

Refer to *Figure 945* for additional information.

Parity error

For each sub-frame, an even number of zeros and ones is expected inside the 28 information bits. If not, the parity error bit PERR is set in the SPDIFRX_SR register and an interrupt is generated if the parity interrupt enable PERRIE bit is set in the SPDIFRX_CR register. The reception of the incoming data is not paused, and the SPDIFRX continue to deliver data to SPDIFRX_FMTx_DR even if the interrupt is still pending.

The interrupt is acknowledged by clearing the PERR flag through PERRCF bit.

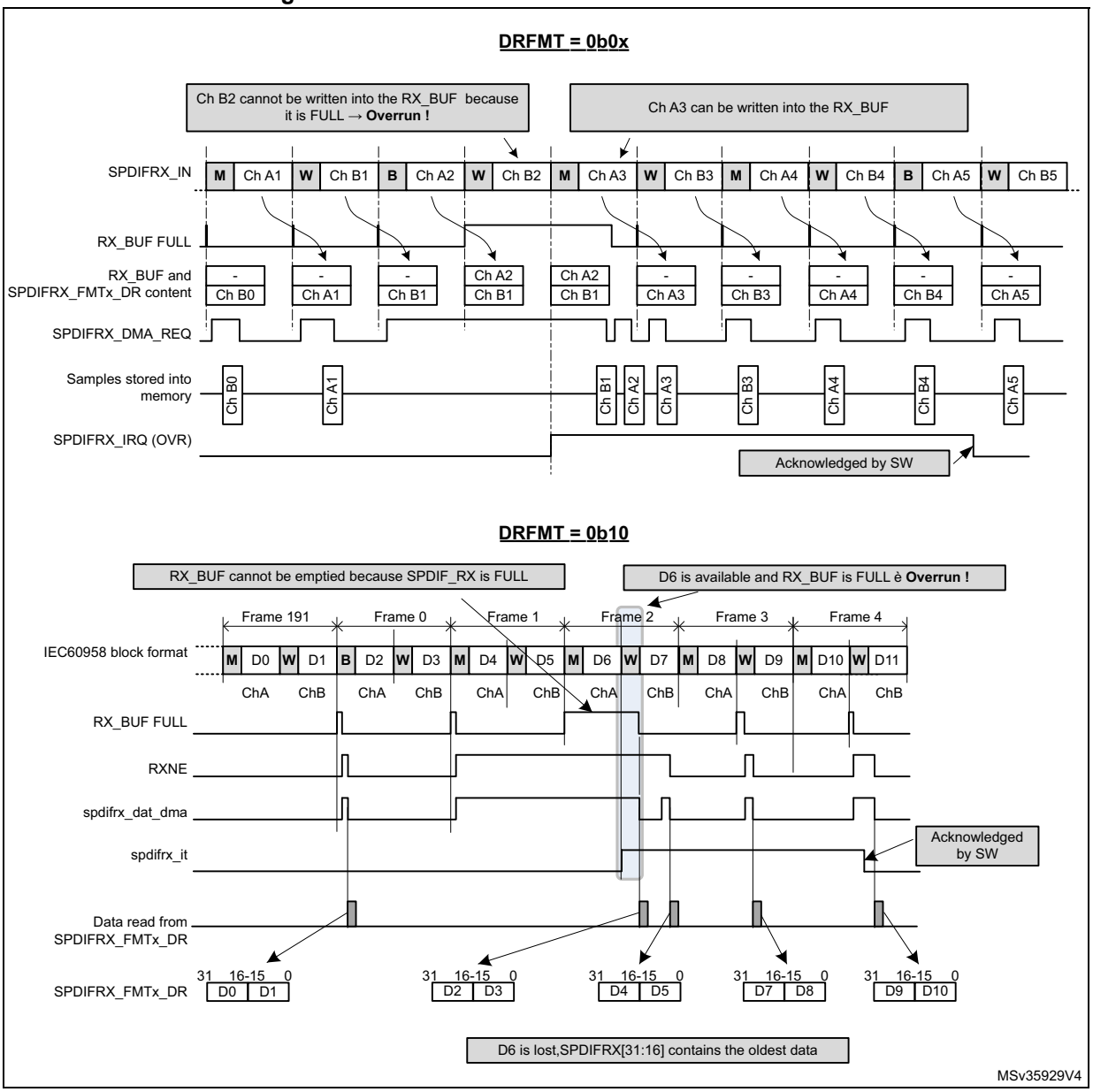
If the software wants to guarantee the coherency between the data read in the SPDIFRX_FMTx_DR register and the value of the bit PERR, the bit PMSK must be set to 0.

Overrun error

If both SPDIFRX_FMTx_DR and RX_BUF are full, while the SPDIFRX_DC needs to write a new sample in RX_BUF, this new sample is dropped, and an overrun condition is triggered. The overrun error flag OVR is set in the SPDIFRX_SR register and an interrupt is generated if the OVRIE bit of the SPDIFRX_CR register is set.

If the RXSTEO bit is set to 0, then as soon as the RX_BUF is empty, the SPDIFRX stores the next incoming data, even if the OVR flag is still pending. The main purpose is to reduce as much as possible the amount of lost samples. Note that the behavior is similar independently of DRFMT value. See *Figure 948*.

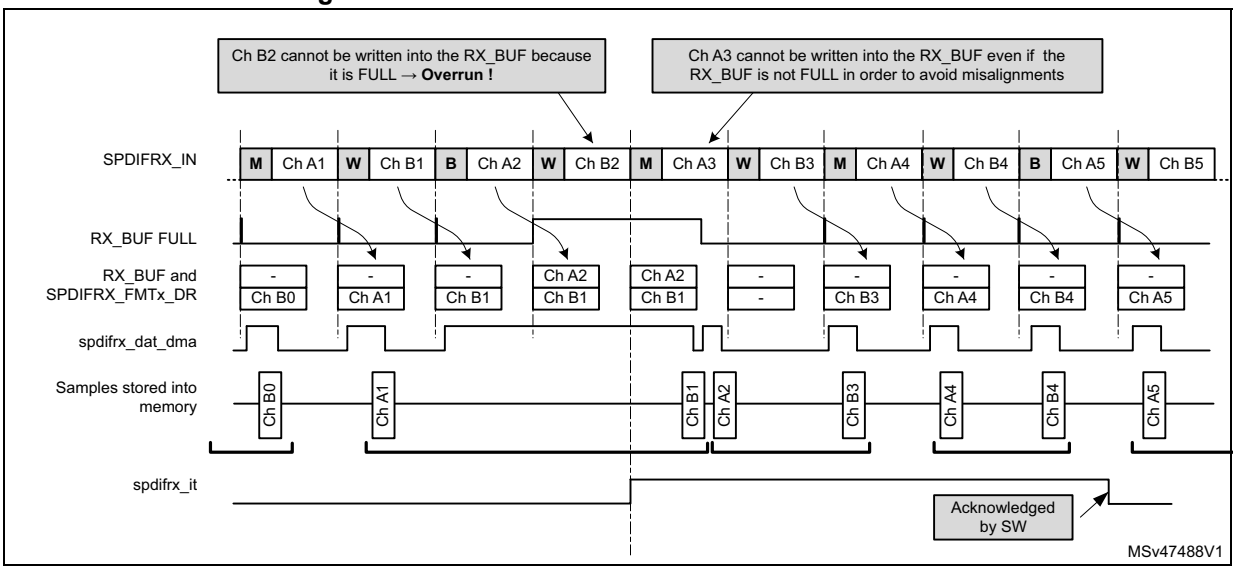
Figure 948. S/PDIF overrun error when RXSTEO = 0



If the RXSTEO bit is set to 1, it means that stereo data are transported, then the SPDIFRX has to avoid misalignment between left and right channels. So the peripheral has to drop a second sample even if there is room inside the RX_BUF in order to avoid misalignment. Then the incoming samples can be written normally into the RX_BUF even if the OVR flag is still pending. Refer to *Figure 949*.

The OVR flag is cleared by software, by setting the OVRCF bit to 1.

Figure 949. S/PDIF overrun error when RXSTEO = 1



69.3.10 Clocking strategy

The SPDIFRX block needs two different clocks:

- The APB clock (spdifrx_pclk), which is used for the register interface,
- The spdifrx_ker_ck which is mainly used by the SPDIFRX_DC part. Those clocks are not supposed to be phase locked, so all signals crossing those clock domains are re-synchronized (SYNC block on *Figure 935*).

In order to decode properly the incoming S/PDIF stream the SPDIFRX_DC must re-sample the received data with a clock at least 11 times higher than the maximum symbol rate, or 704 times higher than the audio sample rate. For example if the user expects to receive a symbol rate to up to 12.288 MHz, the sample rate must be at least 135.2 MHz. The clock used by the SPDIFRX_DC is the spdifrx_ker_ck.

The frequency of the spdifrx_pclk must be at least equal to the symbol rate.

Table 684. Minimum spdifrx_ker_ck frequency versus audio sampling rate(1)

Symbol rate	Minimum spdifrx_ker_ck frequency	Comments
3.072 MHz	33.8 MHz	For 48 kHz stream
6.144 MHz	67.6 MHz	For 96 kHz stream
12.288 MHz	135.2 MHz	For 192 kHz stream

1. Check the RCC capabilities in order to verify which sampling rates can be supported.

69.3.11 Symbol clock generation

The SPDIFRX block provides a symbol clock on signal named **spdifrx_symb_ck**, which can be used as the reference kernel clock for another audio device such as SAI or SPI/I2S. It can be used for SPDIFRX to I2S bridge function.

The symbol clock is built using the values of WIDTH24, WIDTH40 and the symbol boundaries.

- During the reception of the sub-frame sync preambles, the falling and rising edges of the symbol clock are built from the WIDTH24 and WIDTH40 values. Note that WIDTH24 and WIDTH40 are also used for the generation of the symbol clock, when the SPDIFRX is STATE_STOP or STATE_IDLE. See *Table 685* for details.
- During the reception of the sub-frame payload, the SPDIFRX uses the symbols boundaries to generate the rising edge, the WIDTH24 and WIDTH40 values for the generation of the falling edge.

The duty cycle of the symbol clock is close to 50% during the reception of the sub-frame payload. However, the duty cycle can be altered when the SPDIFRX transitions from a symbol clock generated with WIDTH24 and WIDTH40 to a clock generated by the symbol clock boundaries or vice-versa.

The symbol clock has an important jitter mainly due to:

- The re-sampling of the S/PDIF signal with spdifrx_ker_ck clock
- The transition of the symbol clock generation mode

For that reason the application must consider the quality degradation if the symbol clock is used as the reference clock for A/D or D/A converters.

The generation of this symbol clock is controlled by the CKSEN bit. When CKSEN = ‘1’, the clock symbol is generated when the SPDIFRX completes successfully the first fine synchronization (SYNCD = 1), and when it receives correct data from the selected SPDIFRX input.

When the SPDIFRX goes to STATE_STOP, or STATE_IDLE, the symbol clock is gated if the bit CKSBKPEN = ‘0’. If the CKSBKPEN = ‘1’, then a backup symbol clock is still generated if the SPDIFRX is properly synchronized (i.e. valid values available for WIDTH24 and WIDTH40). *Table 685* gives more details on the conditions controlling the generation of the symbol clock.

Table 685. Conditions of spdifrx_symb_ck generation

SPDIFRX states and conditions	CKSEN	CKSBKPEN	spdifrx_symb_ck state
Any state	0	X	Disabled
– SPDIFRX in STATE_SYNC and completing successfully the fine synchronization (SYNCD = '1') or, – SPDIFRX in STATE_RCV, and valid data are received via the selected SPDIFRX input.	1	0	Enabled
– SPDIFRX in STATE_IDLE or, – SPDIFRX in STATE_STOP or, – SPDIFRX did not complete the fine synchronization (on-going) – SPDIFRX is in STATE_RCV, but no data (transitions) detected on the selected SPDIFRX input.		0	Disabled

Table 685. Conditions of spdifrx_symb_ck generation (continued)

SPDIFRX states and conditions	CKSEN	CKSBKPEN	spdifrx_symb_ck state
– SPDIFRX in STATE_IDLE, but with valid values for WIDTH40 and WIDTH24 or – SPDIFRX in STATE_SYNC and completing successfully the fine synchronization (SYNCD = '1') or, – SPDIFRX in STATE_SYNC the on-going fine synchronization is not completed, but WIDTH40 and WIDTH24 contain the valid values from the previous synchronization or, – SPDIFRX in STATE_RCV, and valid data are received via the selected SPDIFRX input or, – SPDIFRX in STATE_STOP, but with valid values for WIDTH40 and WIDTH24.	1	1	Enabled
– SPDIFRX in IDLE, with invalid values for WIDTH40 and WIDTH24 or, – SPDIFRX in STOP with invalid values for WIDTH40 and WIDTH24 (SERR = '1') or, – SPDIFRX in STATE_SYNC with invalid values for WIDTH40 and WIDTH24, and did not completed the on-going fine synchronization or, – SPDIFRX in STATE_RCV and no transitions detected on the selected SPDIFRX input			Disabled

Note that when the flag SERR is set to ‘1’, neither the symbol clock nor the backup clock can be generated, since there is no synchronization.

Note that when both CKSEN and CKSBKPEN are set to ‘1’, the symbol clock looses some transitions when the SPDIFRX switches from STATE_SYNC or STATE_RCV to STATE_STOP, or STATE_IDLE.

The bits CKSEN and CKSBKPEN are located into *SPDIFRX control register (SPDIFRX_CR)*.

69.3.12 DMA interface

The SPDIFRX interface is able to perform communication using the DMA.

Note: The user must refer to product specifications for availability of the DMA controller.

The SPDIFRX offers two independent DMA channels:

- A DMA channel dedicated to the data transfer
- A DMA channel dedicated to the channel status and user data transfer

The DMA mode for the data can be enabled for reception by setting the RXDMAEN bit in the SPDIFRX_CR register. In this case, as soon as the SPDIFRX_FMTx_DR is not empty, the SPDIFRX interface sends a transfer request to the DMA. The DMA reads the data received through the SPDIFRX_FMTx_DR register without CPU intervention.

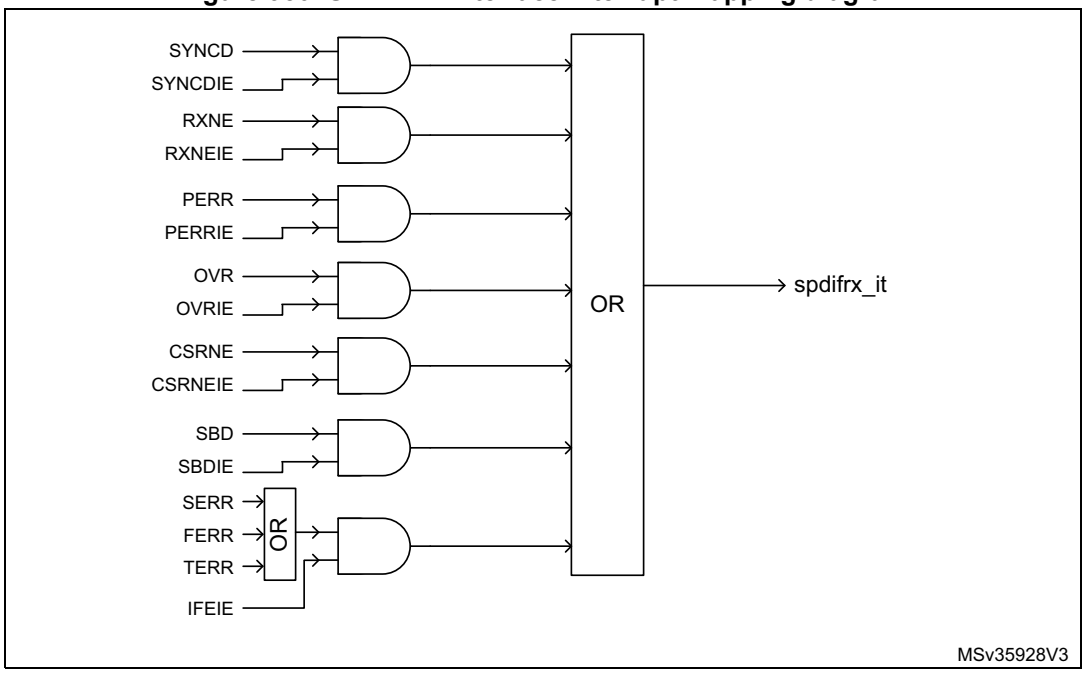
For the use of DMA for the control data refer to *Section 69.3.8*.

69.3.13 Interrupt generation

An interrupt line is shared between:

- Reception events for data flow (RXNE)
- Reception event for control flow (CSRNE)
- Data corruption detection (PERR)
- Transfer flow interruption (OVR)
- Frame structure and synchronization errors (SERR, TERR and FERR)
- Start of new block interrupt (SBD)
- Synchronization done (SYNCD)

Figure 950. SPDIFRX interface interrupt mapping diagram



Clearing interrupt source

- RXNE is cleared when SPDIFRX_FMTx_DR register is read
- CSRNE is cleared when SPDIFRX_CSR register is read
- FERR is cleared when SPDIFRXEN is set to 0
- SERR is cleared when SPDIFRXEN is set to 0
- TERR is cleared when SPDIFRXEN is set to 0
- Others are cleared through SPDIFRX_IFCR register

Note: The SBD event can only occur when the SPDIFRX is synchronized to the input stream (SYNCD = 1).
The SBD flag behavior is not guaranteed when the sub-frame which contains the B preamble is lost due to an overrun.

69.3.14 Register protection

The SPDIFRX block embeds some hardware protection avoid erroneous use of control registers. The table hereafter shows the bit field properties according to the SPDIFRX state.

Table 686. Bit field property versus SPDIFRX state

Registers	Field	SPDIFRXEN		
		00 (STATE_IDLE)	01 (STATE_SYNC)	11 (STATE_RCV)
SPDIFRX_CR	INSEL	rw	r	r
	WFA	rw	r	r
	NBTR	rw	r	r
	CHSEL	rw	r	r
	CBDMAEN	rw	rw	rw
	PTMSK	rw	rw	rw
	CUMSK	rw	rw	rw
	VMSK	rw	rw	rw
	PMSK	rw	rw	rw
	DRFMT	rw	rw	r
	RXSTEO	rw	rw	r
	RXDMAEN	rw	rw	rw
SPDIFRX_IMR	All fields	rw	rw	rw

The table clearly shows that fields such as INSEL must be programmed when the SPDIFRX is in STATE_IDLE. In the others SPDIFRX states, the hardware prevents writing to this field.

Note: Even if the hardware allows the writing of CBDMAEN and RXDMAEN "on-the-fly", it is not recommended to enable the DMA when the SPDIFRX already receives data.

Each of the mask bits (such as PMSK, VMSK) can be changed "on-the-fly" at any SPDIFRX state, but any change does not affect data which are already hold in SPDIFRX_FMTx_DR.

69.4 Programming procedures

The following example illustrates a complete activation sequence of the SPDIFRX block. The data path and channel status and user information both use a dedicated DMA channel. The activation sequence is then split into the following steps:

- Wait for valid data on the selected SPDIFRX_IN input
- Synchronize to the S/PDIF stream
- Read the channel status and user information in order to setup the complete audio path
- Start data acquisition

A simple way to check if valid data are available into the SPDIFRX_IN line is to switch the SPDIFRX into the STATE_SYNC, with bit WFA set to 1. The description hereafter focuses on detection. It is also possible to implement this function as follows:

- The software has to check from time to time (i.e. every 100 ms for example) if the SPDIFRX can find synchronization. This can be done by checking if the bit TERR is set. When it is set it indicates that no activity as been found.
- Connect the SPDIFRX_IN input to an external interrupt event block in order to detect transitions of SPDIFRX_IN line. When activity is detected, then SPDIFRXEN can be set to 01 or 11.

  For those two implementations, the bit WFA is set to 0.

69.4.1 Initialization phase

- The initialization function looks like this:
- Configure the DMA transfer for both audio samples and IEC60958 channel status and user information (DMA channel selection and activation, priority, number of data to transfer, circular/no circular mode, DMA interrupts)
- Configure the destination address:
  - Configure the address of the SPDIFRX_CSR register as source address for IEC60958 channel status and user information
  - Configure the address of the SPDIFRX_FMTx_DR register as source address for audio samples
  - Enable the generation of the spdifrx_ker_ck. Refer to *Table 684* in order to define the minimum clock frequency versus supported audio sampling rate.
    Note that the audio sampling rate of the received stream is not known in advance. This means that the user has to select a spdifrx_ker_ck frequency at least 704 times higher than the maximum audio sampling rate the application is supposed to handle: for example if the application is able to handle streams to up to 96 kHz, then $F_{spdifrx_ker_ck}$ must be at least 704 x 96 kHz = 67.6 MHz
- Enable interrupt for errors and event signaling (IFEIE = SYNCDIE = OVRIE, PERRIE = 1, others set to 0). Note that SYNCDIE can be set to 0.
- Configure the SPDIFRX_CR register:
  - INSEL must select the wanted input
  - NBTR = 2, WFA = 1 (16 re-tries allowed, wait for activity before going to synchronization phase),
  - PTMSK = CUMSK = 1 (Preamble, C and U bits are not mixed with data)
  - VMSK = PMSK = 0 (Parity error and validity bit mixed with data)
  - CHSEL = 0 (channels status are read from sub-frame A)
  - DRFMT = 01 (data aligned to the left)
  - RXSTEO = 1 (expected stereo mode linear)
  - CBDMAEN = RXDMAEN = 1 (enable DMA channels)
  - SPDIFRXEN = 01 (switch SPDIFRX to STATE_SYNC)
- The CPU can enter in WFI mode

Then the CPU receives interrupts coming either from DMA or SPDIFRX.

69.4.2 Handling of interrupts coming from SPDIFRX

When an interrupt from the SPDIFRX is received, then the software has to check what is the source of the interrupt by reading the SPDIFRX_SR register.

- If SYNCD is set to 1, then it means that the synchronization is properly completed. No action has to be performed in our case as the DMA is already programmed. The software just needs to wait for DMA interrupt in order to read channel status information.
  The SYNCD flag must be cleared by setting SYNCDCF bit of SPDIFRX_IFCR register to 1.
- If TERR or SERR or FERR are set to 1, the software has to set SPDIFRXEN to 0, and re-start from the initialization phase.
  - TERR indicates that a time-out occurs either during synchronization phase or after.
  - SERR indicates that the synchronization fails because the maximum allowed re-tries are reached.
  - FERR indicates that the reading of information after synchronization fails (such as unexpected preamble, bad data decoding).
- If PERR is set to 1, it means that a parity error is detected, so one of the received audio sample or the channel status or user data bits are corrupted. The action taken here depends on the application: one action can be to drop the current channel status block as it is not reliable. There is no need to re-start from the initialization phase, as the synchronization is not lost.
  The PERR flag must be cleared by setting PERRCF bit of SPDIFRX_IFCR register to 1.

69.4.3 Handling of interrupts coming from DMA

If an interrupt comes from the DMA channel used of the channel status (SPDIFRX_CSR):

If no error occurred (that is PERR), the CPU can start the decoding of channel information. For example bit 1 of the channel status informs the user if the current stream is linear or not. This information is very important in order to set-up the proper processing chain. In the same way, bits 24 to 27 of the channel status give the sampling frequency of the stream incoming stream.

Thanks to that information, the user can then configure the RXSTEO bit and DRFMT field prior to start the data reception. For example if the current stream is non linear PCM then RXSTEO is set to 0, and DRFMT is set to 10. Then the user can enable the data reception by setting SPDIFRXEN to 11.

The SOB bit, when set to 1 indicates the start of a new block. This information helps the software to identify the bit 0 of the channel status. Note that if the DMA generates an interrupt every time 24 values are transferred into the memory, then the first word always corresponds to the start of a new block.

If an interrupt comes from the DMA channel used of the audio samples (SPDIFRX_FMTx_DR):

The process performed here depends of the data type (linear or non-linear), and on the data format selected.

For example in linear mode, if PE or V bit is set a special processing can be performed locally in order to avoid spurs on output. In non-linear mode those bits are not important as data frame have their own checksum.

69.5 SPDIFRX interface registers

69.5.1 SPDIFRX control register (SPDIFRX_CR)

Only 32-bit accesses are allowed in this register.

Address offset: 0x00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CKSBKPEN	CKSEN	Res.	INSEL[2:0]		
										rw	rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	WFA	NBTR[1:0]		CHSEL	CBDMAEN	PTMSK	CUMSK	VMSK	PMSK	DRFMT[1:0]		RXSTEO	RXDMAEN	SPDIFRXEN[1:0]	
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **CKSBKPEN**: Backup symbol clock enable
This bit is set/reset by software.
1: The SPDIFRX generates a backup symbol clock if CKSEN = 1.
0: The SPDIFRX does not generate a backup symbol clock.

Bit 20 **CKSEN**: Symbol clock enable
This bit is set/reset by software.
1: The SPDIFRX generates a symbol clock.
0: The SPDIFRX does not generate a symbol clock.

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **INSEL[2:0]**: SPDIFRX input selection(1)
000: SPDIFRX_IN0 selected
001: SPDIFRX_IN1 selected
010: SPDIFRX_IN2 selected
011: SPDIFRX_IN3 selected
other: reserved

Bit 15 Reserved, must be kept at reset value.

Bit 14 **WFA**: Wait for activity(1)
This bit is set/reset by software.
1: The SPDIFRX waits for activity on SPDIFRX_IN line (4 transitions) before performing the synchronization.
0: The SPDIFRX does not wait for activity on SPDIFRX_IN line before performing the synchronization.

Bits 13:12 **NBTR[1:0]**: Maximum allowed re-tries during synchronization phase(1)
00: No re-try is allowed (only one attempt)
01: 3 re-tries allowed
10: 15 re-tries allowed
11: 63 re-tries allowed

Bit 11 **CHSEL**: Channel selection(1)
This bit is set/reset by software.
1: The control flow takes the channel status from channel B.
0: The control flow takes the channel status from channel A.

Bit 10 **CBDMAEN**: Control buffer DMA enable for control flow(1)
This bit is set/reset by software.
1: DMA mode is enabled for reception of channel status and used data information.
0: DMA mode is disabled for reception of channel status and used data information.
Note: When this bit is set, the DMA request is made whenever the CSRNE flag is set.

Bit 9 **PTMSK**: Mask of preamble type bits(1)
This bit is set/reset by software.
1: The preamble type bits are not copied into the SPDIFRX_FMTx_DR, zeros are written instead.
0: The preamble type bits are copied into the SPDIFRX_FMTx_DR.

Bit 8 **CUMSK**: Mask of channel status and user bits(1)
This bit is set/reset by software.
1: The channel status and user bits are not copied into the SPDIFRX_FMTx_DR, zeros are written instead.
0: The channel status and user bits are copied into the SPDIFRX_FMTx_DR.

Bit 7 **VMSK**: Mask of validity bit(1)
This bit is set/reset by software.
1: The validity bit is not copied into the SPDIFRX_FMTx_DR, a zero is written instead.
0: The validity bit is copied into the SPDIFRX_FMTx_DR.

Bit 6 **PMSK**: Mask parity error bit(1)
This bit is set/reset by software.
1: The parity error bit is not copied into the SPDIFRX_FMTx_DR, a zero is written instead.
0: The parity error bit is copied into the SPDIFRX_FMTx_DR.

Bits 5:4 **DRFMT[1:0]**: RX data format(1)
This bit is set/reset by software.
11: reserved
10: Data sample are packed by setting two 16-bit sample into a 32-bit word.
01: Data samples are aligned in the left (MSB)
00: Data samples are aligned in the right (LSB).

Bit 3 **RXSTEO**: Stereo mode(1)

This bit is set/reset by software.

1: The peripheral is in stereo mode.

0: The peripheral is in mono mode.

Note: This bit is used in case of overrun situation in order to handle misalignment.

Bit 2 **RXDMAEN**: Receiver DMA enable for data flow(1)

This bit is set/reset by software.

1: DMA mode is enabled for reception.

0: DMA mode is disabled for reception.

Note: When this bit is set, the DMA request is made whenever the RXNE flag is set.

Bits 1:0 **SPDIFRXEN[1:0]**: Peripheral block enable(1)

This field is modified by software.

It must be used to change the peripheral phase among the three possible states: STATE_IDLE, STATE_SYNC and STATE_RCV.

00: Disable SPDIFRX (STATE_IDLE).

01: Enable SPDIFRX synchronization only.

10: Reserved

11: Enable SPDIF receiver.

Note: It is not possible to transition from STATE_RCV to STATE_SYNC, the user must first go the STATE_IDLE.

it is possible to transition from STATE_IDLE to STATE_RCV: in that case the peripheral transitions from STATE_IDLE to STATE_SYNC and as soon as the synchronization is performed goes to STATE_RCV.

1. Refer to *Section 69.3.14: Register protection* for additional information on fields properties.

69.5.2 SPDIFRX interrupt mask register (SPDIFRX_IMR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IFE IE	SYNCD IE	SBLK IE	OVR IE	PERR IE	CSRNE IE	RXNE IE
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bit 6 **IFEIE**: Serial interface error interrupt enable

This bit is set and cleared by software.

0: Interrupt is inhibited.

1: A SPDIFRX interface interrupt is generated whenever SERR = 1, TERR = 1, or FERR = 1 in the SPDIFRX_SR register.

Bit 5 **SYNCDIE**: Synchronization done

This bit is set and cleared by software.

0: Interrupt is inhibited.

1: A SPDIFRX interface interrupt is generated whenever SYNCD = 1 in the SPDIFRX_SR register.

Bit 4 **SBLKIE**: Synchronization block detected interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited.
1: A SPDIFRX interface interrupt is generated whenever SBD = 1 in the SPDIFRX_SR register.

Bit 3 **OVRIE**: Overrun error interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited.
1: A SPDIFRX interface interrupt is generated whenever OVR=1 in the SPDIFRX_SR register.

Bit 2 **PERRIE**: Parity error interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited.
1: A SPDIFRX interface interrupt is generated whenever PERR=1 in the SPDIFRX_SR register.

Bit 1 **CSRNEIE:** Control buffer ready interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited.
1: A SPDIFRX interface interrupt is generated whenever CSRNE = 1 in the SPDIFRX_SR register.

Bit 0 **RXNEIE**: RXNE interrupt enable
This bit is set and cleared by software.
0: Interrupt is inhibited.
1: A SPDIFRX interface interrupt is generated whenever RXNE=1 in the SPDIFRX_SR register.

69.5.3 SPDIFRX status register (SPDIFRX_SR)

Address offset: 0x08

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	WIDTH5[14:0]														
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERR	SERR	FERR	SYNCD	SBD	OVR	PERR	CSRNE	RXNE
							r	r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bits 30:16 **WIDTH5[14:0]**: duration of 5 symbols counted with spdifrx_ker_ck
This value represents the amount of spdifrx_ker_ck clock periods contained on a length of 5 consecutive symbols. This value can be used to estimate the S/PDIF symbol rate. Its accuracy is limited by the frequency of spdifrx_ker_ck.
For example if the spdifrx_ker_ck is fixed to 84 MHz, and WIDTH5 = 147d. The estimated sampling rate of the S/PDIF stream is:
Fs = 5 x $F_{spdifrx_ker_ck}$ / (WIDTH5 x 64) ~ 44.6 kHz, so the closest standard sampling rate is 44.1 kHz.
Note that WIDTH5 is updated by the hardware when SYNCD goes high, and then every frame.

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **TERR**: Time-out error

This bit is set by hardware when the counter TRCNT reaches its max value. It indicates that the time interval between two transitions is too long. It generally indicates that there is no valid signal on SPDIFRX_IN input.

This flag is cleared by writing SPDIFRXEN to 0.

An interrupt is generated if IFEIE=1 in the SPDIFRX_IMR register.

0: No sequence error is detected.

1: Sequence error is detected.

Bit 7 **SERR**: Synchronization error

This bit is set by hardware when the synchronization fails due to amount of re-tries for NBTR.

This flag is cleared by writing SPDIFRXEN to 0.

An interrupt is generated if IFEIE = 1 in the SPDIFRX_IMR register.

0: No synchronization error is detected.

1: Synchronization error is detected.

Bit 6 **FERR**: Framing error

This bit is set by hardware when an error occurs during data reception: such as preamble not at the expected place, short transition not grouped by pairs.

This is set by the hardware only if the synchronization is completed (SYNCD = 1).

This flag is cleared by writing SPDIFRXEN to 0.

An interrupt is generated if IFEIE=1 in the SPDIFRX_IMR register.

0: No Manchester violation detected

1: Manchester violation detected

Bit 5 **SYNCD**: Synchronization done

This bit is set by hardware when the initial synchronization phase is properly completed.

This flag is cleared by writing a 1 to its corresponding bit on SPDIFRX_IFCR register.

An interrupt is generated if SYNCDIE = 1 in the SPDIFRX_IMR register.

0: Synchronization is pending.

1: Synchronization is completed.

Bit 4 **SBD**: Synchronization block detected

This bit is set by hardware when a “B” preamble is detected.

This flag is cleared by writing a 1 to its corresponding bit on SPDIFRX_IFCR register.

An interrupt is generated if SBLKIE = 1 in the SPDIFRX_IMR register.

0: No “B” preamble is detected.

1: “B” preamble is detected.

Bit 3 **OVR**: Overrun error

This bit is set by hardware when a received data is ready to be transferred in the SPDIFRX_FMTx_DR register while RXNE = 1 and both SPDIFRX_FMTx_DR and RX_BUF are full.

This flag is cleared by writing a 1 to its corresponding bit on SPDIFRX_IFCR register.

An interrupt is generated if OVRIE=1 in the SPDIFRX_IMR register.

0: No overrun error

1: Overrun error is detected.

Note: When this bit is set, the SPDIFRX_FMTx_DR register content is not lost but the last data received are.

Bit 2 **PERR**: Parity error
This bit is set by hardware when the data and status bits of the sub-frame received contain an odd number of 0 and 1.
This flag is cleared by writing a 1 to its corresponding bit on SPDIFRX_IFCR register.
An interrupt is generated if PIE = 1 in the SPDIFRX_IMR register.
0: No parity error
1: Parity error

Bit 1 **CSRNE**: Control buffer register not empty
This bit is set by hardware when a valid control information is ready.
This flag is cleared when reading SPDIFRX_CSR register.
An interrupt is generated if CBRDYIE = 1 in the SPDIFRX_IMR register.
0: No control word available on SPDIFRX_CSR register
1: A control word is available on SPDIFRX_CSR register.

Bit 0 **RXNE**: Read data register not empty
This bit is set by hardware when a valid data is available into SPDIFRX_FMTx_DR register.
This flag is cleared by reading the SPDIFRX_FMTx_DR register.
An interrupt is generated if RXNEIE=1 in the SPDIFRX_IMR register.
0: Data is not received.
1: Received data is ready to be read.

69.5.4 SPDIFRX interrupt flag clear register (SPDIFRX_IFCR)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYNCDCF	SBDCF	OVRCF	PERRCF	Res.	Res.
										w	w	w	w		

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **SYNCDCF**: clears the synchronization done flag
Writing 1 in this bit clears the flag SYNCD in the SPDIFRX_SR register.
Reading this bit always returns the value 0.

Bit 4 **SBDCF**: clears the synchronization block detected flag
Writing 1 in this bit clears the flag SBD in the SPDIFRX_SR register.
Reading this bit always returns the value 0.

Bit 3 **OVRCF**: clears the overrun error flag
Writing 1 in this bit clears the flag OVR in the SPDIFRX_SR register.
Reading this bit always returns the value 0.

Bit 2 **PERRCF**: clears the parity error flag
Writing 1 in this bit clears the flag PERR in the SPDIFRX_SR register.
Reading this bit always returns the value 0.

Bits 1:0 Reserved, must be kept at reset value.

69.5.5 SPDIFRX data input register (SPDIFRX_FMT0_DR)

Address offset: 0x10

Reset value: 0x0000 0000

This register can take three different formats according to DRFMT. Here is the format when DRFMT = 00:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	PT[1:0]		C	U	V	PE	DR[23:16]							
		r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **PT[1:0]:** preamble type
These bits indicate the preamble received.
00: not used
01: Preamble B received
10: Preamble M received
11: Preamble W received
Note that if PTMSK = 1, this field is forced to zero

Bit 27 **C**: channel status bit
Contains the received channel status bit, if CUMSK = 0, otherwise it is forced to 0

Bit 26 **U**: user bit
Contains the received user bit, if CUMSK = 0, otherwise it is forced to 0

Bit 25 **V**: validity bit
Contains the received validity bit if VMSK = 0, otherwise it is forced to 0

Bit 24 **PE**: parity error bit
Contains a copy of PERR bit if PMSK = 0, otherwise it is forced to 0

Bits 23:0 **DR[23:0]**: data value
Contains the 24 received data bits, aligned on D[23]

69.5.6 SPDIFRX data input register (SPDIFRX_FMT1_DR)

Address offset: 0x10

Reset value: 0x0000 0000

This register can take three different formats according to DRFMT. Here is the format when DRFMT = 01:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DR[23:8]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR[7:0]								Res.	Res.	PT[1:0]		C	U	V	PE
r	r	r	r	r	r	r	r			r	r	r	r	r	r

Bits 31:8 **DR[23:0]**: data value
Contains the 24 received data bits, aligned on D[23]

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **PT[1:0]:** preamble type
These bits indicate the preamble received.
00: not used
01: preamble B received
10: preamble M received
11: preamble W received
Note that if PTMSK = 1, this field is forced to zero

Bit 3 **C**: channel Status bit
Contains the received channel status bit, if CUMSK = 0, otherwise it is forced to 0

Bit 2 **U**: user bit
Contains the received user bit, if CUMSK = 0, otherwise it is forced to 0

Bit 1 **V**: validity bit
Contains the received validity bit if VMSK = 0, otherwise it is forced to 0

Bit 0 **PE**: parity error bit
Contains a copy of PERR bit if PMSK = 0, otherwise it is forced to 0

69.5.7 SPDIFRX data input register (SPDIFRX_FMT2_DR)

Address offset: 0x10

Reset value: 0x0000 0000

This register can take 3 different formats according to DRFMT.

The data format proposed when DRFMT = 10, is dedicated to non-linear mode, as only 16 bits are used (bits 23 to 8 from S/PDIF sub-frame).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DRNL2[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DRNL1[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **DRNL2[15:0]**: data value
This field contains the channel A

Bits 15:0 **DRNL1[15:0]**: data value
This field contains the channel B

69.5.8 SPDIFRX channel status register (SPDIFRX_CSR)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SOB	CS[7:0]							
							r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **SOB**: start of block
This bit indicates if the bit CS[0] corresponds to the first bit of a new block
0: CS[0] is not the first bit of a new block
1: CS[0] is the first bit of a new block

Bits 23:16 **CS[7:0]**: channel A status information
Bit CS[0] is the oldest value

Bits 15:0 **USR[15:0]**: user data information
Bit USR[0] is the oldest value, and comes from channel A, USR[1] comes channel B.
So USR[n] bits come from channel A is n is even, otherwise they come from channel B.

69.5.9 SPDIFRX debug information register (SPDIFRX_DIR)

Address offset: 0x18

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	TLO[12:0]												
			r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	THI[12:0]												
			r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:16 **TLO[12:0]**: threshold LOW (TLO = 1.5 x UI / $T_{spdifrx_ker_ck}$)
This field contains the current threshold LOW estimation. This value can be used to estimate the sampling rate of the received stream. The accuracy of TLO is limited to a period of the spdifrx_ker_ck. The sampling rate can be estimated as follow:
Sampling Rate = [2 x TLO x $T_{spdifrx_ker_ck}$ +/- $T_{spdifrx_ker_ck}$] x 2/3
Note that TLO is updated by the hardware when SYNCD goes high, and then every frame.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:0 **THI[12:0]**: threshold HIGH (THI = 2.5 x UI / $T_{spdifrx_ker_ck}$)
This field contains the current threshold HIGH estimation. This value can be used to estimate the sampling rate of the received stream. The accuracy of THI is limited to a period of the spdifrx_ker_ck. The sampling rate can be estimated as follow:
Sampling Rate = [2 x THI x $T_{spdifrx_ker_ck}$ +/- $T_{spdifrx_ker_ck}$] x 2/5
Note that THI is updated by the hardware when SYNCD goes high, and then every frame.

69.5.10 SPDIFRX interface register map

Table 687. SPDIFRX interface register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	SPDIFRX_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CKSBKPEN	CKSEN	Res.	INSEL[2:0]			Res.	WFA	NBTR[1:0]		CHSEL	CBDMAEN	PTMSK	CUMSK	VMSK	PMSK	DRFMT[1:0]		RXSTEO	RXDMAEN	SPDIFRXEN[1:0]	
	Reset value											0	0		0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x04	SPDIFRX_IMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IFEIE	SYNCDIE	SBLKIE	OVRIE	PERRIE	CSRNEIE	RXNEIE
	Reset value																										0	0	0	0	0	0	0
0x08	SPDIFRX_SR	Res.	WIDTH5[14:0]															Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERR	SERR	FERR	SYNCD	SBD	OVR	PERR	CSRNE	RXNE
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								0	0	0	0	0	0	0	0	0
0x0C	SPDIFRX_IFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYNCDCF	SBDCF	OVRCF	PERRCF	Res.	Res.
	Reset value																											0	0	0	0		
0x10	SPDIFRX_FMT0_DR	Res.	Res.	PT[1:0]		C	U	V	PE	DR[23:0]																							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	SPDIFRX_FMT1_DR	DR[23:0]																								Res.	PT[1:0]			C	U	V	PE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10	SPDIFRX_FMT2_DR	DRNL2[15:0]																DRNL1[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x14	SPDIFRX_CSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SOB	CS[7:0]								USR[15:0]															
	Reset value								0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18	SPDIFRX_DIR	Res.	Res.	Res.	TLO[12:0]													Res.	Res.	Res.	THI[12:0]												
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

70 Management data input/output (MDIOS)

70.1 MDIOS introduction

An MDIO bus can be useful in systems where a master chip needs to manage (configure and get status data from) one or multiple slave chips. The bus protocol uses only two signals:

- MDC: the management data clock
- MDIO: the data line carrying the opcode (write or read), the slave (port) address, the MDIOS register address, and the data

In each transaction, the master either reads the contents of an MDIOS register in one of its slaves, or it writes data to an MDIOS register in one of its slaves.

The MDIOS peripheral serves as a slave interface to a MDIO bus. A MDIO master can use the MDC/MDIO lines to write and read 32 16-bit MDIOS registers, which are held in the MDIOS. These MDIOS registers are managed by the firmware. This allows the MDIO master to configure the application running on the STM32 and get status information from it.

The MDIOS can operate in Stop mode, optionally waking up the STM32 if the MDIO master performs a read or a write to one of its MDIOS registers.

70.2 MDIOS main features

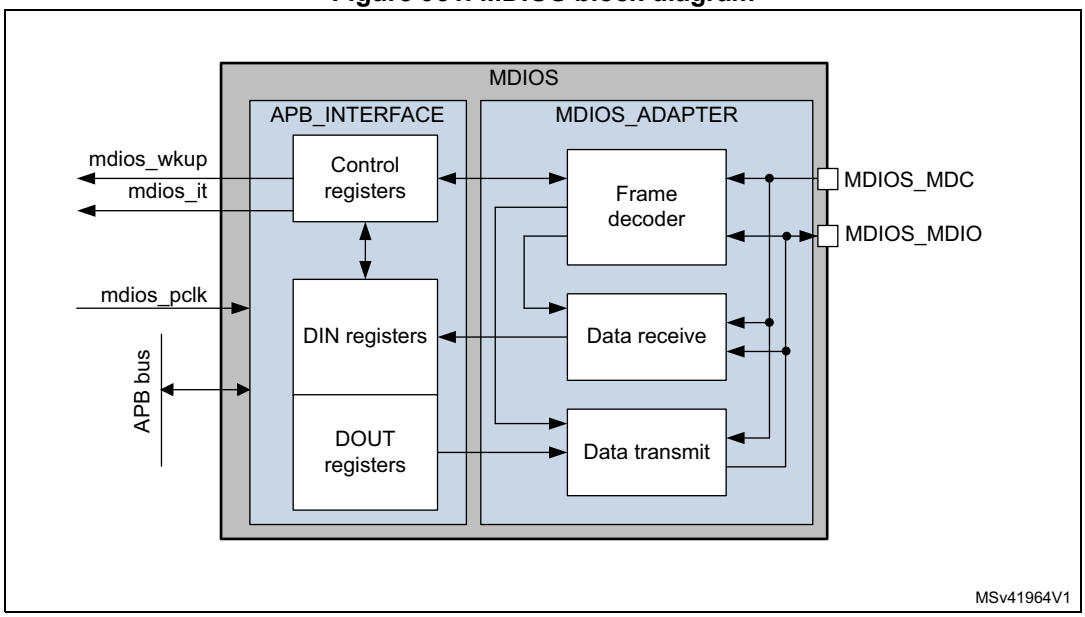
The MDIOS includes the following features:

- 32 MDIOS register addresses, each of which is managed using separate input and output data registers:
  - 32 x 16-bit firmware read/write, MDIOS read-only output data registers
  - 32 x 16-bit firmware read-only, MDIOS write-only input data registers
- Configurable slave (port) address
- Independently maskable interrupts/events:
  - MDIOS register write
  - MDIOS register read
  - MDIOS protocol error
- Able to operate in and wake up from Stop mode

70.3 MDIOS functional description

70.3.1 MDIOS block diagram

Figure 951. MDIOS block diagram



70.3.2 MDIOS pins and internal signals

Table 688 lists the MDIOS inputs and output signals connected to package pins or balls, while *Table 689* shows the internal signals.

Table 688. MDIOS input/output signals connected to package pins or balls

Signal name	Signal type	Description
MDIOS_MDC	Digital input	MDIO master clock
MDIOS_MDIO	Digital input/output	MDIO signal (opcode, address, input/output data)

Table 689. MDIOS internal input/output signals

Signal name	Signal type	Description
mdios_wkup	Digital output	MDIOS wake-up signal
mdios_it		MDIOS interrupt signal
mdios_pclk	Digital input	APB clock

70.3.3 MDIOS protocol

The MDIOS protocol uses two signals:

- MDIOS_MDC: the clock, always driven by the master
- MDIOS_MDIO: signal carrying the opcode, address, and bidirectional data

Each transaction is performed using a frame, which contains 32 bits (each passed serially):

- 14 control bits, driven by the master
  - 2 start bits: always 01
  - 2 opcode bits: read = 10, write = 01
  - 5 port address bits, indicating which slave device is being addressed
  - 5 MDIOS register address bits, up to 32 MDIOS registers can be addressed in each slave
- 2 turnaround state bits
  - On write operations, the master drives 10
  - On read operations, the first bit is high-impedance, and the second bit is driven by the slave to 0
- 16 data bits
  - On write operations, data written to slave MDIOS register is driven by the master
  - On read operations, data read from slave MDIOS register is driven by the slave

Each frame is usually preceded by a preamble, where the MDIO stays at one for 32 MDC clocks. The master can continue to keep MDIO at one, indicating the idle condition, when it has no frame to send.

When MDIO signal is driven by the master, MDIOS samples it using the rising edge of MDC. When MDIOS drives MDIO, the output changes on the rising edge of MDC.

Figure 952. MDIO protocol write frame waveform

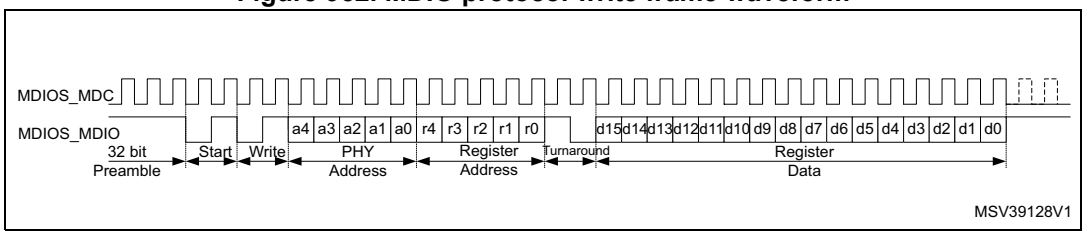
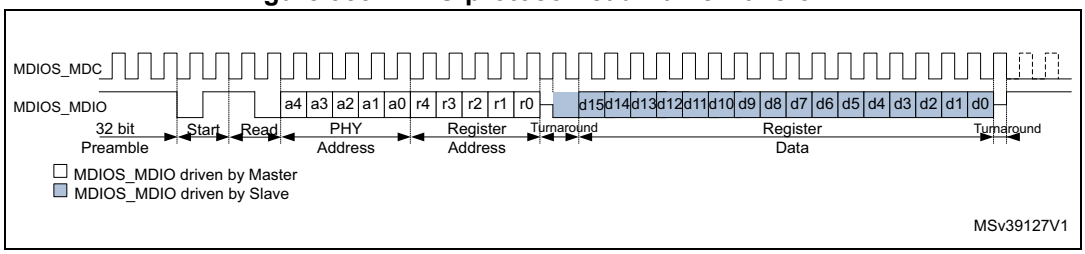


Figure 953. MDIO protocol read frame waveform



70.3.4 MDIOS enabling and disabling

The MDIOS is enabled by setting the EN bit in MDIOS_CR. When EN = 1, the MDIOS monitors the MDIO bus and service frames addressed to one of its MDIOS registers.

When the MDIOS is enabled (setting EN to one), the same write operation to MDIOS_CR must properly set the PORT_ADDRESS[4:0] field to indicate the slave port address. A frame is ignored by the MDIOS if its port address is not the same as PORT_ADDRESS[4:0] (presumably intended for another slave).

When EN = 0, the MDIOS ignores the frames being transmitted on the MDC/MDIO lines, and the IP is in a reduced consumption mode. Clearing EN also clears all of the DIN registers. If EN is cleared while the MDIOS is driving read data, it immediately releases the bus and does not drive the rest of the data. If EN is cleared while the MDIOS is receiving a frame, the frame is aborted and the data is lost.

When the MDIOS is enabled, disabled and reenabled, the status flags are not cleared. For a correct operation, the firmware must clear the status flag before reenabling the MDIOS.

70.3.5 MDIOS data

From the point of view of the MDIO master, there are 32 16-bit MDIOS registers in the MDIOS, which can be written and read. In reality, for each MDIOS register ‘x’ there are two sets of registers: MDIOS_DINRx and MDIOS_DOUTRx.

Input data

When the MDIO master transmits a frame, which writes to MDIOS register ‘x’ in the MDIOS, it is MDIOS_DINRx, which is updated with the incoming data. The registers MDIOS_DINR0 to MDIOS_DINR31 can be read by the firmware, but they can be written only by the MDIO master via the MDIO bus.

The contents of MDIOS_DINRx change immediately after the MDC rising edge when the last data bit is sampled.

If the firmware happens to read the contents of MDIOS_DINRx at the moment that it is being updated, there is a possibility that the value read is corrupted (a bit-by-bit cross between the old value and the new value). For this reason, **the firmware must assure that two subsequent reads from the same MDIOS_DINRx give the same value and assure that the data is stable when it is read**. In the very worst case, the firmware needs to read four times MDIOS_DINRx to:

1. Get the old value
2. Get an incoherent value (when reading at the moment the register changes)
3. Get the new value
4. Confirm the new value

If the firmware uses the WRF interrupt and can guarantee that it reads the MDIOS_DINRx register before any new MDIOS write frame completes, the firmware can perform a single read.

If the MDIO master performs a write operation with a register address, which is greater than 31, the MDIOS ignores the frame (the data is not saved and no flag is set).

Output data

When the MDIOS receives a frame, which requests to read register ‘x’, it returns the value found in MDIOS_DOUTRx. Thus, if the MDIO master expects to read the same value, which it previously wrote to MDIOS register ‘x’, the firmware must copy the data from MDIOS_DINRx to MDIOS_DOUTRx each time new data is written to MDIOS_DINRx. For correct operation, the firmware must copy the data to the MDIOS_DOUTRx register within a preamble (if the master sends preambles before each frame) plus 15 cycles time.

When an MDIOS register is read via the MDIO bus, the MDIOS passes the 16-bit value (from the corresponding MDIOS_DOUTRx) to the MDIOS clock domain during the 15th cycle of the read frame. If the firmware attempts to write MDIOS_DOUTRx while the MDIO

master is currently reading MDIOS register ‘x’, the firmware write operation is ignored if it occurs during the 15th cycle of the frame (during a one-MDC-cycle window). Therefore, **after writing MDIOS_DOUTRx, the firmware must read back the same MDIOS_DOUTRx and confirm that the value was actually written**. If MDIOS_DOUTRx does not contain the value, which was written, the firmware can simply try writing and rereading again.

If the MDIOS frequency is very slow compared to the mdios_pclk frequency, it is better not to tie up the CPU by continuously writing and rereading MDIOS_DOUTRx.

Note: The read flag (RDFx) is set as soon as the MDIOS_DOUTRx DOUT[15:0] value is passed to the MDIOS clock domain. Thus, when a write to MDIOS_DOUTRx is ignored (when the value read back is not the value, which was just written), the firmware can use a read interrupt to know when it is able to write MDIOS_DOUTRx.

Respect the following procedure if the MDC clock is very slow:

1. Write MDIOS_DOUTRx.
2. Assure that all of the read flags are zero (MDIOS_RDFR = 0x0000). Clear the flags if necessary using MDIOS_CRDFR.
3. Read back the same MDIOS_DOUTRx and compare the value with the value, which was written in step 1.
4. If the values are the same, the procedure is done. Otherwise, continue to step 5.
5. Enable read interrupts by setting the RDIE bit in MDIOS_CR1.
6. In the interrupt routine, assure that RDFx is set (no other read flags are set before bit x).
7. There is a 31 cycle + preamble time window (if the master sends a preamble before each frame) to write MDIOS_DOUTRx safely without doing a read-back and compare. If this maximum delay cannot be guaranteed, go back to step 1.

If the MDIO master performs a read operation with a register address, which is greater than 31, the MDIOS returns a data value of all zeros.

70.3.6 MDIOS APB frequency

Whenever the firmware reads from MDIOS_DINRx or writes to MDIOS_DOUTRx, the frequency of the APB bus must be at least 1.5 times the MDC frequency. For example, if MDC is at 20 MHz, the APB must be at 30 MHz or faster.

70.3.7 Write/read flags and interrupts

When MDIOS register ‘x’ is written via the MDIO bus, the WRFx bit in MDIOS_WRFR is set. WRFx becomes one at the moment that MDIOS_DINRx is updated, which is when the last data bit is sampled on a write frame. An interrupt is generated if WRIEN = 1 (in MDIOS_CR). WRFx is cleared by software by writing 1 to CWRFx (in MDIOS_CWRFR).

When MDIOS register ‘x’ is read via the MDIO bus, the RDFx bit in MDIOS_RDFR is set. RDFx becomes one at the moment that MDIOS_DOUTRx is copied to the MDC clock domain, which is on the 15th cycle of a read frame. An interrupt is generated if RDIEN = 1 (in MDIOS_CR). RDFx is cleared by software by writing 1 to CRDFx (in MDIOS_CRDFR).

70.3.8 MDIOS error management

There are three types of errors with their corresponding error flags:

- Preamble error: PERF (bit 0 of MDIOS_SR)
- Start error: SERF (bit 1 of MDIOS_SR)
- Turnaround error: TERF (bit 2 of MDIOS_SR)

Each error flag is set by hardware when the corresponding error condition occurs. Each flag can be cleared by writing ‘1’ to the corresponding bit in the clear flag register (MDIOS_CLRFR).

An interrupt occurs if any of the three error flags is set while EIE = 1 (MDIOS_CR).

Besides, setting an error flag, the MDIOS performs no action for a frame in which an error is detected: MDIOS_DINRx are not updated and the MDIO line is not forced during the data phase.

For a given frame, errors do not accumulate. For example, if a preamble error is detected, no check is done for a start error or a turnaround error for the rest of the current frame.

When DPC = 0, following a detected error, all new frames, and errors are ignored until a complete full preamble has been detected.

When DPC = 1 (disable preamble check, MDIOS_CR[7]), all frames and new errors are ignored as long as one of the error flags is set. As soon as the error bit is cleared, the MDIOS starts looking for a start sequence. Thus, the application must clear the error flag only when it is sure that no frame is currently in progress. Otherwise, the MDIOS likely misinterprets the bits being sent and becomes desynchronized with the master.

Preamble errors

A preamble error occurs when a start sequence begins (with MDIO sampled at 0) without being immediately preceded by a preamble (MDIO sampled at 1 for at least 32 consecutive clocks).

Preamble errors are not reported after the MDIOS is first enabled (EN = 1 in MDIOS_CR) until after a full preamble is received. This is to avoid an error condition when the peripheral frame detection is enabled while a preamble or frame is already in progress. In this case, the MDIOS ignores the first frame (since it did not first detect a full preamble), but does not set PERF.

If the DPC bit (disable preamble check, MDIOS_CR[7]) is set, the MDIO master can send frames without preceding preambles and no preamble error is signaled. When DPC = 1, the application must assure that the master is not in the process of sending a frame at the moment that the MDIOS is enabled (EN is set). Otherwise, the slave might become desynchronized with the master.

Start errors

A start error occurs when an illegal start sequence occurs or if an illegal command is given. The start sequence must always be 01, and the command must be either 01 (write) or 10 (read).

As with preamble errors, start errors are not reported until after a full preamble is received.

Turnaround errors

A turnaround error occurs when an error is detected in the turnaround bits of write frames. The 15th bit of the write frame must be 1 and the 16th bit must be 0.

Turnaround errors are only reported after a full preamble is received, there is no start error, the port address in the current frame matches, and the register address is in the supported range 0 to 31.

70.3.9 MDIOS in Stop mode

The MDIOS can operate in Stop mode, responding to all reads, performing all writes, and causing the STM32 to wake up from Stop mode on MDIOS interrupts.

70.3.10 MDIOS interrupts

There is a single interrupt vector for the three types of interrupts (write, read, and error). Any of these interrupt sources can wake the STM32 up from Stop mode. All interrupt flags need to be cleared in order to clear the interrupt line.

Table 690. Interrupt control bits

Interrupt event	Event flag	Enable control bit
Write interrupt	WRF[31:0]	WRIE
Read interrupt	RDF[31:0]	RDIE
Error interrupt	PERF (preamble), SERF (start), TERF (turnaround)	EIE

70.4 MDIOS registers

70.4.1 MDIOS configuration register (MDIOS_CR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	PORT_ADDRESS[4:0]					DPC	Res.	Res.	Res.	EIE	RDIE	WRIE	EN
			rw	rw	rw	rw	rw	rw				rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:8 **PORT_ADDRESS[4:0]**: slave address
Can be written only when the peripheral is disabled (EN = 0).
If the address given by the MDIO master matches PORT_ADRESS[4:0], the MDIOS services the frame. Otherwise, the frame is ignored.

Bit 7 **DPC**: disable preamble check

0: MDIO master must give preamble before each frame.

1: MDIO master can send each frame without a preceding preamble, and the MDIOS does not signal a preamble error.

Note: When this bit is set, the application must be sure that no frame is currently in progress when the MDIOS is enabled. Otherwise, the MDIOS can become desynchronized with the master.

This bit cannot be changed unless EN = 0 (though it can be changed at the same time that EN is being set).

Bits 6:4 Reserved, must be kept at reset value.

Bit 3 **EIE**: error interrupt enable

0: Interrupt is disabled.

1: Interrupt is enabled.

Note: When this bit is set, an interrupt is generated if any of the error flags (PERF, SERF, or TERF in the MDIOS_SR register) is set.

Bit 2 **RDIE**: register read interrupt enable

0: Interrupt is disabled.

1: Interrupt is enabled.

Note: When this bit is set, an interrupt is generated if any of the read flags (RDF[31:0] in the MDIOS_RDFR register) is set.

Bit 1 **WRIE**: register write interrupt enable

0: Interrupt is disabled.

1: Interrupt is enabled.

Note: When this bit is set, an interrupt is generated if any of the read flags (WRF[31:0] in the MDIOS_WRFR register) is set.

Bit 0 **EN**: peripheral enable

0: MDIOS is disabled.

1: MDIOS is enabled and monitoring the MDIO bus (MDC/MDIO).

70.4.2 MDIOS write flag register (MDIOS_WRFR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WRF[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WRF[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **WRF[31:0]**: write flags for MDIOS registers 0 to 31.

Each bit is set by hardware when the MDIO master performs a write to the corresponding MDIOS register. An interrupt is generated if WRIE (in MDIOS_CR) is set.

Each bit is cleared by software by writing ‘1’ to the corresponding CWRF bit in the MDIOS_CWRFR register.

For WRFx:

0: MDIOS register x is not written by the MDIO master.

1: MDIOS register x is written by the MDIO master and the data is available in DIN[15:0] in the MDIOS_DINRx register.

70.4.3 MDIOS clear write flag register (MDIOS_CWRFR)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CWRF[31:16]															
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CWRF[15:0]															
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:0 **CWRF[31:0]**: clear the write flag

Writing 1 to CWRFx clears the WRFx bit in the MDIOS_WRF register.

70.4.4 MDIOS read flag register (MDIOS_RDFR)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDF[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDF[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RDF[31:0]**: read flags for MDIOS registers 0 to 31.

Each bit is set by hardware when the MDIO master performs a read from the corresponding MDIOS register. An interrupt is generated if RDIE (in MDIOS_CR) is set.

Each bit is cleared by software by writing one to the corresponding CRDF bit in the MDIOS_CRDFR register.

For RDFx:

0: MDIOS register x is not read by the MDIO master.

1: MDIOS register x is read by the MDIO master.

70.4.5 MDIOS clear read flag register (MDIOS_CRDFR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CRDF[31:16]															
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRDF[15:0]															
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:0 **CRDF[31:0]**: clear the read flag
Writing 1 to CRDFx clears the RDFx bit in the MDIOS_RDF register.

70.4.6 MDIOS status register (MDIOS_SR)

Address offset: 0x014

Reset value: 0x0000 0000

Writes to this register have no effect.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERF	SERF	PERF
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **TERF**: turnaround error flag
0: No turnaround error has occurred.
1: A turnaround error has occurred.
Note: Writing 1 to CTERF (MDIOS_CLRFR) clears this bit.

Bit 1 **SERF**: start error flag
0: No start error has occurred.
1: A start error has occurred.
Note: Writing 1 to CSERF (MDIOS_CLRFR) clears this bit.

Bit 0 **PERF**: preamble error flag
0: No preamble error has occurred.
1: A preamble error has occurred.
Note: Writing 1 to CPERF (MDIOS_CLRFR) clears this bit.
This bit is not set if DPC (disable preamble check, MDIOS_CR[7]) is set.

70.4.7 MDIOS clear flag register (MDIOS_CLRFR)

Address offset: 0x018

Reset value: 0x0000 0000

Reads on this register returns all zeros.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTERF	CSERF	CPERF
													rc_w1	rc_w1	rc_w1

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **CTERF**: clear the turnaround error flag
Writing 1 to this bit clears the TERF flag (in MDIOS_SR).
When DPC = 1 (MDIOS_CR[7]), the TERF flag must be cleared only when there is not a frame already in progress.

Bit 1 **CSERF**: clear the start error flag
Writing 1 to this bit clears the SERF flag (in MDIOS_SR).
When DPC = 1 (MDIOS_CR[7]), the SERF flag must be cleared only when there is not a frame already in progress.

Bit 0 **CPERF**: clear the preamble error flag
Writing 1 to this bit clears the PERF flag (in MDIOS_SR).

70.4.8 MDIOS input data register x (MDIOS_DINRx)

Address offset: 0x100 + 0x4 * x, (x = 0 to 31)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIN[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **DIN[15:0]**: input data received from MDIO master during write frames
This field is written by hardware with the 16-bit data received in a write frame, which is addressed to MDIOS register x.

70.4.9 MDIOS output data register x (MDIOS_DOUTRx)

Address offset: 0x180 + 0x4 * x, (x = 0 to 31)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DOUT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **DOUT[15:0]**: output data sent to MDIO master during read frames

This field is written by software. These 16 bits are serially output on the MDIO bus during read frames, which address the MDIOS register x.

70.4.10 MDIOS register map

Table 691. MDIOS register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	MDIOS_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PORT_ADDRESS[4:0]					DPC	Res.	Res.	Res.	EIE	RDIE	WRIE	EN
	Reset value																				0	0	0	0	0	0				0	0	0	0
0x004	MDIOS_WRFR	WRF[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	MDIOS_CWRFR	CWRF[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	MDIOS_RDFR	RDF[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x010	MDIOS_CRDFR	CRDF[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x014	MDIOS_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TERF	SERF	PERF
	Reset value																														0	0	0
0x018	MDIOS_CLRFR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CTERF	CSERF	CPERF
	Reset value																														0	0	0
0x01C-0x0FC	Reserved	Reserved																															
0x100 + 0x04 * x (x=0 to 31) Last address: 0x17C	MDIOS_DINRx	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DIN[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 691. MDIOS register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x180 + 0x4 * x (x=0 to 31) Last address: 0x1FC	MDIOS_DOUTRx	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DOUT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

71 Controller area network with flexible data rate (FDCAN)

71.1 FDCAN introduction

The controller area network (CAN) subsystem (see *Figure 954*) consists of three CAN modules, a shared message RAM, and a clock calibration unit. Refer to the product memory organization for the base address of each of them.

FDCAN modules are compliant with ISO 11898-1: 2015 (CAN protocol specification version 2.0 part A, B) and CAN FD protocol specification version 1.0.

In addition, the first CAN module FDCAN1 supports time triggered CAN (TTCAN), specified in ISO 11898-4, including event synchronized time-triggered communication, global system time, and clock drift compensation. The FDCAN1 contains additional registers, specific to the time triggered feature. The CAN FD option can be used together with event-triggered and time triggered CAN communication.

A 10-Kbyte message RAM implements filters, receive FIFOs, receive buffers, transmit event FIFOs, transmit buffers (and triggers for TTCAN). This memory is shared between the FDCAN modules.

The common clock calibration unit is optional. It can be used to generate a calibrated clock for each FDCAN from the HSI internal RC oscillator and the PLL, by evaluating CAN messages received by the FDCAN1.

The CAN subsystem I/O signals and pins are detailed, respectively, in *Table 692* and *Table 693*.

Table 692. CAN subsystem I/O signals

Name	Type	Description
fdcan_ker_ck	Digital input	CAN subsystem kernel clock input
fdcan_pclk		CAN subsystem APB interface clock input
fdcan_cal_it	Digital output	FDCAN calibration interrupt
fdcan1_intr0_it	Digital output	FDCAN1 interrupt0
fdcan1_intr1_it		FDCAN1 interrupt1
fdcan2_intr0_it	Digital output	FDCAN2 interrupt0
fdcan2_intr1_it		FDCAN2 interrupt1
fdcan3_intr0_it	Digital output	FDCAN3 interrupt0
fdcan3_intr1_it		FDCAN3 interrupt1
fdcan1_swt[0:5]	Digital input	Stop watch trigger input
fdcan1_evt[0:5]		Event trigger input
fdcan1_ts[0:15]		External timestamp vector
fdcan1_soc	Digital output	Start of cycle pulse
fdcan1_rtp		Register time mark pulse
fdcan1_tmp		Trigger time mark pulse

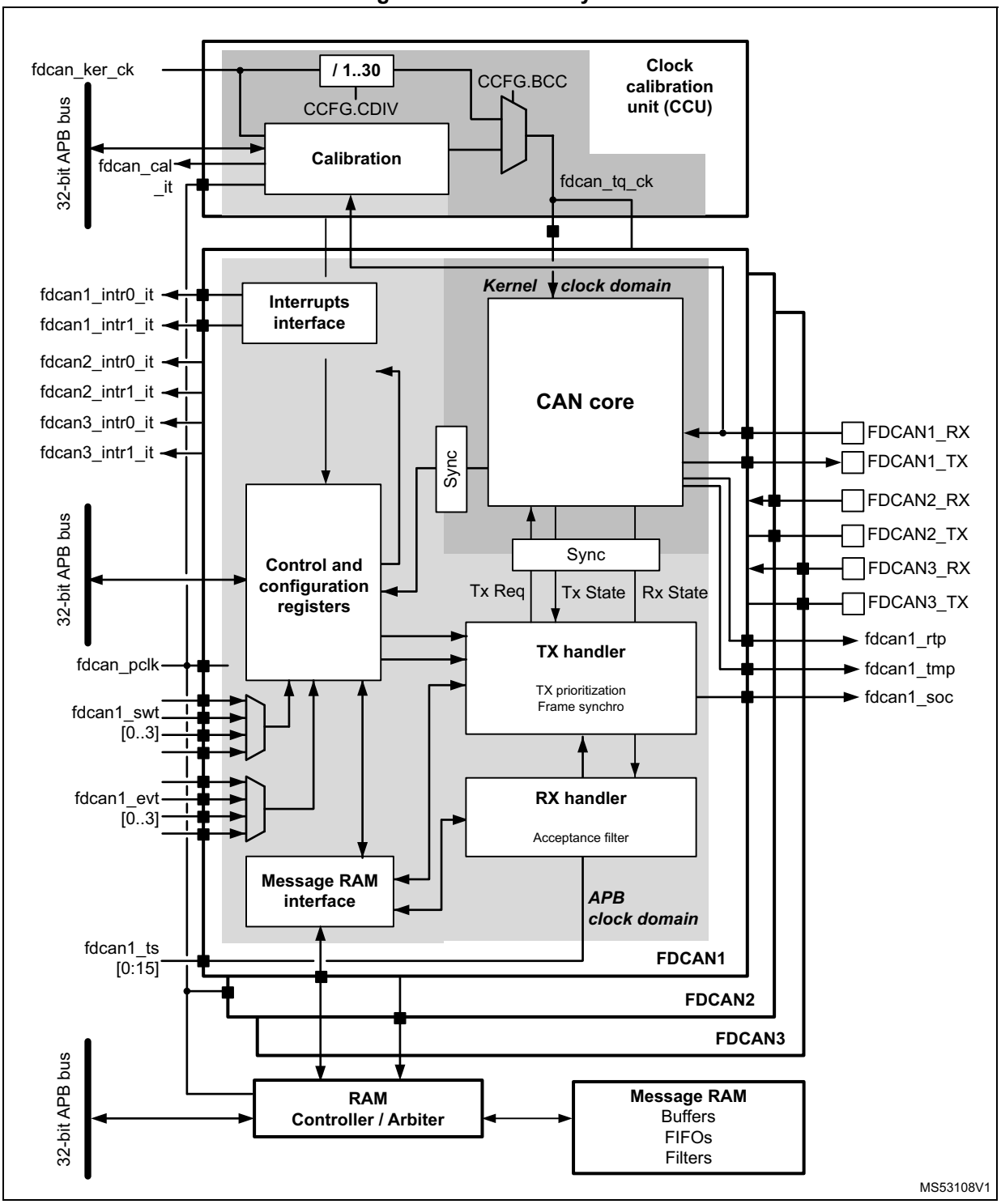
Table 693. CAN subsystem I/O pins

Name	Type	Description
FDCAN1_RX	Digital input	FDCAN1 receive pin
FDCAN1_TX	Digital output	FDCAN1 transmit pin
FDCAN2_RX	Digital input	FDCAN2 receive pin
FDCAN2_TX	Digital output	FDCAN2 transmit pin
FDCAN3_RX	Digital input	FDCAN3 receive pin
FDCAN3_TX	Digital output	FDCAN3 transmit pin

Table 694. CAN triggers

Type	ID	Source	Destination
Internal	SWT0	TIM2_TGO_CKTIM	-
	SWT1	TIM3_TGO_CKTIM	-
	SWT2	ETH1_PPS_OUT	-
	SWT3	-	-
	EVT0	TIM2_TGO_CKTIM	-
	EVT1	TIM3_TGO_CKTIM	-
	EVT2	ETH1_PPS_OUT	-
	EVT3	-	-
External	fdcan1_soc	-	TIM5_ITR13, TIM3_TI1_4
	fdcan1_tmp	-	TIM3_TI1_3, TIM5_TI1_3, ETH1_PTP3
	fdcan1_rtp	-	TIM3_TI1_2, TIM5_TI1_2

Figure 954. CAN subsystem



71.2 FDCAN main features

- Conform with CAN protocol version 2.0 part A, B, and ISO 11898-1: 2015, -4
- CAN FD with max. 64 data bytes supported
- TTCAN protocol level 1 and level 2 completely in hardware (FDCAN1 only)
- Event synchronized time-triggered communication supported (FDCAN1 only)
- CAN error logging
- AUTOSAR and J1939 support
- Improved acceptance filtering
- Two configurable receive FIFOs
- Separate signaling on reception of high priority messages
- Up to 64 dedicated receive buffers
- Up to 32 dedicated transmit buffers
- Configurable transmit FIFO / queue
- Configurable transmit event FIFO
- FDCAN modules share the same message RAM
- Programmable loop-back test mode
- Maskable module interrupts
- Two clock domains: APB bus interface and CAN core kernel clock
- Power-down support

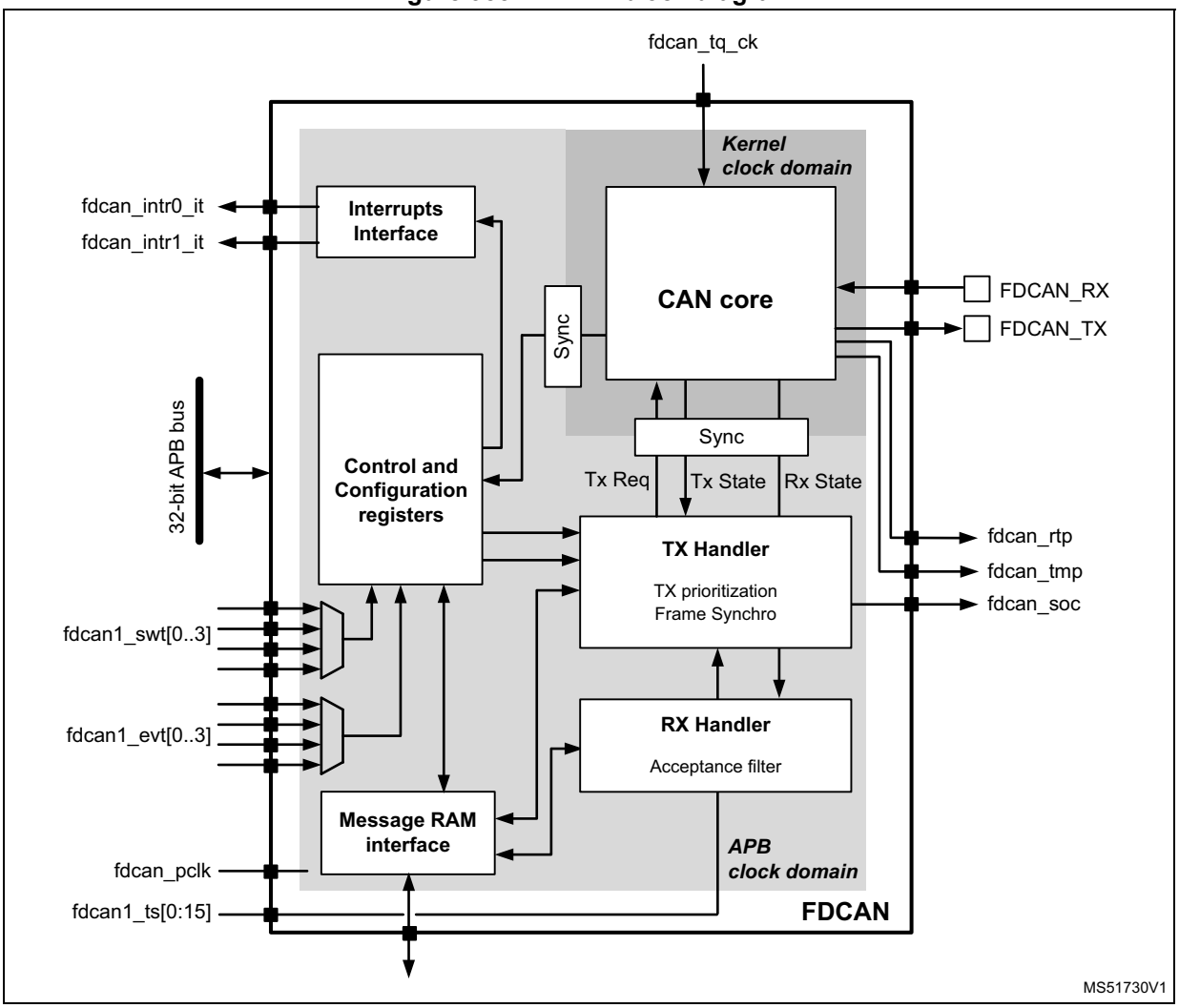
71.3 FDCAN implementation

Table 695. Main features

Module	FDCAN1	FDCAN2	FDCAN3
TTCAN	X	-	-

71.4 FDCAN functional description

Figure 955. FDCAN block diagram



Dual interrupt lines

The FDCAN peripheral provides two interrupt lines, fdcan_intr0_it and fdcan_intr1_it. By programming EINT0 and EINT1 bits in the FDCAN_ILE register, the interrupt lines can be enabled or disabled separately.

CAN core

The CAN core contains the protocol controller and the receive/transmit shift registers. It handles all ISO 11898-1: 2015 protocol functions, and supports both 11-bit and 29-bit identifiers.

Sync

This block synchronizes signals from the APB clock domain to the CAN kernel clock domain, and vice versa.

Tx handler

Controls the message transfer from the message RAM to the CAN core. A maximum of 32 Tx buffers can be configured for transmission. Tx buffers can be used as dedicated Tx buffers, as Tx FIFO, part of a Tx queue, or as a combination of them. A Tx event FIFO stores Tx timestamps together with the corresponding message ID. Transmit cancellation is also supported.

On FDCAN1, the Tx handler also implements the frame synchronization entity (FSE) which controls time-triggered communication according to ISO11898-4. It synchronizes itself with the reference messages on the CAN bus, controls cycle time and global time, and handles transmissions according to the predefined message schedule, the system matrix. It also handles the time marks of the system matrix that are linked to the messages in the message RAM. Stop watch trigger, event trigger, and time mark interrupt are synchronization interfaces.

Rx handler

Controls the transfer of received messages from the CAN core to the external message RAM. The Rx handler supports two receive FIFOs, each of configurable size, and up to 64 dedicated Rx buffers for storage of all messages that have passed acceptance filtering. A dedicated Rx buffer, in contrast to a receive FIFO, is used to store only messages with a specific identifier. An Rx timestamp is stored together with each message. Up to 128 filters can be defined for 11-bit IDs and up to 64 filters for 29-bit IDs.

APB interface

Connects the FDCAN to the APB bus.

Message RAM interface

Connects the FDCAN access to an external 10 Kbytes message RAM through a RAM controller/arbiter.

71.4.1 Operating modes

Software initialization

Software initialization is started by setting the INIT bit in the FDCAN_CCCR register, either by software or by a hardware reset, or by going Bus_Off. While the INIT bit in FDCAN_CCCR register is set, message transfer from and to the CAN bus is stopped, the status of the CAN bus output FDCAN_TX is recessive (high). The counters of the error management logic (EML) are unchanged. Setting INIT bit in FDCAN_CCCR does not change any configuration register. Clearing INIT bit in FDCAN_CCCR finishes the software initialization. Afterwards, the bit stream processor (BSP) synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (Bus_Idle) before it can take part in bus activities and start the message transfer.

Access to the FDCAN configuration registers is only enabled when both INIT bit in the FDCAN_CCCR register and the CCE bit in the FDCAN_CCCR register are set.

The CCE bit in the FDCAN_CCCR register can be set/cleared only while the INIT bit in the FDCAN_CCCR is set. The CCE bit in FDCAN_CCCR register is automatically cleared when the INIT bit in FDCAN_CCCR is cleared.

The following registers are reset when the CCE bit in the FDCAN_CCCR register is set:

- FDCAN_HPMS - high priority message status
- FDCAN_ RXF0S - Rx FIFO 0 status
- FDCAN_RXF1S - Rx FIFO 1 status
- FDCAN_TXFQS - Tx FIFO/queue status
- FDCAN_TXBRP - Tx buffer request pending
- FDCAN_TXBTO - Tx buffer transmission occurred
- FDCAN_TXBCF - Tx buffer cancellation finished
- FDCAN_TXEFS - Tx event FIFO status
- FDCAN_TTOST - TT (time trigger) operation status (FDCAN1 only)
- FDCAN_TTLGT - TT local and global time, only global time FDCAN_TTLGT.GT is reset (FDCAN1 only)
- FDCAN_TTCTC - TT cycle time and count (FDCAN1 only)
- FDCAN_TTCSM - TT cycle sync mark (FDCAN1 only)

The timeout counter value TOC bit in FDCAN_TOCV register is preset to the value configured by TOP bit in FDCAN_TOCC register when CCE bit in FDCAN_CCCR is set.

In addition, the state machines of the Tx handler and Rx handler are held in idle state while the CCE bit in FDCAN_CCCR is set.

The following registers can be written only when the CCE bit in the FDCAN_CCCR register is cleared:

- FDCAN_TXBAR - Tx buffer add request
- FDCAN_TXBCR - Tx buffer cancellation request

TEST bit in FDCAN_CCCR and MON bit in FDCAN_CCCR can be set only by software while both INIT bit and CCE bit in FDCAN_CCCR register are set. Both bits may be reset at any time. DAR bit in FDCAN_CCCR can be set/cleared only while both INIT bit in FDCAN_CCCR and CCE bit in FDCAN_CCCR are set.

Normal operation

The FDCAN1 default operating mode after hardware reset is event-driven CAN communication without time triggers (FDCAN_TTOCF.OM = 00). It is required that both INIT bit and CCE bit in the FDCAN_CCCR register are set before the TT operation mode can be changed.

Once the FDCAN is initialized and the INIT bit in the FDCAN_CCCR register is cleared, the FDCAN synchronizes itself to the CAN bus and is ready for communication.

After passing the acceptance filtering, received messages including message ID and DLC are stored into a dedicated Rx buffer or into the Rx FIFO 0 or Rx FIFO 1.

For messages to be transmitted dedicated Tx buffers and/or a Tx FIFO or a Tx queue can be initialized or updated. Automated transmission on reception of remote frames is not supported.

CAN FD operation

There are two variants in the CAN FD protocol. The first is the long frame mode (LFM), where the data field of a CAN frame can be longer that eight bytes. The second variant is the fast frame mode (FFM), where the control, data, and CRC fields of a CAN frame are

transmitted with a higher bitrate than the beginning and the end of the frame. Fast frame mode can be used in combination with long frame mode.

The previously reserved bit in CAN frames with 11-bit identifiers and the first previously reserved bit in CAN frames with 29-bit identifiers are now decoded as FDF bit. FDF recessive signifies a CAN FD frame, while FDF dominant signifies a classic CAN frame. In a CAN FD frame, the two bits following FDF, res and BRS, decide whether the bitrate inside this CAN FD frame is switched. A CAN FD bitrate switch is signified by res dominant and BRS recessive. The coding of res recessive is reserved for protocol expansions. If the FDCAN receives a frame with FDF recessive and res recessive, it signals a protocol exception event by setting bit FDCAN_PSR.PXE. When protocol exception handling is enabled (FDCAN_CCCR.PXHD = 0), this causes the operation state to change from receiver (FDCAN_PSR.ACT = 10) to integrating (FDCAN_PSR.ACT = 00) at the next sample point. In case protocol exception handling is disabled (FDCAN_CCCR.PXHD = 1), the FDCAN treats a recessive res bit as a form error and responds with an error frame.

CAN FD operation is enabled by programming FDCAN_CCCR.FDOE. If FDCAN_CCCR.FDOE = 1, transmission and reception of CAN FD frames is enabled. Transmission and reception of classic CAN frames is always possible. Whether a CAN FD or a classic CAN frame is transmitted, it can be configured via bit FDF in the respective Tx buffer element. With FDCAN_CCCR.FDOE = 0, received frames are interpreted as classic CAN frames, which leads to the transmission of an error frame when receiving a CAN FD frame. When CAN FD operation is disabled, no CAN FD frames are transmitted, even if bit FDF of a Tx buffer element is set. FDCAN_CCCR.FDOE and FDCAN_CCCR.BRSE can only be changed while FDCAN_CCCR.INIT and FDCAN_CCCR.CCE are both set.

With FDCAN_CCCR.FDOE = 0, the setting of bits FDF and BRS is ignored and frames are transmitted in classic CAN format. With FDCAN_CCCR.FDOE = 1 and FDCAN_CCCR.BRSE = 0, only bit FDF of a Tx buffer element is evaluated. With FDCAN_CCCR.FDOE = 1 and FDCAN_CCCR.BRSE = 1, transmission of CAN FD frames with bitrate switching is enabled. All Tx buffer elements with bits FDF and BRS set are transmitted in CAN FD format with bitrate switching.

A mode change during CAN operation is only recommended under the following conditions:

- The failure rate in the CAN FD data phase is significant higher than in the CAN FD arbitration phase. In this case disable the CAN FD bitrate switching option for transmissions.
- During system startup all nodes are transmitting classic CAN messages until it is verified that they are able to communicate in CAN FD format. If this is true, all nodes switch to CAN FD operation.
- Wake-up messages in CAN partial networking must be transmitted in classic CAN format.
- End-of-line programming in case not all nodes are CAN FD capable. Non CAN FD nodes are held in Silent mode until programming has been completed. Then all nodes switch back to classic CAN communication.

In the CAN FD format, the coding of the DLC differs from the standard CAN format. DLC codes 0 to 8 have the same coding as in standard CAN, codes 9 to 15 (that in standard CAN all code a data field of 8 bytes) are coded according to *Table 696*.

Table 696. DLC coding in FDCAN

DLC	9	10	11	12	13	14	15
Number of data bytes	12	16	20	24	32	48	64

In CAN FD fast frames, the bit timing is switched inside the frame, after the BRS (bitrate switch) bit, if this bit is recessive. Before the BRS bit, in the CAN FD arbitration phase, the nominal CAN bit timing is used as defined by the bit timing and prescaler register FDCAN_NBTP. In the following CAN FD data phase, the fast CAN bit timing is used as defined by the fast bit timing and prescaler register FDCAN_DBTP. The bit timing is switched back from the fast timing at the CRC delimiter or when an error is detected, whichever occurs first.

The maximum configurable bitrate in the CAN FD data phase depends on the FDCAN kernel clock frequency. For example, with an FDCAN kernel clock frequency of 20 MHz and the shortest configurable bit time of four time quanta (tq), the bitrate in the data phase is 5 Mbit/s.

In both data frame formats, CAN FD long frames and CAN FD fast frames, the value of the bit ESI (error status indicator) is determined by the transmitter error state at the start of the transmission. If the transmitter is error passive, ESI is transmitted recessive, else it is transmitted dominant. In CAN FD remote frames the ESI bit is always transmitted dominant, independent of the transmitter error state. The data length code of CAN FD remote frames is transmitted as 0.

In case an FDCAN Tx buffer is configured for CAN FD transmission with DLC > 8, the first eight bytes are transmitted as configured in the Tx buffer while the remaining part of the data field is padded with 0xCC. When the FDCAN receives an FDCAN frame with DLC > 8, the first eight bytes of that frame are stored into the matching Rx buffer or Rx FIFO. The remaining bytes are discarded.

Transceiver delay compensation

During the data phase of a CAN FD transmission only one node is transmitting, all others are receivers. The length of the bus line has no impact. When transmitting via pin FDCAN_TX the protocol controller receives the transmitted data from its local CAN transceiver via pin FDCAN_RX. The received data is delayed by the CAN transceiver loop delay. If this delay is greater than TSEG1 (time segment before sample point), a bit error is detected. Without transceiver delay compensation, the bitrate in the data phase of a CAN FD frame is limited by the transceivers loop delay.

The FDCAN implements a delay compensation mechanism to compensate the CAN transceiver loop delay, thereby enabling transmission with higher bitrates during the CAN FD data phase, independent from the delay of a specific CAN transceiver.

To check for bit errors during the data phase of transmitting nodes, the delayed transmit data is compared against the received data at the Secondary Sample Point SSP. If a bit error is detected, the transmitter reacts on this bit error at the next following regular sample point. During the arbitration phase, the delay compensation is always disabled.

The transmitter delay compensation enables configurations where the data bit time is shorter than the transmitter delay, it is described in detail in the new ISO11898-1. It is enabled by setting bit FDCAN_DBTP.TDC.

The received bit is compared against the transmitted bit at the SSP. The SSP position is defined as the sum of the measured delay from the FDCAN transmit output pin FDCAN_TX through the transceiver to the receive input pin FDCAN_RX plus the transmitter delay compensation offset as configured by FDCAN_TDCR.TDCO. The transmitter delay compensation offset is used to adjust the position of the SSP inside the received bit (for example half of the bit time in the data phase). The position of the secondary sample point is rounded down to the next integer number of minimum time quanta (mtq, that is, one period of fdcan_tq_ck clock).

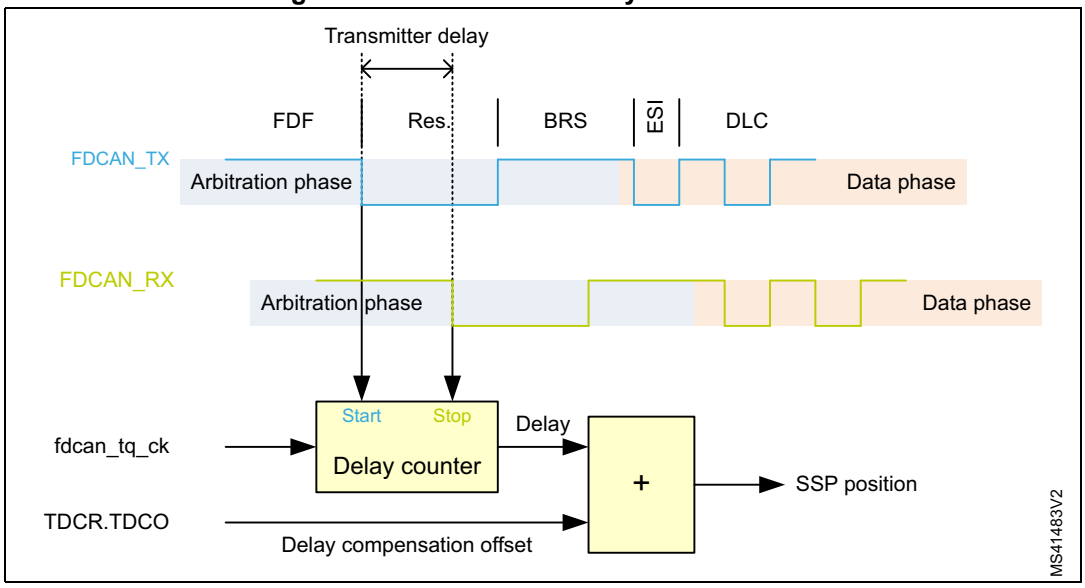
FDCAN_PSR.TDCV shows the actual transmitter delay compensation value, cleared when FDCAN_CCCR.INIT is set, and updated at each transmission of an FD frame while FDCAN_DBTP.TDC is set.

The following boundary conditions must be considered for the transmitter delay compensation implemented in the FDCAN:

- The sum of the measured delay from FDCANx_TX to FDCANx_RX and the configured transmitter delay compensation offset FDCAN_TDCR.TDCO must be less than six bit times in the data phase.
- The sum of the measured delay from FDCANx_TX to FDCANx_RX and the configured transmitter delay compensation offset FDCAN_TDCR.TDCO must be less than or equal to 127 mtq. If this sum exceeds 127 mtq, the maximum value (127 mtq) is used for transmitter delay compensation.
- The data phase ends at the sample point of the CRC delimiter, which stops checking received bits at the SSPs

If transmitter delay compensation is enabled by programming FDCAN_DBTP.TDC = 1, the measurement is started within each transmitted CAN FD frame at the falling edge of bit FDF to bit res. The measurement is stopped when this edge is seen at the receive input pin FDCAN_TX of the transmitter. The resolution of this measurement is 1 mtq.

Figure 956. Transceiver delay measurement



To avoid that a dominant glitch inside the received FDF bit ends the delay compensation measurement before the falling edge of the received res bit (resulting in a too early SSP position), the use of a transmitter delay compensation filter window can be enabled by programming FDCAN_TDCR.TDCF. This defines a minimum value for the SSP position.

Dominant edges on FDCANx_RX that would result in an earlier SSP position are ignored for transmitter delay measurement. The measurement is stopped when the SSP position is at least FDCAN_TDCR.TDCF and FDCAN_RX is low.

Restricted operation mode

In restricted operation mode the node is able to receive data and remote frames and to give acknowledge to valid frames, but it does not send data frames, remote frames, active error frames, or overload frames. In case of an error condition or overload condition, it does not send dominant bits, but waits for the occurrence of bus idle condition to resynchronize itself to the CAN communication. The error counters (FDCAN_ECR.REC, FDCAN_ECR.TEC) are frozen while error logging (FDCAN_ECR.CEL) is active. The software can set the FDCAN into restricted operation mode by setting bit FDCAN_CCCR.ASM. The bit can be set only by software when both FDCAN_CCCR.CCE and FDCAN_CCCR.INIT are set to 1. The bit can be cleared by software at any time.

Restricted operation mode is automatically entered when the Tx handler was not able to read data from the message RAM in time. To leave restricted operation mode, the software must reset FDCAN_CCCR.ASM.

The restricted operation mode can be used in applications that adapt themselves to different CAN bitrates. In this case the application tests different bitrates and leaves the restricted operation mode after it has received a valid frame.

FDCAN_CCCR.ASM is also controlled by the clock calibration unit. When the calibration process is enabled, the restricted operation mode is entered and the FDCAN_CCR.ASM bit is set. Once the calibration is completed, the FDCAN_CCR.ASM bit is cleared.

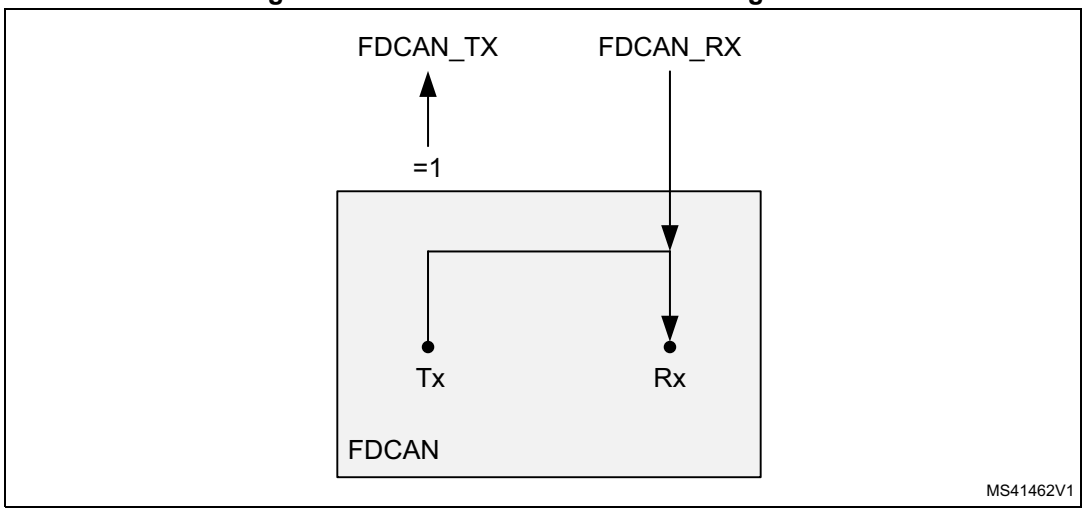
Note: The restricted operation mode must not be combined with the loop back mode (internal or external).

Bus monitoring mode

The FDCAN is set in bus monitoring mode by setting FDCAN_CCCR.MON bit or when error level S3 (FDCAN_TTOST.EL = 11) is entered. In bus monitoring mode (for more details refer to ISO11898-1, 10.12 bus monitoring), the FDCAN is able to receive valid data frames and valid remote frames, but cannot start a transmission. In this mode, it sends only recessive bits on the CAN bus, if the FDCAN is required to send a dominant bit (ACK bit, overload flag, active error flag), the bit is rerouted internally so that the FDCAN monitors this dominant bit, although the CAN bus may remain in recessive state. In bus monitoring mode register FDCAN_TXBRP is held in reset state.

The bus monitoring mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits. *Figure 957* shows the connection of FDCAN_TX and FDCAN_RX signals to the FDCAN in bus monitoring mode.

Figure 957. Pin control in bus monitoring mode



Disabled automatic retransmission (DAR) mode

According to the CAN Specification (see ISO11898-1, 6.3.3 recovery management), the FDCAN provides means for automatic retransmission of frames that have lost arbitration or have been disturbed by errors during transmission. By default, automatic retransmission is enabled.

To support time-triggered communication as described in ISO 11898-1: 2015, chapter 9.2, the automatic retransmission may be disabled via FDCAN_CCCR.DAR.

Frame transmission in disabled automatic retransmission (DAR) mode

In DAR mode all transmissions are automatically canceled after they started on the CAN bus. A Tx buffer Tx request pending bit FDCAN_TXBRP.TRPx is reset after successful transmission, when a transmission has not yet been started at the point of cancellation, has been aborted due to lost arbitration, or when an error occurred during frame transmission.

- Successful transmission:
  - Corresponding Tx buffer transmission occurred bit FDCAN_TXBTO.TOx set
  - Corresponding Tx buffer cancellation finished bit FDCAN_TXBCF.CFx not set
- Successful transmission in spite of cancellation:
  - Corresponding Tx buffer transmission occurred bit FDCAN_TXBTO.TOx set
  - Corresponding Tx buffer cancellation finished bit FDCAN_TXBCF.CFx set
- Arbitration loss or frame transmission disturbed:
  - Corresponding Tx buffer transmission occurred bit FDCAN_TXBTO.TOx not set
  - Corresponding Tx buffer cancellation finished bit FDCAN_TXBCF.CFx set

In case of a successful frame transmission, and if storage of Tx events is enabled, a Tx event FIFO element is written with event type ET = 10 (transmission in spite of cancellation).

Power down (Sleep mode)

The FDCAN can be set into power down mode controlled by clock stop request input via register FDCAN_CCCR.CSR. As long as the clock stop request is active, bit FDCAN_CCCR.CSR is read as 1.

When all pending transmission requests have been completed, the FDCAN waits until the bus idle state is detected. Then the FDCAN sets FDCAN_CCCR.INIT to 1 to prevent any further CAN transfers. Now, the FDCAN acknowledges that it is ready for power down by setting FDCAN_CCCR.CSA to 1. In this state, before the clocks are switched off, further register accesses can be made. A write access to FDCAN_CCCR.INIT has no effect. Now the module clock inputs may be switched off.

To leave power down mode, the application must turn on the module clocks before resetting the CC control register flag FDCAN_CCCR.CSR. The FDCAN acknowledges this by resetting FDCAN_CCCR.CSA. Afterwards, the application can restart CAN communication by resetting bit FDCAN_CCCR.INIT.

Test modes

To enable write access to FDCAN test register (see *Section 71.5.4*), bit FDCAN_CCCR.TEST must be set to 1, thus enabling the configuration of test modes and functions.

Four output functions are available for the CAN transmit pin FDCAN_TX by programming FDCAN_TEST.TX. Additionally, to its default function (the serial data output) it can drive the CAN Sample Point signal to monitor the FDCAN bit timing and it can drive constant dominant or recessive values. The actual value at pin FDCAN_RX can be read from FDCAN_TEST.RX. Both functions can be used to check the CAN bus physical layer.

Due to the synchronization mechanism between CAN kernel clock and APB clock domain, there may be a delay of several APB clock periods between writing to FDCAN_TEST.TX until the new configuration is visible at FDCAN_TX output pin. This applies also when reading FDCAN_RX input pin via FDCAN_TEST.RX.

Note: Test modes must be used for production tests or self test only. The software control for FDCAN_TX pin interferes with all CAN protocol functions. It is not recommended to use test modes for application.

External loop back mode

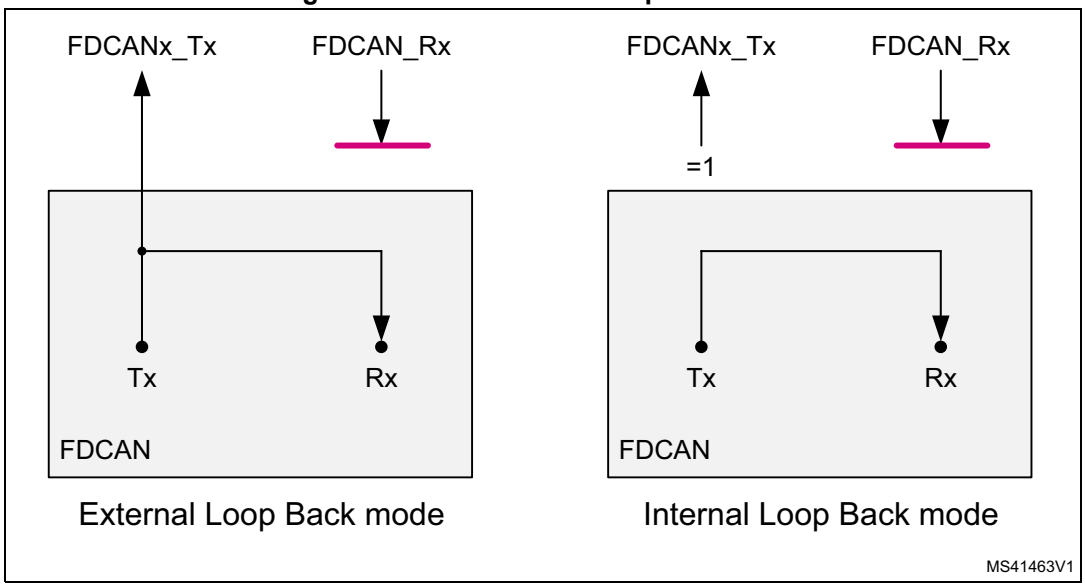
The FDCAN can be set in external loop back mode by programming FDCAN_TEST.LBCK to 1. In loop back mode, the FDCAN treats its own transmitted messages as received messages and stores them (if they pass acceptance filtering) into Rx FIFOs. *Figure 958* (left side) shows the connection of transmit and receive signals FDCAN_TX and FDCAN_RX to the FDCAN in external loop back mode.

This mode is provided for hardware self-test. To be independent from external stimulation, the FDCAN ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/remote frame) in loop back mode. In this mode, the FDCAN performs an internal feedback from its transmit output to its receive input. The actual value of the FDCAN_RX input pin is disregarded by the FDCAN. The transmitted messages can be monitored at the FDCAN_TX transmit pin.

Internal loop back mode

Internal loop back mode is entered by programming bits FDCAN_TEST.LBCK and FDCAN_CCCR.MON to 1. This mode can be used for a “Hot Selftest”, meaning the FDCAN can be tested without affecting a running CAN system connected to the FDCAN_TX and FDCAN_RX pins. In this mode, the FDCAN_RX pin is disconnected from the FDCAN and the FDCAN_TX pin is held recessive. *Figure 958* (right side) shows the connection of FDCAN_TX and FDCAN_RX pins to the FDCAN in case of internal loop back mode.

Figure 958. Pin control in loop back mode



Application watchdog (FDCAN1 only)

The application watchdog is served by reading register FDCAN_TTOST. When the application watchdog is not served in time, bit FDCAN_TTOST.AWE is set, all TTCAN communication is stopped, and the FDCAN1 is set into bus monitoring mode.

The TT application watchdog can be disabled by programming the application watchdog limit FDCAN_TTOCF.AWL to 0x00. The TT application watchdog must not be disabled in a TTCAN application program.

Timestamp generation

For timestamp generation the FDCAN supplies a 16-bit wraparound counter. A prescaler FDCAN_TSCC.TCP can be configured to clock the counter in multiples of CAN bit times (1 … 16). The counter is readable via FDCAN_TSCV.TCV. A write access to register FDCAN_TSCV resets the counter to 0. When the timestamp counter wraps around the interrupt flag FDCAN_IR.TSW is set.

On the start of frame reception/transmission the counter value is captured and stored into the timestamp section of an Rx buffer/Rx FIFO (RXTS[15:0]) or Tx event FIFO (TXTS[15:0]) element.

By programming bit FDCAN_TSCC.TSS, a 16-bit timestamp can be used.

Timeout counter

To signal timeout conditions for Rx FIFO 0, Rx FIFO 1, and the Tx event FIFO the FDCAN supplies a 16-bit timeout counter. It operates as down-counter and uses the same prescaler controlled by FDCAN_TSCC.TCP as the timestamp counter. The timeout counter is configured via register FDCAN_TOCC. The actual counter value can be read from FDCAN_TOCV.TOC. The timeout counter can only be started while FDCAN_CCCR.INIT = 0. It is stopped when FDCAN_CCCR.INIT = 1, for example when the FDCAN enters Bus_Off state.

The operation mode is selected by FDCAN_TOCC.TOS. When operating in Continuous mode, the counter starts when FDCAN_CCCR.INIT is reset. A write to FDCAN_TOCV

presets the counter to the value configured by FDCAN_TOCC.TOP and continues down-counting.

When the timeout counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by FDCAN_TOCC.TOP. Down-counting is started when the first FIFO element is stored. Writing to FDCAN_TOCV has no effect.

When the counter reaches 0, the interrupt flag FDCAN_IR.TOO is set. In Continuous mode, the counter is immediately restarted at FDCAN_TOCC.TOP.

Note: The clock signal for the timeout counter is derived from the CAN core sample point signal. Therefore, the point in time where the timeout counter is decremented can vary due to the synchronization/re-synchronization mechanism of the CAN core. If the baudrate switch feature in FDCAN is used, the timeout counter is clocked differently in arbitration and data fields.

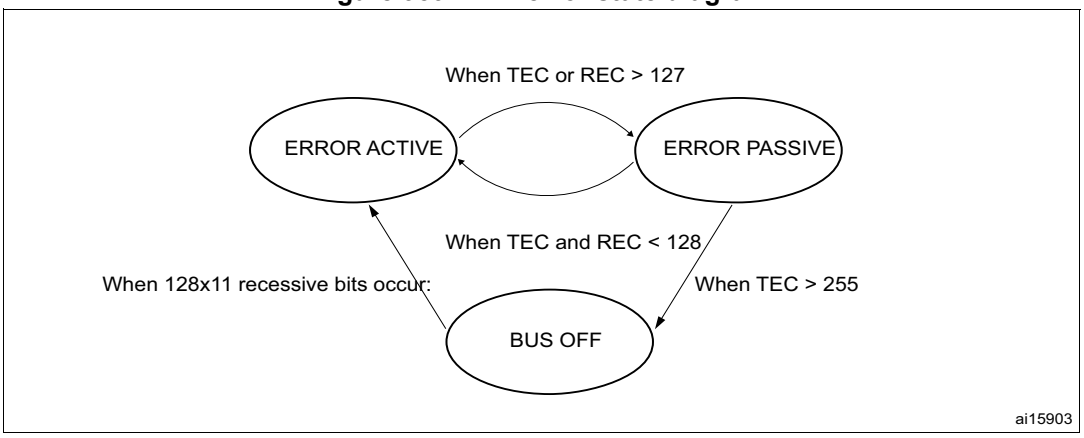
71.4.2 Error management

The error management as described in the CAN protocol is handled entirely by hardware using a transmit error counter (TEC value, in the CAN_ECR register) and a receive error counter (REC value, in the CAN_ECR register), incremented or decremented according to the error condition. Both of them may be read by software to determine the stability of the network. For detailed information about TEC and REC management, refer to the CAN standard.

Bus-Off recovery

The Bus-Off state is reached when TEC is greater than 255, this state is indicated by the BO bit in the CAN_PSR register. In Bus-Off state, the CAN is no longer able to transmit and receive messages. The CAN must wait at least for the recovery sequence specified in the CAN standard (128 occurrences of 11 consecutive recessive bits monitored on CAN RX).

Figure 959. CAN error state diagram

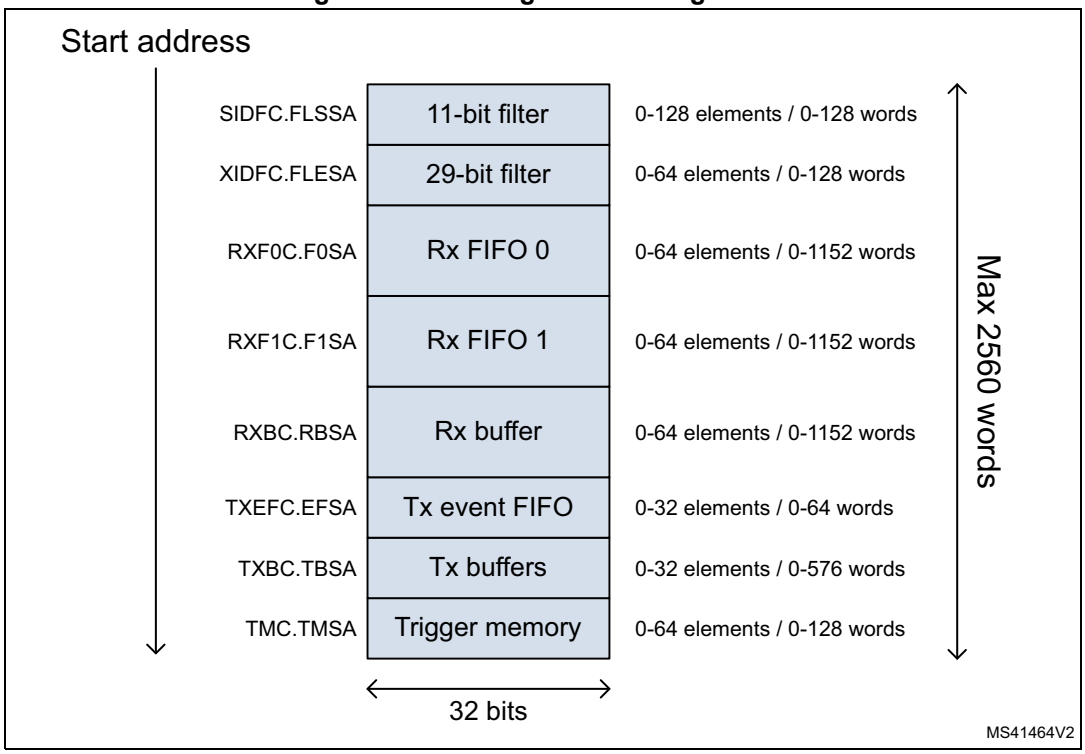


Note: In initialization mode, CAN does not monitor the CAN RX signal, therefore it cannot complete the recovery sequence. To recover, CAN must be in normal mode.

71.4.3 Message RAM

The message RAM has a width of 32 bits. The FDCAN module can be configured to allocate up to 2560 words in the message RAM. It is not necessary to configure each of the sections listed in *Figure 960*, nor is there any restriction with respect to the sequence of the sections.

Figure 960. Message RAM configuration



When the FDCAN addresses the message RAM it addresses 32-bit words, not single bytes. The configured start addresses are 32-bit word addresses, only bits 15 to 2 are evaluated, the two least significant bits are ignored.

Note: The FDCAN does not check for erroneous configuration of the message RAM. In particular, the configuration of the start addresses of the different sections and the number of elements of each section must be done carefully to avoid falsification or loss of data.

Rx handling

The Rx handler controls the acceptance filtering, the transfer of received messages to Rx buffers or to 1 of the two Rx FIFOs, as well as the Rx FIFO put and get Indices.

Acceptance filter

The FDCAN offers the possibility to configure two sets of acceptance filters, one for standard identifiers and one for extended identifiers. These filters can be assigned to Rx buffer, Rx FIFO 0 or Rx FIFO 1. For acceptance filtering each list of filters is executed from element #0 until the first matching element. Acceptance filtering stops at the first matching element. The following filter elements are not evaluated for this message.

The main features are:

- Each filter element can be configured as
  - range filter (from 0 to 128 elements for the 11-bit filter and from 0 to 64 for the 29-bit filter)
  - filter for one or two dedicated IDs
  - classic bit mask filter
- Each filter element is configurable for acceptance or rejection filtering
- Each filter element can be enabled/disabled individually
- Filters are checked sequentially, execution stops with the first matching filter element

Related configuration registers are:

- Global filter configuration (FDCAN_GFC)
- Standard ID filter configuration (FDCAN_SIDFC)
- Extended ID filter configuration (FDCAN_XIDFC)
- Extended ID AND Mask (FDCAN_XIDAM)

Depending on the configuration of the filter element (SFEC / EFEC) a match triggers one of the following actions:

- Store received frame in FIFO 0 or FIFO 1
- Store received frame in Rx buffer
- Store received frame in Rx buffer and generate pulse at filter event pin
- Reject received frame
- Set high priority message interrupt flag FDCAN_IR.HPM
- Set high priority message interrupt flag FDCAN_IR.HPM and store the received frame in FIFO 0 or FIFO 1
- Set high priority message interrupt flag FDCAN_IR.HPM and store the received frame in FIFO 0 or FIFO 1

Acceptance filtering is started after the complete identifier has been received. After acceptance filtering has completed, and if a matching Rx buffer or Rx FIFO has been found, the message handler starts writing the received message data in 32-bit portions to the matching Rx buffer or Rx FIFO. If the CAN protocol controller has detected an error condition (for example CRC error), this message is discarded with the following impact:

- **Rx buffer**
  New data flag of matching Rx buffer is not set, but Rx buffer (partly) overwritten with received data. For error type see FDCAN_PSR.LEC and FDCAN_PSR.DLEC.
- **Rx FIFO**
  Put index of matching Rx FIFO is not updated, but related Rx FIFO element (partly) overwritten with received data. For error type, see FDCAN_PSR.LEC and FDCAN_PSR.DLEC. In case the matching Rx FIFO is operated in overwrite mode, the boundary conditions described in *Rx FIFO overwrite mode* must be considered.

Note: When an accepted message is written to one of the two Rx FIFOs, or into an Rx buffer, the unmodified received identifier is stored independently of the filter(s) used. The result of the acceptance filter process depends strongly upon the sequence of configured filter elements.

Range filter

The filter matches for all received frames with message IDs in the range defined by SF1ID / SF2ID and EF1ID / EF2ID.

There are two possibilities when range filtering is used together with extended frames:

- EFT = 00: The message ID of received frames is AND-ed with the extended ID AND Mask (FDCAN_XIDAM) before the range filter is applied
- EFT = 11: The extended ID AND Mask (FDCAN_XIDAM) is not used for range filtering

Filter for dedicated IDs

A filter element can be configured to filter for one or two specific message IDs. To filter for one specific message ID, the filter element must be configured with SF1ID = SF2ID and EF1ID = EF2ID.

Classic bit mask filter

Classic bit mask filtering is intended to filter groups of message IDs by masking single bits of a received message ID. With classic bit mask filtering SF1ID / EF1ID is used as message ID filter, while SF2ID / EF2ID is used as filter mask.

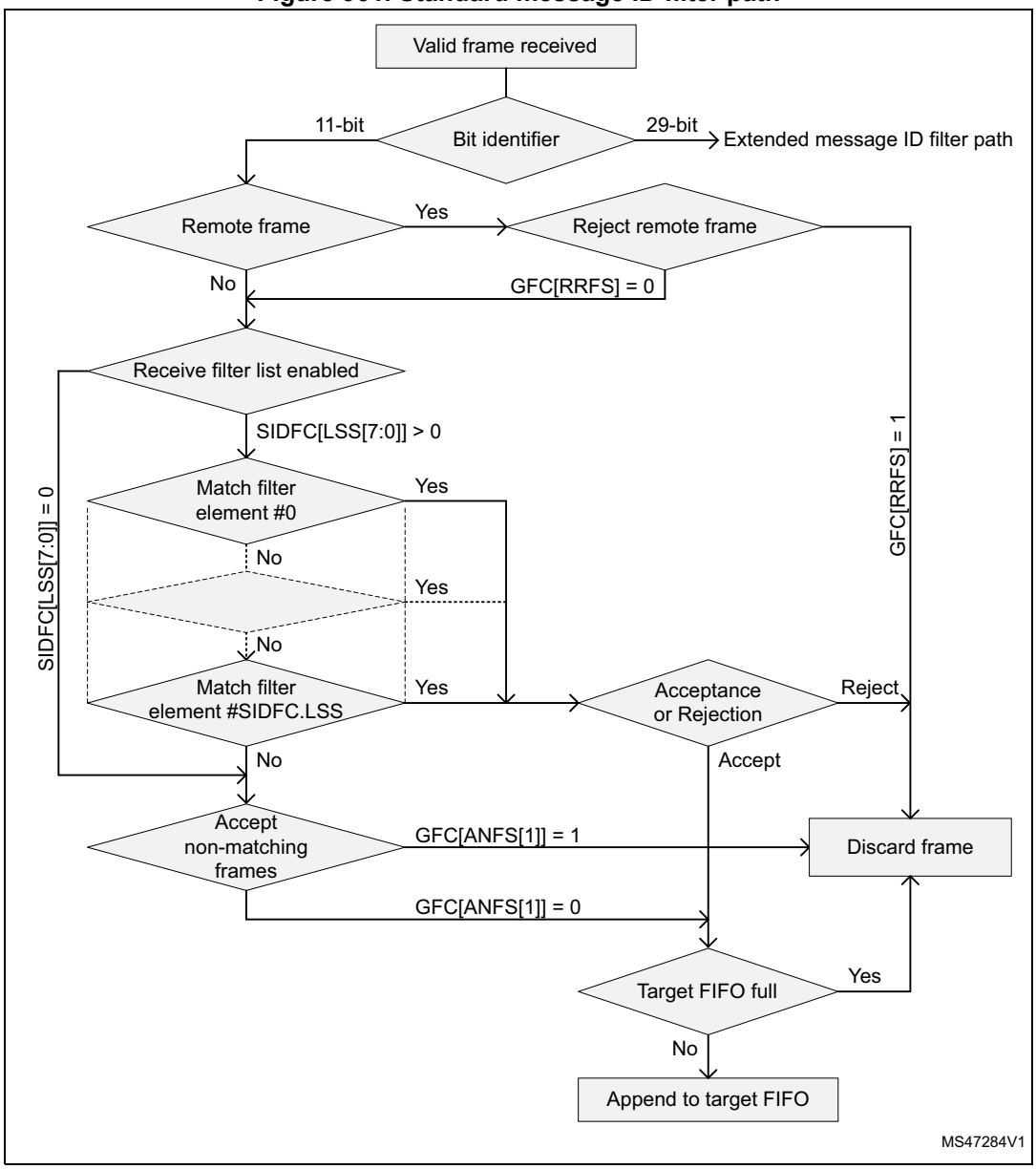
A 0 bit at the filter mask masks out the corresponding bit position of the configured ID filter, for example, the value of the received message ID at that bit position is not relevant for acceptance filtering. Only those bits of the received message ID where the corresponding mask bits are one are relevant for acceptance filtering.

If all mask bits are 1, a match occurs only when the received message ID and the message ID filter are identical. If all mask bits are 0, all message IDs match.

Standard message ID filtering

Figure 961 shows the flow for standard message ID (11-bit Identifier) filtering. The standard message ID filter element is described in *Section 71.4.22*.

Figure 961. Standard message ID filter path

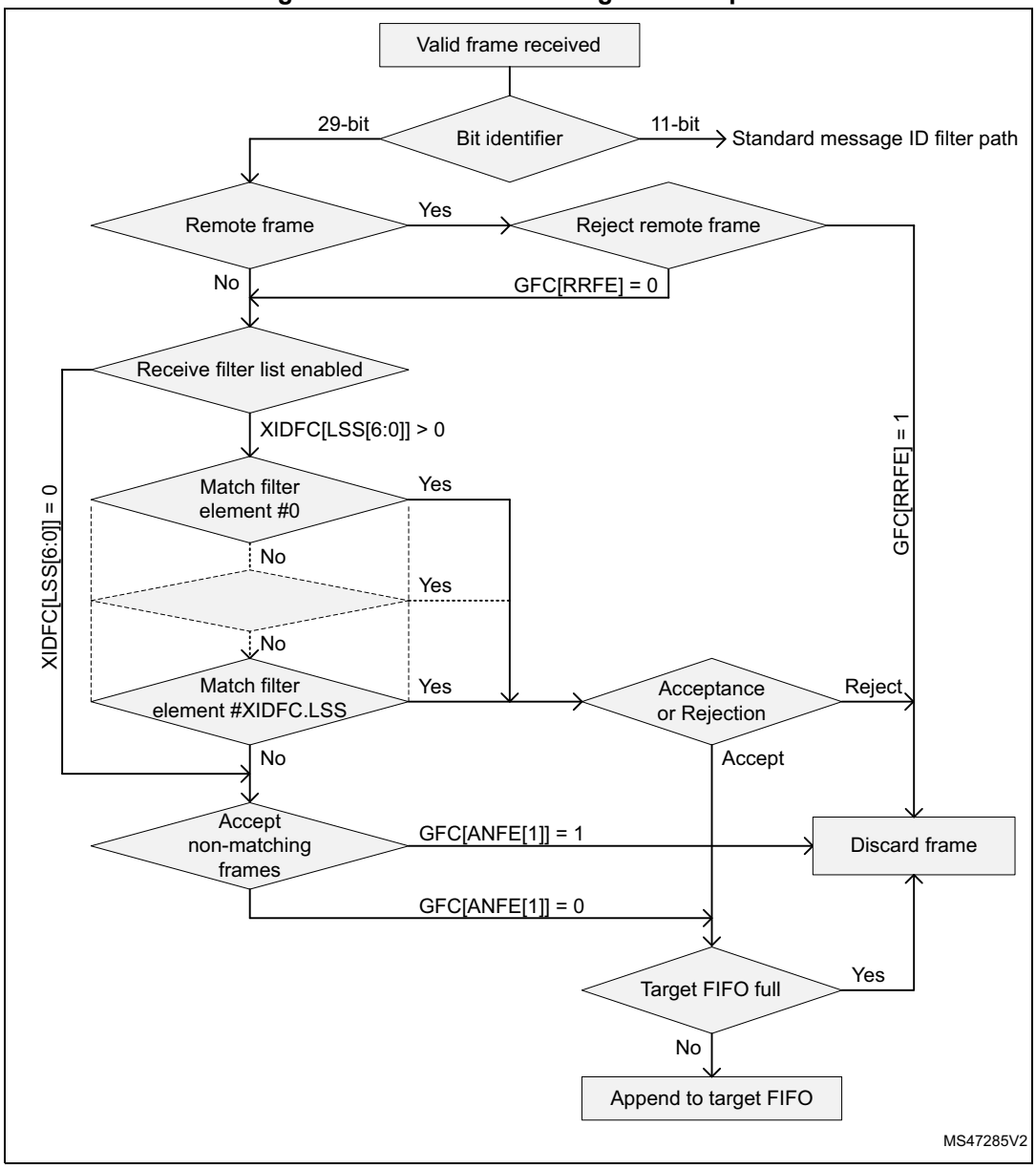


Controlled by the global filter configuration (FDCAN_GFC) and the standard ID filter configuration (FDCAN_SIDFC) message ID, remote transmission request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements.

Extended message ID filtering

Figure 962 shows the flow for extended message ID (29-bit Identifier) filtering. The extended message ID filter element is described in *Section 71.4.23*.

Figure 962. Extended message ID filter path



Controlled by the global filter configuration FDCAN_GFC and the extended ID filter configuration FDCAN_XIDFC message ID, remote transmission request bit (RTR), and the Identifier Extension bit (IDE) of received frames are compared against the list of configured filter elements.

The extended ID AND Mask (FDCAN_XIDAM) is AND-ed with the received identifier before the filter list is executed.

Rx FIFOs

Rx FIFO 0 and Rx FIFO 1 can be configured to hold up to 64 elements each. Configuration of the two Rx FIFOs is done via registers FDCAN_RXF0C and FDCAN_RXF1C.

Received messages that passed acceptance filtering are transferred to the Rx FIFO as configured by the matching filter element. For a description of the filter mechanisms available for Rx FIFO 0 and Rx FIFO 1, see *Acceptance filter*. The Rx buffer and FIFO element are described in *Section 71.4.19*.

When an Rx FIFO full condition is signaled by FDCAN_IR.RFnF, no further messages are written to the corresponding Rx FIFO until at least one message has been read out, and the Rx FIFO get index has been incremented. In case a message is received while the corresponding Rx FIFO is full, this message is discarded and the interrupt flag FDCAN_IR.RFnL is set.

To avoid an Rx FIFO overflow, the Rx FIFO watermark can be used. When the Rx FIFO fill level reaches the Rx FIFO watermark configured by FDCAN_RXFnC.FnWM, interrupt flag FDCAN_IR.RFnW is set.

When reading from an Rx FIFO, Rx FIFO get index RXFnS[FnGI] + FIFO element size must be added to the corresponding Rx FIFO start address RXFnC[FnSA].

Rx FIFO blocking mode

The Rx FIFO blocking mode is configured by RXFnC.FnOM = 0. This is the default operation mode for the Rx FIFOs.

When an Rx FIFO full condition is reached (RXFnS.FnPI = RXFnS.FnGI), no further messages are written to the corresponding Rx FIFO until at least one message has been read out, and the Rx FIFO get index has been incremented. An Rx FIFO full condition is signaled by RXFnS.FnF = 1. In addition, the interrupt flag FDCAN_IR.RFnF is set.

If a message is received while the corresponding Rx FIFO is full, this message is discarded and the message lost condition is signaled by RXFnS.RFnL = 1. In addition, the interrupt flag FDCAN_IR.RFnL is set.

Rx FIFO overwrite mode

The Rx FIFO overwrite mode is configured by RXFnC.FnOM = 1.

When an Rx FIFO full condition (RXFnS.FnPI = RXFnS.FnGI) is signaled by RXFnS.FnF = 1, the next message accepted for the FIFO overwrites the oldest FIFO message. Put and get index are both incremented by one.

When an Rx FIFO is operated in overwrite mode and an Rx FIFO full condition is signaled, reading of the Rx FIFO elements must start at least at get index + 1. The reason for that is that it can happen that a received message is written to the message RAM (put index) while the CPU is reading from the message RAM (get index). In this case inconsistent data may be read from the respective Rx FIFO element. Adding an offset to the get index when reading from the Rx FIFO avoids this problem. The offset depends on how fast the CPU accesses the Rx FIFO. *Figure 964* shows an offset of two with respect to the get index when reading the Rx FIFO. In this case, the two messages stored in elements 1 and 2 are lost.

After reading from the Rx FIFO, the number of the last element read must be written to the Rx FIFO acknowledge index RXFnA.FnA. This increments the get index to that element number. In case the put index has not been incremented to this Rx FIFO element, the Rx FIFO full condition is reset (RXFnS.FnF = 0).

Dedicated Rx buffers

The FDCAN supports up to 64 dedicated Rx buffers. The start address of the dedicated Rx buffer section is configured via FDCAN_RXBC.RBSA.

For each Rx buffer a standard or extended message ID filter element with SFEC / EFEC=111 and SFID2 / EFID2[10:9] = 00 must be configured (see *Section 71.4.22* and *Section 71.4.23*).

After a received message has been accepted by a filter element, the message is stored into the Rx buffer in the message RAM referenced by the filter element. The format is the same as for an Rx FIFO element. In addition, the flag FDCAN_IR.DRX (message stored in dedicated Rx buffer) in the interrupt register is set.

Table 697. Example of filter configuration for Rx buffers

Filter element	SFID1[10:0] EFID1[28:0]	SFID2[10:9] EFID2[10:9]	SFID2[5:0] EFID2[5:0]
0	ID message 1	00	00 0000
1	ID message 2	00	00 0001
2	ID message 3	00	00 0010

After the last word of a matching received message has been written to the message RAM, the respective New data flag in register NDAT1,2 is set. As long as the New data flag is set, the respective Rx buffer is locked against updates from received matching frames. The New data flags must be reset by the user by writing a 1 to the respective bit position.

While an Rx buffer New data flag is set, a message ID filter element referencing this specific Rx buffer is not matched, causing the acceptance filtering to continue. The following message ID filter elements may cause the received message to be stored into another Rx buffer, or into an Rx FIFO, or the message may be rejected, depending on filter configuration.

Rx buffer handling

- Reset interrupt flag FDCAN_IR.DRX
- Read New data registers
- Read messages from message RAM
- Reset New data flags of processed messages

Filtering for Debug messages

Filtering for debug messages is done by configuring one standard/extended message ID filter element for each of the three debug messages. To enable a filter element to filter for debug messages SFEC/EFEC must be programmed to 111. In this case fields SFID1 / SFID2 and EFID1 / EFID2 have a different meaning. While SFID2 / EFID2[10:9] controls the debug message handling state machine, SFID2 / EFID2[5:0] controls the location for storage of a received debug message.

When a debug message is stored, neither the respective New data flag nor FDCAN_IR.DRX are set. The reception of debug messages can be monitored via FDCAN_RXF1S.DMS.

Table 698. Example of filter configuration for Debug messages

Filter element	SFID1[10:0] EFID1[28:0]	SFID2[10:9] EFID2[10:9]	SFID2[5:0] EFID2[5:0]
0	ID debug message A	01	11 1101
1	ID debug message B	10	11 1110
2	ID debug message C	11	11 1111

Tx handling

The Tx handler handles transmission requests for the dedicated Tx buffers, the Tx FIFO, and the Tx queue. It controls the transfer of transmit messages to the CAN core, the put and get Indices, and the Tx event FIFO.Up to 32 Tx buffers can be set up for message transmission (see *Dedicated Tx buffers*). Depending on the configuration of the element size (FDCAN_RXESC), between two and sixteen 32-bit words (Rn = 3 ... 17) are used for storage of a CAN message data field.

Table 699. Possible configurations for frame transmission

FDCAN_CCCR		Tx buffer element		Frame transmission
BRSE	FDOE	FDF	BRS	
Ignored	0	Ignored	Ignored	Classic CAN
0	1	0	Ignored	Classic CAN
0	1	1	Ignored	FD without bitrate switching
1	1	0	Ignored	Classic CAN
1	1	1	0	FD without bitrate switching
1	1	1	1	FD with bitrate switching

Note: AUTOSAR requires at least three Tx queue buffers and support of transmit cancellation.

The Tx handler starts a Tx scan to check for the highest priority pending Tx request (Tx buffer with lowest message ID) when the Tx buffer request pending register FDCAN_TXBRP is updated, or when a transmission has been started.

Transmit pause

This feature is intended for use in CAN systems where the CAN message identifiers are (permanently) specified to specific values and cannot easily be changed. These message identifiers may have a higher CAN arbitration priority than other defined messages, while in a specific application their relative arbitration priority must be inverse. This may lead to a case where one ECU sends a burst of CAN messages that cause another ECU CAN messages to be delayed because that other messages have a lower CAN arbitration priority.

If, as an example, CAN ECU-1 has the feature enabled and is requested by its application software to transmit four messages, after the first successful message transmission, it waits for two CAN bit times of bus idle before it is allowed to start the next requested message. If there are other ECUs with pending messages, those messages are started in the idle time, they would not need to arbitrate with the next message of ECU-1. After having received a

message, ECU-1 is allowed to start its next transmission as soon as the received message releases the CAN bus.

The feature is controlled by the TXP bit in the FDCAN_CCCR register. If the bit is set, the FDCAN, each time it has successfully transmitted a message, pauses for two CAN bit times before starting the next transmission. This enables other CAN nodes in the network to transmit messages even if their messages have lower prior identifiers. Default is disabled (FDCAN_CCCR.TXP = 0).

This feature looses up burst transmissions coming from a single node and it protects against "babbling idiot" scenarios where the application program erroneously requests too many transmissions.

Dedicated Tx buffers

Dedicated Tx buffers are intended for message transmission under complete control of the CPU. Each dedicated Tx buffer is configured with a specific message ID. In case that multiple Tx buffers are configured with the same message ID, the Tx buffer with the lowest buffer number is transmitted first.

If the data section has been updated, a transmission is requested by an add request via FDCAN_TXBAR.ARn. The requested messages arbitrate internally with messages from an optional Tx FIFO or Tx queue and externally with messages on the CAN bus, and are sent out according to their message ID.

A dedicated Tx buffer allocates element size 32-bit words in the message RAM. Therefore, the start address of a dedicated Tx buffer in the message RAM is calculated by adding the element size defined in *Table 700* times the transmit buffer index (0 … 31) to the Tx buffer start address FDCAN_TXBC.TBSA

Table 700. Tx buffer/FIFO - Queue element size

FDCAN_TXESC.TBDS[2:0]	Data field (bytes)	Element size (RAM words)
000	8	4
001	12	5
010	16	6
011	20	7
100	24	8
101	32	10
110	48	14
111	64	18

Tx FIFO

Tx FIFO operation is configured by programming FDCAN_TXBC.TFQM to 0. Messages stored in the Tx FIFO are transmitted starting with the message referenced by the get index FDCAN_TXFQS.TFGI. After each transmission the get index is incremented cyclically until the Tx FIFO is empty. The Tx FIFO enables transmission of messages with the same message ID from different Tx buffers in the order these messages have been written to the Tx FIFO. The FDCAN calculates the Tx FIFO free level FDCAN_TXFQS.TFFL as the difference between get and put index. It indicates the number of available (free) Tx FIFO elements.

New transmit messages must be written to the Tx FIFO starting with the Tx buffer referenced by the put index FDCAN_TXFQS.TFQPI. An add request increments the put index to the next free Tx FIFO element. When the put index reaches the get index, Tx FIFO full (FDCAN_TXFQS.TFQF = 1) is signaled. In this case, no further messages must be written to the Tx FIFO until the next message has been transmitted and the get index has been incremented.

When a single message is added to the Tx FIFO, the transmission is requested by writing a 1 to the FDCAN_TXBAR bit related to the Tx buffer referenced by the Tx FIFO put index.

When multiple (n) messages are added to the Tx FIFO, they are written to n consecutive Tx buffers starting with the put index. The transmissions are then requested via FDCAN_TXBAR. The put index is then cyclically incremented by n. The number of requested Tx buffers must not exceed the number of free Tx buffers as indicated by the Tx FIFO free level.

When a transmission request for the Tx buffer referenced by the get index is canceled, the get index is incremented to the next Tx buffer with pending transmission request and the Tx FIFO free level is recalculated. When transmission cancellation is applied to any other Tx buffer, the get index and the FIFO free level remain unchanged.

A TX FIFO element allocates element size 32-bit words in the message RAM. Therefore, the start address of the next available (free) Tx FIFO buffer is calculated by adding element size times the put index FDCAN_TXFQS.TFQPI (0 … 31) to the Tx buffer start address FDCAN_TXBC.TBSA.

Tx queue

Tx queue operation is configured by programming FDCAN_TXBC.TFQM to 1. Messages stored in the Tx queue are transmitted starting with the message with the lowest message ID (highest priority). If multiple queue buffers are configured with the same message ID, the queue buffer with the lowest buffer number is transmitted first.

New messages must be written to the Tx buffer referenced by the put index FDCAN_TXFQS.TFQPI. An add request cyclically increments the put index to the next free Tx buffer. In case that the Tx queue is full (FDCAN_TXFQS.TFQF = 1), the put index is not valid and no further message must be written to the Tx queue until at least one of the requested messages has been sent out or a pending transmission request has been canceled.

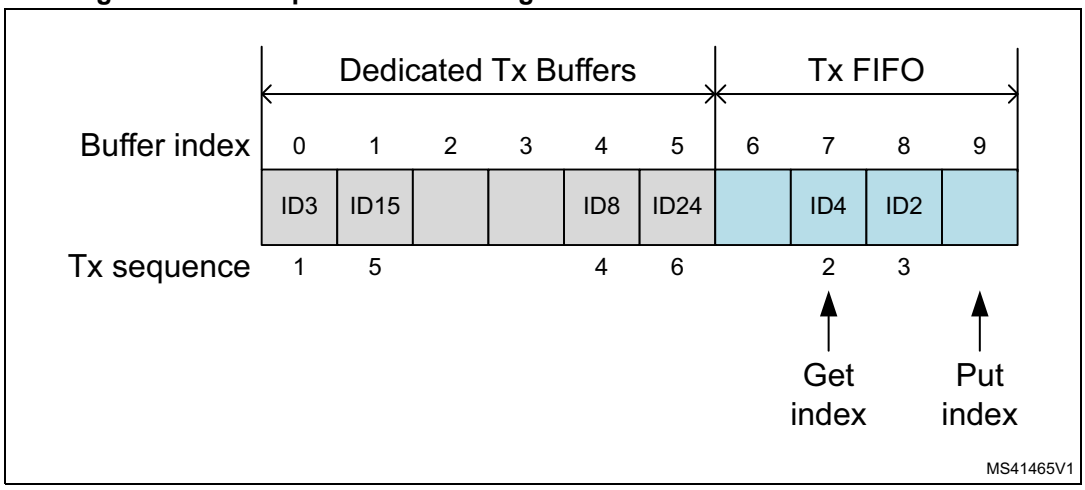
The application may use register FDCAN_TXBRP instead of the put index and may place messages to any Tx buffer without pending transmission request.

A Tx queue buffer allocates element size 32-bit words in the message RAM. Therefore, the start address of the next available (free) Tx queue buffer is calculated by adding element size times the Tx queue put index FDCAN_TXFQS.TFQPI (0 … 31) to the Tx buffer start address FDCAN_TXBC.TBSA

Mixed dedicated Tx buffers / Tx FIFO

In this case the Tx buffers section in the message RAM is subdivided into a set of dedicated Tx buffers and a Tx FIFO. The number of dedicated Tx buffers is configured by FDCAN_TXBC.NDTB. The number of Tx buffers assigned to the Tx FIFO is configured by FDCAN_TXBC.TFQS. In case, FDCAN_TXBC.TFQS is programmed to 0, only dedicated Tx buffers are used.

Figure 963. Example of mixed configuration dedicated Tx buffers / Tx FIFO



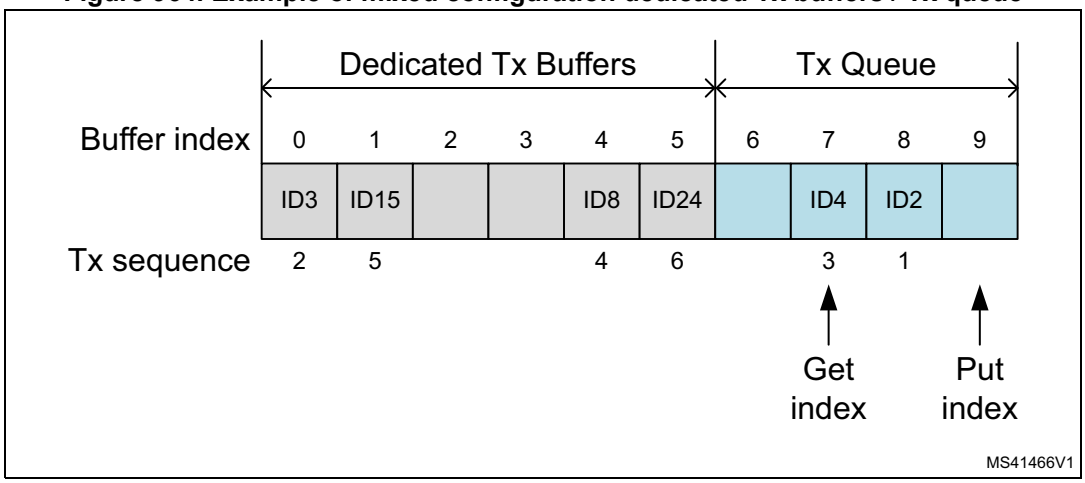
Tx prioritization:

- Scan dedicated Tx buffers and oldest pending Tx FIFO buffer (referenced by FDCAN_TXFS.TFGI)
- Buffer with lowest message ID gets highest priority and is transmitted next

Mixed dedicated Tx buffers / Tx queue

In this case the Tx buffers section in the message RAM is subdivided into a set of dedicated Tx buffers and a Tx queue. The number of dedicated Tx buffers is configured by FDCAN_TXBC.NDTB. The number of Tx queue buffers is configured by FDCAN_TXBC.TFQS. If FDCAN_TXBC.TFQS is programmed to 0, only dedicated Tx buffers are used.

Figure 964. Example of mixed configuration dedicated Tx buffers / Tx queue



Tx priority setting:

- Scan all Tx buffers with activated transmission request
- Tx buffer with lowest message ID gets highest priority and is transmitted next

Transmit cancellation

The FDCAN supports transmit cancellation. To cancel a requested transmission from a dedicated Tx buffer or a Tx queue buffer the user must write a 1 to the corresponding bit position (= number of Tx buffer) of register FDCAN_TXBCR. Transmit cancellation is not intended for Tx FIFO operation.

Successful cancellation is signaled by setting the corresponding bit of register FDCAN_TXBCF to 1.

In case a transmit cancellation is requested while a transmission from a Tx buffer is already ongoing, the corresponding FDCAN_TXBRP bit remains set as long as the transmission is in progress. If the transmission was successful, the corresponding FDCAN_TXBTO and FDCAN_TXBCF bits are set. If the transmission was not successful, it is not repeated and only the corresponding FDCAN_TXBCF bit is set.

Note: If a pending transmission is canceled immediately before this transmission starts, a short time window follows where no transmission is started even if another message is pending in this node. This may enable another node to transmit a message that may have a priority lower than that of the second message in this node.

Tx event handling

To support Tx event handling the FDCAN has implemented a Tx event FIFO. After the FDCAN has transmitted a message on the CAN bus, the message ID and timestamp are stored in a Tx event FIFO element. To link a Tx event to a Tx event FIFO element, the message marker from the transmitted Tx buffer is copied into the Tx event FIFO element.

The Tx event FIFO can be configured to a maximum of 32 elements. The Tx event FIFO element is described in *Tx FIFO*. Depending on the configuration of the element size (FDCAN_TXESC), between two and sixteen 32-bit words (Tn = 3 ..17) are used for storage of a CAN message data field.

The purpose of the Tx event FIFO is to decouple handling transmit status information from transmit message handling, i.e. a Tx buffer holds only the message to transmit, while the transmit status is stored separately in the Tx event FIFO. This has the advantage, especially when operating a dynamically managed transmit queue, that a Tx buffer can be used for a new message immediately after successful transmission. There is no need to save transmit status information from a Tx buffer before overwriting that Tx buffer.

When a Tx event FIFO full condition is signaled by FDCAN_IR.TEFF, no further elements are written to the Tx event FIFO until at least one element has been read out and the Tx event FIFO get index has been incremented. In case a Tx event occurs while the Tx event FIFO is full, this event is discarded and the interrupt flag FDCAN_IR.TEFL is set.

To avoid a Tx event FIFO overflow, the Tx event FIFO watermark can be used. When the Tx event FIFO fill level reaches the Tx event FIFO watermark configured by FDCAN_TXEFC.EFWM, interrupt flag FDCAN_IR.TEFW is set.

When reading from the Tx event FIFO, two times the Tx event FIFO get index FDCAN_TXEFS.EFGI must be added to the Tx event FIFO start address FDCAN_TXEFC.EFSA.

71.4.4 FIFO acknowledge handling

The get indices of Rx FIFO 0, Rx FIFO 1, and the Tx event FIFO are controlled by writing to the corresponding FIFO acknowledge index, see *FDCAN Rx FIFO 0 acknowledge register*

(FDCAN_RXF0A), *FDCAN Rx FIFO 1 acknowledge register (FDCAN_RXF1A)*, and *FDCAN Tx event FIFO configuration register (FDCAN_TXEFC)*. Writing to the FIFO acknowledge index sets the FIFO get index to the FIFO acknowledge index plus one and thereby updates the FIFO fill level. There are two use cases:

- When only a single element has been read from the FIFO (the one being pointed to by the get index), this get index value is written to the FIFO acknowledge index.
- When a sequence of elements has been read from the FIFO, it is sufficient to write the FIFO acknowledge index only once at the end of that read sequence (value: index of the last element read), to update the FIFO get index.

Because the CPU has free access to the FDCAN message RAM, special care must be taken when reading FIFO elements in an arbitrary order (get index not considered). This might be useful when reading a high priority message from one of the two Rx FIFOs. In this case the FIFO acknowledge index must not be written because this would set the get index to a wrong position and also alters the FIFO fill level. In this case, some of the older FIFO elements would be lost.

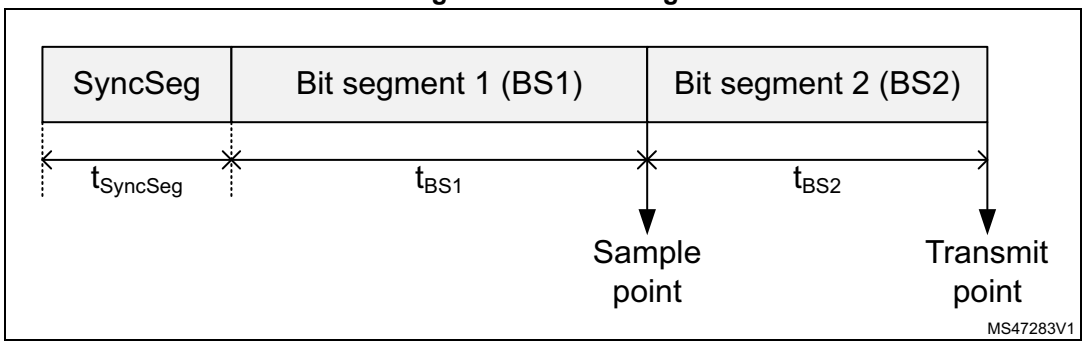
Note: The application must ensure that a valid value is written to the FIFO acknowledge index. The FDCAN does not check for erroneous values.

71.4.5 Bit timing

The bit timing logic monitors the serial bus-line and performs sampling and adjustment of the sample point by synchronizing on the start-bit edge and resynchronizing on the following edges. As shown in *Figure 965*, its operation may be explained simply by splitting the bit time in three segments:

- Synchronization segment (SYNC_SEG): a bit change is expected to occur within this time segment, which has a fixed length of one time quantum (1 x tq).
- Bit segment 1 (BS1): defines the location of the sample point. It includes the PROP_SEG and PHASE_SEG1 of the CAN standard, its duration is programmable between 1 and 16 time quanta, but may be automatically lengthened to compensate for positive phase drifts due to differences in the frequency of the various nodes of the network.
- Bit segment 2 (BS2): defines the location of the transmit point. It represents the PHASE_SEG2 of the CAN standard, its duration is programmable between one and eight time quanta, but may also be automatically shortened to compensate for negative phase drifts.

Figure 965. Bit timing



The baudrate is the inverse of bit time (baudrate = 1 / bit time), which, in turn, is the sum of three components. *Figure 965* indicates that bit time = $t_{SyncSeg}$ + t_{BS1} + t_{BS2}, where:

- for the nominal bit time
  - tq = (FDCAN_NBTP.NBRP[8:0] + 1) * $t_{fdcan_tq_ck}$
  - $t_{SyncSeg}$ = 1 tq
  - t_{BS1} = tq * (FDCAN_NBTP.NTSEG1[7:0] + 1)
  - t_{BS2} = tq * (FDCAN_NBTP.NTSEG2[6:0] + 1)
- for the data bit time
  - tq = (FDCAN_DBTP.DBRP[4:0] + 1) * $t_{fdcan_tq_ck}$
  - $t_{SyncSeg}$ = 1 tq
  - t_{BS1} = tq * (FDCAN_DBTP.DTSEG1[4:0] + 1)
  - t_{BS2} = tq * (FDCAN_DBTP.DTSEG2[3:0] + 1)

The (re)synchronization jump width (SJW) defines an upper bound for the amount of lengthening or shortening of the bit segments. It is programmable between one and four time quanta.

A valid edge is defined as the first transition in a bit time from dominant to recessive bus level, provided the controller itself does not send a recessive bit.

If a valid edge is detected in BS1 instead of SYNC_SEG, BS1 is extended by up to SJW so that the sample point is delayed.

Conversely, if a valid edge is detected in BS2 instead of SYNC_SEG, BS2 is shortened by up to SJW so that the transmit point is moved earlier.

As a safeguard against programming errors, the configuration of the bit timing register is only possible while the device is in Standby mode. Registers FDCAN_DBTP and FDCAN_NBTP (dedicated, respectively, to data and nominal bit timing) are only accessible when FDCAN_CCCR.CCE and FDCAN_CCCR.INIT are set.

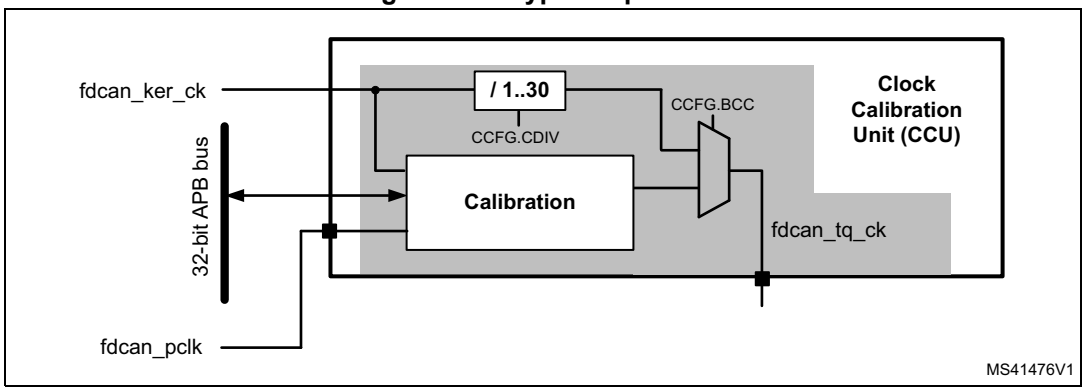
Note: For a detailed description of the CAN bit timing and resynchronization mechanism, refer to the ISO 11898-1 standard.

71.4.6 Clock calibration on CAN

After device reset the clock calibration unit (CCU) does not provide a valid clock signal to the FDCAN(s). The CCU must be initialized via FDCAN_CCFG register. The FDCAN_CCFG register can be written only when FDCAN1 has both FDCAN_CCCR.CCE and FDCAN_CCCR.INIT bits set. In consequence the CCU and the FDCAN1 initialization needs to be completed before any FDCAN module can operate.

Clock calibration is bypassed when FDCAN_CCFG.BCC = 1 (see *Figure 966*).

Figure 966. Bypass operation



Operating conditions

The clock calibration on CAN unit is designed to operate under the following conditions:

- a CAN kernel clock frequency fdcan_ker_ck up of at least 80 MHz
- FDCAN bitrates:
  - Nominal bitrate: up to 1 Mbit/s
  - Data bitrate: between 125 kbps and 1Mbps

The clock calibration on FDCAN unit generates a calibrated time quanta clock fdcan_tq_ck in the range from 0.5 to 25 MHz.

Note: The FDCAN requires that the CAN time quanta clock is always below or equal to the APB clock (fdcan_tq_ck < fdcan_pclk). This must be considered when the clock calibration on CAN unit is bypassed (FDCAN_CCFG.BCC = 1).

Calibration accuracy

The calibration accuracy in state Precision_Calibrated depends upon the factors listed below.

- Dynamic clock tolerance at the CAN kernel clock input fdcan_ker_ck
- Measurement error. For each bit sequence used for calibration measurement, there is a maximum error of one fdcan_pclk period. The number of bits used for measurement of the bit time is 32 or 64-bit, depending on configuration of FDCAN_CCFG.CFL.
- Tolerable error in calibration mechanism

The distance between two calibration messages must be chosen to fit the clock tolerance requirements of the FDCAN1 module.

Note: Dynamic clock tolerance is the clock frequency variation between two calibration messages for example caused by change of temperature or operating voltage.

Functional description

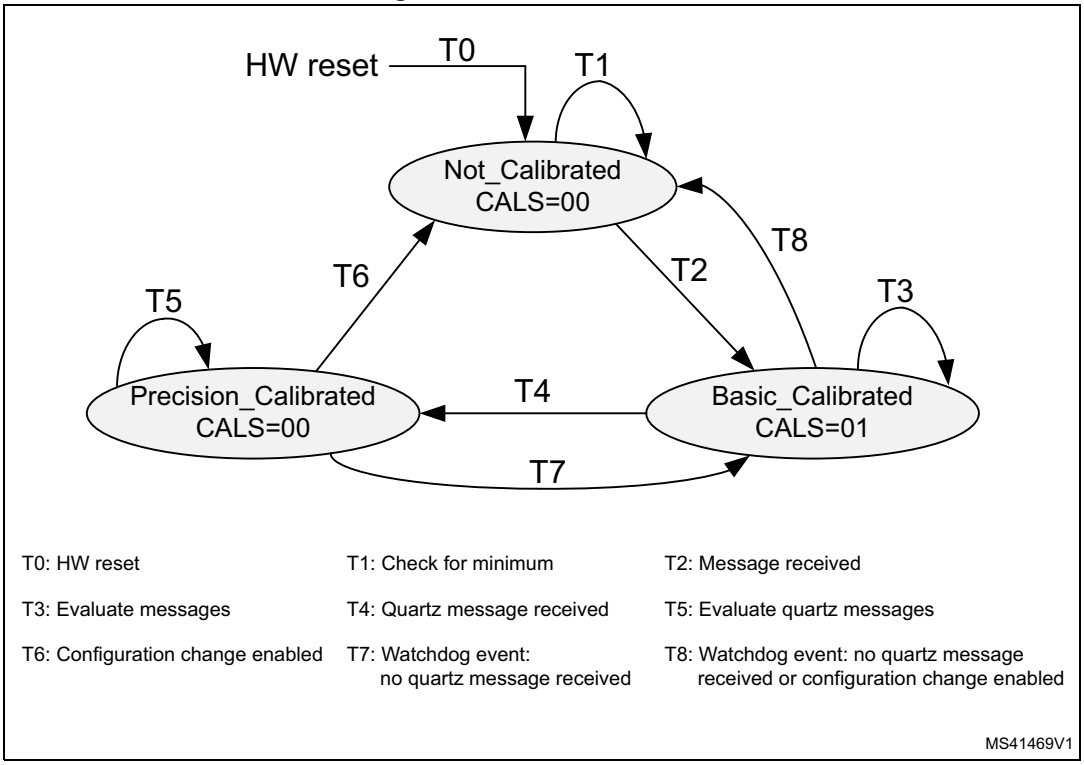
Calibration of the time quanta clock fdcan_tq_ck via CAN messages is performed by adapting a clock divider that generates the CAN protocol time quantum tq from the clock fdcan_ker_ck.

1. First step: basic calibration
   The minimum distance between two consecutive falling edges from recessive to dominant is measured, this time to be assumed two CAN bit times, counted in PLL clock periods. The clock divider (FDCAN_CCFG.CDIV) is updated each time a new

measurement finds a smaller distance between edges. Basic calibration is achieved when the CAN protocol controller detects a valid CAN message.

2. Second step: Precision calibration
The calibration state machine measures the length of a longer bit sequence inside a CAN frame by counting the number of fdcan_ker_ck periods. The length of this bit sequence can be configured to 32 or 64 bits via FDCAN_CCFG.CLF. For a calibration field length of 32/64 bit a calibration message with at least 2/6-byte data field is required. Precision calibration is based on the new clock divider value calculated from the measurement of the longer bit sequence.

Figure 967. FSM calibration



A change in the calibration state sets interrupt flag FDCAN_CCU_IR.CSC, if enabled by the interrupt enable FDCAN_CCU_IE.CSCE. it remains set until cleared by writing 1 in FDCAN_CCU_IR.CSC.

Until precision calibration is achieved, FDCANs operate in a restricted mode (no frame transmission, no error or overload flag transmission, no error counting). In case calibration of the fdcan_ker_clk is done by software by evaluating the calibration status from register FDCAN_CCU_CSTAT, FDCANs must be set to restricted operation mode (FDCAN_CCCR.ASM = 1) until the calibration on CAN unit is in state Precision_Calibrated (see *Application*).

Precision calibration may be performed only on valid CAN frames transmitted by a node with a stable, quartz-controlled clock. calibration frames are detected by the FDCAN1 acceptance filtering A filter element and a Rx buffer must be configured in the FDCAN1 to identify and store calibration messages. After reception of a calibration message the Rx buffer new data flag must be reset to enable signaling of the next calibration message.

In case there is only one CAN transmitter with a quartz clock in the network, this node must transmit its first message after startup with at least one 1010 binary sequence in the data field or in the identifier. This assures that the non-quartz nodes can enter state Basic_Calibrated and then acknowledge the quartz node messages.

Precision calibration must be repeated in predefined maximum intervals supervised by the calibration watchdog.

Note: When the clock calibration on CAN unit transits from state Precision_Calibrated back to Basic_Calibrated, the calibration OK signal is deasserted, the FDCAN1 complete ongoing transmissions, and then enter restricted operation (no frame transmission, no error or overload flag transmission, no error counting).

Configuration

The clock calibration on CAN unit is configured via register FDCAN_CCFG, that is, when FDCAN1 has FDCAN_CCCR.CCE and FDCAN_CCCR.INIT bits set.

For basic calibration the minimum number of oscillator periods between two consecutive falling edges at pin FDCAN1_RX is measured. The number of clock periods depends on the clock frequency applied at input fdcan_ker_ck. In case the measured number of clock periods is below the minimum configured by FDCAN_CCFG.OCPM (as an example, because of a glitch on FDCAN1_RX) the value is discarded and measurement continues.

It is recommended to configure FDCAN_CCFG.OCPM slightly below two CAN bit times:

FDCAN_CCFG.OCPM < ((2 x CAN bit time) / fdcan_ker_ck period) / 32

The length of the bit field used for precision calibration can be configured to 32 or 64 bits via FDCAN_CCFG.CFL. The number of bits used for precision calibration has an impact on calibration accuracy and the maximum distance between two calibration messages.

The number of time quanta per bit time configured by FDCAN_CCFG.TQBT is used together with the measured number of oscillator clock periods FDCAN_CCU_CSTAT.OCPC to define the number of oscillator clocks per bit time.

When the clock calibration is bypassed by configuring FDCAN_CCFG.BCC = 1, the internal clock divider must be configured via FDCAN_CCFG.CDIV to fulfill the condition fdcan_tq_ck < fdcan_pclk.

Note: When clock calibration on CAN is active (FDCAN_CCFG.BCC = 0), the baudrate prescalers of FDCAN modules must be configured to inactive.

Status signaling

The status of the clock calibration on CAN unit can be monitored by reading register FDCAN_CCU_CSTAT. When in state Precision_Calibrated the oscillator clock period counter FDCAN_CCU_CSTAT.OCPC signals the number of oscillator clock periods in the calibration field while FDCAN_CCU_CSTAT.TQC signals the number of time quanta in the calibration field.

The calibration state is monitored by FDCAN_CCU_CSTAT.CALS. A change in the calibration state sets interrupt flag FDCAN_CCU_IR.CSC. If enabled by the interrupt enable FDCAN_CCU_IE.CSCE it remains set until cleared by writing 1 in FDCAN_CCU_IR.CSC.

A calibration watchdog event also sets interrupt flag FDCAN_CCU_IR.CWE. If enabled by FDCAN_CCU_IE.CWEE (set to high), it remains active until reset by FDCAN_CCU_IE.CWE.

71.4.7 Application

Clock calibration bypassed

The CCU internal clock divider is configured for division by one (FDCAN_CCFG.CDIV = 0x0000). In this operation mode the input clock fdcan_ker_ck is directly routed to the clock output fdcan_tq_ck. In this case fdcan_tq_ck is independent from the configuration and status of FDCAN1and FDCAN2 connected to the CCU. CAN FD operation is possible with an FDCAN_ker_ck above 80 MHz.

Software calibration

The clock calibration on CAN unit also supports software calibration of fdcan_ker_ck by trimming of an on-chip oscillator. For calculation of the trimming values the user must read the CCU state from FDCAN_CCU_CSTAT. The clock from fdcan_ker_ck is routed to output fdcan_tq_ck (FDCAN_CCFG.BCC = 1).

The input clock fdcan_ker_ck must be at least 80 MHz. The clock divider of CCU must be configured via FDCAN_CCFG.CDIV to bring fdcan_tq_ck to a valid range. All other configuration parameters must be set via FDCAN_CCFG. For correct operation of FDCAN1 and FDCAN2, the APB clock fdcan_pclk needs to be equal to or higher than the time quanta clock (fdcan_tq_ck). CAN FD operation is not possible.

For startup FDCAN modules must be both configured for restricted operation (FDCAN_CCCR.ASM = 1) before FDCAN_CCCR.INIT is reset. The input clock fdcan_ker_ck must be adjusted until the clock calibration on CAN unit has reached state Precision_Calibrated. Now the software can reset FDCAN_CCCR.ASM, and the CANFD1 and CANFD2 can start normal operation.

During operation the software must check regularly whether the clock calibration on CAN unit is still in state Precision_Calibrated. In case the clock calibration on CAN unit has left state Precision_Calibrated due to drift of fdcan_ker_ck, FDCAN modules must be set into restricted operation mode by programming FDCAN_CCCR.INIT, FDCAN_CCCR.CCE, and FDCAN_CCCR.ASM to 1. After fdcan_ker_ck has been adjusted successfully (clock calibration on CAN unit is in state Precision_Calibrated), FDCAN modules can resume normal operation.

Note: Trimming accuracy must be to sufficient to meet the CAN clock tolerance requirements for the configured bitrate.

Clock calibration active

This operation mode is entered by resetting FDCAN_CCFG.BCC to 0. In this operation mode the fdcan_ker_ck is controlled by the CCU.

The generation of CCU output signal fdcan_tq_ck depends upon the state of the FDCAN1. Input clock fdcan_ker_ck must be above 80 MHz. Configuration of the CCU and FDCAN1 is required. CAN FD operation is not possible.

If FDCAN1 turns to Bus_Off or when its INIT bit is set by the user command (FDCAN_CCCR.INIT = 1), the CCU enters state Not_Calibrated. CANFD1 and CANFD2 enter restricted operation mode.

Note: This is the default operation mode after reset in case the reset value of FDCAN_CCFG.BCC is configured to 0.

71.4.8 TTCAN operations (FDCAN1 only)

Reference message

A reference message is a data frame characterized by a specific CAN identifier. It is received and accepted by all nodes except the time master (sender of the reference message).

For level 1 the data length must be at least one; for levels 0, 2 the data length must be at least four; otherwise, the message is not accepted as reference message. The reference message may be extended by other data up to the sum of eight CAN data bytes. All bits of the identifier except the three LSBs characterize the message as a reference message. The last three bits specify the priorities of up to eight potential time masters. Reserved bits are transmitted as logical 0 and are ignored by the receivers. The reference message is configured via register FDCAN_TTRMC.

The time master transmits the reference message. If the reference message is disturbed by an error, it is retransmitted immediately. In case of a retransmission, the transmitted Master_Ref_Mark is updated. The reference message is sent periodically, but is allowed to stop the periodic transmission (Next_is_Gap bit) and to initiate transmission event-synchronized at the start of the next basic cycle by the current time master or by one of the other potential time masters.

The node transmitting the reference message is the current time master. The time master is allowed to transmit other messages. If the current time master fails, its function is replicated by the potential time master with the highest priority. Nodes that are neither time master nor potential time master are time-receiving nodes.

Level 1

Level 1 operation is configured via FDCAN_TTOCF.OM = 01 and FDCAN_TTOCF.GEN. external clock synchronization is not available in level 1. The information related to the reference message is stored in the first data byte as shown in *Table 701*. Cycle_Count is optional.

Table 701. First byte of level 1 reference message

Bits	0	1	2	3	4	5	6	7
First byte	Next_is_Gap	Reserved	Cycle_Count[5:0]					

Level 2

Level 2 operation is configured via FDCAN_TTOCF.OM = 10 and FDCAN_TTOCF.GEN. The information related to the reference message is stored in the first four data bytes as shown in *Table 702*. Cycle_Count and the lower four bits of FDCAN_NTU_Res are optional. The TTCAN does not evaluate NTU_Res[3:0] from received reference messages, it always transmits these bits as 0.

Table 702. First four bytes of level 2 reference message

Bits	0	1	2	3	4	5	6	7
First byte	Next_is_Gap	Reserved	Cycle_Count[5:0]					
Second byte	NTU_Res[6:4]			NTU_Res[3:0]				Disc_Bit
Third byte	Master_Ref_Mark[7:0]							
Fourth byte	Master_Ref_Mark[15:8]							

Level 0

Level 0 operation is configured via FDCAN_TTOCF.OM = 11. External event-synchronized time-triggered operation is not available in level 0. The information related to the reference message is stored in the first four data bytes as shown in the table below. In level 0 Next_is_Gap is always 0. Cycle_Count and the lower four bits of NTU_Res are optional. The TTCAN does not evaluate NTU_Res[3:0] from received reference messages, it always transmits these bits as 0.

Table 703. First four bytes of level 0 reference message

Bits	0	1	2	3	4	5	6	7
First byte	Next_is_Gap	Reserved	Cycle_Count[5:0]					
Second byte	NTU_Res[6:4]			NTU_Res[3:0]				Disc_Bit
Third byte	Master_Ref_Mark[7:0]							
Fourth byte	Master_Ref_Mark[15:8]							

71.4.9 TTCAN configuration

TTCAN timing

The network time unit (NTU) is the unit in which all times are measured. The NTU is a constant of the whole network and is defined as a priority by the network system designer. In TTCAN level 1 the NTU is the nominal CAN bit time. In TTCAN level 0 and level 2 the NTU is a fraction of the physical second.

The NTU is the time base for the local time. The integer part of the local time (16-bit value) is incremented once each NTU. Cycle time and global time are both derived from local time. The fractional part (3-bit value) of local time, cycle time, and global time is not readable.

In TTCAN level 0 and level 2 the length of the NTU is defined by the time unit Ratio TUR. The TUR is in general a non-integer number, given by
TUR = FDCAN_TURNA.NAV / FDCAN_TURCF.DC. The NTU length is given by
NTU = CAN clock period x TUR.

The TUR numerator configuration NC is an 18-bit number, FDCAN_TURCF[NCL[15:0]] can be programmed in the range 0x0000 to 0xFFFF. FDCAN_TURCF[NCH[17:16]] is hard wired to 0b01. When 0xnnnn is written to FDCAN_TURCF[NCL[15:0]], FDCAN_TURNA.NAV starts with the value 0x10000 + 0x0nnnn = 0x1nnnn. The TUR denominator configuration FDCAN_TURCF.DC is a 14-bit number. FDCAN_TURCF.DC may be programmed in the range 0x0001 to 0x3FFF (0x0000 is an illegal value).

In level 1, NC must be equal to or higher than 4 x FDCAN_TURCF.DC. In level 0 and level 2 NC must be equal to or higher than 8 x FDCAN_TURCF.DC to get the 3-bit resolution for the internal fractional part of the NTU.

A hardware reset presets FDCAN_TURCF.DC to 0x1000 and FDCAN_TURCF.NCL to 0x10000, resulting in an NTU consisting of sixteen CAN clock periods. Local time and application watchdog are not started before either the FDCAN_CCCR.INIT is reset, or FDCAN_TURCF.ELT is set. FDCAN_TURCF.ELT may not be set before the NTU is configured. Setting FDCAN_TURCF.ELT to 1 also locks the write access to register FDCAN_TURCF.

At startup FDCAN_TURNA.NAV is updated from NC (= FDCAN_TURCF.NCL + 0x10000) when FDCAN_TURCF.ELT is set. In TTCAN level 1 there is no drift compensation. FDCAN_TURNA.NAV does not change during operation, it is always equal to NC.

In TTCAN level 0 and level 2 there are two possibilities for FDCAN_TURNA.NAV to change. When operating as time slave or backup time master, and when FDCAN_TTOCF.ECC is set, FDCAN_TURNA.NAV is updated automatically to the value calculated from the monitored global time speed, as long as the TTCAN is in synchronization state In_Schedule or In_Gap. When it loses synchronization, it returns to NC. When operating as the actual time master, and when FDCAN_TTOCF.EECS is set, the user may update FDCAN_TURCF.NCL. When the user sets FDCAN_TTOCN.ECS, FDCAN_TURNA.NAV is updated from the new value of NC at the next reference message. The status flag FDCAN_TTOST.WECS as is set when FDCAN_TTOCN.ECS is set and is cleared when FDCAN_TURNA.NAV is updated. FDCAN_TURCF.NCL is write locked while FDCAN_TTOST.WECS is set.

In TTCAN level 0 and level 2 the clock calibration process adapts FDCAN_TURNA.NAV in the range of the synchronization deviation limit SDL of NC ± 2(FDCAN_TTOCF.LDSDL + 5). FDCAN_TURCF.NCL must be programmed to the largest applicable numerical value in order to achieve the best accuracy in the calculation of FDCAN_TURNA.NAV.

The synchronization deviation SD is the difference between NC and FDCAN_TURNA.NAV (SD = | NC - FDCAN_TURNA.NAV |). It is limited by the synchronization deviation limit SDL, which is configured by its dual logarithm FDCAN_TTOCF.LDSDL (SDL = 2 (FDCAN_TTOCF.LDSDL + 5)) and must not exceed the clock tolerance given by the CAN bit timing configuration. SD is calculated at each new basic cycle. When the calculated TURNA[NAV deviates by more than SDL from NC, or if the Disc_Bit in the reference message is set, the drift compensation is suspended and FDCAN_TTIR.GTE is set and FDCAN_TTOSC.QCS is reset, or in case of the Disc_Bit = 1, FDCAN_TTIR.GTD is set.

Table 704. TUR configuration example

TUR	8	10	24	50	510	125000	32.5	100/12	529/17
NC	0x1FFF8	0x1FFFE	0x1FFF8	0x1FFEA	0x1FFFE	0x1E848	0x1FFE0	0x19000	0x10880
FDCAN _TURCF.DC	0x3FFF	0x3333	0x1555	0x0A3D	0x0101	0x0001	0x0FC0	0x3000	0x0880

FDCAN_TTOCN.ECS schedules NC for activation by the next reference message. FDCAN_TTOCN.SGT schedules FDCAN_TTGTP.TP for activation by the next reference message. Setting FDCAN_TTOCN.ECS and FDCAN_TTOCN.SGT requires FDCAN_TTOCF.EECS to be set (external clock synchronization enabled) while the FDCAN is actual time master.

The TTCAN module provides an application watchdog to verity the function of the application program. The user must serve this watchdog regularly, else all CAN bus activity is stopped. The application watchdog limit FDCAN_TTOCF.AWL specifies the number of NTUs between two times the watchdog must be served. The maximum number of NTUs is 256. The application watchdog is served by reading register FDCAN_TTOST. FDCAN_TTOST.AWE indicates whether the watchdog has been served in time. In case the application failed to serve the application watchdog, the interrupt flag FDCAN_TTIR.AW is set. For software development, the application watchdog may be disabled by programming FDCAN_TTOCF.AWL to 0x00, see *Section 71.6.3*.

Timing of interface signals

The timing events that cause a pulse at output FDCAN trigger time mark interrupt pulse fdcan1_tmp for more than one instance and fdcan_tmp if only one instance; FDCAN register time mark interrupt pulse fdcan1_rtp for more than one instance and fdcan_rtp if only one instance are generated in the CAN clock domain.

There is a clock domain crossing delay to be considered before the same event is visible in the APB clock domain (when FDCAN_TTIR.TTMI is set or FDCAN_TTIR.RTMI is set). As an example, the signals can be connected to the timing input(s) of another FDCAN node (fdcan_swt/fdcan_evt), in order to automatically synchronize two TTCAN networks.

Output FDCAN start of cycle_fdcan1_soc for more than one instance and cycle_fdcan1_soc if only one instance gets active whenever a reference message is completed (either transmitted or received). The output is controlled in the APB clock domain.

71.4.10 Message scheduling

FDCAN_TTOCF.TM controls whether the TTCAN operates as potential time master or as a time slave. If it is a potential time master, the three LSBs of the reference message identifier FDCAN_TTRMC.RID define the master priority, 0 giving the highest and 7 giving the lowest. There cannot be two nodes in the network using the same master priority. FDCAN_TTRMC.RID is used for recognition of reference messages. FDCAN_TTRMC.RMPS is not relevant for time slaves.

The initial reference trigger offset FDCAN_TTOCF.IRTO is a 7-bit-value that defines (in NTUs) how long a backup time master waits before it starts the transmission of a reference message when a reference message is expected but the bus remains idle. The recommended value for FDCAN_TTOCF.IRTO is the master priority multiplied with a factor depending on the expected clock drift between the potential time masters in the network. The sequential order of the backup time masters, when one of them starts the reference message in case the current time master fails, must correspond to their master priority, even with maximum clock drift.

FDCAN_TTOCF.OM decides whether the node operates in TTCAN level 0, level 1, or level 2. In one network, all potential time masters have to operate on the same level. Time slaves may operate on level 1 in a level 2 network, but not vice versa. The configuration of the TTCAN operation mode via FDCAN_TTOCF.OM is the last step in the setup. With FDCAN_TTOCF.OM = 00 (event-driven CAN communication), the FDCAN operates

according to ISO 11898-1: 2015, without time triggers.With FDCAN_TTOCF.OM = 01 (level 1), the FDCAN operates according to ISO 11898-4, but without the possibility to synchronize the basic cycles to external events, the Next_is_Gap bit in the reference message is ignored. With FDCAN_TTOCF.OM = 10 (level 2), the TTCAN operates according to ISO 11898-4, including the event-synchronized start of a basic cycle. With FDCAN_TTOCF.OM = 11 (level 0), the FDCAN operates as event-driven CAN but maintains a calibrated global time base as in level 2.

FDCAN_TTOCF.EECS enables the external clock synchronization, allowing the application program of the current time master to update the TUR configuration during time-triggered operation, to adapt the clock speed and (in levels 0 and level 2 only) the global clock phase to an external reference.

FDCAN_TTMLM.ENTT in the TT matrix limits register specifies the number of expected Tx_Triggers in the system matrix. This is the sum of Tx_Triggers for exclusive, single arbitrating and merged arbitrating windows, excluding the Tx_Ref_Triggers. Note that this is usually not the number of Tx_Trigger memory elements; the number of basic cycles in the system matrix and the trigger repeat factors must be taken into account.

An inaccurate configuration of FDCAN_TTMLM.ENTT results either in a TxCount Underflow (FDCAN_TTIR.TXU = 1 and FDCAN_TTOST.EL = 01, severity 1), or in a Tx count Overflow (FDCAN_TTIR.TXO = 1 and FDCAN_TTOST.EL = 10, severity 2).

Note: In case the first reference message seen by a node does not have Cycle_Count 0, this node may finish its first matrix cycle with its Tx count resulting in a Tx count underflow condition. As long as a node is in state Synchronizing, its Tx_Triggers do not lead to transmissions.

FDCAN_TTMLM.CCM specifies the number of the last basic cycle in the system matrix. The counting of basic cycles starts at 0. In a system matrix consisting of eight basic cycles FDCAN_TTMLM.CCM would be 7. FDCAN_TTMLM.CCM is ignored by time slaves, a receiver of a reference message considers the received cycle count as the valid cycle count for the actual basic cycle.

FDCAN_TTMLM.TXEW specifies the length of the Tx enable window in NTUs. The Tx enable window is that period of time at the beginning of a time window where a transmission may be started. If a transmission of a message cannot be started inside the Tx enable window because of for example, a slight overlap from the previous time window message, the transmission cannot be started in that time window at all. FDCAN_TTMLM.TXEW must be chosen with respect to the network synchronization quality and with respect to the relation between the length of the time windows and the length of the messages.

Trigger memory

The trigger memory is part of the external message RAM to which the TTCAN is connected to (see *Section 71.5.26*). It stores up to 64 trigger elements. A trigger memory element consists of time mark TM, cycle code CC, trigger type TYPE, filter type FTYPE, message number MNR, message status count MSC, time mark event internal TMIN, time mark event external TMEX (see *Section 71.4.24*).

The time mark defines at which cycle time a trigger becomes active. The triggers in the trigger memory must be sorted by their time marks. The trigger element with the lowest time mark is written to the first trigger memory word. Message number and cycle code are ignored for triggers of type Tx_Ref_Trigger, Tx_Ref_Trigger_Gap, Watch_Trigger, Watch_Trigger_Gap, and End_of_List.

When the cycle time reaches the time mark of the actual trigger, the FSE switches to the next trigger and starts to read the following trigger from the trigger memory. In case of a

transmit trigger, the Tx handler starts to read the message from the message RAM as soon as the FSE switches to its trigger. The RAM access speed defines the minimum time step between a transmit trigger and its preceding trigger, the Tx handler must be able to prepare the transmission before the transmit trigger time mark is reached. The RAM access speed also limits the number of non-matching (with regard to their cycle code) triggers between two matching triggers, the next matching trigger must be read before its time mar k is reached. If the reference message is n NTU long, a trigger with a time mark lower than n never becomes active and is treated as a configuration error.

Starting point of the cycle time is the sample point of the reference message start of frame bit. The next reference message is requested when cycle time reaches the Tx_Ref_Trigger time mark. The FDCAN reacts on the transmission request at the next sample point. A new Sync_Mark is captured at the start of frame bit, but the cycle time is incremented until the reference message is successfully transmitted (or received) and the Sync_Mark is taken as the new Ref_Mark. At that point in time, cycle time is restarted. As a consequence, cycle time can never (with the exception of initialization) be seen at a value lower than n, with n being the length of the reference message measured in NTU.

Length of a basic cycle: Tx_Ref_Trigger time mark + 1 NTU + 1 CAN bit time.

The trigger list is different for all nodes in the CAN FD network. Each node knows only the Tx_Triggers for its own transmit messages, the Rx_Triggers for those receive messages that are processed by this node, and the triggers concerning the reference messages.

Trigger types

Tx_Ref_Trigger (TYPE = 0000) and Tx_Ref_Trigger_Gap (TYPE = 0001) cause the transmission of a reference message by a time master. A configuration error (FDCAN_TTOST.EL = 11, severity 3) is detected when a time slave encounters a Tx_Ref_Trigger(_Gap) in its trigger memory. Tx_Ref_Trigger_Gap is only used in external event-synchronized
time-triggered operation mode. In that mode, Tx_Ref_Trigger is ignored when the FDCAN synchronization state is In_Gap (FDCAN_TTOST.SYS = 10).

Tx_Trigger_Single (TYPE = 0010), Tx_Trigger_Continuous (TYPE = 0011), Tx_Trigger_Arbitration (TYPE = 0100), and Tx_Trigger_Merged (TYPE = 0101) cause the start of a transmission. They define the start of a time window.

Tx_Trigger_Single starts a single transmission in an exclusive time window when the message buffer transmission request pending bit is set. After successful transmission the transmission request pending bit is reset.

Tx_Trigger_Continuous starts a transmission in an exclusive time window when the message buffer transmission request pending bit is set. After successful transmission the transmission request pending bit remains set, and the message buffer is transmitted again in the next matching time window.

Tx_Trigger_Arbitration starts an arbitrating time window, Tx_Trigger_Merged a merged arbitrating time window. The last Tx_Trigger of a merged arbitrating time window must be of type Tx_Trigger_Arbitration. A configuration error (FDCAN_TTOST.EL = 11, severity 3) is detected when a trigger of type Tx_Trigger_Merged is followed by any other Tx_Trigger than one of type Tx_Trigger_Merged or Tx_Trigger_Arbitration. Several Tx_Triggers may be defined for the same Tx message buffer. Depending on their cycle code, they may be ignored in some basic cycles. The cycle code must be considered when the expected number of Tx_Triggers (FDCAN_TTMLM.ENTT) is calculated.

Watch_Trigger (TYPE = 0110) and Watch_Trigger_Gap (TYPE = 0111) check for missing reference messages. They are used by both time masters and time slaves. Watch_Trigger_Gap is only used in external event-synchronized time-triggered operation mode. In that mode, a Watch_Trigger is ignored when the FDCAN synchronization state is In_Gap (FDCAN_TTOST.SYS = 10).

Rx_Trigger (TYPE = 1000) is used to check for the reception of periodic messages in exclusive time windows. Rx_Triggers are not active until state In_Schedule or In_Gap is reached. The time mark of an Rx_Trigger must be placed after the end of that message transmission, independent of time window boundaries. Depending on their cycle code, Rx_Triggers may be ignored in some basic cycles. At the time mark of the Rx_Trigger, it is checked whether the last received message before this time mark and after start of cycle or previous Rx_Trigger had matched the acceptance filter element referenced by MNR. Accepted messages are stored in one of the two receive FIFOs, according to the acceptance filtering, independent of the Rx_Trigger. Acceptance filter elements referenced by Rx_Triggers must be placed at the beginning of the filter list to ensure that the filtering is finished before the Rx_Trigger time mark is reached.

Time_Base_Trigger (TYPE = 1001) are used to generate internal/external events depending on the configuration of TMIN and TMEX.

End_of_List (TYPE = 1010 … 1111) is an illegal trigger type, a configuration error (FDCAN_TTOST.EL = 11, severity 3) is detected when an End_of_List trigger is encountered in the trigger memory before the Watch_Trigger and Watch_Trigger_Gap.

Restrictions for the node trigger list

There may not be two triggers that are active at the same cycle time and cycle count, but triggers that are active in different basic cycles (different cycle code) may share the same time mark.

Rx_Triggers and Time_Base_Triggers may not be placed inside the Tx enable windows of Tx_Trigger_Single/Continuous/Arbitration, but they may be placed after Tx_Trigger_Merged.

Triggers that are placed after the Watch_Trigger (or the Watch_Trigger_Gap when FDCAN_TTOST.SYS = 10) never become active. The watch triggers themselves do not become active when the reference messages are transmitted on time.

All unused trigger memory words (after the Watch_Trigger or after the Watch_Trigger_Gap when FDCAN_TTOST.SYS = 10) must be set to trigger type End_of_List.

A typical trigger list for a potential time master begins with a number of Tx_Triggers and Rx_Triggers followed by the Tx_Ref_Trigger and the Watch_Trigger. For networks with external event- synchronized time-triggered communication, this is followed by the Tx_Ref_Trigger_Gap and the Watch_Trigger_Gap. The trigger list for a time slave is the same but without the Tx_Ref_Trigger and the Tx_Ref_Trigger_Gap.

At the beginning of each basic cycle, that is at each reception or transmission of a reference message, the trigger list is processed starting with the first trigger memory element. The FSE looks for the first trigger with a cycle code that matches the current cycle count. The FSE waits until cycle time reaches the trigger time mark and activates the trigger. Afterwards the FSE looks for the next trigger in the list with a cycle code that matches the current cycle count.

Special consideration is needed for the time around Tx_Ref_Trigger and Tx_Ref_Trigger_Gap. In a time master competing for master ship, the effective time mark of

a Tx_Ref_Trigger may be decremented in order to be the first node to start a reference message. In backup time masters the effective time mark of a Tx_Ref_Trigger or Tx_Ref_Trigger_Gap is the sum of its configured time mark and the reference trigger offset FDCAN_TTOCF.IRTO. In case error level 2 is reached (FDCAN_TTOST.EL = 10), the effective time mark is the sum of its time mark and 0x127. No other trigger elements must be placed in this range otherwise it may happen that the time marks appear out of order and are flagged as a configuration error. Trigger elements which are coming after Tx_Ref_Trigger may never become active as long as the reference messages come in time.

There are interdependencies between the following parameters:

- APB clock frequency
- Speed and waiting time for trigger RAM accesses
- Length of the acceptance filter list
- Number of trigger elements
- Complexity of cycle code filtering in the trigger elements
- Offset between time marks of the trigger elements

Example for trigger handling

The example shows how the trigger list is derived from a node system matrix. Assumption is that node A is first time master and has knowledge of the section of the system matrix shown in *Table 705*.

Table 705. System matrix, Node A

Cycle count	Time mark						
	1	2	3	4	5	6	7
0	Tx7	-	-	-	-	TxRef	Error
1	Rx3	-	Tx2, Tx4		-	TxRef	Error
2	-	-	-	-	-	TxRef	Error
3	Tx7	-	Rx5	-	-	TxRef	Error
4	Tx7	-	-	Rx6	-	TxRef	Error

The cycle count starts with 0 and runs until 0, 1, 3, 7, 15, 31, 63 (the corresponding number of basic cycles in the system matrix is, respectively, 1, 2, 4, 8, 16, 32, 64). The maximum cycle count is configured by FDCAN_TTMLM.CCM. The cycle code CC is composed of repeat factor (= value of most significant 1) and the number of the first basic cycle in the system matrix (= bit field after most significant 1).

As an example, with a cycle code of 0b0010011 (repeat factor = 16, first basic cycle = 3) and a maximum cycle count of FDCAN_TTMLM.CCM = 0x3F matches occur at cycle counts 3, 19, 35 and 51.

A trigger element consists of time mark TM, cycle code CC, trigger type TYPE, and message number MNR. For transmission MNR references the Tx buffer number (0 ... 31). For reception MNR references the number of the filter element (0 ... 127) that matched during acceptance filtering. Depending on the configuration of the filter type FTYPE, the 11-bit or 29-bit message ID filter list is referenced.

In addition, a trigger element can be configured for generation of time mark event internal TMIN, and time mark event external TMEX. The message status count MSC holds the counter value (0 ... 7) for scheduling errors for periodic messages in exclusive time windows at the point in time when the time mark of the trigger element became active.

Table 706. Trigger list, Node A

Trigger	Time mark TM[15:0]	Cycle code CC [6:0]	Trigger type TYPE [3:0]	Mess No. MNR [6:0]
0	Mark1	0b0000100	Tx_Trigger_Single	7
1	Mark 1	0b1000000	Rx_Trigger	3
2	Mark 1	0b1000011	Tx_Trigger_Single	7
3	Mark 3	0b1000001	Tx_Trigger_Merged	2
4	Mark 3	0b1000011	Rx_Trigger	5
5	Mark 4	0b1000001	Tx_Trigger_Arbitration	4
6	Mark 4	0b1000100	Rx_Trigger	6
7	Mark 6	N/A	Tx_Ref_Trigger	0 (Ref)
8	Mark 7	N/A	Watch_Trigger	N/A
9	N/A	N/A	End_of_List	N/A

Tx_Trigger_Single, Tx_Trigger_Continuous, Tx_Trigger_Merged, Tx_Trigger_Arbitration, Rx_Trigger, and Time_Base_Trigger are only valid for the specified cycle code. For all other trigger types the cycle code is ignored.

The FSE starts the basic cycle with scanning the trigger list starting from 0 until a trigger with time mark higher than cycle time and with its cycle code CC matching the actual cycle count is reached, or a trigger of type Tx_Ref_Trigger, Tx_Ref_Trigger_Gap, Watch_Trigger, or Watch_Trigger_Gap is encountered.

When the cycle time reached the time mark TM, the action defined by trigger type TYPE and message number MNR is started. There is an error in the configuration when End_of_List is reached.

At mark 6 the reference message (always TxRef) is transmitted. After transmission of the reference message the FSE returns to the beginning of the trigger list. When the watch trigger at mark 7 is reached, the node was not able to transmit the reference message; error treatment is started.

Detection of configuration errors

A configuration error is signaled via FDCAN_TTOST.EL = 11 (severity 3) when:

- The FSE comes to a trigger in the list with a cycle code that matches the current cycle count but with a time mark that is less than the cycle time.
- The previous active trigger was a Tx_Trigger_Merged and the FSE comes to a trigger in the list with a cycle code that matches the current cycle count but that is neither a

Tx_Trigger_Merged nor a Tx_Trigger_Arbitration nor a Time_Base_Trigger nor an Rx_Trigger.

- The FSE of a node with FDCAN_TTOCF.TM=0 (time slave) encounters a Tx_Ref_Trigger or a Tx_Ref_Trigger_Gap.
- Any time mark placed inside the Tx enable window (defined by FDCAN_TTMLM.TXEW) of a Tx_Trigger with a matching cycle code.
- A time mark is placed near the time mark of a Tx_Ref_Trigger and the reference trigger offset FDCAN_TTOST.RTO causes a reversal of their sequential order measured in cycle time.

TTCAN schedule initialization

The synchronization to the TTCAN message schedule starts when FDCAN_CCCR.INIT is reset. The TTCAN can operate strictly time-triggered (FDCAN_TTOCF.GEN = 0) or external event-synchronized time-triggered (FDCAN_TTOCF.GEN = 1). All nodes start with cycle time 0 at the beginning of their trigger list with FDCAN_TTOST.SYS = 00 (out of synchronization), no transmission is enabled with the exception of the reference message. Nodes in external event-synchronized time-triggered operation mode ignore Tx_Ref_Trigger and Watch_Trigger and use instead Tx_Ref_Trigger_Gap and Watch_Trigger_Gap until the first reference message decides whether a Gap is active.

Time slaves

After configuration, a time slave ignores its Watch_Trigger and Watch_Trigger_Gap if it does not receive any message before reaching the Watch_Triggers. When it reaches Init_Watch_Trigger, interrupt flag FDCAN_TTIR.IWT is set, the FSE is frozen, and the cycle time becomes invalid, but the node is still able to take part in CAN bus communication (to acknowledge or to send error flags). The first received reference message restarts the FSE and the cycle time.

Note: Init_Watch_Trigger is not part of the trigger list. It is implemented as an internal counter that counts up to 0xFFFF = maximum cycle time.

When a time slave has received any message but the reference message before reaching the Watch_Triggers, it assumes a fatal error (FDCAN_TTOST.EL = 11, severity 3), set interrupt flag FDCAN_TTIR.WT, switch off its CAN bus output, and enter the bus monitoring mode (FDCAN_CCCR.MON set to 1). In the bus monitoring mode it is still able to receive messages, but cannot send any dominant bits and therefore cannot give acknowledge.

Note: To leave the fatal error state, the user must set FDCAN_CCCR.INIT = 1. After reset of FDCAN_CCCR.INIT, the node restarts TTCAN communication.

When no error is encountered during synchronization, the first reference message sets FDCAN_TTOST.SYS = 01 (Synchronizing), the second sets the FDCAN synchronization state (depending on its Next_is_Gap bit) to FDCAN_TTOST.SYS = 11 (In_Schedule) or FDCAN_TTOST.SYS = 10 (In_Gap), enabling all Tx_Triggers and Rx_Triggers.

Potential time masters

After configuration, a potential time master starts the transmission of a reference message when it reaches its Tx_Ref_Trigger (or its Tx_Ref_Trigger_Gap when in external event-synchronized time-triggered operation). It ignores its Watch_Trigger and Watch_Trigger_Gap when it does not receive any message or transmit the reference message successfully before reaching the Watch_Triggers (assumed reason: all other nodes still in reset or configuration, giving no acknowledge). When it reaches

Init_Watch_Trigger, the attempted transmission is aborted, interrupt flag FDCAN_TTIR.IWT is set, the FSE is frozen, and the cycle time becomes invalid, but the node is still able to take part in CAN bus communication (to give acknowledge or to send error flags). Resetting FDCAN_TTIR.IWT re-enables the transmission of reference messages until the next time the Init_Watch_Trigger condition is met, or another CAN message is received. The FSE is not be restarted by the reception of a reference message.

When a potential time master reaches the Watch_Triggers after it has received any message but the reference message, it assumes a fatal error (FDCAN_TTOST.EL = 11, severity 3), sets interrupt flag FDCAN_TTIR.WT, switches off its CAN bus output, and enters the bus monitoring mode (FDCAN_CCCR.MON set to 1). In bus monitoring mode, it is still able to receive messages, but cannot send any dominant bits and therefore cannot give acknowledge.

When no error is detected during initialization, the first reference message sets FDCAN_TTOST.SYS = 01 (synchronizing), the second sets the FDCAN synchronization state (depending on its Next_is_Gap bit) to FDCAN_TTOST.SYS = 11 (In_Schedule) or FDCAN_TTOST.SYS = 10 (In_Gap), enabling all Tx_Triggers and Rx_Triggers.

A potential time master is current time master (FDCAN_TTOST.MS = 11) when it was the transmitter of the last reference message, else it is backup time master (FDCAN_TTOST.MS = 10).

When all potential time masters have finished configuration, the node with the highest time master priority in the network becomes the current time master.

71.4.11 TTCAN gap control

All functions related to gap control apply only when the FDCAN is operated in external event synchronized time-triggered mode (FDCAN_TTOCF.GEN = 1). In this operation mode the FDCAN message schedule may be interrupted by inserting gaps between the basic cycles of the system matrix. All nodes connected to the CAN network must be configured for external event- synchronized time-triggered operation.

During a gap, all transmissions are stopped and the CAN bus remains idle. A gap is finished when the next reference message starts a new basic cycle. A gap starts at the end of a basic cycle that itself was started by a reference message with bit Next_is_Gap = 1 for example gaps are initiated by the current time master.

The current time master has two options to initiate a gap. A gap can be initiated under software control when the application program writes FDCAN_TTOCN.NIG = 1. The Next_is_Gap bit is transmitted as 1 with the next reference message. A gap can also be initiated under hardware control when the application program enables the event trigger input pin fdcan_evt by writing FDCAN_TTOCN.GCS = 1. When a reference message is started and FDCAN_TTOCN.GCS is set, a HIGH level at event trigger pin fdcan_evt sets Next_is_Gap = 1.

As soon as that reference message is completed, the FDCAN_TTOST.WFE bit announces the gap to the time master as well as to the time slaves. The current basic cycle continues until its last time window. The time after the last time window is the gap time.

For the actual time master and the potential time masters, FDCAN_TTOST.GSI is set when the last basic cycle has finished and the gap time starts. In nodes that are time slaves, bit FDCAN_TTOST.GSI remains at 0.

When a potential time master is in synchronization state In_Gap (FDCAN_TTOST.SYS = 10), it has four options to intentionally finish a gap:

1. Under software control by writing FDCAN_TTOCN.FGP = 1.
2. Under hardware control (FDCAN_TTOCN.GCS = 1) an edge from HIGH to LOW at the event-trigger input pin fdcan_evt sets FDCAN_TTOCN.FGP and restarts the schedule.
3. The third option is a time-triggered restart. When FDCAN_TTOCN.TMG = 1, the next register time mark interrupt (FDCAN_TTIR.RTMI = 1) sets FDCAN_TTOCN.FGP and starts the reference message.
4. Finally any potential time master finishes a gap when it reaches its Tx_Ref_Trigger_Gap, assuming that the event to synchronize on did not occur in time.

None of these options can cause a basic cycle to be interrupted with a reference message.

Setting of FDCAN_TTOCN.FGP after the gap time begins starts the transmission of a reference message immediately and thereby synchronizes the message schedule. When FDCAN_TTOCN.FGP is set before the gap time has started (while the basic cycle is still in progress), the next reference message is started at the end of the basic cycle, at the Tx_Ref_Trigger – there is no gap time in the message schedule.

In strictly time-triggered operation, bit Next_is_Gap = 1 in the reference message is ignored, as well as the event-trigger input pin fdcan_evt and the bits FDCAN_TTOCN.NIG, FDCAN_TTOCN.FGP, and FDCAN_TTOCN.TMG.

71.4.12 Stop watch

The stop watch function enables capturing of FDCAN internal time values (local time, cycle time, or global time) triggered by an external event.

To enable the stop watch function, the application program first must define local time, cycle time, or global time as the stop watch source via FDCAN_TTOCN.SWS. When FDCAN_TTOCN.SWS is different from 00 and TT interrupt register flag FDCAN_TTIR.SWE is 0, the actual value of the time selected by FDCAN_TTTOCN.SWS is copied into FDCAN_TTCPT.SWV on the next rising / falling edge (as configured via FDCAN_TTOCN.SWP) on stop watch trigger pin fdcan_swt. This sets interrupt flag FDCAN_TTIR.SWE. After the application program has read FDCAN_TTCPT.SWV, it may enable the next stop watch event by resetting FDCAN_TTIR.SWE to 0.

71.4.13 Local time, cycle time, global time, and external clock synchronization

There are two possible levels in time-triggered CAN:

1. Level 1 only provides time-triggered operation using cycle time.
2. Level 2 additionally provides increased synchronization quality, global time and external clock synchronization. In both levels, all timing features are based on a local time base - the local time.

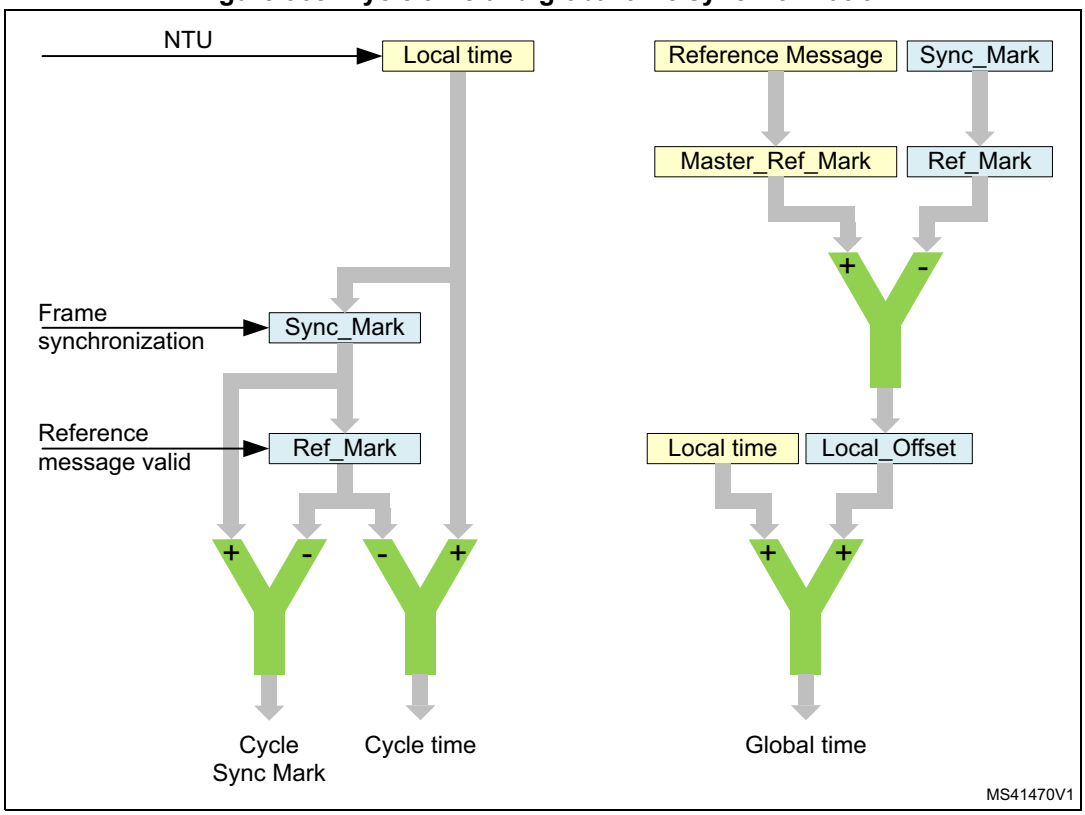
The local time is a 16-bit cyclic counter, it is incremented once each NTU. Internally the NTU is represented by a 3-bit counter which can be regarded as a fractional part (three binary digits) of the local time. Generally, the 3-bit NTU counter is incremented eight times each NTU. If the length of the NTU is shorter than eight CAN clock periods (as may be configured in level 1, or as a result of clock calibration in level 2), the length of the NTU fraction is adapted, and the NTU counter is incremented only four times each NTU.

Figure 968 describes the synchronization of the cycle time and global time, performed in the same manner by all FDCAN nodes, including the time master. Any message received or transmitted invokes a capture of the local time taken at the message is frame

synchronization event. This frame synchronization event occurs at the sample point of each start of frame (SoF) bit and causes the local time to be stored as Sync_Mark. Sync_Marks and Ref_Marks are captured including the 3-bit fractional part.

Whenever a valid reference message is transmitted or received, the internal Ref_Mark is updated from the Sync_Mark. The difference between Ref_Mark and Sync_Mark is the cycle sync mark (cycle sync mark = Sync_Mark - Ref_Mark) stored in register FDCAN_TTCSM. The most significant 16 bits of the difference between Ref_Mark and the actual value of the local time is the cycle time (cycle time = local time - Ref_Mark).

Figure 968. Cycle time and global time synchronization



The cycle time that can be read from FDCAN_TTCTC.CT is the difference of the node local time and Ref_Mark, both synchronized into the APB clock domain and truncated to 16 bit.

The global time exists for TTCAN level 0 and level 2 only, in level 1 it is invalid. The node view of the global time is the local image of the global time in (local) NTUs. After configuration, a potential time master uses its own local time as global time. The time master establishes its own local time as global time by transmitting its own Ref_Marks as Master_Ref_Marks in the reference message (bytes 3 and 4). The global time that can be read from FDCAN_TTLGT.GT is the sum of the node local time and its local offset, both synchronized into the APB clock domain and truncated to 16 bit. The fractional part is used for clock synchronization only.

A node that receives a reference message calculates its local offset to the global time by comparing its local Ref_Mark with the received Master_Ref_Mark (see *Figure 969*). The node view of the global time is local time plus local offset. In a potential time master that has never received another time master reference message, Local_Offset is 0. When a node becomes the current time master after first having received other reference messages,

Local_Offset is frozen at its last value. In the time receiving nodes, Local_Offset may be subject to small adjustments, due to clock drift, when another node becomes time master, or when there is a global time discontinuity, signaled by Disc_Bit in the reference message. With the exception of global time discontinuity, the global time provided to the application program by register FDCAN_TTLGT is smoothed by a low-pass filtering to have a continuous monotonic value.

Figure 969. TTCAN level 0 and level 2 drift compensation

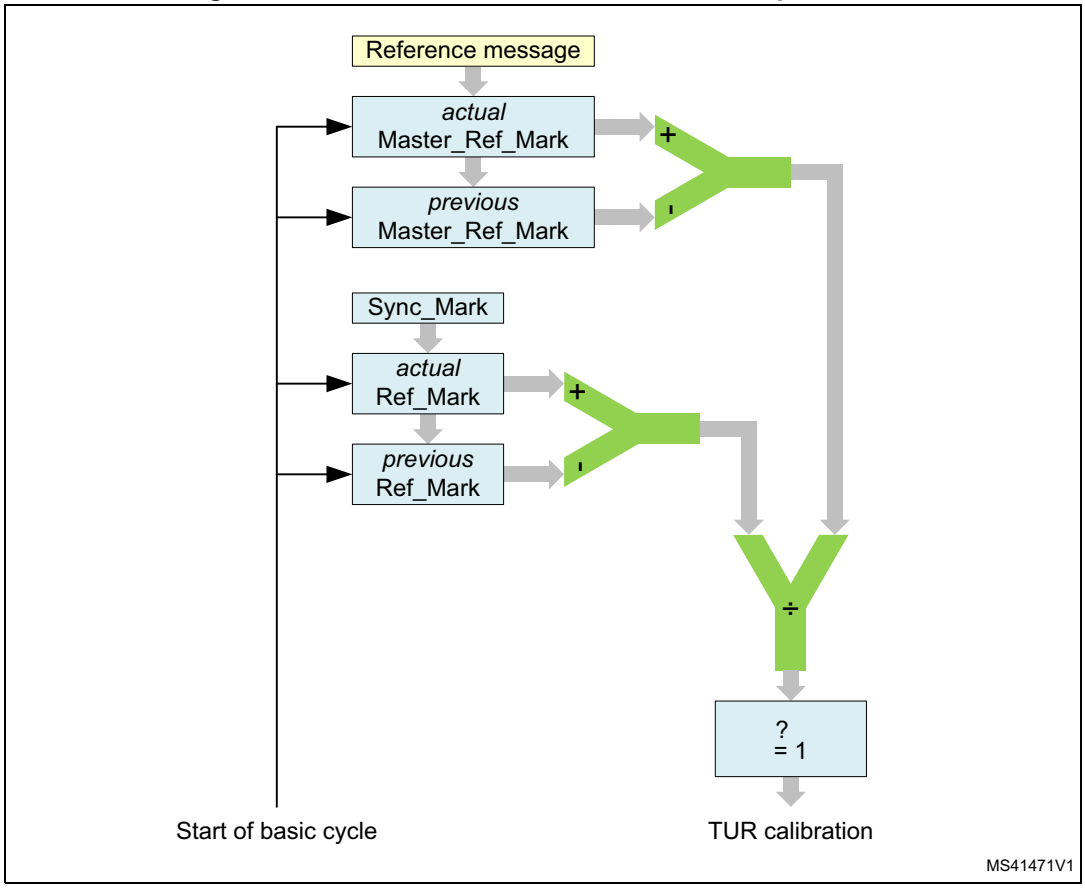


Figure 969 describes how in TTCAN levels 0 and 2 each time receiving node compensates the drift between its own local clock and the time master clock by comparing the length of a basic cycle in local time and in global time. If there is a difference between the two values and the Disc_Bit in the reference message is not set, a new value for FDCAN_TURNA.NAV is calculated. If the synchronization deviation SD = | NC - FDCAN_TURNA.NAV | ≤ SDL (synchronization deviation limit), the new value for FDCAN_TURNA.NAV takes effect. Else the automatic drift compensation is suspended.

In TTCAN level 0 and level 2, FDCAN_TTOST.QCS indicates whether the automatic drift compensation is active or suspended. In TTCAN level 1, FDCAN_TOST.QCS is always 1.

The current time master may synchronize its local clock speed and the global time phase to an external clock source. This is enabled by bit FDCAN_TTOCF.EECS.

The stop watch function (see *Section 71.4.12*) may be used to measure the difference in clock speed between the local clock and the external clock. The local clock speed is adjusted by first writing the newly calculated numerator configuration low to

FDCAN_TURCF.NCL (FDCAN_TURCF.DC cannot be updated during operation). The new value takes effect by writing FDCAN_TTOCN.ECS to 1.

The global time phase is adjusted by first writing the phase offset into the TT global time Preset register FDCAN_TTGTP. The new value takes effect by writing FDCAN_TTOCN.SGT to 1. The first reference message transmitted after the global time phase adjustment has the Disc_Bit set to 1.

FDCAN_TTOST.QGTP shows whether the node global time is in phase with the time master global time. FDCAN_TTOST.QGTP is permanently 0 in TTCAN level 1 and when the synchronization deviation limit is exceeded in TTCAN levels 0, 2 (FDCAN_TTOST.QCS = 0). It is temporarily 0 while the global time is low-pass filtered to supply the application with a continuous monotonic value. There is no low-pass filtering when the last reference message contained a Disc_Bit = 1 or when FDCAN_TTOST.QCS = 0.

71.4.14 TTCAN error level

The ISO 11898-4 specifies four levels of error severity:

1. S0 - No error
2. S1 - Warning - Only notification of application, reaction application-specific.
3. S2 error - Notification of application. All transmissions in exclusive or arbitrating time windows are disabled (i.e. no data or remote frames may be started). Potential time masters still transmit reference messages with the reference trigger offset FDCAN_TTOST.RTO set to the maximum value of 127.
4. S3 - Severe error - Notification of application. All CAN bus operations are stopped, that is, transmission of dominant bits is not allowed, and FDCAN_CCCR.MON is set. The S3 error condition remains active until the application updates the configuration (set FDCAN_CCCR.CCE).

If several errors are detected at the same time, the highest severity prevails. When an error is detected, the application is notified by FDCAN_TTIR.ELC. The error level is monitored by FDCAN_TTOST.EL.

The TTCAN signals the following error conditions as required by ISO 11898-4:

- Config_error (S3)
  - Sets error level FDCAN_TTOST.EL to 11 when a merged arbitrating time window is not properly closed or when there is a Tx_Trigger with a time mark beyond the Tx_Ref_Trigger.
- Watch_Trigger_Reached (S3)
  - Sets error level FDCAN_TTOST.EL to 11 when a watch trigger was reached because the reference message is missing.
- Application_Watchdog (S3)
  - Sets error level FDCAN_TTOST.EL to 11 when the application failed to serve the application watchdog.The application watchdog is configured via FDCAN_TTOCF.AWL. It is served by reading register FDCAN_TTOST. When the watchdog is not served in time, bit FDCAN_TTOST.AWE and interrupt flag

FDCAN_TTIR.AW are set, all FDCAN communication is stopped, and the FDCAN is set into bus monitoring mode (FDCAN_CCCR.MON set to 1).

- CAN_Bus_Off (S3)
  - Entering CAN_Bus_Off state sets error level FDCAN_TTOST.EL to 11. CAN_Bus_Off state is signaled by FDCAN_PSR.BO = 1 and FDCAN_CCCR.INIT = 1.
- Scheduling_Error_2 (S2)
  - Sets error level FDCAN_TTOST.EL to 10 if the MSC of one Tx_Trigger has reached 7. In addition, the interrupt flag FDCAN_TTIR.SE2 is set. The error level FDCAN_TTOST.EL is reset to 00 at the beginning of a matrix cycle when no Tx_Trigger has an MSC of 7 in the preceding matrix cycle.
- Tx_Overflow (S2)
  - Sets error level FDCAN_TTOST.EL to 10 when the Tx count is equal to or higher than the expected number of Tx_T riggers FDCAN_TTMLM.ENTT and a Tx_Trigger event occurs. In addition, the interrupt flag FDCAN_TTIR.TXO is set. The error level FDCAN_TTOST.EL is reset to 00 when the Tx count is no more than FDCAN_TTMLM.ENTT at the start of a new matrix cycle.
- Scheduling_Error_1 (S1)
  - Sets error level FDCAN_TTOST.EL to 01 if within one matrix cycle the difference between the maximum MSC and the minimum MSC for all trigger memory elements (of exclusive time windows) is larger than two, or if one of the MSCs of an exclusive Rx_Trigger has reached seven. In addition, the interrupt flag FDCAN_TTIR.SE1 is set. If within one matrix cycle none of these conditions is valid, the error level FDCAN_TTOST.EL is reset to 00.
- Tx_Underflow (S1)
  - Sets error level FDCAN_TTOST.EL to 01 when the Tx count is less than the expected number of Tx_Triggers FDCAN_TTMLM.ENTT at the start of a new matrix cycle. In addition, the interrupt flag FDCAN_TTIR.TXU is set. The error level FDCAN_TTOST.EL is reset to 00 when the Tx count is at least FDCAN_TTMLM.ENTT at the start of a new matrix cycle.

71.4.15 TTCAN message handling

Reference message

For potential time masters the identifier of the reference message is configured via FDCAN_TTRMC.RID. No dedicated Tx buffer is required for transmission of the reference message. When a reference message is transmitted, the first data byte for TTCAN level 1 (that is, the first four data bytes for TTCAN level 0 and the first four data bytes for TTCAN level 2) is provided by the FSE.

In case the reference message Payload select FDCAN_TTRMC.RMPS is set, the rest of the reference message payload (level 1: bytes 2-8, levels 0, 2: bytes 5-6) is taken from Tx buffer 0. In this case the data length DLC code from message buffer 0 is used.

Table 707. Number of data bytes transmitted with a reference message

FDCAN_TTRMC.RMPS	FDCAN_TXBRP.TRP0	Level 0	Level 1	Level 2
0	0	4	1	4
0	1	4	1	4

Table 707. Number of data bytes transmitted with a reference message (continued)

FDCAN_TTRMC.RMPS	FDCAN_TXBRP.TRP0	Level 0	Level 1	Level 2
1	0	4	1	4
1	1	4 + MBO	1 + MBO	4 + MBO

To send additional payload with the reference message in level 1 a DLC > 1 must be configured, for level 0 and level 2 a DLC > 4 is required. In addition, the transmission request pending bit FDCAN_TXBRP.TRP0 of message buffer 0 must be set (see *Table 707*). In case bit FDCAN_TXBRP.TRP0 is not set when a reference message is started, the reference message is transmitted with the data bytes supplied by the FSE only.

For acceptance filtering of reference messages the reference identifier FDCAN_TTRMC.RID is used.

Message reception

Message reception is done via the two Rx FIFOs in the same way as for event-driven CAN communication (see *Rx handler*).

The message status count MSC is part of the corresponding trigger memory element and must be initialized to 0 during configuration. It is updated while the TTCAN is in synchronization states In_Gap or In_Schedule. The update happens at the message Rx_Trigger. At this point in time it is checked at which acceptance filter element the latest message received in this basic cycle had matched. The matching filter number is stored as the acceptance filter result. If this is the same the filter number as defined in this trigger memory element, the MSC is decremented by one. If the acceptance filter result is not the same filter number as defined for this filter element, or if the acceptance filter result is cleared, the MSC is incremented by one. At each Rx_Trigger and at each start of cycle, the last acceptance filter result is cleared.

The time mark of an Rx_Trigger must be set to a value where it is ensured that reception and acceptance filtering for the targeted message has been completed. This must take into consideration the RAM access time and the order of the filter list. It is recommended, that filters which are used for Rx_Triggers are placed at the beginning of the filter list. It is not recommended to use an Rx_Trigger for the reference message.

Message transmission

For time-triggered message transmission the TTCAN supplies 32 dedicated Tx buffers (see *Transmit pause*). A Tx FIFO or Tx queue is not available when the FDCAN is configured for time-triggered operation (FDCAN_TTOCF.OM = 01 or 10).

Each Tx_Trigger in the trigger memory points to a particular Tx buffer containing a specific message. There may be more than one Tx_Trigger for a given Tx buffer if that Tx buffer contains a message that is to be transmitted more than once in a basic cycle or matrix cycle.

The application program must update the data regularly and on time, synchronized to the cycle time. The user is responsible that no partially updated messages are transmitted. To assure this the user must proceed in the following way:

Tx_Trigger_Single / Tx_Trigger_Merged / Tx_Trigger_Arbitration

- Check whether the previous transmission has been completed by reading FDCAN_TXBTO
- Update the Tx buffer configuration and/or payload
- Issue an add request to set the Tx buffer request pending bit

Tx_Trigger_Continous

- Issue a cancellation request to reset the Tx buffer request pending bit
- Check whether the cancellation has finished by reading FDCAN_TXBCF
- Update Tx buffer configuration and/or payload
- Issue an add request to set the Tx buffer request pending bit

The message MSC stored with the corresponding Tx_Trigger provides information on the success of the transmission.

The MSC is incremented by one when the transmission cannot be started because the CAN bus was not idle within the corresponding transmit enable window or when the message was started and cannot be completed successfully. The MSC is decremented by one when the message was transmitted successfully or when the message may have been started within its transmit enable window but was not started because transmission was disabled (TTCAN in error level S2 or user has disabled this particular message).

The Tx buffers may be managed dynamically, that is, several messages with different identifiers may share the same Tx buffer element. In this case the user must ensure that no transmission request is pending for the Tx buffer element to be reconfigured by checking FDCAN_TXBRP.

If a Tx buffer with pending transmission request must be updated, the user must first issue a cancellation request and check whether the cancellation has been completed by reading FDCAN_TXBCF before it starts updating.

The Tx handler transfers a message from the message RAM to its intermediate output buffer at the trigger element, which becomes active immediately before the Tx_Trigger element (which defines the beginning of the transmit window). During and after the transfer time the transmit message may not be updated and its FDCAN_TXBRP bit may not be changed. To control this transfer time, an additional trigger element may be placed before the Tx_Trigger. This may be example of a Time_Base_Trigger which need not cause any other action. The difference in time marks between the Tx_Trigger and the preceding trigger must be large enough to guarantee that the Tx handler can read four words from the message RAM even at high RAM access load from other modules.

Transmission in exclusive time windows

A transmission is started time-triggered when the cycle time reaches the time mark of a Tx_Trigger_Single or Tx_Trigger_Continuous. There is no arbitration on the bus with messages from other nodes. The MSC is updated according the result of the transmission attempt. After successful transmission started by a Tx_Trigger_Single the respective Tx buffer request pending bit is reset. After successful transmission started by a Tx_Trigger_Continuous the respective Tx buffer request pending remains set. When the transmission was not successful due to disturbances, it is repeated next time (one of) its Tx_Trigger(s) become(s) active.

Transmission in arbitrating time windows

A transmission is started time-triggered when the cycle time reaches the time mark of a Tx_Trigger_Arbitration. Several nodes may start to transmit at the same time. In this case the message must arbitrate with the messages from other nodes. The MSC is not updated. When the transmission was not successful (lost arbitration or disturbance), it is repeated next time (one of) its Tx_Trigger(s) become(s) active.

Transmission in merged arbitrating time windows

The purpose of a merged arbitrating time window is, to enable multiple nodes to send a limited number of frames which are transmitted in immediate sequence, the order given by CAN arbitration. It is not intended for burst transmission by a node. Since the node does not have exclusive access within this time window, it may happen that not all requested transmissions are successful.

Messages which have lost arbitration or were disturbed by an error, may be re-transmitted inside the same merged arbitrating time window. The re-transmission is not started if the corresponding transmission request pending flag is reset by a successful Tx cancellation.

In single transmit windows, the Tx handler transmits the message indicated by the message number of the trigger element. In merged arbitrating time windows, it can handle up to three message numbers from the trigger list. Their transmission is attempted in the sequence defined by the trigger list. If the time mark of a fourth message is reached before the first is transmitted (or canceled by the user), the fourth request is ignored.

The transmission inside a merged arbitrating time window is not time-triggered. The transmission of a message may start before its time mark, or after the time mark if the bus was not idle.

The messages transmitted by a specific node inside a merged arbitrating time window are started in the order of their Tx_Triggers, so a message with low CAN priority may prevent the successful transmission of a following message with higher priority, if their is compelling bus traffic. This must be considered for the configuration of the trigger list. Time_Base_Triggers may be placed between consecutive Tx_Triggers to define the time until the data of the corresponding Tx buffer needs to be updated.

71.4.16 TTCAN interrupt and error handling

The TT interrupt register FDCAN_TTIR consists of four segments. Each interrupt can be enabled separately by the corresponding bit in the TT interrupt enable register FDCAN_TTIE. The flags remain set until the user clears them. A flag is cleared by writing a 1 to the corresponding bit position.

The first segment consists of flags CER, AW, WT, and IWT. Each flag indicates a fatal error condition where the CAN communication is stopped. With the exception of IWT, these error conditions require a re-configuration of the FDCAN module before the communication can be restarted.

The second segment consists of flags ELC, SE1, SE2, TXO, TXU, and GTE. Each flag indicates an error condition where the CAN communication is disturbed. If they are caused by a transient failure, for example by disturbances on the CAN bus, they are handled by the FDCAN protocol failure handling and do not require intervention by the application program.

The third segment consists of flags GTD, GTW, SWE, TTMI, and RTMI. The first two flags are controlled by global time events (levels 0, 2 only) that require a reaction by the application program. With a stop watch event triggered by a rising edge on pin fdcan_swt

internal time values are captured. The trigger time mark interrupt notifies the application that a specific Time_Base_Trigger is reached. The register time mark interrupt signals that the time referenced by FDCAN_TTOCN.TMC (cycle, local, or global) is equal to the time mark FDCAN_TTTMK.TM. It can also be used to finish a gap.

The fourth segment consists of flags SOG, CSM, SMC, and SBC. These flags provide a means to synchronize the application program to the communication schedule.

71.4.17 Level 0

TTCAN level 0 is not part of ISO11898-4. This operation mode makes the hardware, that in TTCAN level 2 maintains the calibrated global time base, also available for event-driven CAN according to ISO11898-1.

Level 0 operation is configured via FDCAN_TTOCF.OM = 11. In this mode the FDCAN operates in event driven CAN communication, there is no fixed schedule, the configuration of FDCAN_TTOCF.GEN is ignored. External event-synchronized operation is not available in level 0. A synchronized time base is maintained by the transmission of reference messages.

In level 0 the trigger memory is not active and therefore needs not to be configured. The time mark interrupt flag (FDCAN_TTIR.TTMI) is set when the cycle time has reached FDCAN_TTOCF.IRTO ´ 0x200, it reminds the user to set a transmission request for message buffer 0. The Watch_Trigger interrupt flag (FDCAN_TTIR.WT) is set when the cycle time has reached 0xFF00. These values were chosen to have enough margin for a stable clock calibration. There are no further TT-error-checks.

Register time mark interrupts (FDCAN_TTIR.RTMI) are also possible.

The reference message is configured as for level 2 operation. Received reference messages are recognized by the identifier configured in register FDCAN_TTRMC. For the transmission of reference messages only message buffer 0 may be used. The node transmits reference messages any time the user sets a transmission request for message buffer 0, there is no reference trigger offset.

Level 0 operation is configured via:

- FDCAN_TTRMC
- FDCAN_TTOCF except EVTP, AWL, GEN
- FDCAN_TTMLM except ENTT, TXEW
- FDCAN_TURCF

Level 0 operation is controlled via:

- FDCAN_TTOCN except NIG, TMG, FGP, GCS, TTMIE
- FDCAN_TTGTP
- FDCAN_TTTMK
- FDCAN_TTIR excluding bits CER, AW, IWT SE2, SE1, TXO, TXU, SOG (no function)
- FDCAN_TTIR the following bits have changed function
  - TTMI not defined by trigger memory - activated at cycle time FDCAN_TTOCF.IRTO 0x200
  - WT not defined by trigger memory - activated at cycle time 0xFF00

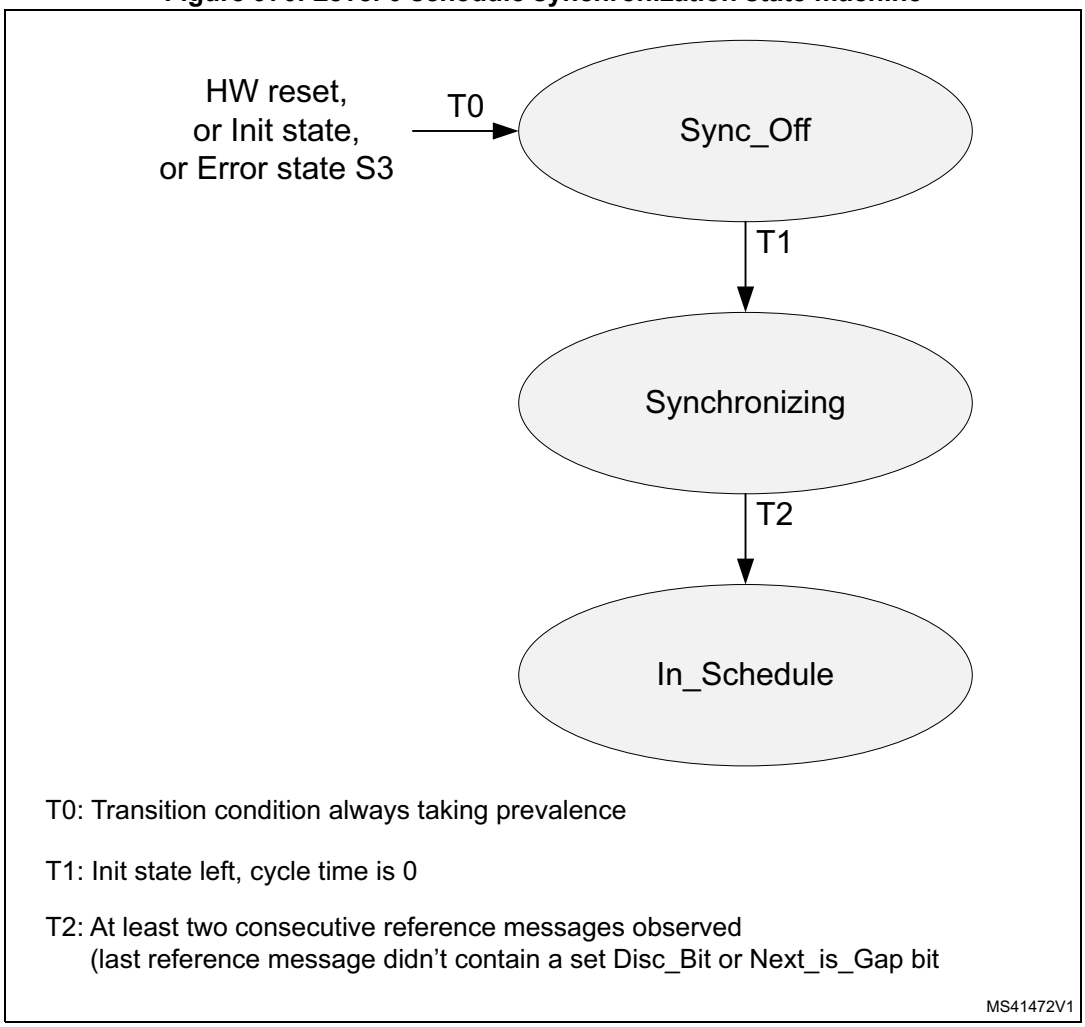
Level 0 operation is signaled via:

- TTOST excluding bits AWE, WFE, GSI, GFI, RTO (no function)

Synchronizing

Figure 970 describes the states and the state transitions in TTCAN level 0 operation. level 0 has no In_Gap state.

Figure 970. Level 0 schedule synchronization state machine



Handling of error levels

During level 0 operation only the following error conditions may occur:

- Watch_Trigger_Reached (S3), reached cycle time 0xFF00
- CAN_Bus_Off (S3)

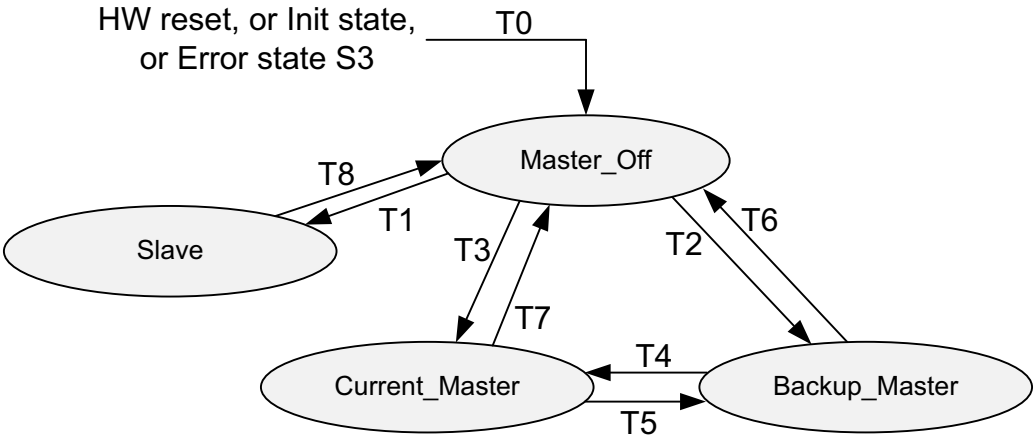
Since no S1 and S2 errors are possible, the error level can only switch between S0 (No error) and S3 (Severe error). In TTCAN level 0 an S3 error is handled differently. When error level S3 is reached, both FDCAN_TTOST.SYS and FDCAN_TTOST.MS are reset, and interrupt flags FDCAN_TTIR.GTE and FDCAN_TTIR.GTD are set.

When error level S3 (FDCAN_TTOST.EL = 11) is entered, bus monitoring mode is (contrary to TTCAN level 1 and level 2) not entered. S3 error level is left automatically after transmission (time master) or reception (time slave) of the next reference message.

Master slave relation

Figure 971 describes the master slave relation in TTCAN level 0. In case of an S3 error the FDCAN returns to state Master_Off.

Figure 971. Level 0 master to slave relation



T0: Transition condition, always taking prevalence

T1: Reference message observed when not potential time master

T2: Reference message observed with master priority different from own master priority, Error state ≠ S3

T3: Reference message observed with master priority equal to own master priority, Error state ≠ S3

T4: Reference message observed with own master priority

T5: Reference message observed with master priority higher than own master priority

T6: Error state S3

T7: Error state S3

T8: Error state S3

MS41473V1

71.4.18 Synchronization to external time schedule

This feature can be used to synchronize the phase of the FDCAN schedule to an external schedule (for example that of a second TTCAN network or FlexRay network). It is applicable only when the FDCAN is current time master (FDCAN_TTOST.MS = 11).

External synchronization is controlled by event trigger input pin fdcan_evt. If bit FDCAN_TTOCN.ESCN is set, a rising edge at event trigger pin fdcan_evt the FDCAN compares its actual cycle time with the target phase value configured by FDCAN_TTGTP.CTP.

Before setting FDCAN_TTOCN.ESCN the user must adapt the phases of the two time schedules for example by using the FDCAN gap control (see *Section 71.4.11*). When the user sets FDCAN_TTOCN.ESCN, FDCAN_TTOSTSPL is set.

If the difference between the cycle time and the target phase value FDCAN_TTGTP.CTP at the rising edge at event trigger pin fdcan_evt is greater than 9 NTU, the phase lock bit FDCAN_TTOST.SPL is reset, and interrupt flag FDCAN_TTIR.CSM is set.

FDCAN_TTOST.SPL is also reset (and FDCAN_TTIR.CSM is set), when another node becomes time master.

If both FDCAN_TTOST.SPL and FDCAN_TTOCN.ESCN are set, and if the difference between the cycle time and the target phase value FDCAN_TTGTP.CTP at the rising edge at event trigger pin fdcan_evt is lower or equal to nine NTU, the phase lock bit FDCAN_TTOST.SPL remains set, and the measured difference is used as the reference trigger offset value to adjust the phase at the next transmitted reference message.

Note: The rising edge detection at event trigger pin fdcan_evt is enabled with the start of each basic cycle. The first rising edge triggers the compare of the actual cycle time with FDCAN_*TTGTP.CTP. All further edges until the beginning of the next basic cycle are ignored.*

71.4.19 FDCAN Rx buffer and FIFO element

Up to 64 Rx buffers and two Rx FIFOs can be configured in the message RAM. Each Rx FIFO section can be configured to store up to 64 received messages. The structure of a Rx buffer / FIFO element is shown in *Table 708*, the description is provided in *Table 709*.

Table 708. Rx buffer and FIFO element

Bit	31			24	23			16	15	8	7	0
R0	ESI	XTD	RTR	ID[28:0]								
R1	ANMF	FIDX[6:0]			Res.	FDF	BRS	DLC[3:0]	RXTS[15:0]			
R2	DB3[7:0]				DB2[7:0]				DB1[7:0]		DB0[7:0]	
R3	DB7[7:0]				DB6[7:0]				DB5[7:0]		DB4[7:0]	
⋮	⋮				⋮				⋮			
Rn	DBm[7:0]				DBm-1[7:0]				DBm-2[7:0]		DBm-3[7:0]	

The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via register FDCAN_RXESC.

Table 709. Rx buffer and FIFO element description

Field	Description
R0 bit 31 ESI	Error state indicator 0: Transmitting node is error active 1: Transmitting node is error passive
R0 bit 30 XTD	Extended identifier Signals to the user whether the received frame has a standard or extended identifier. 0: 11-bit standard identifier 1: 29-bit extended identifier
R0 bit 29 RTR	Remote transmission request Signals to the user whether the received frame is a data frame or a remote frame. 0: Received frame is a data frame 1: Received frame is a remote frame

Table 709. Rx buffer and FIFO element description (continued)

Field	Description
R0 bits 28:0 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier is stored into ID[28:18].
R1 bit 31 ANMF	Accepted non-matching frame Acceptance of non-matching frames may be enabled via FDCAN_GFC.ANFS and FDCAN_GFC.ANFE. 0: Received frame matching filter index FIDX 1: Received frame did not match any Rx filter element
R1 bits 30:24 FIDX[6:0]	Filter index 0-127 = index of matching Rx acceptance filter element (invalid if ANMF = 1). Range is 0 to FDCAN_SIDFC.LSS. - 1 or FDCAN_XIDFC.LSE. - 1.
R1 bit 21 FDF	FD Format 0: Standard frame format 1: FDCAN frame format (new DLC-coding and CRC)
R1 bit 20 BRS	Bitrate Switch 0: Frame received without bitrate switching 1: Frame received with bitrate switching
R1 bits 19:16 DLC[3:0]	Data length code 0-8: Classic CAN + CAN FD: received frame has 0-8 data bytes 9-15: Classic CAN: received frame has 8 data bytes 9-15: CAN FD: received frame has 12/16/20/24/32/48/64 data bytes
R1 bits 15:0 RXTS[15:0]	Rx timestamp Timestamp counter value captured on start of frame reception. Resolution depending on configuration of the timestamp counter prescaler FDCAN_TSCC.TCP.
R2 bits 31:24 DB3[7:0]	Data Byte 3
R2 bits 23:16 DB2[7:0]	Data Byte 2
R2 bits 15:8 DB1[7:0]	Data Byte 1
R2 bits 7:0 DB0[7:0]	Data Byte 0
R3 bits 31:24 DB7[7:0]	Data Byte 7
R3 bits 23:16 DB6[7:0]	Data Byte 6
R3 bits 15:8 DB5[7:0]	Data Byte 5
R3 bits 7:0 DB4[7:0]	Data Byte 4
⋮	⋮

Table 709. Rx buffer and FIFO element description (continued)

Field	Description
Rn bits 31:24 DBm[7:0]	Data Byte m
Rn bits 23:16 DBm-1[7:0]	Data Byte m-1
Rn bits 15:8 DBm-2[7:0]	Data Byte m-2
Rn bits 7:0 DBm-3[7:0]	Data Byte m-3

71.4.20 FDCAN Tx buffer element

The Tx buffers section can be configured to hold dedicated Tx buffers as well as a Tx FIFO / Tx queue. In case that the Tx buffers section is shared by dedicated Tx buffers and a Tx FIFO / Tx queue, the dedicated Tx buffers start at the beginning of the Tx buffers section followed by the buffers assigned to the Tx FIFO or Tx queue. The Tx handler distinguishes between dedicated Tx buffers and Tx FIFO / Tx queue by evaluating the Tx buffer configuration FDCAN_TXBC.TFQS and FDCAN_TXBC.NDTB. The element size can be configured for storage of CAN FD messages with up to 64 bytes data field via register FDCAN_TXESC.

Table 710. Tx buffer and FIFO element

Bit	31			24	23					16	15	8	7	0
T0	ESI	XTD	RTR	ID[28:0]										
T1	MM[7:0]				EFC	Res.	FDF	BPS	DLC[3:0]		Reserved			
T2	DB3[7:0]				DB2[7:0]						DB1[7:0]		DB0[7:0]	
T3	DB7[7:0]				DB6[7:0]						DB5[7:0]		DB4[7:0]	
⋮	⋮				⋮						⋮			
Tn	DBm[7:0]				DBm-1[7:0]						DBm-2[7:0]		DBm-3[7:0]	

Table 711. Tx buffer element description

Field	Description
T0 bit 31 ESI(1)	Error state indicator 0: ESI bit in CAN FD format depends only on error passive flag 1: ESI bit in CAN FD format transmitted recessive
T0 bit 30 XTD	Extended identifier 0: 11-bit standard identifier 1: 29-bit extended identifier
T0 bit 29 RTR(2)	Remote transmission request 0: Transmit data frame 1: Transmit remote frame

Table 711. Tx buffer element description (continued)

Field	Description
T0 bits 28:0 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier must be written to ID[28:18].
T1 bits 31:24 MM[7:0]	Message marker Written by CPU during Tx buffer configuration. Copied into Tx event FIFO element for identification of Tx message status.
T1 bit 23 EFC	Event FIFO control 0: Don’t store Tx events 1: Store Tx events
T1 bit 21 FDF	FD Format 0: Frame transmitted in classic CAN format 1: Frame transmitted in CAN FD format
T1 bit 20 BRS(3)	Bitrate switching 0: CAN FD frames transmitted without bitrate switching 1: CAN FD frames transmitted with bitrate switching
T1 bits 19:16 DLC[3:0]	Data length code 0 - 8: Classic CAN + CAN FD: received frame has 0-8 data bytes 9-15: Classic CAN: received frame has 8 data bytes 9 - 15: CAN FD: received frame has 12/16/20/24/32/48/64 data bytes
T2 bits 31:24 DB3[7:0]	Data Byte 3
T2 bits 23:16 DB2[7:0]	Data Byte 2
T2 bits 15:8 DB1[7:0]	Data Byte 1
T2 bits 7:0 DB0[7:0]	Data Byte 0
T3 bits 31:24 DB7[7:0]	Data Byte 7
T3 bits 23:16 DB6[7:0]	Data Byte 6
T3 bits 15:8 DB5[7:0]	Data Byte 5
T3 bits 7:0 DB4[7:0]	Data Byte 4
⋮	⋮
Tn bits 31:24 DBm[7:0]	Data Byte m
Tn bits 23:16 DBm-1[7:0]	Data Byte m-1

Table 711. Tx buffer element description (continued)

Field	Description
Tn bits 15:8 DBm-2[7:0]	Data Byte m-2
Tn bits 7:0 DBm-3[7:0]	Data Byte m-3

1. The ESI bit of the transmit buffer is OR-ed with the error passive flag to decide the value of the ESI bit in the transmitted FD frame. As required by the CAN FD protocol specification, an error active node may optionally transmit the ESI bit recessive, but an error passive node always transmit the ESI bit recessive.
2. When RTR = 1, the FDCAN transmits a remote frame according to ISO11898-1, even if FDCAN_CCCR.FDOE enables the transmission in CAN FD format.
3. Bits ESI, FDF, and BRS are only evaluated when CAN FD operation is enabled FDCAN_CCCR.FDOE = 1. Bit BRS is only evaluated when in addition FDCAN_CCCR.BRSE = 1.

71.4.21 FDCAN Tx event FIFO element

Each element stores information about transmitted messages. By reading the Tx event FIFO the user gets this information in the order the messages were transmitted. Status information about the Tx event FIFO can be obtained from register FDCAN_TXEFS.

Table 712. Tx Event FIFO element

Bit	31			24	23				16	15	8	7	0
E0	ESI	XTD	RTR	ID[28:0]									
E1	MM[7:0]				ET[1:0]	EDL	BRS	DLC[3:0]		TXTS[15:0]			

Table 713. Tx Event FIFO element description

Field	Description
E0 bit 31 ESI	Error state indicator – 0: Transmitting node is error active – 1: Transmitting node is error passive
E0 bit 30 XTD	Extended Identifier – 0: 11-bit standard identifier – 1: 29-bit extended identifier
E0 bit 29 RTR	Remote transmission request – 0: Transmit data frame – 1: Transmit remote frame
E0 bits 28:0 ID[28:0]	Identifier Standard or extended identifier depending on bit XTD. A standard identifier must be written to ID[28:18].
E1 bits 31:24 MM[7:0]	Message marker Copied from Tx buffer into Tx event FIFO element for identification of Tx message status.

Table 713. Tx Event FIFO element description (continued)

Field	Description
E1 bits 23:22 EFC	Event type – 00: Reserved – 01: Tx event – 10: Transmission in spite of cancellation (always set for transmissions in DAR mode) – 11: Reserved
E1 bit 21 EDL	Extended data length – 0: Standard frame format – 1: FDCAN frame format (new DLC-coding and CRC)
E1 bit 20 BRS	Bitrate switching – 0: Frame transmitted without bitrate switching – 1: Frame transmitted with bitrate switching
T1 bits 19:16 DLC[3:0]	Data length code – 0 - 8: Frame with 0-8 data bytes transmitted – 9 - 15: Frame with eight data bytes transmitted
E1 bits 15:0 TXTS[15:0]	Tx timestamp Timestamp counter value captured on start of frame transmission. Resolution depending on configuration of the timestamp counter prescaler FDCAN_TSCC.TCP.

71.4.22 FDCAN standard message ID filter element

Up to 128 filter elements can be configured for 11-bit standard IDs. When accessing a standard message ID filter element, its address is the filter list standard start address FDCAN_SIDFC.FLSSA plus the index of the filter element (0 … 127).

Table 714. Standard message ID filter element

Bit	31		24	23	16	15	8	7	0
S0	SFT[1:0]	SFEC[2:0]		SFID1[10:0]		Res.		SFID2[10:0]	

Table 715. Standard message ID filter element field description

Field		Description
Bit 31:30 SFT[1:0](1)		Standard filter type – 00: Range filter from SFID1 to SFID2 – 01: Dual ID filter for SFID1 or SFID2 – 10: Classic filter: SFID1 = filter, SFID2 = mask – 11: Filter element disabled
Bit 29:27 SFEC[2:0]		Standard filter element configuration All enabled filter elements are used for acceptance filtering of standard frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If SFEC = “100”, “101”, or “110” a match sets interrupt flag FDCAN_IR.HPM and, if enabled, an interrupt is generated. In this case register FDCAN_HPMS is updated with the status of the priority match. – 000: Disable filter element – 001: Store in Rx FIFO 0 if filter matches – 010: Store in Rx FIFO 1 if filter matches – 011: Reject ID if filter matches – 100: Set priority if filter matches – 101: Set priority and store in FIFO 0 if filter matches – 110: Set priority and store in FIFO 1 if filter matches – 111: Store into Rx buffer or as debug message, configuration of FDCAN_SFT[1:0] ignored
Bits 26:16 SFID1[10:0]		Standard filter ID 1 First ID of standard ID filter element. When filtering for Rx buffers or for debug messages this field defines the ID of a standard message to be stored. The received identifiers must match exactly, no masking mechanism is used.
Bits 15:0	SFID2[15:10]	Standard filter ID 2 This bit field has a different meaning depending on the configuration of SFEC: – SFEC = 001 ... 110 Second ID of standard ID filter element – SFEC = 111 Filter for Rx buffers or for debug messages
	SFID2[10:9]	Decides whether the received message is stored into an Rx buffer or treated as message A, B, or C of the debug message sequence. – 00: Store message into an Rx buffer – 01: Debug message A – 10: Debug message B – 11: Debug message C
	SFID2[8:6]	Is used to control the filter event pins at the Extension Interface. A 1 at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one fdcan_pclk period in case the filter matches. SFID2[8] is used by the calibration unit.
	SFID2[5:0]	Defines the offset to the Rx buffer start address FDCAN_RXBC.RBSA for storage of a matching message.

1. With SFT = “11” the filter element is disabled and the acceptance filtering continues (same behavior as with SFEC = “000”).

Note: In case a reserved value is configured, the filter element is considered disabled.

71.4.23 FDCAN extended message ID filter element

Up to 64 filter elements can be configured for 29-bit extended IDs. When accessing an Extended message ID filter element, its address is the filter list extended start address FDCAN_XIDFC.FLESA plus two times the index of the filter element (0 … 63).

Table 716. Extended message ID filter element

Bit	31	30	29	28	0
F0	EFEC[2:0]			EFID1[28:0]	
F1	EFT[1:0]		Reserved	EFID2[28:0]	

Table 717. Extended message ID filter element field description

Field	Description
F0 bits 31:29 EFEC[2:0]	Extended filter element configuration All enabled filter elements are used for acceptance filtering of extended frames. Acceptance filtering stops at the first matching enabled filter element or when the end of the filter list is reached. If EFEC = 100, 101, or 110 a match sets interrupt flag FDCAN_IR.HPM and, if enabled, an interrupt is generated. In this case register FDCAN_HPMS is updated with the status of the priority match. – 000: Disable filter element – 001: Store in Rx FIFO 0 if filter matches – 010: Store in Rx FIFO 1 if filter matches – 011: Reject ID if filter matches – 100: Set priority if filter matches – 101: Set priority and store in FIFO 0 if filter matches – 110: Set priority and store in FIFO 1 if filter matches – 111: Store into Rx buffer, configuration of EFT[1:0] ignored
F0 bits 28:0 EFID1[28:0]	Extended filter ID 1 First ID of extended ID filter element. When filtering for Rx buffers or for debug messages this field defines the ID of an extended message to be stored. The received identifiers must match exactly, only FDCAN_XIDAM masking mechanism.
F1 bits 31:30 EFT[1:0]	Extended filter type – 00: Range filter from EF1ID to EF2ID (EF2ID ≥ EF1ID) – 01: Dual ID filter for EF1ID or EF2ID – 10: Classic filter: EF1ID = filter, EF2ID = mask – 11: Range filter from EF1ID to EF2ID (EF2ID ≥ EF1ID), FDCAN_XIDAM mask not applied

Table 717. Extended message ID filter element field description (continued)

Field		Description
F1 bits 28:0	EFID2[10:0]	Extended filter ID 2 This bit field has a different meaning depending on the configuration of EFEC: – SFEC = 001 ... 110 Second ID of extended ID filter element – SFEC = 111 Filter for Rx buffers or for debug messages
	EFID2[10:9]	Decides whether the received message is stored into an Rx buffer or treated as message A, B, or C of the debug message sequence. – 00: Store message into an Rx buffer – 01: Debug message A – 10: Debug message B – 11: Debug message C
	EFID2[8:6]	Is used to control the filter event pins at the Extension Interface. A 1 at the respective bit position enables generation of a pulse at the related filter event pin with the duration of one fdcan_pclk period in case the filter matches. EFID2[8] interface is used by the calibration unit.
	EFID2[5:0]	Defines the offset to the Rx buffer start address FDCAN_RXBC.RBSA for storage of a matching message.

71.4.24 FDCAN trigger memory element

Up to 64 trigger memory elements can be configured. When accessing a trigger memory element, its address is the trigger memory start address FDCAN_TTTMC.TMSA plus the index of the trigger memory element (0 … 63).

Table 718. Trigger memory element

Bit	31 24	23 16	15 8	7 0
T0	TM[15:0]		Res. CC[6:0]	Res. TMIN TMEX TYPE[3:0]
T1	Res.	FTYPE MNR[6:0]	Res.	MSC[2:0]

Table 719. Trigger memory element description

Field	Description
T0 bits 31:16 TM[15:0]	Time mark Cycle time for which the trigger becomes active.
T0 bit 14:8 CC[6:0]	Cycle code Cycle count for which the trigger is valid. Ignored for trigger types Tx_Ref_Trigger, Tx_Ref_Trigger_Gap, Watch_Trigger, Watch_Trigger_Gap, End_of_List. – 0b000000x valid for all cycles – 0b000001c valid every 2^{nd} cycle at cycle count mod2 = c – 0b00001cc valid every 4^{th} cycle at cycle count mod4 = cc – 0b0001ccc valid every 8^{th} cycle at cycle count mod8 = ccc – 0b001cccc valid every 16^{th} cycle at cycle count mod16 = cccc – 0b01ccccc valid every 32^{nd} cycle at cycle count mod32 = ccccc – 0b1cccccc valid every 64^{th} cycle at cycle count mod64 = cccccc

Table 719. Trigger memory element description (continued)

Field	Description
T0 bit 5 TMIN	Time mark event internal – 0: No action – 1: FDCAN_TTIR.TTMI is set when trigger memory element becomes active
T0 bit 4 TMEX	Time mark event external – 0: No action – 1: Pulse at output fdcan1_tmp for more than one instance and fdcan_tmp if only one instance with the length of one period is generated when the time arc of the trigger memory element becomes active and FDCAN_TTOCN.TTMIE = 1
T0 bit 3:0 TYPE[3:0]	Trigger type – 0000 Tx_Ref_Trigger - valid when not in gap – 0001 Tx_Ref_Trigger_Gap - valid when in gap – 0010 Tx_Trigger_Single - starts a single transmission in an exclusive time window – 0011 Tx_Trigger_Continuous - starts continuous transmission in an exclusive time window – 0100 Tx_Trigger_Arbitration - starts a transmission in an arbitrating time window – 0101 Tx_Trigger_Merged - starts a merged arbitration window – 0110 Watch_Trigger - valid when not in gap – 0111 Watch_Trigger_Gap - valid when in gap – 1000 Rx_Trigger - check for reception – 1001 Time_Base_Trigger - only control TMIN, TMEX – 1010 … 1111=End_of_List - illegal type, causes configuration error
T1 bit 23FTYPE	Filter type – 0: 11-bit standard message ID – 1: 29-bit extended message ID
T1 bit 22:16 MNR[6:0](1)	Message number – Transmission: trigger is valid for configured Tx buffer number. Valid values are 0 to 31. – Reception: trigger is valid for standard/extended message ID filter element number. Valid values are, respectively 0 to 63 and 0 to 127.
T1 bits 2:0 MSC[2:0]	Message status count Counts scheduling errors for periodic messages in exclusive time windows. It has no function for arbitrating messages and in event-driven CAN communication (ISO11898-1). – 0-7= Actual status

1. The trigger memory elements must be written when the FDCAN is in INIT state. Write access to the trigger memory elements outside INIT state is not allowed. There is an exception for TMIN and TMEX when they are defined as part of a trigger memory element of TYPE Tx_Ref_Trigger. In this case they become active at the time mark modified by the actual reference trigger offset (TTOST[RTO]).

71.5 FDCAN registers

71.5.1 FDCAN core release register (FDCAN_CREL)

Address offset: 0x0000

Reset value: 0x3214 1218

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REL[3:0]				STEP[3:0]				SUBSTEP[3:0]				YEAR[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MON[7:0]								DAY[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 **REL[3:0]**: Core release = 3

Bits 27:24 **STEP[3:0]**: Step of core release = 2

Bits 23:20 **SUBSTEP[3:0]**: Sub-step of core release = 1

Bits 19:16 **YEAR[3:0]**: Timestamp year = 4

Bits 15:8 **MON[7:0]**: Timestamp month = 12

Bits 7:0 **DAY[7:0]**: Timestamp day =18

71.5.2 FDCAN Endian register (FDCAN_ENDN)

Address offset: 0x0004

Reset value: 0x8765 4321

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ETV[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETV[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ETV[31:0]**: Endianness test value

The endianness test value is 0x8765 4321.

71.5.3 FDCAN data bit timing and prescaler register (FDCAN_DBTP)

Address offset: 0x000C

Reset value: 0x0000 0A33

This register is dedicated to data bit timing phase and only writable if bits FDCAN_CCCR.CCE and FDCAN_CCCR.INIT are set. The CAN time quantum may be programmed in the range from 1 to 32 FDCAN clock periods. tq = (DBRP + 1) FDCAN clock periods.

DTSEG1 is the sum of Prop_Seg and Phase_Seg1. DTSEG2 is Phase_Seg2. Therefore the length of the bit time is (DTSEG1 + DTSEG2 + 3) tq for programmed values, or (Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2) tq for functional values.

The information processing time (IPT) is zero, meaning the data for the next bit is available at the first clock edge after the sample point.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDC	Res.	Res.	DBRP[4:0]				
								rw			rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DTSEG1[4:0]					DTSEG2[3:0]				DSJW[3:0]			
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TDC**: Transceiver delay compensation
0: Transceiver delay compensation disabled
1: Transceiver delay compensation enabled

Bits 22:21 Reserved, must be kept at reset value.

Bits 20:16 **DBRP[4:0]**: Data bitrate prescaler
The value by which the oscillator frequency is divided to generate the bit time quanta. The bit time is built up from a multiple of these quanta. Valid values for the baudrate prescaler are 0 to 31. The hardware interpreters this value as the programmed value plus 1.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **DTSEG1[4:0]**: Data time segment before sample point
Valid values are 0 to 31. The value used by the hardware is the one programmed, incremented by 1, that is, t_{BS1} = (DTSEG1 + 1) x tq.

Bits 7:4 **DTSEG2[3:0]**: Data time segment after sample point
Valid values are 0 to 15. The value used by the hardware is the one programmed, incremented by 1, that is, t_{BS2} = (DTSEG2 + 1) x tq.

Bits 3:0 **DSJW[3:0]**: Synchronization jump width
Valid values are 0 to 15. The value used by the hardware is the one programmed, incremented by 1, that is, t_{SJW} = (DSJW + 1) x tq.

Note: With a CAN kernel clock of 8 MHz, the reset value of 0x00000A33 configures the FDCAN for a bit rate of 500 kbit/s.

Note: The data phase bit rate must be higher than or equal to the nominal bit rate.

71.5.4 FDCAN test register (FDCAN_TEST)

Write access to this register must be enabled by setting bit FDCAN_CCCR.TEST to 1. All register functions are set to their reset values when bit FDCAN_CCCR.TEST is reset.

Loop back mode and software control of Tx pin FDCANx_TX are hardware test modes. Programming TX differently from 00 may disturb the message transfer on the CAN bus.

Address offset: 0x0010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX	TX[1:0]		LBCK	Res.	Res.	Res.	Res.
								r	rw	rw	rw				

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **RX**: Receive pin

Monitors the actual value of transmit pin FDCANx_RX. This bit is a synchronized version of the FDCANx_RX, it goes back to 1 after reset if connected to a network.

0: The CAN bus is dominant (FDCANx_RX = 0)

1: The CAN bus is recessive (FDCANx_RX = 1)

Bits 6:5 **TX[1:0]**: Control of transmit pin

00: Reset value, FDCANx_TX TX is controlled by the CAN core, updated at the end of the CAN bit time

01: Sample point can be monitored at pin FDCANx_TX

10: Dominant (0) level at pin FDCANx_TX

11: Recessive (1) at pin FDCANx_TX

Bit 4 **LBCK**: Loop back mode

0: Reset value, loop back mode is disabled

1: Loop back mode is enabled (see *Test modes*)

Bits 3:0 Reserved, must be kept at reset value.

71.5.5 FDCAN RAM watchdog register (FDCAN_RWD)

The RAM watchdog monitors the READY output of the message RAM. A message RAM access starts the message RAM watchdog counter with the value configured by the FDCAN_RWD.WDC bits.

The counter is reloaded with FDCAN_RWD.WDC bits when the message RAM signals successful completion by activating its READY output. In case there is no response from the message RAM until the counter has counted down to 0, the counter stops and the interrupt flag FDCAN_IR.WDI bit is set. The RAM watchdog counter is clocked by the fdcan_pclk clock.

Address offset: 0x0014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDV[7:0]								WDC[7:0]							
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **WDV[7:0]**: Watchdog value
Actual message RAM watchdog counter value.

Bits 7:0 **WDC[7:0]**: Watchdog configuration
Start value of the message RAM watchdog counter. With the reset value of 00 the counter is disabled.
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

71.5.6 FDCAN CC control register (FDCAN_CCCR)

Address offset: 0x0018

Reset value: 0x0000 0001

For details about setting and resetting of single bits, see *Software initialization*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NISO	TXP	EFBI	PXHD	Res.	Res.	BRSE	FDOE	TEST	DAR	MON	CSR	CSA	ASM	CCE	INIT
rw	rw	rw	rw			rw	rw	rw	rw	rw	rw	r	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **NISO**: Non ISO operation
If this bit is set, the FDCAN uses the CAN FD frame format as specified by the Bosch CAN FD Specification V1.0.
0: CAN FD frame format according to ISO11898-1
1: CAN FD frame format according to Bosch CAN FD Specification V1.0

Bit 14 **TXP**:
If this bit is set, the FDCAN pauses for two CAN bit times before starting the next transmission after successfully transmitting a frame.
0: Disabled
1: Enabled

Bit 13 **EFBI**: Edge filtering during bus integration
0: Edge filtering disabled
1: Two consecutive dominant tq required to detect an edge for hard synchronization

Bit 12 **PXHD**: Protocol exception handling disable
0: Protocol exception handling enabled
1: Protocol exception handling disabled

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **BRSE**: FDCAN bitrate switching
0: Bitrate switching for transmissions disabled
1: Bitrate switching for transmissions enabled

Bit 8 **FDOE**: FD operation enable
0: FD operation disabled
1: FD operation enabled

Bit 7 **TEST**: Test mode enable
0: Normal operation, register TEST holds reset values
1: Test mode, write access to register TEST enabled

Bit 6 **DAR**: Disable automatic retransmission
0: Automatic retransmission of messages not transmitted successfully enabled
1: Automatic retransmission disabled

Bit 5 **MON**: Bus monitoring mode
Bit MON can be set only by software when both CCE and INIT are set to 1. The bit can be reset by the user at any time.
0: Bus monitoring mode is disabled
1: Bus monitoring mode is enabled

Bit 4 **CSR**: Clock stop request
0: No clock stop is requested
1: Clock stop requested. When clock stop is requested, first INIT and then CSA is set after all pending transfer requests have been completed and the CAN bus reached idle.

Bit 3 **CSA**: Clock stop acknowledge
0: No clock stop acknowledged
1: FDCAN may be set in power down by stopping APB clock and kernel clock

Bit 2 **ASM**: ASM restricted operation mode
The restricted operation mode is intended for applications that adapt themselves to different CAN bitrates. The application tests different bitrates and leaves the restricted operation mode after it has received a valid frame. In the optional restricted operation mode the node is able to transmit and receive data and remote frames and it gives acknowledge to valid frames, but it does not send active error frames or overload frames. In case of an error condition or overload condition, it does not send dominant bits, instead it waits for the occurrence of bus idle condition to resynchronize itself to the CAN communication. The error counters are not incremented. Bit ASM can be set only by software when both CCE and INIT are set to 1. The bit can be reset by the software at any time. This bit is set automatically set to 1 when the Tx handler was not able to read data from the message RAM in time. If the FDCAN is connected to a clock calibration on CAN unit, ASM bit is set by hardware as long as the calibration is not completed.
0: Normal CAN operation
1: Restricted operation mode active

Bit 1 **CCE**: Configuration change enable
0: The CPU has no write access to the protected configuration registers
1: The CPU has write access to the protected configuration registers (while FDCAN_CCCR.INIT = 1 CCE bit is automatically cleared when INIT bit is cleared)

Bit 0 **INIT**: Initialization
0: Normal operation
1: Initialization is started (while FDCAN_CCCR.INIT = 1 CCE bit is automatically cleared when INIT bit is cleared)

Note: Due to the synchronization mechanism between the two clock domains, there may be a delay until the value written to INIT can be read back. Therefore the programmer must ensure that the previous value written to INIT has been accepted by reading INIT before setting INIT to a new value.

71.5.7 FDCAN nominal bit timing and prescaler register (FDCAN_NBTP)

Address offset: 0x001C

Reset value: 0x0600 0A03

This register is dedicated to the nominal bit timing used during the arbitration phase, and is only writable if bits FDCAN_CCCR.CCE and FDCAN_CCCR.INIT are set. The CAN bit time may be programed in the range of 4 to 81 tq. The CAN time quantum may be programmed in the range of [1 … 1024] FDCAN kernel clock periods.

tq = (BRP + 1) FDCAN clock period fdcan_tq_ck

NTSEG1 is the sum of Prop_Seg and Phase_Seg1. NTSEG2 is Phase_Seg2. Therefore the length of the bit time is (programmed values) [NTSEG1 + NTSEG2 + 3] tq or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] tq.

The information processing time (IPT) is zero, meaning the data for the next bit is available at the first clock edge after the sample point.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NSJW[6:0]							NBRP[8:0]								
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NTSEG1[7:0]								Res.	NTSEG2[6:0]						
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:25 **NSJW[6:0]**: Nominal (re)synchronization jump width

Must be smaller than NTSEG2, valid values are 0 to 127. The value used by the hardware is the one programmed, incremented by 1, that is, t_{SJW} = (NSJW + 1) x tq.

These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 24:16 **NBRP[8:0]**: Bitrate prescaler

Value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values are 0 to 511. The value used by the hardware is the one programmed, incremented by 1.

These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:8 **NTSEG1[7:0]**: Nominal time segment before sample point

Valid values are 0 to 255. The value used by the hardware is the one programmed, incremented by 1 (t_{BS1} = (NTSEG1 + 1) x tq).

These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **NTSEG2[6:0]**: Nominal time segment after sample point

Valid values are 0 to 127. The value used by the hardware is the one programmed, incremented by 1 (t_{BS2} = (NTSEG2 + 1) x tq).

Note: With a CAN kernel clock of 8 MHz, the reset value of 0x06000A03 configures the FDCAN for a bit rate of 500 kbit/s.

71.5.8 FDCAN timestamp counter configuration register (FDCAN_TSCC)

Address offset: 0x0020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCP[3:0]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSS[1:0]	
														rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **TCP[3:0]**: Timestamp counter prescaler

Configures the timestamp and timeout counters time unit in multiples of CAN bit times [1 … 16]. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

In CAN FD mode the internal timestamp counter TCP does not provide a constant time base due to the different CAN bit times between arbitration phase and data phase. Thus CAN FD requires an external counter for timestamp generation (TSS = 10).

These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:2 Reserved, must be kept at reset value.

Bits 1:0 **TSS[1:0]**: Timestamp select

00: Timestamp counter value always 0x0000

01: Timestamp counter value incremented according to TCP

10: External timestamp counter from TIM3 value used (tim3_cnt[0:15])

11: Same as 00.

These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

71.5.9 FDCAN timestamp counter value register (FDCAN_TSCV)

Address offset: 0x0024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSC[15:0]															
rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TSC[15:0]**: Timestamp counter

The internal/external timestamp counter value is captured on start of frame (both Rx and Tx). When FDCAN_TSCC.TSS = 01, the timestamp counter is incremented in multiples of CAN bit times [1 … 16] depending on the configuration of FDCAN_TSCC.TCP. A wrap around sets interrupt flag FDCAN_IR.TSW. Write access resets the counter to 0. When FDCAN_TSCC.TSS = 10, TSC reflects the external timestamp counter value. A write access has no impact.

Note: A "wrap around" is a change of the timestamp counter value from non-0 to 0 not caused by write access to FDCAN_TSCV.

71.5.10 FDCAN timeout counter configuration register (FDCAN_TOCC)

Address offset: 0x0028

Reset value: 0xFFFF 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TOP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOS[1:0]		ETOC
													rw	rw	rw

Bits 31:16 **TOP[15:0]**: Timeout period

Start value of the timeout counter (down-counter). Configures the timeout period.

These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:3 Reserved, must be kept at reset value.

Bits 2:1 **TOS[1:0]**: Timeout select

When operating in Continuous mode, a write to FDCAN_TOCV presets the counter to the value configured by FDCAN_TOCC.TOP and continues down-counting. When the timeout counter is controlled by one of the FIFOs, an empty FIFO presets the counter to the value configured by FDCAN_TOCC.TOP. Down-counting is started when the first FIFO element is stored.

00: Continuous operation

01: Timeout controlled by Tx event FIFO

10: Timeout controlled by Rx FIFO 0

11: Timeout controlled by Rx FIFO 1

These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 0 **ETOC**: Enable timeout counter

0: Timeout counter disabled

1: Timeout counter enabled

This is a write-protected bit, write access is possible only when the bit 1 (CCE) and bit 0 (INIT) of FDCAN_CCCR register are set to 1.

For more details see *Timeout counter*.

71.5.11 FDCAN timeout counter value register (FDCAN_TOCV)

Address offset: 0x002C

Reset value: 0x0000 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TOC[15:0]															
rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w	rc_w

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **TOC[15:0]**: Timeout counter

The timeout counter is decremented in multiples of CAN bit times [1 … 16] depending on the configuration of FDCAN_TSCC.TCP. When decremented to 0, interrupt flag FDCAN_IR.TOO is set and the timeout counter is stopped. Start and reset/restart conditions are configured via FDCAN_TOCC.TOS.

71.5.12 FDCAN error counter register (FDCAN_ECR)

Address offset: 0x0040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CEL[7:0]							
								rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RP	REC[6:0]							TEC[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **CEL[7:0]**: CAN error logging

The counter is incremented each time when a CAN protocol error causes the transmit error counter or the receive error counter to be incremented. It is reset by read access to CEL. The counter stops at 0xFF; the next increment of TEC or REC sets interrupt flag FDCAN_IR.ELO.

Bit 15 **RP**: Receive error passive

0: The receive error counter is below the error passive level of 128

1: The receive error counter has reached the error passive level of 128

Bits 14:8 **REC[6:0]:** Receive error counter

Actual state of the receive error counter, values between 0 and 127.

Bits 7:0 **TEC[7:0]**: Transmit error counter

Actual state of the transmit error counter, values between 0 and 255.

When FDCAN_CCCR.ASM is set, the CAN protocol controller does not increment TEC and REC when a CAN protocol error is detected, but CEL is still incremented.

71.5.13 FDCAN protocol status register (FDCAN_PSR)

Address offset: 0x0044

Reset value: 0x0000 0707

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDCV[6:0]						
									r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	PXE	REDL	RBRS	RESI	DLEC[2:0]			BO	EW	EP	ACT[1:0]		LEC[2:0]		
	rc_r	rc_r	rc_r	rc_r	rs_r	rs_r	rs_r	r	r	r	r	r	rs_r	rs_r	rs_r

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **TDCV[6:0]**: Transmitter delay compensation value

Position of the secondary sample point, defined by the sum of the measured delay from FDCAN_TX to FDCAN_RX and FDCAN_TDCR.TDCO. The SSP position is, in the data phase, the number of minimum time quanta (mtq) between the start of the transmitted bit and the secondary sample point. Valid values are 0 to 127 mtq.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **PXE**: Protocol exception event

0: No protocol exception event occurred since last read access
1: Protocol exception event occurred

Bit 13 **REDL**: Received FDCAN message

This bit is set independent of acceptance filtering.
0: Since this bit was reset by the CPU, no FDCAN message has been received
1: Message in FDCAN format with EDL flag set has been received
Access type is RX: reset on read.

Bit 12 **RBRS:** BRS flag of last received FDCAN message

This bit is set together with REDL, independent of acceptance filtering.
0: Last received FDCAN message did not have its BRS flag set
1: Last received FDCAN message had its BRS flag set
Access type is RX: reset on read.

Bit 11 **RESI**: ESI flag of last received FDCAN message

This bit is set together with REDL, independent of acceptance filtering.
0: Last received FDCAN message did not ha ve its ESI flag set
1: Last received FDCAN message had its ESI flag set
Access type is RX: reset on read.

Bits 10:8 **DLEC[2:0]**: Data last error code

Type of last error that occurred in the data phase of an FDCAN format frame with its BRS flag set. Coding is the same as for LEC. This field is cleared to 0 when an FDCAN format frame with its BRS flag set has been transferred (reception or transmission) without error.
Access type is RS: set on read.

Bit 7 **BO**: Bus_Off status

0: The FDCAN is not Bus_Off
1: The FDCAN is in Bus_Off state

Bit 6 **EW**: Warning status
0: Both error counters are below the Error_Warning limit of 96
1: At least one of error counter has reached the Error_Warning limit of 96

Bit 5 **EP**: Error passive
0: The FDCAN is in the Error_Active state. It normally takes part in bus communication and sends an active error flag when an error has been detected
1: The FDCAN is in the Error_Passive state

Bits 4:3 **ACT[1:0]**: Activity
Monitors the module CAN communication state.
00: Synchronizing: node is synchronizing on CAN communication
01: Idle: node is neither receiver nor transmitter
10: Receiver: node is operating as receiver
11: Transmitter: node is operating as transmitter

Bits 2:0 **LEC[2:0]:** Last error code
The LEC indicates the type of the last error to occur on the CAN bus. This field is cleared to 0 when a message has been transferred (reception or transmission) without error.
000: No error: No error occurred since LEC has been reset by successful reception or transmission.
001: Stuff error: More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed.
010: Form error: A fixed format part of a received frame has the wrong format.
011: AckError: The message transmitted by the FDCAN was not acknowledged by another node.
100: Bit1Error: During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value 1), but the monitored bus value was dominant.
101: Bit0Error: During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (data or identifier bit logical value 0), but the monitored bus value was recessive. During Bus_Off recovery this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the Bus_Off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed).
110: CRCError: The CRC check sum of a received message was incorrect. The CRC of an incoming message does not match with the CRC calculated from the received data.
111: NoChange: Any read access to the protocol status register re-initializes the LEC to ‘7. When the LEC shows the value 7, no CAN bus event was detected since the last CPU read access to the protocol status register.
Access type is RS: set on read.

Note: When a frame in FDCAN format has reached the data phase with BRS flag set, the next CAN event (error or valid frame) is shown in FLEC instead of LEC. An error in a fixed stuff bit of an FDCAN CRC sequence is shown as a Form error, not Stuff error

Note: The Bus_Off recovery sequence (see CAN Specification Rev. 2.0 or ISO11898-1) cannot be shortened by setting or resetting FDCAN_CCCR.INIT. If the device goes Bus_Off, it sets FDCAN_CCCR.INIT of its own, stopping all bus activities. Once FDCAN_CCCR.INIT has been cleared by the CPU, the device waits for 129 occurrences of bus Idle (129 × 11 consecutive recessive bits) before resuming normal operation. At the end of the Bus_Off recovery sequence, the error management counters are reset. During the waiting time after the reset of FDCAN_CCCR.INIT, each time a sequence of 11 recessive bits has been monitored, a Bit0 error code is written to FDCAN_PSR.LEC, enabling the CPU to readily check up whether the CAN bus is stuck at dominant or continuously disturbed and to monitor the Bus_Off recovery sequence. FDCAN_ECR.REC is used to count these sequences.

71.5.14 FDCAN transmitter delay compensation register (FDCAN_TDCR)

Address offset: 0x0048

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TDCO[6:0]							Res.	TDCF[6:0]						
	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bits 14:8 **TDCO[6:0]**: Transmitter delay compensation offset

Offset value defining the distance between the measured delay from FDCAN_TX to FDCAN_RX and the secondary sample point. Valid values are 0 to 127 mtq.

These are write-protected bits, which means that write access by the bits is possible only when the bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **TDCF[6:0]**: Transmitter delay compensation filter window length

Defines the minimum value for the SSP position, dominant edges on FDCAN_RX that would result in an earlier SSP position are ignored for transmitter delay measurements.

These are write-protected bits, which means that write access by the bits is possible only when the bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

71.5.15 FDCAN interrupt register (FDCAN_IR)

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the user clears them. A flag is cleared by writing a 1 to the corresponding bit position.

Writing a 0 has no effect. A hard reset clears the register. The configuration of IE controls whether an interrupt is generated. The configuration of ILS controls on which interrupt line an interrupt is signaled.

Address offset: 0x0050

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	ARA	PED	PEA	WDI	BO	EW	EP	ELO	Res.	Res.	DRX	TOO	MRAF	TSW
		rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TEFL	TEFF	TEFW	TEFN	TFE	TCF	TC	HPM	RF1L	RF1F	RF1W	RF1N	RF0L	RF0F	RF0W	RF0N
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **ARA**: Access to reserved address
0: No access to reserved address occurred
1: Access to reserved address occurred

Bit 28 **PED**: Protocol error in data phase (data bit time is used)
0: No protocol error in data phase
1: Protocol error in data phase detected (PSR.DLEC different from 0,7)

Bit 27 **PEA**: Protocol error in arbitration phase (nominal bit time is used)
0: No protocol error in arbitration phase
1: Protocol error in arbitration phase detected (PSR.LEC different from 0,7)

Bit 26 **WDI**: Watchdog interrupt
0: No message RAM watchdog event occurred
1: Message RAM watchdog event due to missing READY

Bit 25 **BO**: Bus_Off status
0: Bus_Off status unchanged
1: Bus_Off status changed

Bit 24 **EW**: Warning status
0: Error_Warning status unchanged
1: Error_Warning status changed

Bit 23 **EP**: Error passive
0: Error_Passive status unchanged
1: Error_Passive status changed

Bit 22 **ELO**: Error logging overflow
0: CAN error logging counter did not overflow
1: Overflow of CAN error logging counter occurred

Bits 21:20 Reserved, must be kept at reset value.

Bit 19 **DRX**: Message stored to dedicated Rx buffer
The flag is set whenever a received message has been stored into a dedicated Rx buffer.
0: No Rx buffer updated
1: At least one received message stored into a Rx buffer

Bit 18 **TOO**: Timeout occurred
0: No timeout
1: Timeout reached

Bit 17 **MRAF**: Message RAM access failure
The flag is set when the Rx handler
- Has not completed acceptance filtering or storage of an accepted message until the arbitration field of the following message has been received. In this case acceptance filtering or message storage is aborted and the Rx handler starts processing of the following message.
- Was unable to write a message to the message RAM. In this case message storage is aborted.

In both cases the FIFO put index is not updated or the New data flag for a dedicated Rx buffer is not set. The partly stored message is overwritten when the next message is stored to this location.
The flag is also set when the Tx handler was not able to read a message from the message RAM in time. In this case message transmission is aborted. In case of a Tx handler access failure the FDCAN is switched into restricted operation mode (see *Restricted operation mode*). To leave restricted operation mode, the user must reset FDCAN_CCCR.ASM.
0: No message RAM access failure occurred
1: Message RAM access failure occurred

Bit 16 **TSW**: Timestamp wraparound
0: No timestamp counter wraparound
1: Timestamp counter wraparound

Bit 15 **TEFL**: Tx event FIFO element lost
0: No Tx event FIFO element lost
1: Tx event FIFO element lost, also set after write attempt to Tx event FIFO of size 0

Bit 14 **TEFF**: Tx event FIFO full
0: Tx event FIFO not full
1: Tx event FIFO full

Bit 13 **TEFW**: Tx event FIFO watermark reached
0: Tx event FIFO fill level below watermark
1: Tx event FIFO fill level reached watermark

Bit 12 **TEFN**: Tx event FIFO new entry
0: Tx event FIFO unchanged
1: Tx handler wrote Tx event FIFO element

Bit 11 **TFE**: Tx FIFO empty
0: Tx FIFO non-empty
1: Tx FIFO empty

Bit 10 **TCF**: Transmission cancellation finished
0: No transmission cancellation finished
1: Transmission cancellation finished

Bit 9 **TC**: Transmission completed
0: No transmission completed
1: Transmission completed

Bit 8 **HPM**: High priority message
0: No high priority message received
1: High priority message received

Bit 7 **RF1L**: Rx FIFO 1 message lost
0: No Rx FIFO 1 message lost
1: Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size 0

Bit 6 **RF1F**: Rx FIFO 1 full
0: Rx FIFO 1 not full
1: Rx FIFO 1 full

Bit 5 **RF1W**: Rx FIFO 1 watermark reached
0: Rx FIFO 1 fill level below watermark
1: Rx FIFO 1 fill level reached watermark

Bit 4 **RF1N**: Rx FIFO 1 new message
0: No new message written to Rx FIFO 1
1: New message written to Rx FIFO 1

Bit 3 **RF0L**: Rx FIFO 0 message lost
0: No Rx FIFO 0 message lost
1: Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size 0

Bit 2 **RF0F**: Rx FIFO 0 full
0: Rx FIFO 0 not full
1: Rx FIFO 0 full

Bit 1 **RF0W**: Rx FIFO 0 watermark reached
0: Rx FIFO 0 fill level below watermark
1: Rx FIFO 0 fill level reached watermark

Bit 0 **RF0N**: Rx FIFO 0 New message
0: No new message written to Rx FIFO 0
1: New message written to Rx FIFO 0

71.5.16 FDCAN interrupt enable register (FDCAN_IE)

The settings in the interrupt enable register determine which status changes in the interrupt register is signaled on an interrupt line.

Address offset: 0x0054

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	ARAE	PEDE	PEAE	WDIE	BOE	EWE	EPE	ELOE	Res.	Res.	DRXE	TOOE	MRAFE	TSWE
		rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TEFLE	TEFFE	TEFWE	TEFNE	TFEE	TCFE	TCE	HPME	RF1LE	RF1FE	RF1WE	RF1NE	RF0LE	RF0FE	RF0WE	RF0NE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **ARAE**: Access to Reserved address enable

Bit 28 **PEDE**: Protocol error in data phase enable

Bit 27 **PEAE**: Protocol error in Arbitration phase enable

Bit 26 **WDIE**: Watchdog interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 25 **BOE**: Bus_Off status
0: Interrupt disabled
1: Interrupt enabled

Bit 24 **EWE**: Warning status interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 23 **EPE**: Error passive interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 22 **ELOE**: Error logging overflow interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bits 21:20 Reserved, must be kept at reset value.

Bit 19 **DRXE**: Message stored to dedicated Rx buffer interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 18 **TOOE**: Timeout occurred interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 17 **MRAFE:** Message RAM access failure interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 16 **TSWE**: Timestamp wraparound interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 15 **TEFLE**: Tx event FIFO element lost interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 14 **TEFFE**: Tx event FIFO full interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 13 **TEFWE**: Tx event FIFO watermark reached interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 12 **TEFNE**: Tx event FIFO new entry interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 11 **TFEE**: Tx FIFO empty interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 10 **TCFE**: Transmission cancellation finished interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 9 **TCE**: Transmission completed interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 8 **HPME**: High priority message interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 7 **RF1LE**: Rx FIFO 1 message lost interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 6 **RF1FE**: Rx FIFO 1 full interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 5 **RF1WE**: Rx FIFO 1 watermark reached interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 4 **RF1NE**: Rx FIFO 1 new message interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 3 **RF0LE**: Rx FIFO 0 message lost interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 2 **RF0FE**: Rx FIFO 0 full interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 1 **RF0WE**: Rx FIFO 0 watermark reached interrupt enable
0: Interrupt disabled
1: Interrupt enabled

Bit 0 **RF0NE**: Rx FIFO 0 new message interrupt enable
0: Interrupt disabled
1: Interrupt enabled

71.5.17 FDCAN interrupt line select register (FDCAN_ILS)

This register assigns an interrupt generated by a specific interrupt flag from the interrupt register to one of the two module interrupt lines. For interrupt generation the respective interrupt line must be enabled via FDCAN_ILE.EINT0 and FDCAN_ILE.EINT1.

Address offset: 0x0058

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	ARAL	PEDL	PEAL	WDIL	BOL	EWL	EPL	ELOL	Res.	Res.	DRXL	TOOL	MRAFL	TSWL
		rw	rw	rw	rw	rw	rw	rw	rw			rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TEFLL	TEFFL	TEFWL	TEFNL	TFEL	TCFL	TCL	HPML	RF1LL	RF1FL	RF1WL	RF1NL	RF0LL	RF0FL	RF0WL	RF0NL
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **ARAL**: Access to reserved address line

Bit 28 **PEDL**: Protocol error in data phase line

Bit 27 **PEAL**: Protocol error in arbitration phase line

Bit 26 **WDIL**: Watchdog interrupt line

Bit 25 **BOL**: Bus_Off status

Bit 24 **EWL**: Warning status interrupt line

Bit 23 **EPL**: Error passive interrupt line

Bit 22 **ELOL**: Error logging overflow interrupt line

Bits 21:20 Reserved, must be kept at reset value.

Bit 19 **DRXL**: Message stored to dedicated Rx buffer interrupt line

Bit 18 **TOOL**: Timeout occurred interrupt line

Bit 17 **MRAFL**: Message RAM access failure interrupt line

Bit 16 **TSWL**: Timestamp wraparound interrupt line

Bit 15 **TEFLL**: Tx event FIFO element Lost interrupt line

Bit 14 **TEFFL**: Tx event FIFO full interrupt line

Bit 13 **TEFWL**: Tx event FIFO watermark reached interrupt line

Bit 12 **TEFNL**: Tx event FIFO new entry interrupt line

Bit 11 **TFEL**: Tx FIFO empty interrupt line

Bit 10 **TCFL**: Transmission cancellation finished interrupt line

Bit 9 **TCL**: Transmission completed interrupt line

Bit 8 **HPML**: High priority message interrupt line

Bit 7 **RF1LL**: Rx FIFO 1 message lost interrupt line

Bit 6 **RF1FL**: Rx FIFO 1 full interrupt line

Bit 5 **RF1WL**: Rx FIFO 1 watermark reached interrupt line

Bit 4 **RF1NL**: Rx FIFO 1 new message interrupt line

Bit 3 **RF0LL**: Rx FIFO 0 message lost interrupt line

Bit 2 **RF0FL**: Rx FIFO 0 full interrupt line

Bit 1 **RF0WL**: Rx FIFO 0 watermark reached interrupt line

Bit 0 **RF0NL**: Rx FIFO 0 new message interrupt line

71.5.18 FDCAN interrupt line enable register (FDCAN_ILE)

Each of the two interrupt lines to the CPU can be enabled/disabled separately by programming bits EINT0 and EINT1.

Address offset: 0x005C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EINT1	EINT0
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **EINT1**: Enable interrupt line 1

0: Interrupt line fdcan_intr1_it disabled

1: Interrupt line fdcan_intr1_it enabled

Bit 0 **EINT0**: Enable interrupt line 0
0: Interrupt line fdcan_intr0_it disabled
1: Interrupt line fdcan_intr0_it enabled

71.5.19 FDCAN global filter configuration register (FDCAN_GFC)

Global settings for message ID filtering. The global filter configuration register controls the filter path for standard and extended messages as described in *Figure 961* and *Figure 962*.

Address offset: 0x0080

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ANFS[1:0]		ANFE[1:0]		RRFS	RRFE
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:4 **ANFS[1:0]**: Accept non-matching frames standard
Defines how received messages with 11-bit ID that do not match any element of the filter list are treated.
00: Accept in Rx FIFO 0
01: Accept in Rx FIFO 1
10: Reject
11: Reject
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 3:2 **ANFE[1:0]**: Accept non-matching frames extended
Defines how received messages with 29-bit ID that do not match any element of the filter list are treated.
00: Accept in Rx FIFO 0
01: Accept in Rx FIFO 1
10: Reject
11: Reject
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 1 **RRFS**: Reject remote frames standard
0: Filter remote frames with 11-bit standard ID
1: Reject all remote frames with 11-bit standard ID
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 0 **RRFE**: Reject remote frames extended
0: Filter remote frames with 29-bit standard ID
1: Reject all remote frames with 29-bit standard ID
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

71.5.20 FDCAN standard ID filter configuration register (FDCAN_SIDFC)

Settings for 11-bit standard message ID filtering.The standard ID filter configuration register controls the filter path for standard messages as described in *Figure 961*.

Address offset: 0x0084

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSS[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLSSA[13:0]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **LSS[7:0]**: List size standard
0: No standard message ID filter
1-128: Number of standard message ID filter elements
>128: Values greater than 128 are interpreted as 128.
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:2 **FLSSA[13:0]**: Filter list standard start address
Start address of standard message ID filter list (32-bit word address, see *Table 714*).These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 1:0 Reserved, must be kept at reset value.

71.5.21 FDCAN extended ID filter configuration register (FDCAN_XIDFC)

Settings for 29-bit extended message ID filtering. The FDCAN extended ID filter configuration register controls the filter path for standard messages as described in *Figure 962*.

Address offset: 0x0088

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSE[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLESA[13:0]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **LSE[7:0]**: List size extended
0: No extended message ID filter
1-128: Number of extended ID filter elements
>128: Values greater than 128 are interpreted as 128.
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:2 **FLESA[13:0]**: Filter list extended start address
Start address of extended message ID filter list (32-bit word address, see *Table 716: Extended message ID filter element*).
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 1:0 Reserved, must be kept at reset value.

71.5.22 FDCAN extended ID and mask register (FDCAN_XIDAM)

Address offset: 0x0090

Reset value: 0x1FFF FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	EIDM[28:16]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EIDM[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:0 **EIDM[28:0]**: Extended ID Mask
For acceptance filtering of extended frames the extended ID AND Mask is AND-ed with the message ID of a received frame. Intended for masking of 29-bit IDs in SAE J1939. With the reset value of all bits set to 1 the mask is not active.
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

71.5.23 FDCAN high priority message status register (FDCAN_HPMS)

This register is updated every time a message ID filter element configured to generate a priority event match. This can be used to monitor the status of incoming high priority messages and to enable fast access to these messages.

Address offset: 0x0094

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLST	FIDX[6:0]							MSI[1:0]		BIDX[5:0]					
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **FLST**: Filter list
Indicates the filter list of the matching filter element.
0: Standard filter list
1: Extended filter list

Bits 14:8 **FIDX[6:0]**: Filter index
Index of matching filter element. Range is 0 to FDCAN_SIDFC[LSS] - 1 or FDCAN_XIDFC[LSE] - 1.

Bits 7:6 **MSI[1:0]**: Message storage indicator
00: No FIFO selected
01: FIFO overrun
10: Message stored in FIFO 0
11: Message stored in FIFO 1

Bits 5:0 **BIDX[5:0]**: Buffer index
Index of Rx FIFO element to which the message was stored. Only valid when MSI[1] = 1.

71.5.24 FDCAN new data 1 register (FDCAN_NDAT1)

Address offset: 0x0098

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ND31	ND30	ND29	ND28	ND27	ND26	ND25	ND24	ND23	ND22	ND21	ND20	ND19	ND18	ND17	ND16
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ND15	ND14	ND13	ND12	ND11	ND10	ND9	ND8	ND7	ND6	ND5	ND4	ND3	ND2	ND1	ND0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ND[31:0]**: New data[31:0]

The register holds the new data flags of Rx buffers 0 to 31. The flags are set when the respective Rx buffer has been updated from a received frame. The flags remain set until the user clears them. A flag is cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect. A hard reset clears the register.

0: Rx buffer not updated

1: Rx buffer updated from new message

71.5.25 FDCAN new data 2 register (FDCAN_NDAT2)

Address offset: 0x009C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ND63	ND62	ND61	ND60	ND59	ND58	ND57	ND56	ND55	ND54	ND53	ND52	ND51	ND50	ND49	ND48
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ND47	ND46	ND45	ND44	ND43	ND42	ND41	ND40	ND39	ND38	ND37	ND36	ND35	ND34	ND33	ND32
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ND[63:32]**: New data[63:32]

The register holds the new data flags of Rx buffers 32 to 63. The flags are set when the respective Rx buffer has been updated from a received frame. The flags remain set until the user clears them. A flag is cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect. A hard reset clears clear the register.

0: Rx buffer not updated

1: Rx buffer updated from new message

71.5.26 FDCAN Rx FIFO 0 configuration register (FDCAN_RXF0C)

Address offset: 0x00A0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
F0OM	F0WM[6:0]							Res.	F0S[6:0]						
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F0SA[13:0]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bit 31 **F0OM**: FIFO 0 operation mode

FIFO 0 can be operated in blocking or in overwrite mode.

0: FIFO 0 blocking mode

1: FIFO 0 overwrite mode

Bits 30:24 **F0WM[6:0]**: FIFO 0 watermark
0: Watermark interrupt disabled
1-64: Level for Rx FIFO 0 watermark interrupt (FDCAN_IR.RF0W)
>64: Watermark interrupt disabled
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 23 Reserved, must be kept at reset value.

Bits 22:16 **F0S[6:0]**: Rx FIFO 0 size
0: No Rx FIFO 0
1-64: Number of Rx FIFO 0 elements
>64: Values greater than 64 are interpreted as 64
The Rx FIFO 0 elements are indexed from 0 to F0S-1.

Bits 15:2 **F0SA[13:0]**: Rx FIFO 0 start address
Start address of Rx FIFO 0 in message RAM (32-bit word address, see *Figure 960*).

Bits 1:0 Reserved, must be kept at reset value.

71.5.27 FDCAN Rx FIFO 0 status register (FDCAN_RXF0S)

Address offset: 0x00A4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	RF0L	F0F	Res.	Res.	F0PI[5:0]					
						rw	rw			rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	F0GI[5:0]						Res.	F0FL[6:0]						
		rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **RF0L**: Rx FIFO 0 message lost
This bit is a copy of interrupt flag FDCAN_IR.RF0L. When FDCAN_IR.RF0L is reset, this bit is also reset.
0: No Rx FIFO 0 message lost
1: Rx FIFO 0 message lost, also set after write attempt to Rx FIFO 0 of size 0

Bit 24 **F0F**: Rx FIFO 0 full
0: Rx FIFO 0 not full
1: Rx FIFO 0 full

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **F0PI[5:0]**: Rx FIFO 0 put index
Rx FIFO 0 write index pointer, range 0 to 63.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **F0GI[5:0]**: Rx FIFO 0 get index
Rx FIFO 0 read index pointer, range 0 to 63.

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **F0FL**[6:0]: Rx FIFO 0 fill level
Number of elements stored in Rx FIFO 0, range 0 to 64.

71.5.28 FDCAN Rx FIFO 0 acknowledge register (FDCAN_RXF0A)

Address offset: 0x00A8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F0AI[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **F0AI[5:0]**: Rx FIFO 0 acknowledge index
After the user has read a message or a sequence of messages from Rx FIFO 0 it must write the buffer index of the last element read from Rx FIFO 0 to F0AI. This sets the Rx FIFO 0 get index FDCAN_RXF0S.F0GI to F0AI + 1 and update the FIFO 0 fill level FDCAN_RXF0S.F0FL.

71.5.29 FDCAN Rx buffer configuration register (FDCAN_RXBC)

Address offset: 0x00AC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RBSA[13:0]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:2 **RBSA[13:0]**: Rx buffer start address
Configures the start address of the Rx buffers section in the message RAM (32-bit word address). Also used to reference debug messages A, B, C.
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 1:0 Reserved, must be kept at reset value.

71.5.30 FDCAN Rx FIFO 1 configuration register (FDCAN_RXF1C)

Address offset: 0x00B0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
F1OM	F1WM[6:0]							Res.	F1S[6:0]						
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
F1SA[13:0]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bit 31 **F1OM**: FIFO 1 operation mode

FIFO 1 can be operated in blocking or in overwrite mode.

0: FIFO 1 blocking mode

1: FIFO 1 overwrite mode

Bits 30:24 **F1WM[6:0]**: Rx FIFO 1 watermark

0: Watermark interrupt disabled

1-64: Level for Rx FIFO 1 watermark interrupt (FDCAN_IR.RF1W)

>64: Watermark interrupt disabled.

These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 23 Reserved, must be kept at reset value.

Bits 22:16 **F1S[6:0]**: Rx FIFO 1 size

0: No Rx FIFO 1

1-64: Number of Rx FIFO 1 elements

>64: Values greater than 64 are interpreted as 64

The Rx FIFO 1 elements are indexed from 0 to F1S - 1.

These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:2 **F1SA[13:0]**: Rx FIFO 1 start address

start address of Rx FIFO 1 in message RAM (32-bit word address, see *Figure 960*).

These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 1:0 Reserved, must be kept at reset value.

71.5.31 FDCAN Rx FIFO 1 status register (FDCAN_RXF1S)

Address offset: 0x00B4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMS[1:0]		Res.	Res.	Res.	Res.	RF1L	F1F	Res.	Res.	F1PI[5:0]					
r	r					r	r			r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	F1GI[5:0]						Res.	F1FL[6:0]						
		r	r	r	r	r	r		r	r	r	r	r	r	r

Bits 31:30 **DMS[1:0]**: Debug message status
00: Idle state, wait for reception of debug messages
01: Debug message A received
10: Debug messages A, B received
11: Debug messages A, B, C received

Bits 29:26 Reserved, must be kept at reset value.

Bit 25 **RF1L**: Rx FIFO 1 message lost
This bit is a copy of interrupt flag FDCAN_IR.RF1L. When FDCAN_IR.RF1L is reset, this bit is also reset.
0: No Rx FIFO 1 message lost
1: Rx FIFO 1 message lost, also set after write attempt to Rx FIFO 1 of size 0.

Bit 24 **F1F**: Rx FIFO 1 full
0: Rx FIFO 1 not full
1: Rx FIFO 1 full

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **F1PI[5:0]**: Rx FIFO 1 put index
Rx FIFO 1 write index pointer, range 0 to 63.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:8 **F1GI[5:0]**: Rx FIFO 1 get index
Rx FIFO 1 read index pointer, range 0 to 63.

Bit 7 Reserved, must be kept at reset value.

Bits 6:0 **F1FL[6:0]**: Rx FIFO 1 fill level
Number of elements stored in Rx FIFO 1, range 0 to 64

71.5.32 FDCAN Rx FIFO 1 acknowledge register (FDCAN_RXF1A)

Address offset: 0x00B8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F1AI[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **F1AI[5:0]**: Rx FIFO 1 acknowledge index

After the user has read a message or a sequence of messages from Rx FIFO 1 it must write the buffer index of the last element read from Rx FIFO 1 to F1AI. This sets the Rx FIFO 1 get index FDCAN_RXF1S.F1GI. to F1AI + 1 and update the FIFO 1 fill level FDCAN_RXF1S.F1FL.

71.5.33 FDCAN Rx buffer element size configuration register (FDCAN_RXESC)

Configures the number of data bytes belonging to an Rx buffer / Rx FIFO element. Data field sizes higher than 8 bytes are intended for CAN FD operation only.

Address offset: 0x00BC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	RBDS[2:0]			Res.	F1DS[2:0]			Res.	F0DS[2:0]		
					rw	rw	rw		rw	rw	rw		rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:8 **RBDS[2:0]**: Rx buffer data field size

000: 8-byte data field
001: 12-byte data field
010: 16-byte data field
011: 20-byte data field
100: 24-byte data field
101: 32-byte data field
110: 48-byte data field
111: 64-byte data field

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **F1DS[2:0]**: Rx FIFO 0 data field size
000: 8-byte data field
001: 12-byte data field
010: 16-byte data field
011: 20-byte data field
100: 24-byte data field
101: 32-byte data field
110: 48-byte data field
111: 64-byte data field

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **F0DS[2:0]**: Rx FIFO 1 data field size
000: 8-byte data field
001: 12-byte data field
010: 16-byte data field
011: 20-byte data field
100: 24-byte data field
101: 32-byte data field
110: 48-byte data field
111: 64-byte data field

71.5.34 FDCAN Tx buffer configuration register (FDCAN_TXBC)

Address offset: 0x00C0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TFQM	TFQS[5:0]						Res.	Res.	NDTB[5:0]					
	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TBSA[13:0]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bit 31 Reserved, must be kept at reset value.

Bit 30 **TFQM**: Tx FIFO/queue mode
0: Tx FIFO operation
1: Tx queue operation.
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 29:24 **TFQS[5:0]**: Transmit FIFO/queue size
0: No Tx FIFO/queue
1-32: Number of Tx buffers used for Tx FIFO/queue
>32: Values greater than 32 are interpreted as 32.
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **NDTB[5:0]**: Number of dedicated transmit buffers
0: No dedicated Tx buffers
1-32: Number of dedicated Tx buffers
>32: Values greater than 32 are interpreted as 32.
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:2 **TBSA[13:0]**: Tx buffers start address
Start address of Tx buffers section in message RAM (32-bit word address, see *Figure 960*).
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 1:0 Reserved, must be kept at reset value.

Note: The sum of TFQS and NDTB cannot be larger than 32. There is no check for erroneous configurations. The Tx buffers section in the message RAM starts with the dedicated Tx buffers.

71.5.35 FDCAN Tx FIFO/queue status register (FDCAN_TXFQS)

The Tx FIFO/queue status is related to the pending Tx requests listed in register FDCAN_TXBRP. Therefore the effect of add/cancellation requests may be delayed due to a running Tx scan (FDCAN_TXBRP not yet updated).

Address offset: 0x00C4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFQF	TFQPI[4:0]				
										r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	TFGI[4:0]					Res.	Res.	TFFL[5:0]					
			r	r	r	r	r			r	r	r	r	r	r

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **TFQF**: Tx FIFO/queue full
0 Tx FIFO/queue not full
1 Tx FIFO/queue full

Bits 20:16 **TFQPI[4:0]**: Tx FIFO/queue put index
Tx FIFO/queue write index pointer, range 0 to 31

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **TFGI[4:0]**:
Tx FIFO get index.
Tx FIFO read index pointer, range 0 to 31. Read as 0 when Tx queue operation is configured (FDCAN_TXBC.TFQM = 1)

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **TFFL[5:0]**: Tx FIFO free level
Number of consecutive free Tx FIFO elements starting from TFGI, range 0 to 32. Read as 0 when Tx queue operation is configured (FDCAN_TXBC.TFQM = 1).

Note: In case of mixed configurations where dedicated Tx buffers are combined with a Tx FIFO or a Tx queue, the put and get index indicate the number of the Tx buffer starting with the first dedicated Tx buffers. For example: For a configuration of 12 dedicated Tx buffers and a Tx FIFO of 20 buffers a put index of 15 points to the fourth buffer of the Tx FIFO.

71.5.36 FDCAN Tx buffer element size configuration register (FDCAN_TXESC)

Configures the number of data bytes belonging to a Tx buffer element. Data field sizes >8 bytes are intended for CAN FD operation only.

Address offset: 0x00C8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TBDS[2:0]		
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bits 2:0 **TBDS[2:0]**: Tx buffer data field size:
000: 8-byte data field
001: 12-byte data field
010: 16-byte data field
011: 20-byte data field
100: 24-byte data field
101: 32-byte data field
110: 48-byte data field
111: 64-byte data field

71.5.37 FDCAN Tx buffer request pending register (FDCAN_TXBRP)

Address offset: 0x00CC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TRP[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRP[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TRP[31:0]**:Transmission request pending

Each Tx buffer has its own transmission request pending bit. The bits are set via register FDCAN_TXBAR. The bits are reset after a requested transmission has completed or has been canceled via register FDCAN_TXBCR.

FDCAN_TXBRP bits are set only for those Tx buffers configured via FDCAN_TXBC. After an FDCAN_TXBRP bit has been set, a Tx scan (see *Filtering for Debug messages*) is started to check for the pending Tx request with the highest priority (Tx buffer with lowest message ID).

A cancellation request resets the corresponding transmission request pending bit of register FDCAN_TXBRP. In case a transmission has already been started when a cancellation is requested, this is done at the end of the transmission, regardless whether the transmission was successful or not. The cancellation request bits are reset directly after the corresponding FDCAN_TXBRP bit has been reset.

After a cancellation has been requested, a finished cancellation is signaled via FDCAN_TXBCF

after successful transmission together with the corresponding FDCAN_TXBTO bit

when the transmission has not yet been started at the point of cancellation

when the transmission has been aborted due to lost arbitration

when an error occurred during frame transmission

In DAR mode all transmissions are automatically canceled if they are not successful. The corresponding FDCAN_TXBCF bit is set for all unsuccessful transmissions.

0: No transmission request pending

1: Transmission request pending

Note: FDCAN_TXBRP bits set while a Tx scan is in progress are not considered during this particular Tx scan. In case a cancellation is requested for such a Tx buffer, this add request is canceled immediately, the corresponding FDCAN_TXBRP bit is reset.

71.5.38 FDCAN Tx buffer add request register (FDCAN_TXBAR)

Address offset: 0x00D0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **AR[31:0]**:Add request

Each Tx buffer has its own add request bit. Writing a 1 sets the corresponding add request bit; writing a 0 has no impact. This enables the user to set transmission requests for multiple Tx buffers with one write to FDCAN_TXBAR. FDCAN_TXBAR bits are set only for those Tx buffers configured via FDCAN_TXBC. When no Tx scan is running, the bits are reset immediately, else the bits remain set until the Tx scan process has completed.

0: No transmission request added

1: Transmission requested added.

Note: If an add request is applied for a Tx buffer with pending transmission request (corresponding FDCAN_TXBRP bit already set), the request is ignored.

71.5.39 FDCAN Tx buffer cancellation request register (FDCAN_TXBCR)

Address offset: 0x00D4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CR[31:0]**: Cancellation request

Each Tx buffer has its own cancellation request bit. Writing a 1 sets the corresponding cancellation request bit; writing a 0 has no impact.

This enables the user to set cancellation requests for multiple Tx buffers with one write to FDCAN_TXBCR. FDCAN_TXBCR bits are set only for those Tx buffers configured via FDCAN_TXBC. The bits remain set until the corresponding FDCAN_TXBRP bit is reset.

0: No cancellation pending

1: Cancellation pending

71.5.40 FDCAN Tx buffer transmission occurred register (FDCAN_TXBTO)

Address offset: 0x00D8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TO[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TO[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TO[31:0]**: Transmission occurred

Each Tx buffer has its own transmission occurred bit. The bits are set when the corresponding FDCAN_TXBRP bit is cleared after a successful transmission. The bits are reset when a new transmission is requested by writing a 1 to the corresponding bit of register FDCAN_TXBAR.

0: No transmission occurred

1: Transmission occurred

71.5.41 FDCAN Tx buffer cancellation finished register (FDCAN_TXBCF)

Address offset: 0x00DC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CF[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CF[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CF[31:0]**: Cancellation finished

Each Tx buffer has its own cancellation finished bit. The bits are set when the corresponding FDCAN_TXBRP bit is cleared after a cancellation was requested via FDCAN_TXBCR. In case the corresponding FDCAN_TXBRP bit was not set at the point of cancellation, CF is set immediately. The bits are reset when a new transmission is requested by writing a 1 to the corresponding bit of register FDCAN_TXBAR.

0: No transmit buffer cancellation

1: Transmit buffer cancellation finished

71.5.42 FDCAN Tx buffer transmission interrupt enable register (FDCAN_TXBTIE)

Address offset: 0x00E0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TIE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TIE[31:0]**: Transmission interrupt enable

Each Tx buffer has its own transmission interrupt enable bit.

0: Transmission interrupt disabled

1: Transmission interrupt enable

71.5.43 FDCAN Tx buffer cancellation finished interrupt enable register (FDCAN_TXBCIE)

Address offset: 0x00E4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CFIE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CFIE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CFIE[31:0]**: Cancellation finished interrupt enable
Each Tx buffer has its own cancellation finished interrupt enable bit.
0: Cancellation finished interrupt disabled
1: Cancellation finished interrupt enabled

71.5.44 FDCAN Tx event FIFO configuration register (FDCAN_TXEFC)

Address offset: 0x00F0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	EFWM[5:0]						Res.	Res.	EFS[5:0]					
		rw	rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EFSA[13:0]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:24 **EFWM[5:0]**: Event FIFO watermark
0: Watermark interrupt disabled
1-32: Level for Tx event FIFO watermark interrupt (FDCAN_IR.TEFW)
>32: Watermark interrupt disabled
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **EFS[5:0]**: Event FIFO size.
0: Tx event FIFO disabled
1-32: Number of Tx event FIFO elements
>32: Values greater than 32 are interpreted as 32
The Tx event FIFO elements are indexed from 0 to EFS-1.
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:2 **EFSA[13:0]**: Event FIFO start address
Start address of Tx event FIFO in message RAM (32-bit word address, see *Figure 960*).
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 1:0 Reserved, must be kept at reset value.

71.5.45 FDCAN Tx event FIFO status register (FDCAN_TXEFS)

Address offset: 0x00F4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	TEFL	EFF	Res.	Res.	Res.	EFPI[4:0]				
						r	r				r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	EFGI[4:0]					Res.	Res.	EFFL[5:0]					
			r	r	r	r	r			r	r	r	r	r	r

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **TEFL**: Tx event FIFO element lost
This bit is a copy of interrupt flag FDCAN_IR.TEFL. When FDCAN_IR.TEFL is reset, this bit is also reset.
0 No Tx event FIFO element lost
1 Tx event FIFO element lost, also set after write attempt to Tx event FIFO of size 0.

Bit 24 **EFF**: Event FIFO full
0: Tx event FIFO not full
1: Tx event FIFO full

Bits 23:21 Reserved, must be kept at reset value.

Bits 20:16 **EFPI[4:0]**: Event FIFO put index
Tx event FIFO write index pointer, range 0 to 31.

Bits 15:13 Reserved, must be kept at reset value.

Bits 12:8 **EFGI[4:0]**: Event FIFO get index
Tx event FIFO read index pointer, range 0 to 31.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:0 **EFFL[5:0]**: Event FIFO fill level
Number of elements stored in Tx event FIFO, range 0 to 31.

71.5.46 FDCAN Tx event FIFO acknowledge register (FDCAN_TXEFA)

Address offset: 0x00F8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EFAI[4:0]				
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **EFAI[4:0]**: Event FIFO acknowledge index

After the user has read an element or a sequence of elements from the Tx event FIFO, it must write the index of the last element read from Tx event FIFO to EFAI. This sets the Tx event FIFO get index FDCAN_TXEFS.EFGI to EFAI + 1 and update the FIFO 0 fill level FDCAN_TXEFS.EFFL.

71.5.47 FDCAN register map

Table 720. FDCAN register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	FDCAN_CREL	REL[3:0]				STEP[3:0]				SUBSTEP [3:0]				YEAR[3:0]				MON[7:0]								DAY[7:0]							
	Reset value	0	0	1	1	0	0	1	0	0	0	0	1	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	1	1	0	0	0
0x0004	FDCAN_ENDN	ETV[31:0]																															
	Reset value	1	0	0	0	0	1	1	1	0	1	1	0	0	1	1	1	0	1	0	0	0	0	1	1	0	0	1	0	0	0	0	1
0x0008	Reserved	Reserved																															
0x000C	FDCAN_DBTP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDC	Res.	Res.	DBRP[4:0]					Res.	Res.	Res.	DTSEG1[4:0]					DTSEG2 [3:0]				DSJW[3:0]			
	Reset value									0			0	0	0	0	0				0	1	0	1	0	0	0	1	1	0	0	1	1
0x0010	FDCAN_TEST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RX	TX [1:0]		LBCK	Res.	Res.	Res.	Res.
	Reset value																									0	0	0	0				
0x0014	FDCAN_RWD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WDV[7:0]								WDC[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0018	FDCAN_CCCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NISO	TXP	EFBI	PXHD	Res.	Res.	BRSE	FDOE	TEST	DAR	MON	CSR	CSA	ASM	CCE	INIT
	Reset value																	0	0	0	0			0	0	0	0	0	0	0	0	0	1
0x001C	FDCAN_NBTP	NSJW[6:0]							NBRP[8:0]									NTSEG1[7:0]								Res.	NTSEG2[6:0]							
	Reset value	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0		0	0	0	0	0	1	1
0x0020	FDCAN_TSCC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCP[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSS [1:0]	
	Reset value													0	0	0	0															0	0

Table 720. FDCAN register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0024	FDCAN_TSCV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSC[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0028	FDCAN_TOCC	TOP[15:0]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOS [1:0]		ETOC
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1														0	0	0
0x002C	FDCAN_TOCV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOC[15:0]															
	Reset value																	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0030-0x003F	Reserved	Reserved																															
0x0040	FDCAN_ECR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CEL[7:0]								RP	REC[6:0]							TEC[7:0]							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0044	FDCAN_PSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDCV[6:0]							Res.	PXE	REDL	RBRS	RESI	DLEC [2:0]			BO	EW	EP	ACT [1:0]		LEC[2:0]		
	Reset value										0	0	0	0	0	0	0		0	0	0	0	1	1	1	0	0	0	0	0	1	1	1
0x0048	FDCAN_TDCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDCO[6:0]							Res.	TDCF[6:0]						
	Reset value																		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0050	FDCAN_IR	Res.	Res.	ARA	PED	PEA	WDI	BO	EW	EP	ELO	Res.	Res.	DRX	TOO	MRAF	TSW	TEFL	TEFF	TEFW	TEFN	TFE	TCF	TC	HPM	RF1L	RF1F	RF1W	RF1N	RF0L	RF0F	RF0W	RF0N
	Reset value			0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0054	FDCAN_IE	Res.	Res.	ARAE	PEDE	PEAE	WDIE	BOE	EWE	EPE	ELOE	Res.	Res.	DRXE	TOOE	MRAFE	TSWE	TEFLE	TEFFE	TEFWE	TEFNE	TFEE	TCFE	TCE	HPME	RF1LE	RF1FE	RF1WE	RF1NE	RF0LE	RF0FE	RF0WE	RF0NE
	Reset value			0	0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0058	FDCAN_ILS	Res.	Res.	ARAL	PEDL	PEAL	WDIL	BOL	EWL	EPL	ELOL	Res.	Res.	DRXL	TOOL	MRAFL	TSWL	TEFLL	TEFFL	TEFWL	TEFNL	TFEL	TCFL	TCL	HPML	RF1LL	RF1FL	RF1WL	RF1NL	RF0LL	RF0FL	RF0WL	RF0NL
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x005C	FDCAN_ILE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EINT1	EINT0
	Reset value																															0	0
0x006-0x007F	Reserved	Reserved																															
0x0080	FDCAN_GFC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ANFS [1:0]		ANFE [1:0]		RRFS	RRFE
	Reset value																											0	0	0	0	0	0
0x0084	FDCAN_SIDFC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSS[7:0]								FLSSA[13:0]														Res.	Res.
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x0088	FDCAN_XIDFC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSE[7:0]								FLESA[13:0]														Res.	Res.
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		

Table 720. FDCAN register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0090	FDCAN_XIDAM	Res.	Res.	Res.	EIDM[28:0]																												
	Reset value				1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0094	FDCAN_HPMS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FLST	FIDX[6:0]							MISI [1:0]		BIDX[5:0]					
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0098	FDCAN_NDAT1	ND31	ND30	ND29	ND28	ND27	ND26	ND25	ND24	ND23	ND22	ND21	ND20	ND19	ND18	ND17	ND16	ND15	ND14	ND13	ND12	ND11	ND10	ND9	ND8	ND7	ND6	ND5	ND4	ND3	ND2	ND1	ND0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x009C	FDCAN_NDAT2	ND63	ND62	ND61	ND60	ND59	ND58	ND57	ND56	ND55	ND54	ND53	ND52	ND51	ND50	ND49	ND48	ND47	ND46	ND45	ND44	ND43	ND42	ND41	ND40	ND39	ND38	ND37	ND36	ND35	ND34	ND33	ND32
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00A0	FDCAN_RXF0C	F0OM	F0WM[6:0]							Res.	F0S[6:0]							F0SA[13:0]														Res.	Res.
	Reset value	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x00A4	FDCAN_RXF0S	Res.	Res.	Res.	Res.	Res.	Res.	RF0	F0F	Res.	Res.	F0PI[5:0]						Res.	Res.	F0GI[5:0]						Res.	RF0FL[6:0]						
	Reset value							0	0			0	0	0	0	0	0			0	0	0	0	0	0		0	0	0	0	0	0	0
0x00A8	FDCAN_RXF0A	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F0AI[5:0]					
	Reset value																											0	0	0	0	0	0
0x00AC	FDCAN_RXBC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RBSA[13:0]														Res.	Res.
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x00B0	FDCAN_RXF1C	F1OM	F1WM[6:0]							Res.	F1S[6:0]							F1SA[13:0]														Res.	Res.
	Reset value	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x00B4	FDCAN_RXF1S	DMS [1:0]		Res.	Res.	Res.	Res.	RF1	F1F	Res.	Res.	F1PI[5:0]						Res.	Res.	F1GI[5:0]						Res.	RF1FL[6:0]						
	Reset value	0	0					0	0			0	0	0	0	0	0			0	0	0	0	0	0		0	0	0	0	0	0	0
0x00B8	FDCAN_RXF1A	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	F1AI[5:0]					
	Reset value																											0	0	0	0	0	0
0x00BC	FDCAN_RXESC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RBDS [2:0]			Res.		F1DS [2:0]			Res.	F0DS [2:0]			
	Reset value																				0	0	0			0	0	0			0	0	0
0x00C0	FDCAN_TXBC	Res.	TFQM	TFQS[5:0]						Res.	Res.	NDTB[5:0]						TBSA[13:0]														Res.	Res.
	Reset value		0	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x00C4	FDCAN_TXFQS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFQF	TFQPI[4:0]					Res.	Res.	Res.	TFGI[4:0]					Res.	Res.	TFFL[5:0]					
	Reset value											0	0	0	0	0	0				0	0	0	0	0			0	0	0	0	0	0
0x00C8	FDCAN_TXESC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TBDS [2:0]		
	Reset value																														0	0	0

Table 720. FDCAN register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00CC	FDCAN _TXBRP	TRP31	TRP30	TRP29	TRP28	TRP27	TRP26	TRP25	TRP24	TRP23	TRP22	TRP21	TRP20	TRP19	TRP18	TRP17	TRP16	TRP15	TRP14	TRP13	TRP12	TRP11	TRP10	TRP9	TRP8	TRP7	TRP6	TRP5	TRP4	TRP3	TRP2	TRP1	TRP0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00D0	FDCAN _TXBAR	AR31	AR30	AR29	AR28	AR27	AR26	AR25	AR24	AR23	AR22	AR21	AR20	AR19	AR18	AR17	AR16	AR15	AR14	AR13	AR12	AR11	AR10	AR9	AR8	AR7	AR6	AR5	AR4	AR3	AR2	AR1	AR0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00D4	FDCAN _TXBCR	CR31	CR30	CR29	CR28	CR27	CR26	CR25	CR24	CR23	CR22	CR21	CR20	CR19	CR18	CR17	CR16	CR15	CR14	CR13	CR12	CR11	CR10	CR9	CR8	CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00D8	FDCAN _TXBTO	TO31	TO30	TO29	TO28	TO27	TO26	TO25	TO24	TO23	TO22	TO21	TO20	TO19	TO18	TO17	TO16	TO15	TO14	TO13	TO12	TO11	TO10	TO9	TO8	TO7	TO6	TO5	TO4	TO3	TO2	TO1	TO0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00DC	FDCAN _TXBCF	CF31	CF30	CF29	CF28	CF27	CF26	CF25	CF24	CF23	CF22	CF21	CF20	CF19	CF18	CF17	CF16	CF15	CF14	CF13	CF12	CF11	CF10	CF9	CF8	CF7	CF6	CF5	CF4	CF3	CF2	CF1	CF0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00E0	FDCAN _TXBTIE	TIE31	TIE30	TIE29	TIE28	TIE27	TIE26	TIE25	TIE24	TIE23	TIE22	TIE21	TIE20	TIE19	TIE18	TIE17	TIE16	TIE15	TIE14	TIE13	TIE12	TIE11	TIE10	TIE9	TIE8	TIE7	TIE6	TIE5	TIE4	TIE3	TIE2	TIE1	TIE0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00E4	FDCAN _TXBCIE	CFIE31	CFIE30	CFIE29	CFIE28	CFIE27	CFIE26	CFIE25	CFIE24	CFIE23	CFIE22	CFIE21	CFIE20	CFIE19	CFIE18	CFIE17	CFIE16	CFIE15	CFIE14	CFIE13	CFIE12	CFIE11	CFIE10	CFIE9	CFIE8	CFIE7	CFIE6	CFIE5	CFIE4	CFIE3	CFIE2	CFIE1	CFIE0
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00F0	FDCAN _TXEFC	Res.	Res.	EFWM[5:0]						Res.	Res.	EFS[5:0]						EFSA[15:2]														Res.	Res.
	Reset value			0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x00F4	FDCAN _TXEFS	Res.	Res.	Res.	Res.	Res.	Res.	TEF	EFF	Res.	Res.	Res.	EFPI[4:0]					Res.	Res.	Res.	EFGI[4:0]					Res.	Res.	EFFL[5:0]					
	Reset value							0	0				0	0	0	0	0				0	0	0	0	0			0	0	0	0	0	0
0x00F8	FDCAN _TXEFA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EFAI[4:0]				
	Reset value																												0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

71.6 TTCAN registers

These registers are available only for FDCAN1.

71.6.1 FDCAN TT trigger memory configuration register (FDCAN_TTTMC)

Address offset: 0x100

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TME[6:0]						
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TMSA[13:0]														Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:16 **TME[6:0]**: Trigger memory elements
0: No trigger memory
1-64: Number of trigger memory elements
>64: Values greater than 64 are interpreted as 64
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:2 **TMSA[13:0]**: Trigger memory start address.
Start address of trigger memory in message RAM (32-bit word address, see *Figure 960*).
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 1:0 Reserved, must be kept at reset value.

71.6.2 FDCAN TT reference message configuration register (FDCAN_TTRMC)

Address offset: 0x0104

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RMPS	XTD	Res.	RID[28:16]												
rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RID[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RMPS**: Reference message payload select
Ignored in case of time slaves.
0: Reference message has no additional payload
1: The following elements are taken from Tx buffer 0:
Message marker MM,
Event FIFO control EFC,
Data length code DLC,
Data Bytes DB (level 1: bytes 2-8, levels 0, 2: bytes 5-8)
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 30 **XTD**: Extended identifier
0: 11-bit standard identifier
1: 29-bit extended identifier
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 29 Reserved, must be kept at reset value.

Bits 28:0 **RID[28:0]**: Reference identifier.
Identifier transmitted with reference message and used for reference message filtering. Standard or extended reference identifier depending on bit XTD. A standard identifier must be written to ID[28:18].
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

71.6.3 FDCAN TT operation configuration register (FDCAN_TTOCF)

Address offset: 0x0108

Reset value: 0x0001 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	EVTP	ECC	EGTF	AWL[7:0]							
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EECS	IRTO[6:0]							LDSDL[2:0]			TM	GEN	Res.	OM[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw

Bits 31:27 Reserved, must be kept at reset value.

Bit 26 **EVTP**: Event trigger polarity.
0: Rising edge trigger
1: Falling edge trigger
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 25 **ECC**: Enable clock calibration.
0: Automatic clock calibration in FDCAN levels 0, 2 is disabled
1: Automatic clock calibration in FDCAN levels 0, 2 is enabled
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 24 **EGTF**: Enable global time filtering.
0: Global time filtering in FDCAN levels 0, 2 is disabled
1: Global time filtering in FDCAN levels 0, 2 is enabled
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 23:16 **AWL[7:0]**: Application watchdog limit.
The application watchdog can be disabled by programming AWL to 0x00.
0x00 to FF: Maximum time after which the application must serve the application watchdog. The application watchdog is incremented once each 256 NTUs.
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 15 **EECS**: Enable external clock synchronization
If enabled, TUR configuration (FDCAN_TURCF.NCL only) may be updated during FDCAN operation.
0: External clock synchronization in FDCAN levels 0, 2 disabled
1: External clock synchronization in FDCAN levels 0, 2 enabled
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 14:8 **IRTO[6:0]**: Initial reference trigger offset.
0x00 to 7F Positive offset, range from 0 to 127
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 7:5 **LDSDL[2:0]**: LD of synchronization deviation limit.
The synchronization deviation limit SDL is configured by its dual logarithm LDSDL with SDL = 2 * (LDSDL + 5). SDL is comprised between 32 and 4096. It must not exceed the clock tolerance given by the CAN bit timing configuration.
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 4 **TM**: Time master.
0: Time master function disabled
1: Potential time master
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 3 **GEN**: Gap enable.
0: Strictly time-triggered operation
1: External event-synchronized time-triggered operation
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **OM[1:0]**: Operation mode.
00: Event-driven CAN communication, default
01: TTCAN level 1
10: TTCAN level 2
11: TTCAN level 0
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

71.6.4 FDCAN TT matrix limits register (FDCAN_TTMLM)

Address offset: 0x010C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	ENTT[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TXEW[3:0]				CSS[1:0]		CCM[5:0]					
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **ENTT[11:0]**: Expected number of Tx triggers

0x000 to FFF Expected number of Tx triggers in one matrix cycle.

These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TXEW[3:0]**: Tx enable window

0x0 to F Length of Tx enable window, 1-16 NTU cycles

These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 7:6 **CSS[1:0]**: Cycle start synchronization

Enables sync pulse output.

00: No sync pulse

01: Sync pulse at start of basic cycle

10: Sync pulse at start of matrix cycle

11: Reserved

These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 5:0 **CCM[5:0]:** Cycle count Max

0x00: 1 basic cycle per matrix cycle

0x01: 2 basic cycles per matrix cycle

0x03: 4 basic cycles per matrix cycle

0x07: 8 basic cycles per matrix cycle

0x0F: 16 basic cycles per matrix cycle

0x1F: 32 basic cycles per matrix cycle

0x3F: 64 basic cycles per matrix cycle

Others: Reserved

These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Note: ISO 11898-4, Section 5.2.1 requires that only the listed cycle count values are configured. Other values are possible, but may lead to inconsistent matrix cycles.

71.6.5 FDCAN TUR configuration register (FDCAN_TURCF)

The length of the NTU is given by: NTU = CAN clock period x NC/DC.

NC is an 18-bit value. Its high part, NCH[17:16] is hard wired to 0b01. Therefore the range of NC extends from 0x10000 to 0x1FFFF. The value configured by NCL is the initial value for FDCAN_TURNA.NAV[15:0]. DC is set to 0x1000 by hardware reset and it may not be written to 0x0000.

- Level 1: NC 4 × DC and NTU = CAN bit time
- Levels 0 and 2: NC 8 × DC

The actual value of FDCAN_TUR may be changed by the clock drift compensation function of TTCAN level 0 and level 2 in order to adjust the node local view of the NTU to the time master view of the NTU. DC is not changed by the automatic drift compensation, FDCAN_TURNA.NAV may be adjusted around NC in the range of the synchronization deviation limit given by FDCAN_TTOCF.LDSDL. NC and DC must be programmed to the largest suitable values in achieve the best computational accuracy for the drift compensation process.

Address offset: 0x0110

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ELT	Res.	DC[13:0]													
rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NCL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **ELT**: Enable local time.

0: Local time is stopped, default

1: Local time is enabled

These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Note: The local time is started by setting ELT, it remains active until ELT is reset or until the next hardware reset. FDCAN_TURCF.DC is locked when FDCAN_TURCF.ELT = 1. If ELT is written to 0, the readable value stays at 1 until the new value has been synchronized into the CAN clock domain. During this time write access to the other bits of the register is locked.

Bit 30 Reserved, must be kept at reset value.

Bits 29:16 **DC[13:0]**: Denominator configuration.

0x0000: Illegal value

0x0001 to 0x3FFF: Denominator configuration

These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Bits 15:0 **NCL[15:0]**: Numerator configuration low.
Write access to the TUR numerator configuration low is only possible during configuration with FDCAN_TURCF.ELT = 0 or if FDCAN_TTOCF.EECS (external clock synchronization enabled) is set. When a new value for NCL is written outside TT configuration mode, the new value takes effect when FDCAN_TTOST.WECS is cleared to 0. NCL is locked FDCAN_TTOST.WECS is 1.
0x0000 to FFFF Numerator configuration low
These are write-protected bits, write access is possible only when bit CCE and bit INIT of FDCAN_CCCR register are set to 1.

Note: If NC < 7 × DC in TTCAN level 1, subsequent time marks in the trigger memory must differ by at least two NTUs.

71.6.6 FDCAN TT operation control register (FDCAN_TTOCN)

Address offset: 0x0114

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LCKC	Res.	ESCN	NIG	TMG	FGP	GCS	TTIE	TMC[1:0]		RTIE	SWS[1:0]		SWP	ECS	SGT
r		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **LCKC**: TT operation control register locked.
Set by a write access to register FDCAN_TTOCN. Reset when the updated configuration has been synchronized into the CAN clock domain.
0: Write access to FDCAN_TTOCN enabled
1: Write access to FDCAN_TTOCN locked

Bit 14 Reserved, must be kept at reset value.

Bit 13 **ESCN**: External synchronization control
If enabled the FDCAN synchronizes its cycle time phase to an external event signaled by a rising edge at event trigger pin (see *Section 71.4.18: Synchronization to external time schedule*).
0: External synchronization disabled
1: External synchronization enabled

Bit 12 **NIG**: Next is gap.
This bit can be set only when the FDCAN is the actual time master and when it is configured for external event-synchronized time-triggered operation (FDCAN_TTOCF.GEN = 1)
0: No action, reset by reception of any reference message
1: Transmit next reference message with Next_is_Gap = 1

Bit 11 **TMG**: Time mark gap.
0: Reset by each reference message
1: Next reference message started when register time mark interrupt FDCAN_TTIR.RTMI is activated

Bit 10 **FGP**: Finish gap.
Set by the CPU, reset by each reference message
0: No reference message requested
1: Application requested start of reference message

Bit 9 **GCS**: Gap control select
0: Gap control independent from event trigger
1: Gap control by input event trigger pin

Bit 8 **TTIE**: Trigger time mark interrupt pulse enable
External time mark events are configured by trigger memory element TMEX. A trigger time mark interrupt pulse is generated when the trigger memory element becomes active, and the FDCAN is in synchronization state In_Schedule or In_Gap.
0: Trigger time mark interrupt output fdcan1_tmp for more than one instance and fdcan_tmp if only one instance disabled
1: Trigger time mark interrupt output fdcan1_tmp for more than one instance and fdcan_tmp if only one instance enabled

Bits 7:6 **TMC[1:0]**: Register time mark compare.
00: No register time mark interrupt generated
01: Register time mark interrupt if time mark = cycle time
10: Register time mark interrupt if time mark = local time
11: Register time mark interrupt if time mark = global time

Note: When changing the time mark reference (cycle, local, global time), it is recommended to first write TMC = 00, then reconfigure FDCAN_TTTMK, and finally set FDCAN_TMC to the intended time reference.

Bit 5 **RTIE**: Register time mark interrupt pulse enable.
Register time mark interrupts are configured by register FDCAN_TTTMK. A register time mark interrupt pulse with the length of one fdcan_tq_ck period is generated when time referenced by FDCAN_TTOCN.TMC (cycle, local, or global) is equal to FDCAN_TTTMK.TM, independently from the synchronization state.
0: Register time mark interrupt output disabled
1: Register time mark interrupt output enabled

Bits 4:3 **SWS[1:0]**: Stop watch source.
00: Stop watch disabled
01: Actual value of cycle time is copied to FDCAN_TTCPT.SWV
10: Actual value of local time is copied to FDCAN_TTCPT.SWV
11: Actual value of global time is copied to FDCAN_TTCPT.SWV

Bit 2 **SWP**: Stop watch polarity.
0: Rising edge trigger
1: Falling edge trigger

Bit 1 **ECS**: External clock synchronization.
Writing a 1 to ECS sets FDCAN_TTOST.WECS if the node is the actual time master. ECS is reset after one APB clock period. The external clock synchronization takes effect at the start of the next basic cycle.

Bit 0 **SGT**: Set global time.
Writing a 1 to SGT sets FDCAN_TTOST.WGDT if the node is the actual time master. SGT is reset after one APB clock period. The global time preset takes effect when the node transmits the next reference message with the Master_Ref_Mark modified by the preset value written to FDCAN_TTGTP.

71.6.7 FDCAN TT global time preset register (FDCAN_TTGTP)

If TTOST.WGDT is set, the next reference message is transmitted with the Master_Ref_Mark modified by the preset value and with Disc_Bit = 1, presetting the global time in all nodes simultaneously.

TP is reset to 0x0000 each time a reference message with Disc_Bit = 1 becomes valid or if the node is not the current time master. TP is locked while FDCAN_TTOST.WGTD = 1 after setting FDCAN_TTOCN.SGT until the reference message with Disc_Bit = 1 becomes valid or until the node is no longer the current time master.

Address offset: 0x0118

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CTP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **CTP[15:0]**: Cycle time target phase
CTP is write-protected while FDCAN_TTOCN.ESCN or FDCAN_TTOST.SPL are set (see *Section 71.4.18: Synchronization to external time schedule*).
0x0000 to FFFF defines the target value of cycle time when a rising edge of event trigger is expected

Bits 15:0 **TP[15:0]**: Time preset
TP is write-protected while FDCAN_TTOST.WGTD is set.
0x0000 to 7FFF next master reference mark = master reference mark + TP
0x8000 reserved
0x8001–FFFF Next master reference mark = master reference mark - (0x10000 - TP).

71.6.8 FDCAN TT time mark register (FDCAN_TTTMK)

A time mark interrupt (FDCAN_TTIR.TMI = 1) is generated when the time base indicated by FDCAN_TTOCN.TMC (cycle time, local time, or global time) has the same value as TM.

Address offset: 0x011C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LCKM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TICC[6:0]						
r									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TM[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **LCKM**: TT time mark register locked

Always set by a write access to registers FDCAN_TTOCN. Set by write access to register FDCAN_TTTMK when FDCAN_TTOCN.TMC 00. Reset when the registers have been synchronized into the CAN clock domain.

0: Write access to FDCAN_TTTMK enabled

1: Write access to FDCAN_TTTMK locked

Bits 30:23 Reserved, must be kept at reset value.

Bits 22:16 **TICC[6:0]**: Time mark cycle code

Cycle count for which the time mark is valid.

0b000000x valid for all cycles

0b000001c valid every second cycle at cycle count mod2 = c

0b00001cc valid every fourth cycle at cycle count mod4 = cc

0b0001ccc valid every eighth cycle at cycle count mod8 = ccc

0b001cccc valid every sixteenth cycle at cycle count mod16 = cccc

0b01ccccc valid every thirty-second cycle at cycle count mod32 = ccccc

0b1cccccc valid every sixty-fourth cycle at cycle count mod64 = cccccc

Bits 15:0 **TM[15:0]**: Time mark

0x0000 to FFFF time mark

Note: When using byte access to register FDCAN_TTTMK it is recommended to first disable the time mark compare function (FDCAN_TTOCN.TMC = 00) to avoid comparisons on inconsistent register values.

71.6.9 FDCAN TT interrupt register (FDCAN_TTIR)

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the user clears them. A flag is cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect. A hard reset clears the register.

Address offset: 0x0120

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CER	AW	WT
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IWTG	ELC	SE2	SE1	TXO	TXU	GTE	GTD	GTW	SWE	TTMI	RTMI	SOG	CSM	SMC	SBC
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **CER**: Configuration error

Trigger out of order.

0: No error found in trigger list

1: Error found in trigger list

Bit 17 **AW**: Application watchdog

0: Application watchdog served in time

1: Application watchdog not served in time

Bit 16 **WT**: Watch trigger
0: No missing reference message
1: Missing reference message (level 0: cycle time 0xFF00)

Bit 15 **IWTG**: Initialization watch trigger
The initialization is restarted by resetting IWT.
0 No missing reference message during system startup
1 No system startup due to missing reference message

Bit 14 **ELC**: Error level changed
Not set when error level changed during initialization.
0: No change in error level
1: Error level changed

Bit 13 **SE2**: Scheduling error 2
0: No scheduling error 2
1: Scheduling error 2 occurred

Bit 12 **SE1**: Scheduling error 1
0: No scheduling error 1
1: Scheduling error 1 occurred

Bit 11 **TXO**: Tx count overflow
0: Number of Tx trigger as expected
1: More Tx trigger than expected in one cycle

Bit 10 **TXU**: Tx count underflow
0: Number of Tx trigger as expected
1: Less Tx trigger than expected in one cycle

Bit 9 **GTE**: Global time error
Synchronization deviation SD exceeds limit specified by FDCAN_TTOCF.LDSDL, TTCAN levels 0, 2 only.
0: Synchronization deviation within limit
1: Synchronization deviation exceeded limit

Bit 8 **GTD**: Global time discontinuity
0: No discontinuity of global time
1: Discontinuity of global time

Bit 7 **GTW**: Global time wrap
0: No global time wrap occurred
1: Global time wrap from 0xFFFF to 0x0000 occurred

Bit 6 **SWE**: Stop watch event
0: No rising/falling edge at stop watch trigger pin detected
1 :Rising/falling edge at stop watch trigger pin detected

Bit 5 **TTMI:** Trigger time mark event internal
Internal time mark events are configured by trigger memory element TMIN (see *Section 71.4.24*). Set when the trigger memory element becomes active, and the FDCAN is in synchronization state In_Gap or In_Schedule.
0: Time mark not reached
1: Time mark reached (level 0: cycle time FDCAN_TTOCF.RTO x 0x200)

Bit 4 **RTMI**: Register time mark interrupt
Set when time referenced by TTOCN.TMC (cycle, local, or global) is equal to FDCAN_TTTMK.TM, independently from the synchronization state.
0: Time mark not reached
1: Time mark reached

Bit 3 **SOG**: Start of gap
0 No reference message seen with Next_is_Gap bit set
1 Reference message with Next_is_Gap bit set becomes valid

Bit 2 **CSM**: Change of synchronization mode
0: No change in master to slave relation or schedule synchronization
1: Master to slave relation or schedule synchronization changed,
also set when FDCAN_TTOST.SPL is reset

Bit 1 **SMC**: Start of matrix cycle
0: No matrix cycle started since bit has been reset
1: Matrix cycle started

Bit 0 **SBC**: Start of basic cycle
0: No basic cycle started since bit has been reset
1: Basic cycle started

71.6.10 FDCAN TT interrupt enable register (FDCAN_TTIE)

The settings in the TT interrupt enable register determine which status changes in the TT interrupt register result in an interrupt.

Address offset: 0x0124

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CERE	AWE	WTE
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IWTE	ELCE	SE2E	SE1E	TXOE	TXUE	GTEE	GTDE	GTWE	SWEE	TTMIE	RTMIE	SOGE	CSME	SMCE	SBCE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **CERE**: Configuration error interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 17 **AWE**: Application watchdog interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 16 **WTE**: Watch trigger interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 15 **IWTE**: Initialization watch trigger interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 14 **ELCE**: Change error level interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 13 **SE2E**: Scheduling error 2 interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 12 **SE1E**: Scheduling error 1 interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 11 **TXOE**: Tx count overflow interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 10 **TXUE**: Tx count underflow interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 9 **GTEE**: Global time error interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 8 **GTDE**: Global time discontinuity interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 7 **GTWE**: Global time wrap interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 6 **SWEE**: Stop watch event interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 5 **TTMIE:** Trigger time mark event internal interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 4 **RTMIE**: Register time mark interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 3 **SOGE**: Start of gap interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 2 **CSME**: Change of synchronization mode interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 1 **SMCE**: Start of matrix cycle interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

Bit 0 **SBCE**: Start of basic cycle interrupt enable
0: TT interrupt disabled
1: TT interrupt enabled

71.6.11 FDCAN TT interrupt line select register (FDCAN_TTILS)

The TT interrupt line select register assigns an interrupt generated by a specific interrupt flag from the TT interrupt register to one of the two module interrupt lines. For interrupt generation the respective interrupt line must be enabled via FDCAN_ILE.EINT0 and FDCAN_ILE.EINT1.

Address offset: 0x0128

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CERL	AWL	WTL
													rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IWTL	ELCL	SE2L	SE1L	TXOL	TXUL	GTEL	GTDL	GTWL	SWEL	TTMIL	RTMIL	SOGL	CSML	SMCL	SBCL
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **CERL**: Configuration error interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 17 **AWL**: Application watchdog interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 16 **WTL:** Watch trigger interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 15 **IWTL:** Initialization watch trigger interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 14 **ELCL:** Change error level interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 13 **SE2L:** Scheduling error 2 interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 12 **SE1L:** Scheduling error 1 interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 11 **TXOL:** Tx count overflow interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 10 **TXUL:** Tx count underflow interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 9 **GTEL:** Global time error interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 8 **GTDL:** Global time discontinuity interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 7 **GTWL:** Global time wrap interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 6 **SWEL:** Stop watch event interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 5 TTMIL: Trigger time mark event internal interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 4 **RTMIL:** Register time mark interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 3 **SOGL:** Start of gap interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 2 **CSML:** Change of synchronization mode interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 1 **SMCL:** Start of matrix cycle interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

Bit 0 **SBCL:** Start of basic cycle interrupt line
0: TT interrupt assigned to interrupt line 0
1: TT interrupt assigned to interrupt line 1

71.6.12 FDCAN TT operation status register (FDCAN_TTOST)

Address offset: 0x012C

Reset value: 0x0000 0080

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SPL	WECS	AWE	WFE	GSI	TMP[2:0]			GFI	WGTD	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	r	r	r						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTO[7:0]								QCS	QGTP	SYS[1:0]		MS[1:0]		EL[1:0]	
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **SPL**: Schedule phase lock

The bit is valid only when external synchronization is enabled (FDCAN_TTOCN.ESCN = 1). In this case it signals that the difference between cycle time configured by FDCAN_TTGTP.CTP and the cycle time at the rising edge at event trigger pin is less than or equal to 9 NTU (see *Section 71.4.18: Synchronization to external time schedule*).

0: Phase outside range

1: Phase inside range

Bit 30 **WECS**: Wait for external clock synchronization.

0: No external clock synchronization pending

1: Node waits for external clock synchronization to take effect. The bit is reset at the start of the next basic cycle.

Bit 29 **AWE**: Application watchdog event

The application watchdog is served by reading FDCAN_TTOST. When the watchdog is not served in time, bit AWE is set, all FDCAN communication is stopped, and the FDCAN is set into bus monitoring mode.

0: Application watchdog served in time

1: Failed to serve application watchdog in time

Bit 28 **WFE**: Wait for event

0: No gap announced, reset by a reference message with Next_is_Gap = 0

1: Reference message with Next_is_Gap = 1 received

Bit 27 **GSI**: Gap started indicator

0: No gap in schedule, reset by each reference message and for all time slaves

1: Gap time after basic cycle has started

Bits 26:24 **TMP[2:0]**: Time master priority

0x0-7 Priority of actual time master

Bit 23 **GFI**: Gap finished indicator

Set when the CPU writes FDCAN_TTOCN.FGP, or by a time mark interrupt if TMG = 1, or via input pin (event trigger) if FDCAN_TTOCN.GCS = 1. Not set by Ref_Trigger_Gap or when Gap is finished by another node sending a reference message.

0: Reset at the end of each reference message

1: Gap finished by FDCAN

Bit 22 **WGTD**: Wait for global time discontinuity

0: No global time preset pending

1: Node waits for the global time preset to take effect. The bit is reset when the node has transmitted a reference message with Disc_Bit = 1 or after it received a reference message.

Bits 21:16 Reserved, must be kept at reset value.

Bits 15:8 **RTO[7:0]**: Reference trigger offset

The reference trigger offset value is a signed integer with a range from -127 (0x81) to 127 (0x7F). There is no notification when the lower limit of-127 is reached. In case the FDCAN becomes time master (MS[1:0] = 11), the reset of RTO is delayed due to synchronization between user and CAN clock domain. For time slaves the value configured by FDCAN_TTOCF.IRTO is read.

0x00-FF Actual reference trigger offset value

Bit 7 **QCS**: Quality of clock speed

Only relevant in TTCAN level 0 and level 2, otherwise fixed to 1.

0: Local clock speed not synchronized to time master clock speed

1: Synchronization deviation ≤ SDL

Bit 6 **QGTP**: Quality of global time phase
Only relevant in TTCAN level 0 and level 2, otherwise fixed to 0.
0: Global time not valid
1: Global time in phase with time master

Bits 5:4 **SYS[1:0]**: Synchronization state
00: Out of Synchronization
01: Synchronizing to FDCAN communication
10: Schedule suspended by gap (In_Gap)
11: Synchronized to schedule (In_Schedule)

Bits 3:2 **MS[1:0]**: Master state
00: Master_Off, no master properties relevant
01: Operating as time Slave
10: Operating as backup time master
11: Operating as current time master

Bits 1:0 **EL[1:0]**: Error level
00: Severity 0 - No error
01: Severity 1 - Warning
10: Severity 2 - error
11: Severity 3 - Severe error

71.6.13 FDCAN TUR numerator actual register (FDCAN_TURNA)

There is no drift compensation in TTCAN level 1 (NAV = NC). In TTCAN level 0 and level 2, the drift between the node local clock and the time master local clock is calculated. The drift is compensated when the synchronization deviation (difference between NC and the calculated NAV) is lower than 2 * (FDCAN_TTOCF.LDSDL + 5). With FDCAN_TTOCF.LDSDL < 7, this results in a maximum range for NAV of (NC - 0x1000) ≤ NAV ≤ (NC + 0x1000).

Address offset: 0x0130

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NAV[17:16]	
														r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NAV[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:0 **NAV[17:0]**: Numerator actual value
0x0EFFF: illegal value
0x0F000 to 20FFF: actual numerator value
0x21000: illegal value

71.6.14 FDCAN TT local and global time register (FDCAN_TTLGT)

Address offset: 0x0134

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **GT[15:0]**: Global time

Non-fractional part of the sum of the node local time and its local offset (see *Section 71.4.13*).
0x0000 to FFFF Global time value of FDCAN network

Bits 15:0 **LT[15:0]**: Local time

Non-fractional part of local time, incremented once each local NTU (see *Section 71.4.13*).
0x0000 to FFFF Local time value of FDCAN node

71.6.15 FDCAN TT cycle time and count register (FDCAN_TTCTC)

Address offset: 0x0138

Reset value: 0x003F 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC[5:0]					
										r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:16 **CC[5:0]**: Cycle count

0x00 to 3F Number of actual basic cycle in the system matrix

Bits 15:0 **CT[15:0]**: Cycle time

Non-fractional part of the difference of the node local time and Ref_Mark (see *Section 71.4.13*).
0x0000 to FFFF: cycle time value of FDCAN basic cycle

71.6.16 FDCAN TT capture time register (FDCAN_TTCPT)

Address offset: 0x013C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWV[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCV[5:0]					
										r	r	r	r	r	r

Bits 31:16 **SWV[15:0]**: Stop watch value

On a rising / falling edge (as configured via FDCAN_TTOCN.SWP) at the stop watch trigger pin, when FDCAN_TTOCN.SWS] is different from 00 and FDCAN_TTIR.SWE is 0, the actual time value as selected by FDCAN_TTOCN.SWS (cycle, local, global) is copied to SWV and TFDCAN_TIR.SWE is set to 1.Capturing of the next stop watch value is enabled by resetting FDCAN_TTIR.SWE.

0x0000 to FFFF Captured stop watch value

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:0 **CCV[5:0]**: Cycle count value

Cycle count value captured together with SWV.

0x00 to 3F: captured cycle count value

71.6.17 FDCAN TT cycle sync mark register (FDCAN_TTCSM)

Address offset: 0x0140

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CSM[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **CSM[15:0]**: Cycle sync mark

The cycle sync mark is measured in cycle time. It is updated when the reference message becomes valid and retains its value until the next reference message becomes valid.

0x0000 to FFFF Captured cycle time

71.6.18 FDCAN TT trigger select register (FDCAN_TTTS)

The settings in the FDCAN_TTTS register select the input to be used as event trigger and stop watch trigger.

Address offset: 0x0300

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EVTSEL[1:0]		Res.	Res.	SWTDEL[1:0]	
										rw	rw			rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:4 **EVTSEL[1:0]**: Event trigger input selection
These bits are used to select the input to be used as event trigger
00: fdcan1_evt0
01: fdcan1_evt1
10: fdcan1_evt2
11: fdcan1_evt3

Bits 3:2 Reserved, must be kept at reset value.

Bits 1:0 **SWTDEL[1:0]**: Stop watch trigger input selection
These bits are used to select the input to be used as stop watch trigger
00: fdcan1_swt0
01: fdcan1_swt1
10: fdcan1_swt2
11: fdcan1_swt3

71.6.19 FDCAN TT register map

Table 721. FDCAN TT register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0100	FDCAN _TTTMC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TME[6:0]							TMSA[13:0]														Res.	Res.
	Reset value										0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
0x0104	FDCAN _TTRMC	RMPS	XTD	Res.	RID[28:0]																												
	Reset value	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0108	FDCAN _TTOCF	Res.	Res.	Res.	Res.	Res.	EVTP	ECC	EGTF	AWL[7:0]								EECS	IRTO[6:0]							LDSDL [2:0]			TM	GEN	Res.	OM [1:0]	
	Reset value						0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0
0x010C	FDCAN _TTMLM	Res.	Res.	Res.	Res.	ENTT[11:0]												Res.	Res.	Res.	Res.	TXEW[3:0]				CSS [1:0]		CCM[5:0]					
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 721. FDCAN TT register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0110	FDCAN_TURCF	ELT	Res.	DC[13:0]														NCL[15:0]															
	Reset value	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0114	FDCAN_TTOCN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LCKC	Res.	ESCN	NIG	TMG	FGP	GCS	TTIE	TMC[1:0]		RTIE	SWS[1:0]		SWP	ECS	SGT
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0118	FDCAN_TTGTP	CTP[15:0]																TP[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x011C	FDCAN_TTTMK	LCKM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TICC[6:0]							TM[15:0]															
	Reset value	0									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0120	FDCAN_TTIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CER	AW	WT	IWTG	ELC	SE2	SE1	TXO	TXU	GTE	GTD	GTW	SWE	TTMI	RTMI	SOG	CSM	SMC	SBC
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0124	FDCAN_TTIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CERE	AWE	WTE	IWTE	ELCE	SE2E	SE1E	TXOE	TXUE	GTEE	GTDE	GTWE	SWEE	TTMIE	RTMIE	SOGE	CSME	SMCE	SBCE
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0128	FDCAN_TTILS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CERL	AWL	WTL	IWTL	ELCL	SE2L	SE1L	TXOL	TXUL	GTEL	GTDL	GTWL	SWEL	TTMIL	RTMIL	SOGL	CSML	SMCL	SBCL
	Reset value														0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x012C	FDCAN_TTOST	SPL	WECS	AWE	WFE	GSI	TMP [2:0]			GFI	WGTD	Res.	Res.	Res.	Res.	Res.	Res.	RTO[7:0]								QCS	QGTP	SYS [1:0]		MS [1:0]		EL [1:0]	
	Reset value	0	0	0	0	0	0	0	0	0	0							0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0x0130	FDCAN_TURNA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NAV[17:0]																	
	Reset value															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0134	FDCAN_TTLGT	GT[15:0]																LT[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0138	FDCAN_TTCTC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC[5:0]						CT[15:0]															
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x013C	FDCAN_TTCPT	SWV[15:0]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CCV[5:0]					
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0											0	0	0	0	0	0
0x0140	FDCAN_TTCSM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSM[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0144 to 0x01FC	Reserved																																
	Reset value																																

Table 721. FDCAN TT register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0300	FDCAN_TTTS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EVTSEL [1:0]		Res.	Res.	SWTSEL [1:0]	
	Reset value																											0	0			0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

71.7 CCU registers

71.7.1 Clock calibration unit core release register (FDCAN_CCU_CREL)

Address offset: 0x0000

Reset value: 0x1114 1218

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REL[3:0]				STEP[3:0]				SUBSTEP[3:0]				YEAR[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MON[7:0]								DAY[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 **REL[3:0]**: Core release = 1

Bits 27:24 **STEP[3:0]**: Step of core release = 1

Bits 23:20 **SUBSTEP[3:0]**: Sub-step of core release = 1

Bits 19:16 **YEAR[3:0]**: Timestamp year = 4

Bits 15:8 **MON[7:0]**: Timestamp month = 12

Bits 7:0 **DAY[7:0]**: Timestamp day = 18

71.7.2 Calibration configuration register (FDCAN_CCU_CCFG)

Address offset: 0x0004

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SWR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CDIV[3:0]			
rw												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OCPM[7:0]								CFL	BCC	Res.	TQBT[4:0]				
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw

Bit 31 **SWR**: Software reset

Writing a 1 to this bit resets the calibration FSM to state Not_Calibrated (FDCAN_CCU_CSTAT.CALS = 00). The calibration watchdog value CWD.WDV is also reset. Registers FDCAN_CCFG, FDCAN_CCU_CSTAT and the calibration watchdog configuration CWD.WDC are unchanged. The bit remains set until reset is completed.

Write access by the user to registers/bits marked with “P = Protected Write” is possible only when FDCAN control bits FDCAN_CCCR.CCE = 1 AND FDCAN_CCCR.INIT = 1.

Bits 30:20 Reserved, must be kept at reset value.

Bits 19:16 **CDIV[3:0]**: Clock divider

The clock divider must be configured when the clock calibration is bypassed (BCC = 1) to ensure that the FDCAN requirement is fulfilled.

0000: Divide by 1
0001: Divide by 2
0010: Divide by 4
0011: Divide by 6
0100: Divide by 8
0101: Divide by 10
0110: Divide by 12
0111: Divide by 14
1000: Divide by 16
1001: Divide by 18
1010: Divide by 20
1011: Divide by 22
1100: Divide by 24
1101: Divide by 26
1110: Divide by 28
1111: Divide by 30

Write access by the user to registers/bits marked with “P = Protected Write” is possible only when FDCAN control bit FDCAN_CCCR.CCE = 1.

Bits 15:8 **OCPM[7:0]**: Oscillator clock periods minimum

Configures the minimum number of periods in two CAN bit times. OCPM is used in basic calibration to avoid false measurements in case of glitches on the bus line. The configured number of periods is OCPM × 32. The configuration depends on the frequency and the bitrate configured in FDCAN modules (from 125 kbit/s up to 1 Mbit/s). It is recommended to configure a value slightly below two CAN bit times. The reset value is 1.6 bit times at 80 MHz fdcan_ker_ck and 1 Mbit/s CAN bitrate.

Write access by the user to registers/bits marked with “P = Protected Write” is possible only when FDCAN control bits FDCAN_CCCR.CCE = 1 AND FDCAN_CCCR.INIT = 1.

Bit 7 **CFL**: Calibration field length

0: Calibration field length is 32 bits
1: Calibration field length is 64 bits

Write access by the user to registers/bits marked with “P = Protected Write” is possible only when FDCAN control bits FDCAN_CCCR.CCE = 1 AND FDCAN_CCCR.INIT = 1.

Bit 6 **BCC**: Bypass clock calibration

If this bit is set, the clock input fdcan_ker_ck is routed to the time quanta clock through a clock divider configurable via CDIV. In this case the baudrate prescaler of the connected FDCANs must be configured to generate the FDCAN internal time quanta clock.

0: Clock calibration unit generates time quanta clock
1: Clock calibration unit bypassed (default configuration)

Bit 5 Reserved, must be kept at reset value.

Bits 4:0 **TQBT[4:0]**: Time quanta per bit time

Configures the number of time quanta per bit time. Same value as configured in FDCAN modules. The range of the resulting time quanta clock fdcan_tq_ck is from 0.5 MHz (bitrate of 125 kbit/s with 4 tq per bit time) to 25 MHz (bitrate of 1 Mbit/s with 25 tq per bit time). Valid values are 4 to 25. Configured values below 4 are interpreted as 4, values above 25 are interpreted as 25.
Write access by the user to registers/bits marked with “P = Protected Write” is possible only when FDCAN control bits FDCAN_CCCR.CCE = 1 AND FDCAN_CCCR.INIT = 1.

71.7.3 Calibration status register (FDCAN_CCU_CSTAT)

Address offset: 0x0008

Reset value: 0x0203 FFFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CALS[1:0]		Res.	TQC[10:0]											OCPC[17:16]	
r	r		r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OCPC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:30 **CALS[1:0]**: Calibration state

00: Not_Calibrated
01: Basic_Calibrated
10: Precision_Calibrated
11: Reserved

Bit 29 Reserved, must be kept at reset value.

Bits 28:18 **TQC[10:0]**: Time quanta counter

Captured number of time quanta in calibration field (32 or 64 bits). Only valid when the clock calibration unit is in state Precision_Calibrated.

Bits 17:0 **OCPC[17:0]**: Oscillator clock period counter

Captured number of oscillator clock periods in calibration field (32 or 64 bits). Only valid when the clock calibration unit is in state Precision_Calibrated.

71.7.4 Calibration watchdog register (FDCAN_CCU_CWD)

Address offset: 0x000C

Reset value: 0x0000 0000

The calibration watchdog is started after the first falling edge when the calibration FSM is in state Not_Calibrated (FDCAN_CCU_CSTAT.CALS = 00). In this state the calibration watchdog monitors the message received. In case no message was received until the calibration watchdog has counted down to 0, the calibration FSM stays in state Not_Calibrated (FDCAN_CCU_CSTAT.CALS = 00), the counter is reloaded with FDCAN_RWD.WDC and basic calibration is restarted after the next falling edge.

When in state Basic_Calibrated (FDCAN_CCU_CSTAT.CALS = 01), the calibration watchdog is restarted with each received message . In case no message was received until the calibration watchdog has counted down to 0, the calibration FSM returns to state

Not_Calibrated (FDCAN_CCU_CSTAT.CALS = 00), the counter is reloaded with FDCAN_RWD.WDC and basic calibration is restarted after the next falling edge.

When a quartz message is received, state Precision_Calibrated (FDCAN_CCU_CSTAT.CALS = 10) is entered and the calibration watchdog is restarted. In this state the calibration watchdog monitors the quartz message received input. In case no message from a quartz controlled node is received by the attached TTCAN until the calibration watchdog has counted down to 0, the calibration FSM transits back to state Basic_Calibrated (FDCAN_CCU_CSTAT.CALS = 01). The signal is active when the CAN protocol engine on the attached TTCAN is started i.e. when the INIT bit is reset.

A calibration watchdog event also sets interrupt flag FDCAN_CCU_IR.CWE. If enabled by FDCAN_CCU_IE.CWEE, the interrupt line is activated (set to high). Interrupt line remains active until the interrupt flag FDCAN_CCU_IR.CWE is reset.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WDV[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WDC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **WDV[15:0]**: Watchdog value

Actual calibration watchdog counter value.

Bits 15:0 **WDC[15:0]**: WDC

Watchdog configuration

Start value of the calibration watchdog counter. With the reset value of 00 the counter is disabled.

Write access by the user to registers/bits marked with “P = Protected Write” is possible only when FDCAN control bits FDCAN_CCCR.CCE = 1 AND FDCAN_CCCR.INIT = 1.

71.7.5 Clock calibration unit interrupt register (FDCAN_CCU_IR)

The flags are set when one of the listed conditions is detected (edge-sensitive). The flags remain set until the user clears them. A flag is cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect. A hard reset clears the register. The configuration of FDCAN_CCU_IE controls whether an interrupt is generated or not.

Address offset: 0x0010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSC	CWE
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **CSC**: Calibration state changed
0: Calibration state unchanged
1: Calibration state has changed

Bit 0 **CWE**: Calibration watchdog event
0: No calibration watchdog event
1: Calibration watchdog event occurred

71.7.6 Clock calibration unit interrupt enable register (FDCAN_CCU_IE)

Address offset: 0x0014

Reset value: 0x0000 0000

The settings in the CU interrupt enable register determine whether a status change in the CU interrupt register is signaled on an interrupt line.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSCE	CWEE
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **CSCE**: Calibration state changed enable
0: Interrupt disabled
1: Interrupt enabled

Bit 0 **CWEE**: Calibration watchdog event enable
0: Interrupt disabled
1: Interrupt enabled

71.7.7 CCU register map

Table 722. CCU register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	FDCAN_CCU_CREL	REL[3:0]				STEP[3:0]				SUBSTEP[3:0]				YEAR[3:0]				MON[7:0]								DAY[7:0]							
	Reset value	0	0	0	1	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	1	1	0	0	0
0x0004	FDCAN_CCU_CCFG	SWR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CDIV[3:0]				OCPM[7:0]								CFL	BCC	Res.	TQBT[4:0]				
	Reset value	0												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0008	FDCAN_CCU_CSTAT	CALS[1:0]		Res.	TQC[10:0]													OCPC[17:0]															
	Reset value	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x000C	FDCAN_CCU_CWD	WDV[15:0]																WDC[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 722. CCU register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0010	FDCAN_ CCU_IR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSC	CWE
	Reset value																															0	0
0x0014	FDCAN_ CCU_IE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSCE	CWEE
	Reset value																															0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

72 USB subsystem (USBSS)

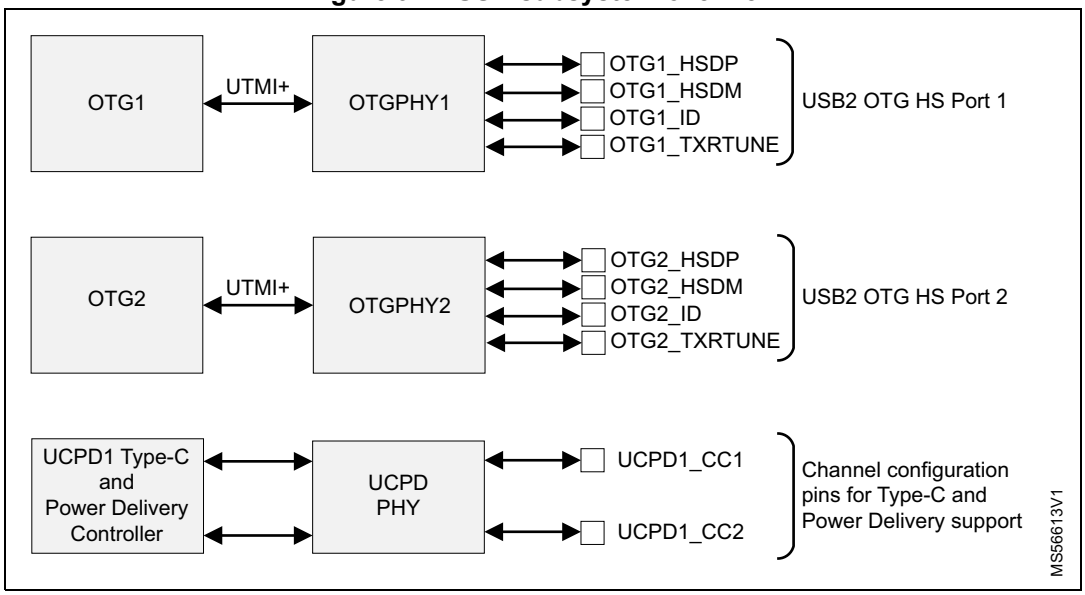
72.1 Overview

The USB subsystem includes two USB2 OTG high-speed ports and two channel configuration lines for Type-C and power delivery support. A high level block diagram is shown in *Figure 972*.

The USB2 OTG high-speed ports support Full Speed and Low Speed modes when acting as Host, and Full Speed mode when acting as Device.

The configuration channels support Type-C and power delivery features in applications where one of the USB2 port is connected to a Type-C receptacle.

Figure 972. USB subsystem overview



72.2 USB2 OTG high-speed Port 1

72.2.1 USB2 OTG high-speed Port 1 main features

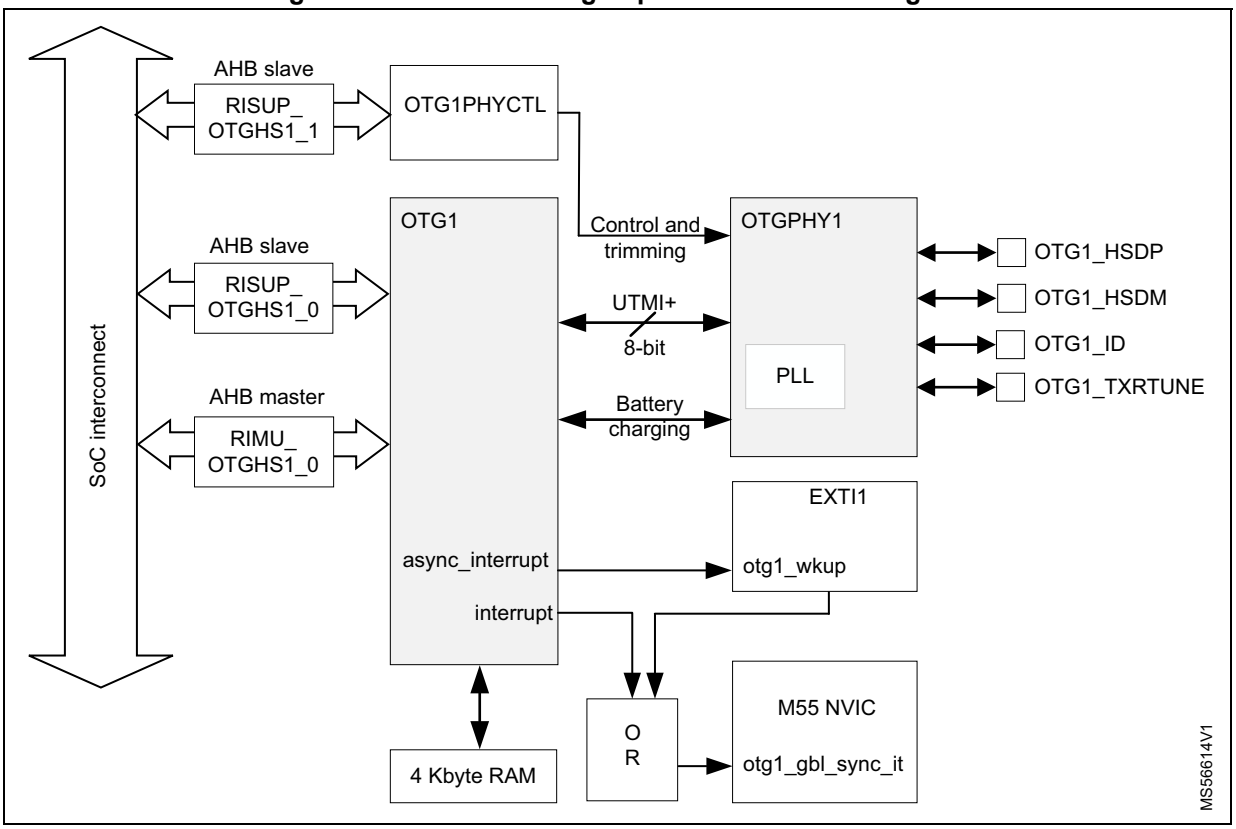
- Can be configured statically as Host or Device (DRD configuration)
- Dynamic switching between the two roles is not supported: Host Negotiation Protocol (HNP) and Session Request Protocol (SRP) are not supported
- Embedded PHY, supporting high-speed (HS), Full Speed (FS) and Low Speed (LS) modes (LS mode is not supported when configured as Device)
- LPM (Link Power Management) support
- DMA engine with local 4-Kbyte buffer
- Supports Battery Charging v1.2, with the exception of the Accessory Charger Adapter mode
- Supports suspend, sleep, resume, and remote wake-up operations
- Provides electrical parameter override bits for optimal interoperability

The supported standards are:

- Universal Serial Bus Specification, Revision 2.0, USB Implementers Forum, Inc., April 27, 2000
- Engineering Change Notice: USB2.0 Link Power Management Addendum, July 16, 2007
- Battery Charging Specification, Revision 1.2, December 7, 2010

72.2.2 USB2 OTG high-speed Port 1 functional description

Figure 973. USB2 OTG high-speed Port 1 block diagram



RISUP_OTGHS1_0 and RISUP_OTGHS1_1 correspond to the same RIFSC index, namely OTG1_HS. They are connected to the same sec/priv bits from the RIFSC, and their ilac outputs are OR-ed together, to go to the same IAC index.

USB2 OTG high-speed port pins and internal signals

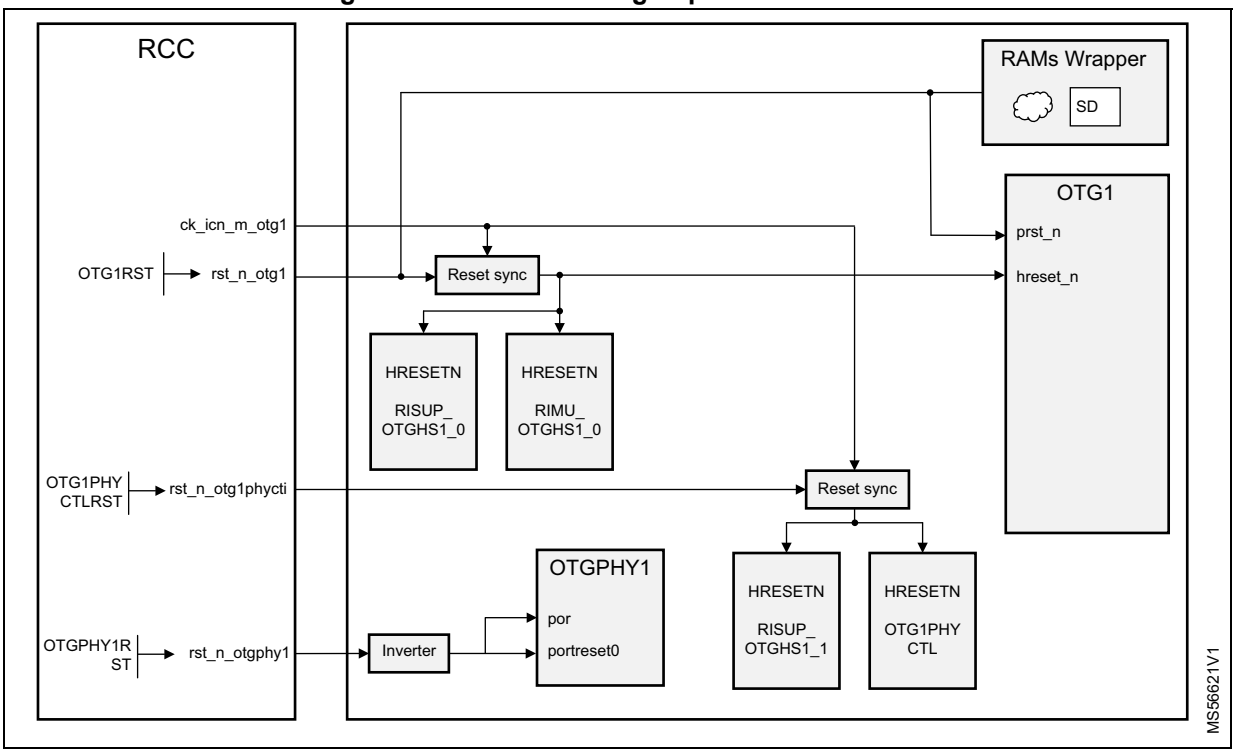
Table 723. USB2 OTG high-speed Port 1 pins

Signal name	Signal type	Description
OTG1_HSDP	3V3 analog	USB D+ signal
OTG1_HSDM		USB D- signal
OTG1_ID	1V8 analog	USB Mini-Receptacle Identifier. Differentiates a mini-A from a mini-B plug.
OTG1_TXRTUNE		Transmitter resistor tune pin. Connects to GND through an external 200±1% Ω resistor that adjusts the high-speed source impedance of OTGPHY1.

Table 724. USB2 OTG high-speed Port 1 internal signals

Signal name	Signal type	Description
OTG1 AHB Master	Digital	AHB Master interface to the SoC interconnect for memory access. AHBLite protocol, 32-bit data width.
OTG1 AHB Slave	Digital	AHB Slave interface to access OTG1 registers. AHBLite protocol, 32-bit data width.
OTG1PHYCTL AHB Slave	Digital	AHB Slave interface for access to OTG1PHYCTL registers.

Figure 974. USB2 OTG high-speed Port 1 resets



There are three resets for the USB2 OTG high-speed Port 1:

- rst_n_otg1, controlled by the OTG1RST bit in RCC_AHB5RSTR register.
- rst_n_otg1phyctl, controlled by the OTG1PHYCTLRST bit in RCC_AHB5RSTR register.
- rst_n_otgphy1, controlled by the OTGPHY1RST bit in RCC_AHB5RSTR register.

When configuring the USB2 OTG high-speed Port 1, the reset sequence is:

1. rst_n_otgphy1, rst_n_otg1 and rst_n_otg1phyctl resets are asserted
2. clocks coming from RCC and HSE are enabled
3. rst_n_otg1phyctl reset is released
4. the frequency of OTGPHY1 reference clock is selected in OTG1PHYCTL_CR register
5. OTGPHY1 trimming bits are programmed (if needed)
6. rst_n_otgphy1 is released
7. wait 50 µs (OTGPHY1 PLL lock time and start of generation of phyclock0)
8. rst_n_otg1 is released

Description of the USB2 OTG high-speed controller (OTG1)

Refer to *Section 73: USB on-the-go high-speed (OTG)*.

Description of the OTGPHY1

OTGPHY1 is controlled by OTG1PHYCTL registers. Refer to *Section 74: USB HS PHY controller (USBPHYC)*.

72.2.3 USB2 OTG high-speed Port 1 interrupts

The USB2 OTG high-speed Port 1 generates one interrupt and one wake-up event that can be used to take the system out of Stop mode.

Table 725. USB2 OTG high-speed Port 1 interrupt

OTG1 interrupt	IP pin name	CPU interrupt
OTG1_GBL_SYNC_IT	OTG1/interrupt	OTG1_GBL_SYNC_IT

Table 726. USB2 OTG high-speed Port 1 wake-up events

OTG1 event	IP pin name	EXTI event
OTG1_WKUP	OTG1/async_interrupt	OTG1_WKUP

72.2.4 Battery charging

The OTGPHY1 integrates the voltage and current sources, as well as the voltage comparators required to support Battery Charging v1.2. The relevant control bits are implemented in OTG_GCCFG register of the OTG1 controller.

The control sequence depends on whether the OTG HS port 1 is Host or Device.

Host case

If the application wants the Host port to behave as a Standard Downstream Port (SDP), the default values of H_CDPEN, H_CDPDETEN, H_VDMSRCEN bits in OTG_GCCFG register must be kept.

If the application wants the Host port to behave as a Charging Downstream Port (CDP), one of the following two control sequences must be implemented:

Control sequence 1:

- Set the H_CDPEN bit to 1
- Decode the disconnect information transmitted by the OTGPHY1 to the OTG1 controller (HOSTDISCONNECT0 output of OTGPHY1).
  - If disconnect = 1, set the H_VDMSRCEN bit to 1
  - else set the H_VDMSRCEN bit to 0

Control sequence 2:

- Set the H_CDPEN bit to 1 and the H_CDPDETEN bit to 1
- Monitor CHGDET and FSVPLUS status bits:
  - If CHGDET = 1 and FSVPLUS = 0, set the H_VDMSRCEN bit to 1
  - else, set the H_VDMSRCEN bit to 0

Device case

- Data contact detection
  - Set the PD_DCDET bit to 1. This enables a current source on OTG1_HSDP and pulls OTG1_HSDM low.
  - If FSVPLUS is at 0 for more than tDCD_DBNC (as defined in Battery Charging v1.2), Data Contact has happened. PD_DCDET must be reset.
- Primary detection
  - Set the PD_PDET bit to 1. This enables a voltage source on OTG1_HSDP and a voltage comparator on OTG1_HSDM.
  - If CHGDET = 0, the Device is connected to a Standard Downstream Port (SDP).
  - If CHGDET = 1, the Device is connected to a Charging Downstream Port (CDP) or a Dedicated Charging Port (DCP).
  - Reset PD_PDET.
- Secondary detection
  - Set the PD_SDET bit to 1. This enables a voltage source on OTG1_HSDM and a voltage comparator on OTG1_HSDP.
  - If CHGDET = 0, the Device is connected to a Charging Downstream Port (CDP).
  - If CHGDET = 1, the Device is connected to a Dedicated Charging Port (DCP).
  - Reset PD_SDET.

72.2.5 ID management

If IDPULLUP_DIS control bit in OTG_GCCFG register is 0, the ID information is captured on OTG1_ID pin and provided by the OTGPHY1 to the OTG1 controller. If the OTG1_ID pin is unused and left floating, PULLDOWN_EN bit in the OTG_GCCFG register of OTG1 can be used to control the pull-down resistors on OTG1_HSDP and OTG1_HSDM pins.

72.2.6 VBUS detection

OTGPHY1 is configured in VBUS external comparator mode. As a consequence:

- OTGDISABLE0 and DRVVBUS0 bits of OTG1PHYCTL_CR register must be kept to their reset values.
- VBUSVLD and SESSVLD bits of OTG_GCCFG register must be ignored.

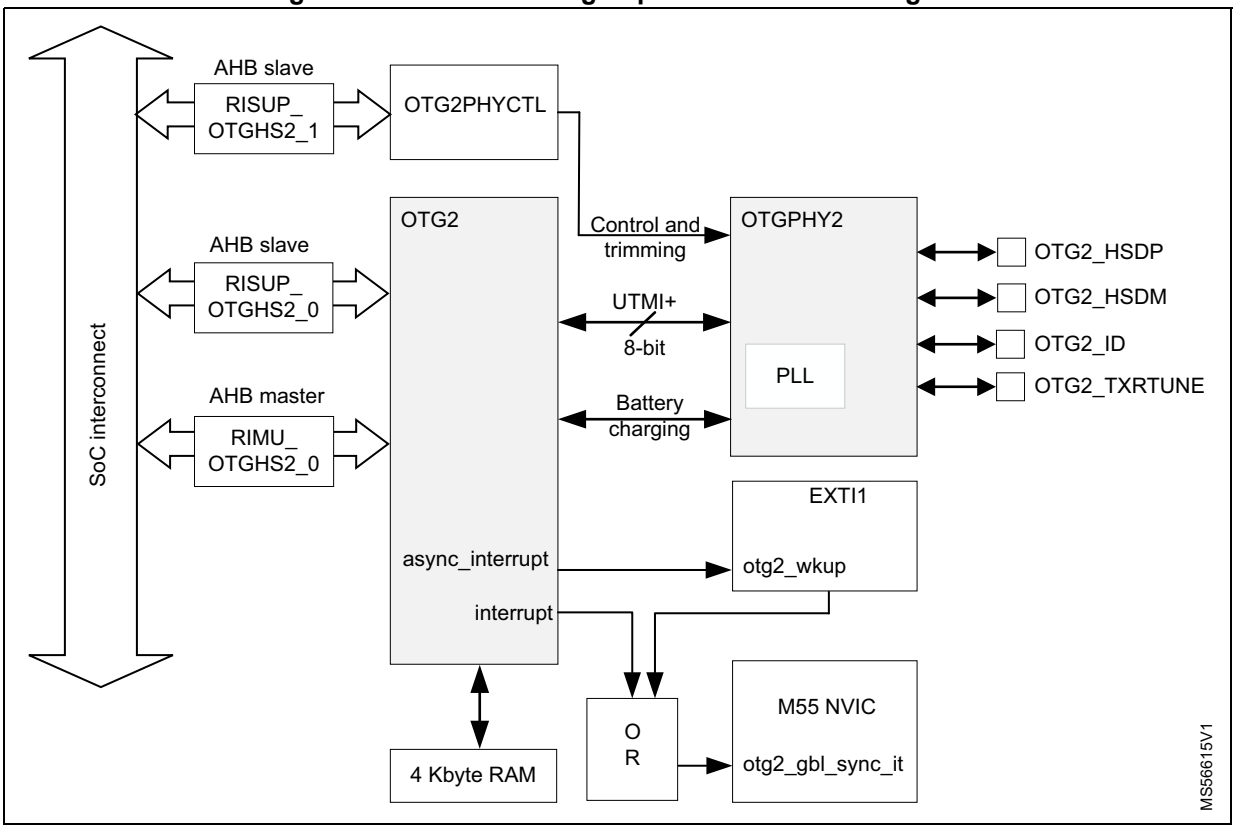
72.3 USB2 OTG high-speed Port 2

72.3.1 USB2 OTG high-speed Port 2 main features

Features are identical to those of the USB2 OTG high-speed Port 1.

72.3.2 USB2 OTG high-speed Port 2 functional description

Figure 975. USB2 OTG high-speed Port 2 block diagram



RISUP_OTGHS2_0 and RISUP_OTGHS2_1 correspond to the same RIFSC index (OTG2_HS). They are connected to the same sec/priv bits from the RIFSC, and their ilac outputs are OR-ed together, to go to the same IAC index.

Table 727. USB2 OTG high-speed Port 2 pins

Signal name	Signal type	Description
OTG2_HSDP	3V3 analog	USB D+ signal
OTG2_HSDM		USB D- signal
OTG2_ID	1V8 analog	USB Mini-Receptacle identifier. Differentiates a mini-A from a mini-B plug.
OTG2_TXRTUNE		Transmitter resistor tune pin. Connects to GND through an external 200 200±1% Ω resistor that adjusts the high-speed source impedance of OTGPHY2.

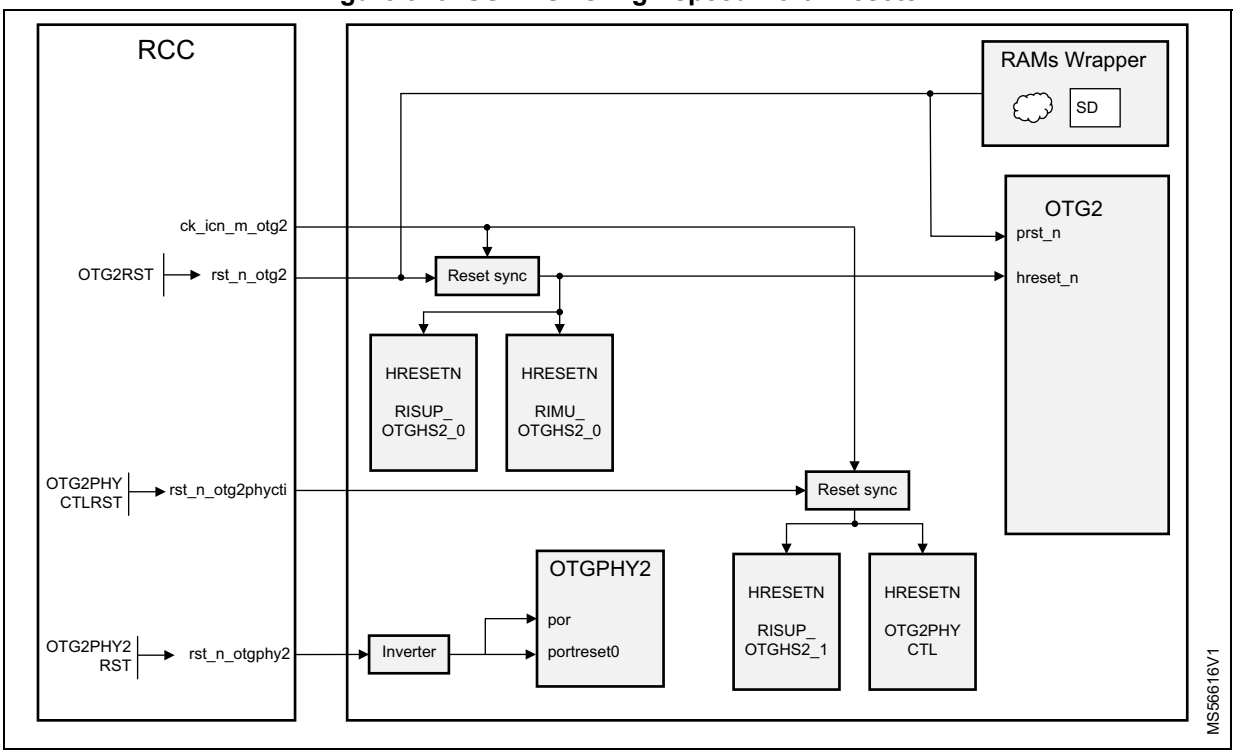
Table 728. USB2 OTG high-speed Port 2 internal signals

Signal name	Signal type	Description
OTG2 AHB Master	Digital	AHB Master interface to the SoC interconnect for memory access. AHBLite protocol, 32-bit data width.

Table 728. USB2 OTG high-speed Port 2 internal signals (continued)

Signal name	Signal type	Description
OTG2 AHB Slave	Digital	AHB Slave interface for access to OTG2 registers. AHBLite protocol, 32-bit data width.
OTG2PHYCTL AHB Slave	Digital	AHB Slave interface for access to OTG2PHYCTL registers.

Figure 976. USB2 OTG high-speed Port 2 resets



There are three resets for the USB2 OTG high-speed Port 2:

- rst_n_otg2, controlled by the OTG2RST bit in RCC_AHB5RSTR register.
- rst_n_otg2phyctl, controlled by the OTG2PHYCTLRST bit in RCC_AHB5RSTR register.
- rst_n_otgphy2, controlled by the OTGPHY2RST bit in RCC_AHB5RSTR register.

When configuring the USB2 OTG high-speed Port 2, the reset sequence is:

1. rst_n_otgphy2, rst_n_otg2 and rst_n_otg2phyctl resets are asserted.
2. clocks coming from RCC and HSE are enabled.
3. rst_n_otg2phyctl reset is released
4. the frequency of OTGPHY2 reference clock is selected in OTG2PHYCTL_CR register.
5. OTGPHY2 trimming bits are programmed (if needed)
6. rst_n_otgphy2 is released
7. wait 50 µs (OTGPHY2 PLL lock time and start of generation of phyclock0)
8. rst_n_otg2 is released

Description of the USB2 OTG HS controller (OTG2)

Refer to *Section 73: USB on-the-go high-speed (OTG)*.

Description of the OTGPHY2

OTGPHY2 is controlled by OTG2PHYCTL registers. Refer to *Section 74: USB HS PHY controller (USBPHYC)*.

72.3.3 USB2 OTG high-speed Port 2 interrupts

The USB2 OTG high-speed Port 2 generates one interrupt and one wake-up event, which can be used to take the system out of Stop mode.

Table 729. USB2 OTG high-speed Port 2 interrupts

OTG2 interrupt	IP pin name	CPU interrupt
OTG2_GBL_SYNC_IT	OTG2/interrupt	OTG2_GBL_SYNC_IT

Table 730. USB2 OTG high-speed Port 2 wake-up events

OTG2 event	IP pin name	EXTI event
OTG2_WKUP	OTG2/async_interrupt	OTG2_WKUP

72.3.4 Battery charging

The OTGPHY2 integrates the voltage and current sources, as well as the voltage comparators required to support Battery Charging v1.2. The relevant control bits are implemented in OTG_GCCFG register of OTG2.

The control sequence is the same as for the USB2 OTG high-speed Port 1.

72.3.5 ID management

This is similar to the ID management for OTGPHY1. The control bits are in the OTG_GCCFG register of OTG2.

72.3.6 VBUS management

OTGPHY2 is configured in ‘VBUS external comparator’ mode. As a consequence:

- OTGDISABLE0 and DRVVBUS0 bits in OTG2PHYCTL_CR register must be kept to their reset values.

The bits VBUSVLD and SESSVLD of OTG_GCCFG register must be ignored.

72.4 USB Type-C and power delivery interface

72.4.1 USB Type-C and power delivery interface main features

- The USB Type-C and power delivery interface is made of CC1 and CC2 channel configuration pins. It enables the implementation of one Type-C port, associated to OTG HS Port 1 or to OTG HS Port 2.
- It supports the CC signaling of the Type-C specification:
  - attachment detection
  - cable orientation resolution
  - source/sink role advertisement
  - detection of current modes
- It supports the physical layer of the power delivery specification and enables power negotiation, power role swap and data role swap.
- Generation of a Fast Role Swap request is supported: FRSTX1 and FRSTX2 signals from UCPD1 IP are mapped as alternate functions of GPIOs for control of external NMOS transistors.
- detection of a Fast Role Swap request is not supported
- ADC channel are used for VBUS comparison with thresholds defined in the standard
- Support for sink, source or dual role power (DRP) platforms.
- Independent DMA interface for Tx and Rx functionality.
- Not supported: Dead battery support, Over voltage protection on CC signals and VBUS, Over current detection on VBUS, VCONN switches, VBUS switch gate control and discharge. These features must be supported with external components.

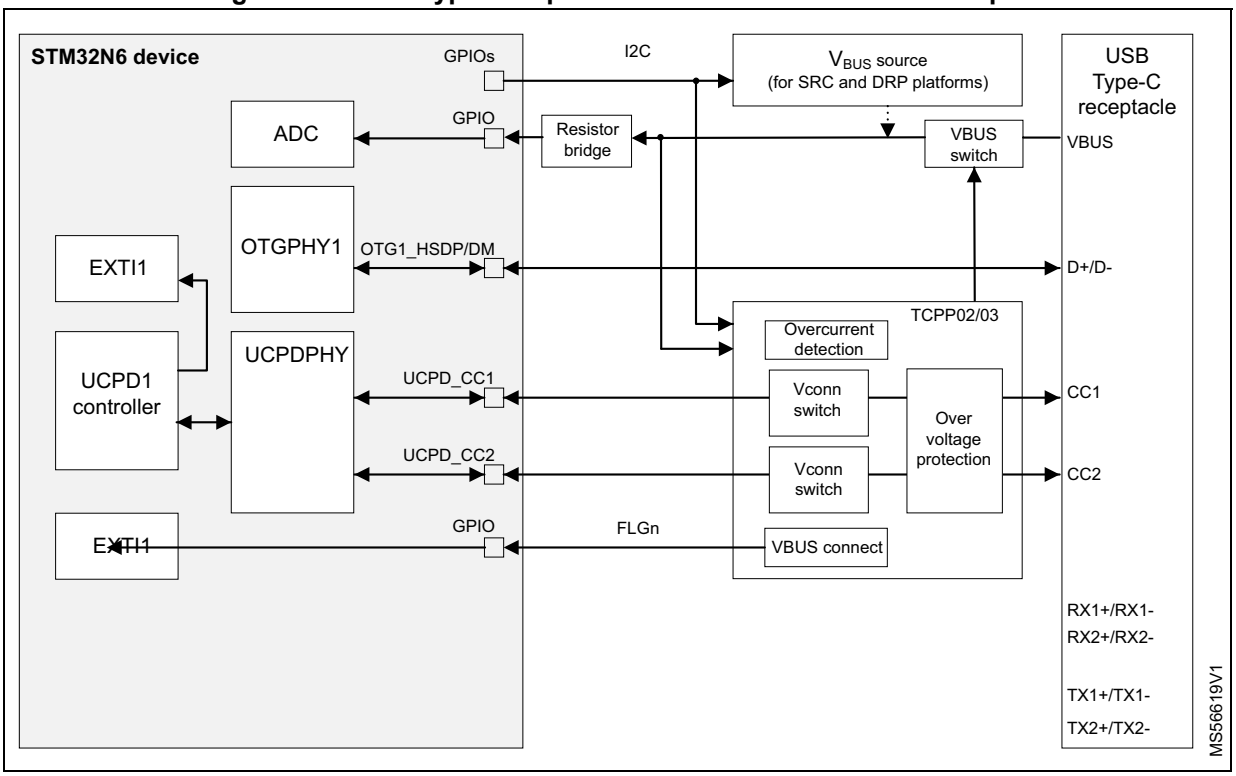
The standards supported are:

- Universal Serial Bus Type-C Cable and Connector specification, release 2.0, August 2019
- Universal Serial Bus Power Delivery specification:
  - Revision 2.0, v1.3, January 12 2017
  - Revision 3.0, v2.0, August 29 2019

72.4.2 USB Type-C and power delivery interface implementation

Figure 977 gives an overview of how OTG HS Port 1 connects to a Type-C receptacle.

Figure 977. USB Type-C implementation: OTG HS Port 1 example



The CC IOs (UCPDPHY) integrate the Rp and Rd resistors, and the voltage threshold comparators needed to define the local port attributes and detect the attributes of the remote port being attached.

The UCPD digital controller handles Type-C level detection with de-bounce, CRC generation/checking, 4b5b encode/decode, BMC encode/decode.

If needed, VBUS detection must be managed by an ADC channel or an external comparator. It cannot be managed by OTGPHY1 since it is forced in external comparator mode and not connected to VBUS.

VBUS comparison with Vsafe0v and Vsafe5v thresholds is performed by an ADC. An external resistor bridge is required to scale down VBUS to the 1V8 range of the ADC channel input.

An external component is required to protect the CC lines from overvoltage condition and VBUS line from overvoltage and overcurrent conditions. TCPP02/03 is well suited for this purpose and also implements Vconn switches, Dead battery support, the control for VBUS switch and VBUS discharge.

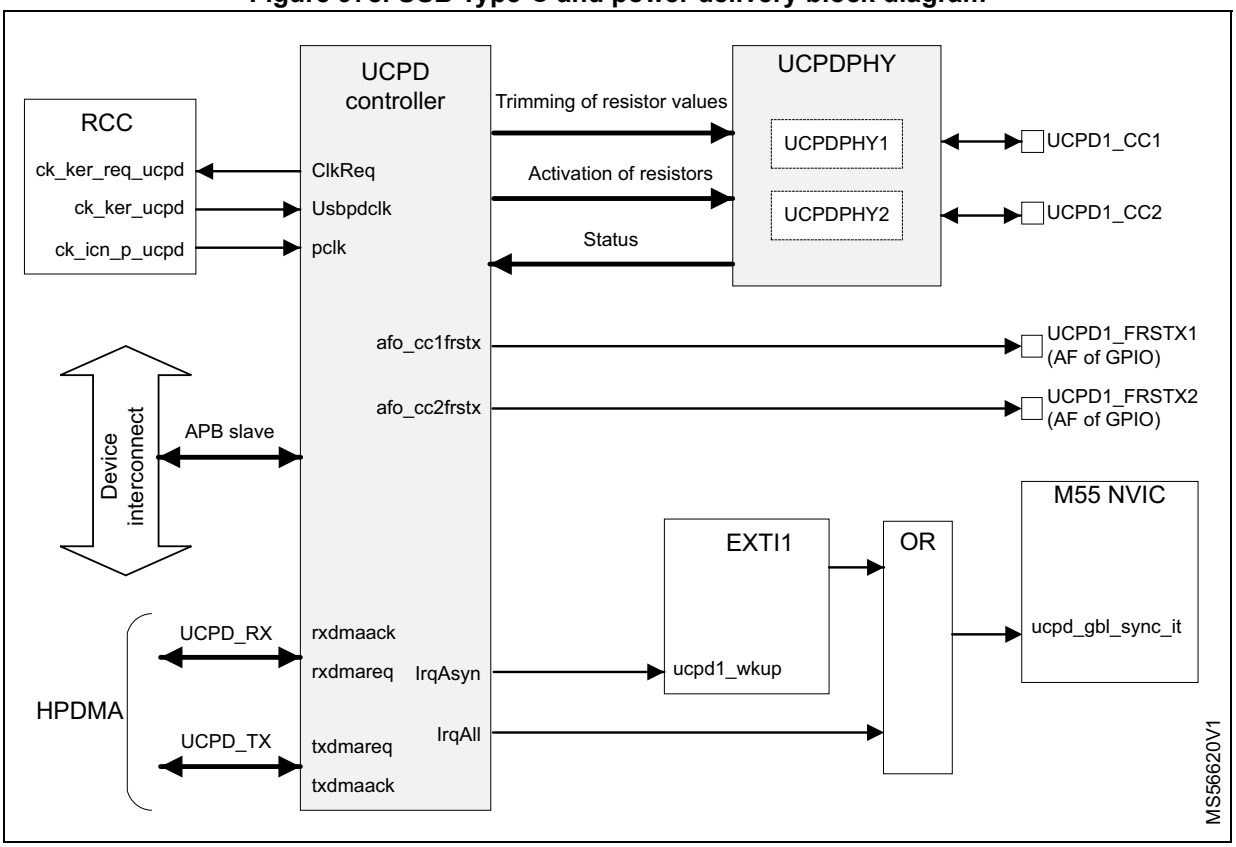
A VBUS source is required in source (SRC) and DRP platforms.

In a SRC or DRP platform, the device can be woken up from Stop mode by the UCPD digital controller upon asynchronous detection of events on the CC lines. In a sink (SNK) platform, the device can be woken up from Stdby mode by the detection of VBUS connection performed in TCPP02/03 and routed to a GPIO (FLG_n signal in the figure above).

Note: The Type-C interface can be associated either to OTG HS Port 1 or to OTG HS Port 2. If an application requires two Type-C ports, an external Type-C controller is required.

72.4.3 USB Type-C and power delivery interface functional description

Figure 978. USB Type-C and power delivery block diagram



Note: afo_cc1frstx and afo_cc2frstx connect directly to the pad ring without being OR-ed together. Be careful when using these alternate functions to control external transistors: UCPD1_FRSTX1 is associated with CC1, while UCPD1_FRSTX2 is associated with CC2.

USB Type-C and power delivery pins and internal signals

Table 731. USB Type-C and power delivery pins

Signal name	Signal type	Description
UCPD1_CC1	3V3 analog, 5V tolerant	Channel configuration
UCPD1_CC2		
UCPD1_FRSTX1	1V8 digital	Alternate function of GPIO
UCPD1_FRSTX2		

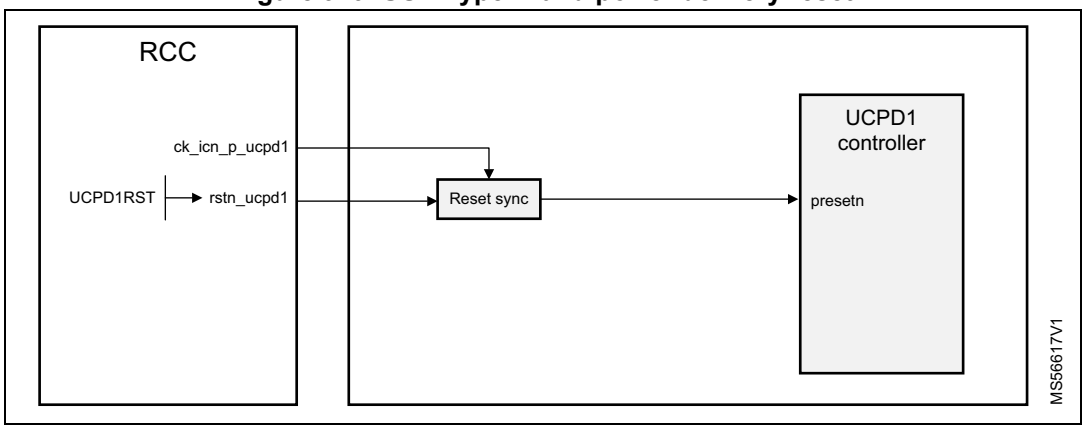
Table 732. USB Type-C and power delivery internal signals

Signal name	Signal type	Description
UCPD APB Slave	Digital	APB Slave interface for access to the UCPD controller registers. 32-bit data width.
rxdmareq	Digital	Request of UCPD_RX DMA interface

Table 732. USB Type-C and power delivery internal signals (continued)

Signal name	Signal type	Description
rxdmaack	Digital	Acknowledge of UCPD_RX DMA interface
txdmareq	Digital	Request of UCPD_TX DMA interface
txdmaack	Digital	Acknowledge of UCPD_TX DMA interface

Figure 979. USB Type-C and power delivery reset



There is only one reset for the USB Type-C and power delivery interface:

- rstn_ucpd1, controlled by the UCPD1RST bit in RCC_APB1HRSTR register.

Description of the USB Type-C and power delivery controller (UCPD1)

Refer to *Section 75: USB Type-C®/USB Power Delivery interface (UCPD)*.

72.4.4 USB Type-C and power delivery interrupts

The USB Type-C and power delivery interface involves one interrupt and one event generated by the UCPD controller.

Table 733. USB Type-C and power delivery interrupt

USB Type-C and PD interrupt	UCPD1 pin name	CPU interrupt
UCPD_GBL_SYNC_IT	IrqAll	UCPD_GBL_SYNC_IT

Table 734. USB Type-C and power delivery events

USB Type-C and PD event	UCPD1 pin name	EXTI event
UCPD1_WKUP	IrqAsyn	UCPD1_WKUP

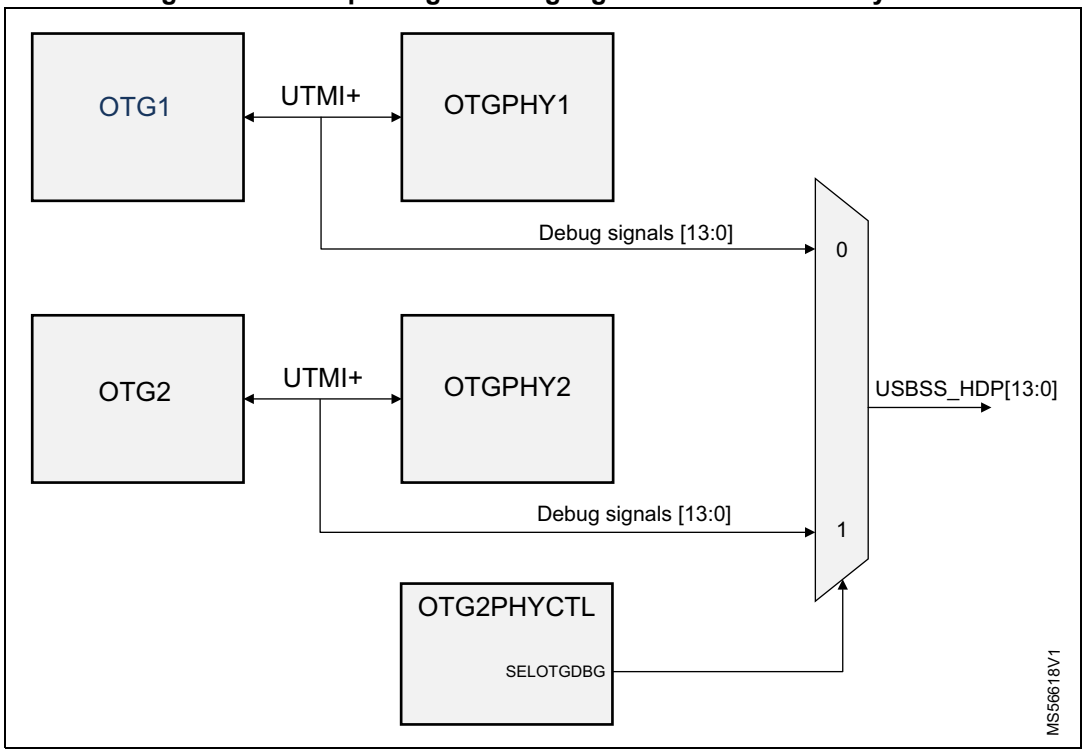
When the system is in Stop mode, it can be woken up by the UCPD1 WAKEUP event, generated when the UCPD1 controller detects some activity on one of the CC lines.

When the system is in Stdby mode, it can be woken up by an EXTI event generated by a GPIO. This is meant for Sink platforms where an external component detects VBUS presence and wakes up the device.

72.5 Debug signals

Signals of the OTG high-speed Port 1 and Port 2 are multiplexed and routed to the hardware debug port (HDP). The SELOTGDBG bit in OTG2PHYCTL_CR register controls the multiplexer. These debug signals are tapped on the UTMI+ interface between the OTG controller (OTG1, OTG2) and the USB2 PHYs (OTGPHY1, OTGPHY2). The list of debug signals is described in the Hardware Debug Port User Specifications (table 3).

Figure 980. Multiplexing of debug signals in the USB subsystem



73 USB on-the-go high-speed (OTG)

73.1 OTG introduction

Portions Copyright (c) Synopsys, Inc. All rights reserved. Used with permission.

This section presents the architecture and the programming model of the OTG controller.

The following acronyms are used throughout the section:

FS	Full-speed
LS	Low-speed
HS	High-speed
MAC	Media access controller
OTG	On-the-go
PFC	Packet FIFO controller
PHY	Physical layer
USB	Universal serial bus
UTMI	USB 2.0 Transceiver Macrocell interface (UTMI)
LPM	Link power management
BCD	Battery charging detector
HNP	Host negotiation protocol
SRP	Session request protocol

References are made to the following documents:

- USB On-The-Go Supplement, Revision 2.0
- Universal Serial Bus Revision 2.0 Specification
- USB 2.0 Link Power Management Addendum Engineering Change Notice to the USB 2.0 specification, July 16, 2007
- Errata for USB 2.0 ECN: Link Power Management (LPM) - 7/2007
- Battery Charging Specification, Revision 1.2

The USB OTG is a dual-role device (DRD) controller that supports both device and host functions and is fully compliant with the *On-The-Go Supplement to the USB 2.0 Specification*. It can also be configured as a host-only or device-only controller, fully compliant with the *USB 2.0 Specification.* OTG supports the speeds defined in the *Table 735: OTG speeds supported* below. The only external device required is a charge pump for V_{BUS} in OTG mode.

Table 735. OTG speeds supported

-	HS (480 Mbit/s)	FS (12 Mbit/s)	LS (1.5 Mbit/s)
Host mode	X	X	X
Device mode	X	X	-

73.2 OTG main features

The main features can be divided into three categories: general, host-mode, and device-mode features.

73.2.1 General features

The OTG interface general features are the following:

- It is USB-IF certified to the Universal Serial Bus Specification Rev 2.0.
- OTG supports the following PHY interfaces:
  - A UTMI interface for internal HS PHY
- It includes support (PHY) for the optional On-The-Go (OTG) protocol detailed in the On-The-Go Supplement Rev 2.0 specification
  - Integrated support for A-B device identification (ID line)
  - It allows host to turn V_{BUS} off to conserve battery power in OTG applications.
- It is software-configurable to operate as:
  - USB On-The-Go Full-Speed Dual Role device
- It supports HS SOF and LS Keep-alives with
  - SOF pulse internal connection to timer (TIMx)
  - Configurable framing period
  - Configurable end of frame interrupt
- OTG embeds an internal DMA with thresholding support and software selectable AHB burst type in DMA mode.
- It includes power saving features such as system stop during USB suspend, switch-off of clock domains internal to the digital core, PHY and DFIFO power management.
- It features a dedicated RAM of 4 Kbytes with advanced FIFO control:
  - Configurable partitioning of RAM space into different FIFOs for flexible and efficient use of RAM
  - Each FIFO can hold multiple packets.
  - Dynamic memory allocation
  - Configurable FIFO sizes that are not powers of 2 to allow the use of contiguous memory locations
- It guarantees max USB bandwidth for up to one frame (1 ms) without system intervention.
- It supports charging port detection as described in Battery Charging Specification Revision 1.2.

73.2.2 Host-mode features

The OTG interface main features and requirements in host-mode are the following:

- External charge pump for V_{BUS} voltage generation
- Up to 16 host channels (pipes): each channel is dynamically reconfigurable to allocate any type of USB transfer.
- Built-in hardware scheduler holding:
  - Up to 16 interrupt plus isochronous transfer requests in the periodic hardware queue
  - Up to 16 control plus bulk transfer requests in the non-periodic hardware queue
- Management of a shared Rx FIFO, a periodic Tx FIFO and a nonperiodic Tx FIFO for efficient usage of the USB data RAM.

73.2.3 Peripheral-mode features

The OTG interface main features in peripheral-mode are the following:

- 1 bidirectional control endpoint0
- 8 IN endpoints (EPs) configurable to support bulk, interrupt or isochronous transfers
- 8 OUT endpoints configurable to support bulk, interrupt or isochronous transfers
- Management of a shared Rx FIFO and a Tx-OUT FIFO for efficient usage of the USB data RAM
- Management of up to 9 dedicated Tx-IN FIFOs (one for each active IN EP) to put less load on the application
- Support for the soft disconnect feature.

73.3 OTG implementation

Table 736. OTG implementation(1)

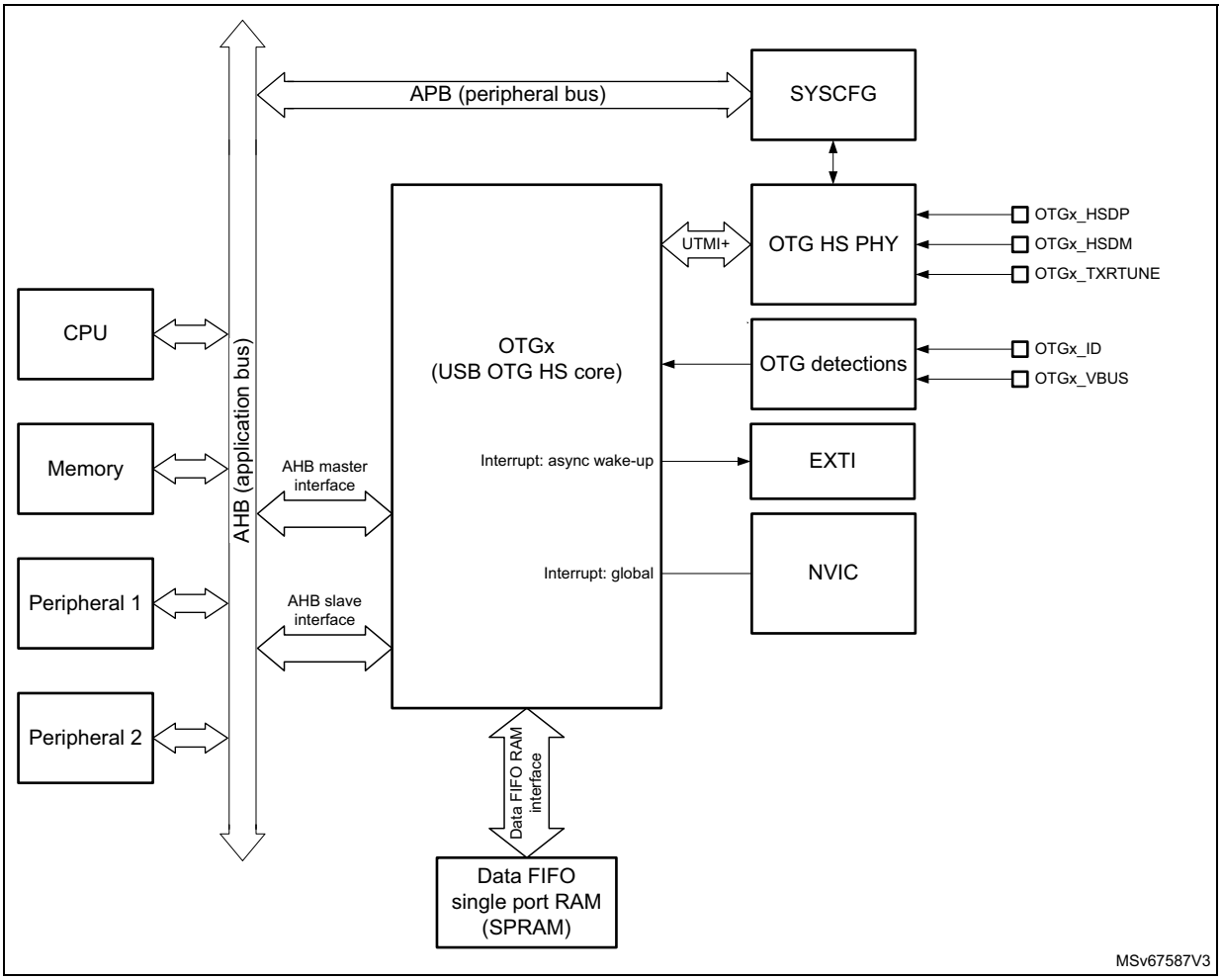
USB features	OTG1, OTG2
Device bidirectional endpoints (including EP0)	9
Host mode channels	16
Size of dedicated SRAM	4 Kbytes
USB 2.0 link power management (LPM) support	X
OTG revision supported	2.0
Battery charging detection (BCD) support	X
Integrated PHY	HS

1. “X” = supported, “-” = not supported, “FS” = supported in FS mode, “HS” = supported in HS mode.

73.4 OTG functional description

73.4.1 OTG block diagram

Figure 981. OTG1/OTG2 high-speed block diagram



73.4.2 OTG pin and internal signals

Table 737. OTG input/output pins

Signal name	Signal type	Description
OTGx_HSDP	Digital input/output	USB OTG D+ line
OTGx_HSDM	Digital input/output	USB OTG D- line
OTGx_ID	Digital input	USB OTG ID
OTGx_TXRTUNE	Analog input	USB OTG Tx tune resistor

Table 738. OTG input/output signals

Signal name	Signal type	Description
usb_sof	Digital output	USB OTG start-of-frame event for on chip peripherals
usb_wkup	Digital output	USB OTG wake-up event output
usb_gbl_it	Digital output	USB OTG global interrupt

73.4.3 OTG core

The CPU reads and writes from/to the OTG core registers through the AHB peripheral bus. It is informed of USB events through the single USB OTG interrupt line described in *Section 73.12: OTG interrupts*.

The CPU submits data over the USB by writing 32-bit words to dedicated OTG locations (push registers). The data are then automatically stored into Tx-data FIFOs configured within the USB data RAM. There is one Tx FIFO push register for each in-endpoint (peripheral mode) or out-channel (host mode).

The CPU receives the data from the USB by reading 32-bit words from dedicated OTG addresses (pop registers). The data are then automatically retrieved from a shared Rx FIFO configured within the 4-Kbyte USB data RAM. There is one Rx FIFO pop register for each out-endpoint or in-channel.

The USB protocol layer is driven by the serial interface engine (SIE) and serialized over the USB by the transceiver module within the on-chip physical layer (PHY).

Caution: To guarantee a correct operation for the USB OTG peripheral, the AHB frequency must be higher than 30 MHz.

73.4.4 OTG detections

Additionally the OTG uses the following functions:

- Integrated ID pull-up resistor used to sample the ID line for A/B device identification.

73.4.5 High-speed OTG PHY connected to OTG

Note: Refer to implementation table to determine if an HS PHY is embedded.

The USB OTG core includes an internal UTMI interface which is connected to the embedded HS PHY (see *Section 73.4.1: OTG block diagram*).

73.4.6 Battery charging detection

Support is available for the USB battery charging specification (v1.2).

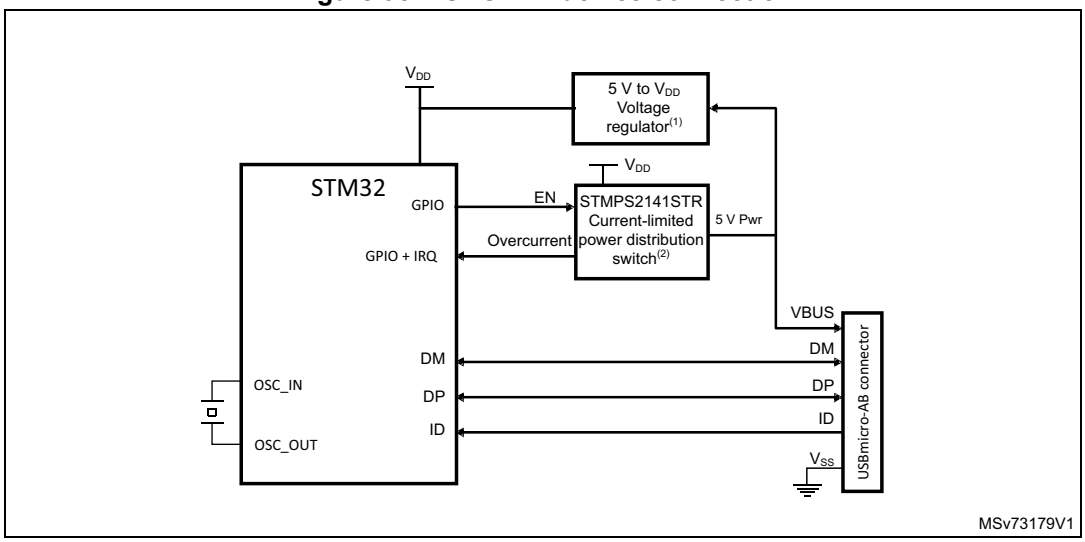
The OTG includes the necessary circuits to enable battery charging detection (portable device).

There is also a possibility to behave as a charging data port (the port advertises 1.5 A available on it's VBUS pin).

Various fields control this functions and are implemented in register OTG_GCCFG.

73.5 OTG dual role device (DRD)

Figure 982. OTG A-B device connection



1. External voltage regulator only needed when building a VBUS powered device.
2. STMPS2141STR needed only if the application has to support a VBUS powered device. A basic power switch can be used if 5 V are available on the application board.

73.5.1 ID line detection

The host or peripheral (the default) role is assumed depending on the ID input pin. The ID line status is determined on plugging in the USB cable, depending on whether a MicroA or MicroB plug is connected to the micro-AB receptacle.

- If the B-side of the USB cable is connected with a floating ID wire, the integrated pull-up resistor detects a high ID level and the default peripheral role is confirmed. In this configuration the OTG complies with the standard FSM described in section 4.2.4: ID pin of the On-the-Go specification Rev2.0, supplement to the USB2.0.
- If the A-side of the USB cable is connected with a grounded ID, the OTG issues an ID line status change interrupt (CIDSCHG bit in OTG_GINTSTS) for host software initialization, and automatically switches to the host role. In this configuration the OTG complies with the standard FSM described by section 4.2.4: ID pin of the On-the-Go specification Rev2.0, supplement to the USB2.0.

73.6 OTG as a USB peripheral

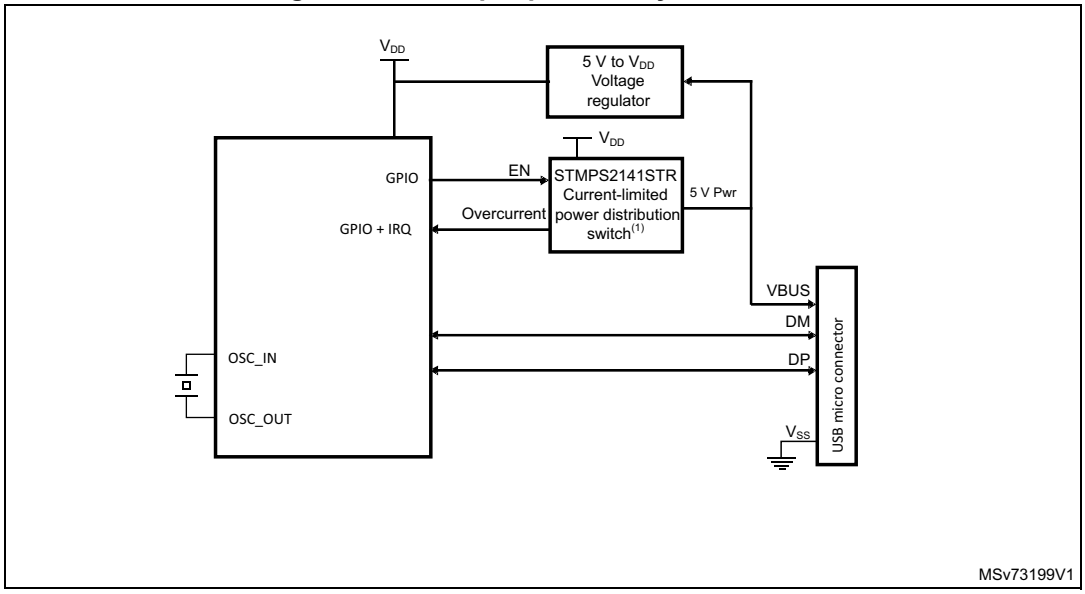
This section gives the functional description of the OTG in the USB peripheral mode. The OTG works as an USB peripheral in the following circumstances:

- OTG B-Peripheral
  - OTG B-device default state if B-side of USB cable is plugged in
- B-device
  - If the ID line is present, functional and connected to the B-side of the USB cable.
- Peripheral only
  - The force device mode bit (FDMOD) in the *Section 73.14.4: OTG USB configuration register (OTG_GUSBCFG)* is set to 1, forcing the OTG core to work

as an USB peripheral-only. In this case, the ID line is ignored even if it is present on the USB connector.

Note: To build a bus-powered device implementation in case of the B-device or peripheral-only configuration, an external regulator has to be added, that generates the necessary power-supply from V_{BUS}.

Figure 983. OTG peripheral-only connection



1. Use a regulator to build a bus-powered device.

73.6.1 Peripheral states

Powered state

External circuitry should be used to detect presence of a voltage on the VBUS connector pin. Refer to AN4879 "Introduction to USB hardware and PCB guidelines using STM32 MCUs" for recommended components. The B-peripheral V_{BUS} detection should be overridden with OTG_GOTGCTL.BVALOEN = 1. The result of this detection should be maintained by writing the current value to OTG_GOTGCTL.BVALOVAL.

In the powered state, the OTG expects to receive some reset signaling from the host. No other USB operation is possible. When a reset signaling is received the reset detected interrupt (USBRST in OTG_GINTSTS) is generated. When the reset signaling is complete, the enumeration done interrupt (ENUMDNE bit in OTG_GINTSTS) is generated and the OTG enters the Default state.

Soft disconnect

The powered state can be exited by software with the soft disconnect feature. The DP pull-up resistor is removed by setting the soft disconnect bit in the device control register (SDIS bit in OTG_DCTL), causing a device disconnect detection interrupt on the host side even though the USB cable was not really removed from the host port.

Default state

In the Default state the OTG expects to receive a SET_ADDRESS command from the host. No other USB operation is possible. When a valid SET_ADDRESS command is decoded on the USB, the application writes the corresponding number into the device address field in the device configuration register (DAD bit in OTG_DCFG). The OTG then enters the address state and is ready to answer host transactions at the configured USB address.

Suspended state

The OTG peripheral constantly monitors the USB activity. After counting 3 ms of USB idleness, the early suspend interrupt (ESUSP bit in OTG_GINTSTS) is issued, and confirmed 3 ms later, if appropriate, by the suspend interrupt (USBSUSP bit in OTG_GINTSTS). The device suspend bit is then automatically set in the device status register (SUSPSTS bit in OTG_DSTS) and the OTG enters the suspended state.

The suspended state may optionally be exited by the device itself. In this case the application sets the remote wake-up signaling bit in the device control register (RWUSIG bit in OTG_DCTL) and clears it after 1 to 15 ms.

When a resume signaling is detected from the host, the resume interrupt (WKUPINT bit in OTG_GINTSTS) is generated and the device suspend bit is automatically cleared.

73.6.2 Peripheral endpoints

The OTG core instantiates the following USB endpoints:

- Control endpoint 0:
  - Bidirectional and handles control messages only
  - Separate set of registers to handle in and out transactions
  - Dedicated control (OTG_DIEPCTL0/OTG_DOEPCTL0), transfer configuration (OTG_DIEPTSIZ0/OTG_DOEPTSIZ0), and status-interrupt (OTG_DIEPINT0/)OTG_DOEPINT0) registers. The available set of bits inside the control and transfer size registers slightly differs from that of other endpoints
- eight IN endpoints
  - Each of them can be configured to support the isochronous, bulk or interrupt transfer type
  - Each of them has dedicated control (OTG_DIEPCTL*x*), transfer configuration (OTG_DIEPTSIZ*x*), and status-interrupt (OTG_DIEPINT*x*) registers
  - The device IN endpoints common interrupt mask register (OTG_DIEPMSK) is available to enable/disable a single kind of endpoint interrupt source on all of the IN endpoints (EP0 included)
  - Support for incomplete isochronous IN transfer interrupt (IISOIXFR bit in OTG_GINTSTS), asserted when there is at least one isochronous IN endpoint on

which the transfer is not completed in the current frame. This interrupt is asserted along with the end of periodic frame interrupt (OTG_GINTSTS/EOPF).

- eight OUT endpoints
  - Each of them can be configured to support the isochronous, bulk or interrupt transfer type
  - Each of them has a dedicated control (OTG_DOEPCTL*x*), transfer configuration (OTG_DOEPTSIZ*x*) and status-interrupt (OTG_DOEPINT*x*) register
  - Device OUT endpoints common interrupt mask register (OTG_DOEPMSK) is available to enable/disable a single kind of endpoint interrupt source on all of the OUT endpoints (EP0 included)
  - Support for incomplete isochronous OUT transfer interrupt (INCOMPISOOUT bit in OTG_GINTSTS), asserted when there is at least one isochronous OUT endpoint on which the transfer is not completed in the current frame. This interrupt is asserted along with the end of periodic frame interrupt (OTG_GINTSTS/EOPF).

Endpoint control

- The following endpoint controls are available to the application through the device endpoint-*x* IN/OUT control register (OTG_DIEPCTL*x*/OTG_DOEPCTL*x*):
  - endpoint enable/disable
  - endpoint activate in current configuration
  - program USB transfer type (isochronous, bulk, interrupt)
  - program supported packet size
  - program Tx FIFO number associated with the IN endpoint
  - program the expected or transmitted data0/data1 PID (bulk/interrupt only)
  - program the even/odd frame during which the transaction is received or transmitted (isochronous only)
  - optionally program the NAK bit to always negative-acknowledge the host regardless of the FIFO status
  - optionally program the STALL bit to always stall host tokens to that endpoint
  - optionally program the SNOOP mode for OUT endpoint not to check the CRC field of received data

Endpoint transfer

The device endpoint-*x* transfer size registers (OTG_DIEPTSIZ*x*/OTG_DOEPTSIZ*x*) allow the application to program the transfer size parameters and read the transfer status. Programming must be done before setting the endpoint enable bit in the endpoint control register. Once the endpoint is enabled, these fields are read-only as the OTG core updates them with the current transfer status.

The following transfer parameters can be programmed:

- transfer size in bytes
- number of packets that constitute the overall transfer size

Endpoint status/interrupt

The device endpoint-*x* interrupt registers (OTG_DIEPINT*x*/OTG_DOPEPINT*x)* indicate the status of an endpoint with respect to USB- and AHB-related events. The application must read these registers when the OUT endpoint interrupt bit or the IN endpoint interrupt bit in

the core interrupt register (OEPINT bit in OTG_GINTSTS or IEPINT bit in OTG_GINTSTS, respectively) is set. Before the application can read these registers, it must first read the device all endpoints interrupt (OTG_DAINT) register to get the exact endpoint number for the device endpoint-*x* interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_DAINT and OTG_GINTSTS registers

The peripheral core provides the following status checks and interrupt generation:

- transfer completed interrupt, indicating that data transfer was completed on both the application (AHB) and USB sides
- setup stage has been done (control-out only)
- associated transmit FIFO is half or completely empty (in endpoints)
- NAK acknowledge has been transmitted to the host (isochronous-in only)
- IN token received when Tx FIFO was empty (bulk-in/interrupt-in only)
- Out token received when endpoint was not yet enabled
- babble error condition has been detected
- endpoint disable by application is effective
- endpoint NAK by application is effective (isochronous-in only)
- more than 3 back-to-back setup packets were received (control-out only)
- timeout condition detected (control-in only)
- isochronous out packet has been dropped, without generating an interrupt

73.7 OTG as a USB host

This section gives the functional description of the OTG in the USB host mode. The OTG works as a USB host in the following circumstances:

Automatic host mode direct from ID pin:

The ID pin is not always available, refer to product datasheet for available pins.

- Use cases:
  - Micro connector: A-Device with a Micro-A plug inserted, and connector ID pin connected to the STM32 ID pin which is then detected in a low state.
- Note that the integrated pull-down resistors are automatically set on the DP/DM lines.

Manual forcing of host mode when not possible via ID pin:

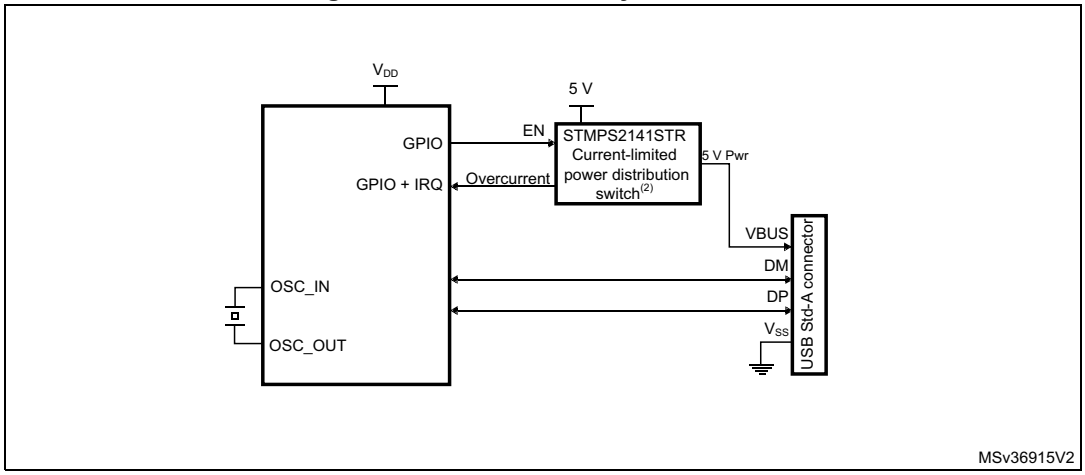
The force host mode bit (FHMOD) in the *OTG USB configuration register (OTG_GUSBCFG)* forces the OTG core to work as a USB host-only when required.

- Use cases:
  - Standard-A connector which always implies Host mode.
  - Micro connector, so ID present on connector, but ID pin not available in pinout, at A-plug insertion (detected by another means, for example: GPIO).
  - A Type-C connector, when host functionality is determined at cable attachment or via power delivery messages.
- Note that the integrated pull-down resistors are set on the DP/DM lines by setting bit FORCEHOSTPD.

Note: On-chip 5 V V_{BUS} generation is not supported. For this reason, a charge pump or, if 5 V are available on the application board, a basic power switch must be added externally to drive

the 5 V V_{BUS} line. The external charge pump can be driven by any GPIO output. This is required for the OTG A-host, A-device and host-only configurations.

Figure 984. OTG host-only connection



1. V_{DD} range is between 2 V and 3.6 V.

73.7.1 USB host states

Host port power

On-chip 5 V V_{BUS} generation is not supported. For this reason, a charge pump or, if 5 V are available on the application board, a basic power switch, must be added externally to drive the 5 V V_{BUS} line. The external charge pump can be driven by any GPIO output or via an I^2C interface connected to an external PMIC (power management IC). When the application decides to power on V_{BUS}, it must also set the port power bit in the host port control and status register (PPWR bit in OTG_HPRT).

The charge pump overcurrent flag can also be used to prevent electrical damage. Connect the overcurrent flag output from the charge pump to any GPIO input and configure it to generate a port interrupt on the active level. The overcurrent ISR must promptly disable the V_{BUS} generation and clear the port power bit.

Host detection of a peripheral connection

USB peripherals or B-device are detected as soon as they are connected. The OTG core issues a host port interrupt triggered by the device connected bit in the host port control and status (PCDET bit in OTG_HPRT).

Host detection of peripheral a disconnection

The peripheral disconnection event triggers the disconnect detected interrupt (DISCINT bit in OTG_GINTSTS).

Host enumeration

After detecting a peripheral connection the host must start the enumeration process by sending USB reset and configuration commands to the new peripheral.

The application drives a USB reset signaling (single-ended zero) over the USB by keeping the port reset bit set in the host port control and status register (PRST bit in OTG_HPRT) for

a minimum of 10 ms and a maximum of 20 ms. The application takes care of the timing count and then of clearing the port reset bit.

Once the USB reset sequence has completed, the host port interrupt is triggered by the port enable/disable change bit (PENCHNG bit in OTG_HPRT). This informs the application that the speed of the enumerated peripheral can be read from the port speed field in the host port control and status register (PSPD bit in OTG_HPRT) and that the host is starting to drive SOFs (FS) or Keep alives (LS). The host is now ready to complete the peripheral enumeration by sending peripheral configuration commands.

Host suspend

The application decides to suspend the USB activity by setting the port suspend bit in the host port control and status register (PSUSP bit in OTG_HPRT). The OTG core stops sending SOFs and enters the suspended state.

The suspended state can be optionally exited on the remote device's initiative (remote wake-up). In this case the remote wake-up interrupt (WKUPINT bit in OTG_GINTSTS) is generated upon detection of a remote wake-up signaling, the port resume bit in the host port control and status register (PRES bit in OTG_HPRT) self-sets, and resume signaling is automatically driven over the USB. The application must time the resume window and then clear the port resume bit to exit the suspended state and restart the SOF.

If the suspended state is exited on the host initiative, the application must set the port resume bit to start resume signaling on the host port, time the resume window and finally clear the port resume bit.

73.7.2 Host channels

The OTG core instantiates 16 host channels. Each host channel supports an USB host transfer (USB pipe). The host is not able to support more than 16 transfer requests at the same time. If more than 16 transfer requests are pending from the application, the host controller driver (HCD) must re-allocate channels when they become available from previous duty, that is, after receiving the transfer completed and channel halted interrupts.

Each host channel can be configured to support in/out and any type of periodic/nonperiodic transaction. Each host channel makes us of dedicated control (OTG_HCCHAR*x*), transfer configuration (OTG_HCTSIZ*x*) and status/interrupt (OTG_HCINT*x*) registers with associated mask (OTG_HCINTMSK*x*) registers.

Host channel control

- The following host channel controls are available to the application through the host channel-*x* characteristics register (OTG_HCCHAR*x*):
  - Channel enable/disable
  - Program the HS/FS/LS speed of target USB peripheral
  - Program the address of target USB peripheral
  - Program the endpoint number of target USB peripheral
  - Program the transfer IN/OUT direction
  - Program the USB transfer type (control, bulk, interrupt, isochronous)
  - Program the maximum packet size (MPS)
  - Program the periodic transfer to be executed during odd/even frames

Host channel transfer

The host channel transfer size registers (OTG_HCTSIZ*x*) allow the application to program the transfer size parameters, and read the transfer status. Programming must be done before setting the channel enable bit in the host channel characteristics register. Once the endpoint is enabled the packet count field is read-only as the OTG core updates it according to the current transfer status.

- The following transfer parameters can be programmed:
  - transfer size in bytes
  - number of packets making up the overall transfer size
  - initial data PID

Host channel status/interrupt

The host channel-*x* interrupt register (OTG_HCINT*x*) indicates the status of an endpoint with respect to USB- and AHB-related events. The application must read these register when the host channels interrupt bit in the core interrupt register (HCINT bit in OTG_GINTSTS) is set. Before the application can read these registers, it must first read the host all channels interrupt (OTG_HAINT) register to get the exact channel number for the host channel-x interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_HAINT and OTG_GINTSTS registers. The mask bits for each interrupt source of each channel are also available in the OTG_HCINTMSKx register.

- The host core provides the following status checks and interrupt generation:
  - Transfer completed interrupt, indicating that the data transfer is complete on both the application (AHB) and USB sides
  - Channel has stopped due to transfer completed, USB transaction error or disable command from the application
  - Associated transmit FIFO is half or completely empty (IN endpoints)
  - ACK response received
  - NAK response received
  - STALL response received
  - USB transaction error due to CRC failure, timeout, bit stuff error, false EOP
  - Babble error
  - frame overrun
  - data toggle error

73.7.3 Host scheduler

The host core features a built-in hardware scheduler which is able to autonomously re-order and manage the USB transaction requests posted by the application. At the beginning of each frame the host executes the periodic (isochronous and interrupt) transactions first, followed by the nonperiodic (control and bulk) transactions to achieve the higher level of priority granted to the isochronous and interrupt transfer types by the USB specification.

The host processes the USB transactions through request queues (one for periodic and one for nonperiodic). Each request queue can hold up to 8 entries. Each entry represents a pending transaction request from the application, and holds the IN or OUT channel number along with other information to perform a transaction on the USB. The order in which the

requests are written to the queue determines the sequence of the transactions on the USB interface.

At the beginning of each frame, the host processes the periodic request queue first, followed by the nonperiodic request queue. The host issues an incomplete periodic transfer interrupt (IPXFR bit in OTG_GINTSTS) if an isochronous or interrupt transaction scheduled for the current frame is still pending at the end of the current frame. The OTG core is fully responsible for the management of the periodic and nonperiodic request queues.The periodic transmit FIFO and queue status register (OTG_HPTXSTS) and nonperiodic transmit FIFO and queue status register (OTG_HNPTXSTS) are read-only registers which can be used by the application to read the status of each request queue. They contain:

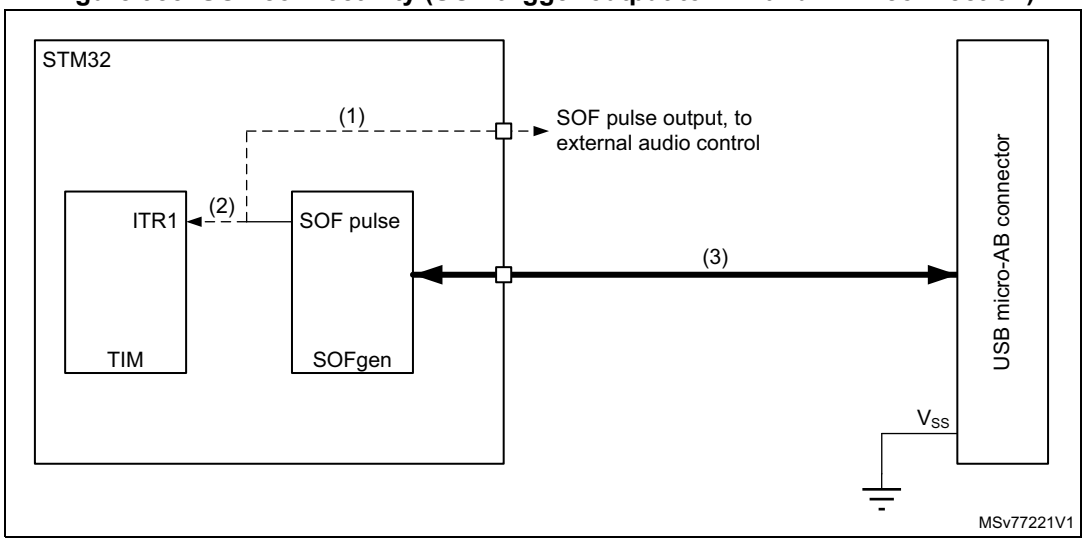
- The number of free entries currently available in the periodic (nonperiodic) request queue (8 max)
- Free space currently available in the periodic (nonperiodic) Tx FIFO (out-transactions)
- IN/OUT token, host channel number and other status information.

As request queues can hold a maximum of eight entries each, the application can push to schedule host transactions in advance with respect to the moment they physically reach the SB for a maximum of eight pending periodic transactions plus 8 pending non-periodic transactions.

To post a transaction request to the host scheduler (queue) the application must check that there is at least 1 entry available in the periodic (nonperiodic) request queue by reading the PTXQSAV bits in the OTG_HNPTXSTS register or NPTQXSAV bits in the OTG_HNPTXSTS register.

73.8 OTG SOF trigger

Figure 985. SOF connectivity (SOF trigger output to TIM and ITR1 connection)



1. SOF pulse output is not available on this product.
2. SOF timer connection is available on this product.
3. See *Table 737: OTG input/output pins* for full pin list.

The OTG core provides means to monitor, track and configure SOF framing in the host and peripheral, as well as an SOF pulse output connectivity feature.

Such utilities are especially useful for adaptive audio clock generation techniques, where the audio peripheral needs to synchronize to the isochronous stream provided by the PC, or the host needs to trim its framing rate according to the requirements of the audio peripheral.

73.8.1 Host SOFs

In host mode the number of PHY clocks occurring between the generation of two consecutive SOF (HS/FS) or Keep-alive (LS) tokens is programmable in the host frame interval register (HFIR), thus providing application control over the SOF framing period. An interrupt is generated at any start of frame (SOF bit in OTG_GINTSTS). The current frame number and the time remaining until the next SOF are tracked in the host frame number register (HFNUM).

A SOF pulse signal, is generated at any SOF starting token and with a width of 20 HCLK cycles. The SOF pulse is also internally connected to the input trigger of the timer, so that the input capture feature, the output compare feature and the timer can be triggered by the SOF pulse.

73.8.2 Peripheral SOFs

In device mode, the start of frame interrupt is generated each time an SOF token is received on the USB (SOF bit in OTG_GINTSTS). The corresponding frame number can be read from the device status register (FNSOF bit in OTG_DSTS). A SOF pulse signal with a width of 20 HCLK cycles is also generated. The SOF pulse signal is also internally connected to the TIM input trigger, so that the input capture feature, the output compare feature and the timer can be triggered by the SOF pulse.

The end of periodic frame interrupt (OTG_GINTSTS/EOPF) is used to notify the application when 80%, 85%, 90% or 95% of the time frame interval elapsed depending on the periodic frame interval field in the device configuration register (PFIVL bit in OTG_DCFG). This feature can be used to determine if all of the isochronous traffic for that frame is complete.

73.9 OTG low-power modes

Table 739 below defines the STM32 low power modes and their compatibility with the OTG.

Table 739. Compatibility of STM32 low power modes with the OTG

Mode	Description	USB compatibility
Run	MCU fully active	Required when USB not in suspend state.
Sleep	USB suspend exit causes the device to exit Sleep mode. Peripheral registers content is kept.	Available while USB is in suspend state.
Stop	USB suspend exit causes the device to exit Stop mode. Peripheral registers content is kept(1).	Available while USB is in suspend state.
Standby	Powered-down. The peripheral must be reinitialized after exiting Standby mode.	Not compatible with USB applications.

1. Within Stop mode there are different possible settings. Some restrictions may also exist, refer to to understand which (if any) restrictions apply when using OTG.

The following bits and procedures reduce power consumption.

Power reduction techniques are available while in the USB suspended state, when the USB session is not yet valid or the device is disconnected.

- Stop PHY clock (STPPCLK bit in OTG_PCGCCTL)

  When setting the stop PHY clock bit in the clock gating control register, most of the transceiver is disabled, and only the part in charge of detecting the asynchronous resume or remote wake-up event is kept alive.
- Gate HCLK (GATEHCLK bit in OTG_PCGCCTL)

  When setting the Gate HCLK bit in the clock gating control register, most of the system clock domain internal to the OTG core is switched off by clock gating. Only the register read and write interface is kept alive. The dynamic power consumption due to the USB clock switching activity is cut even if the system clock is kept running by the application for other purposes.
- USB system stop

  When the OTG is in the USB suspended state, the application may decide to drastically reduce the overall power consumption by a complete shut down of all the clock sources in the system. USB System Stop is activated by first setting the Stop PHY clock bit and then configuring the system deep sleep mode in the power control system module (PWR).

  The OTG core automatically reactivates both system and USB clocks by asynchronous detection of remote wake-up (as an host) or resume (as a device) signaling on the USB.

To save dynamic power, the USB data FIFO is clocked only when accessed by the OTG core.

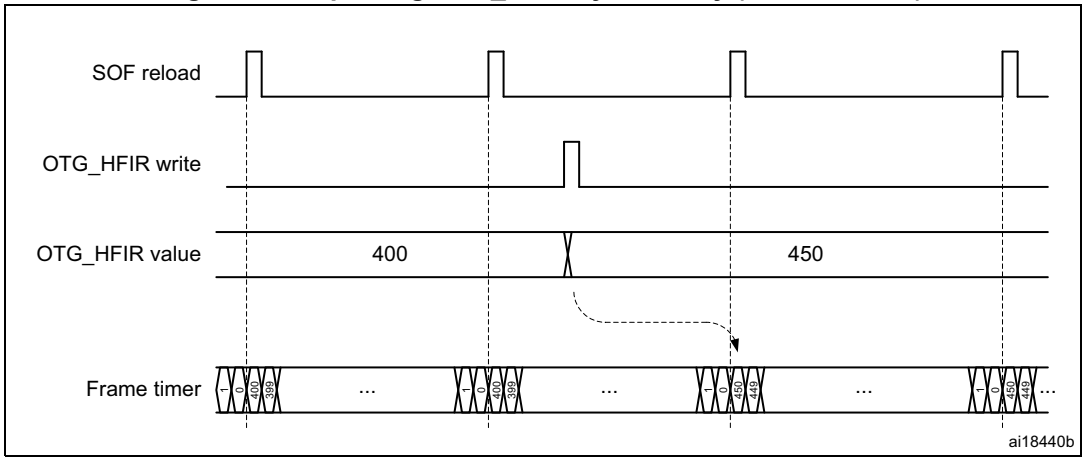
73.10 OTG Dynamic update of the OTG_HFIR register

The USB core embeds a dynamic trimming capability of micro-SOF framing period in host mode allowing to synchronize an external device with the micro-SOF frames.

When the OTG_HFIR register is changed within a current micro-SOF frame, the SOF period correction is applied in the next frame as described in *Figure 986*.

For a dynamic update, it is required to set RLDCTRL=1.

Figure 986. Updating OTG_HFIR dynamically (RLDCTRL = 1)

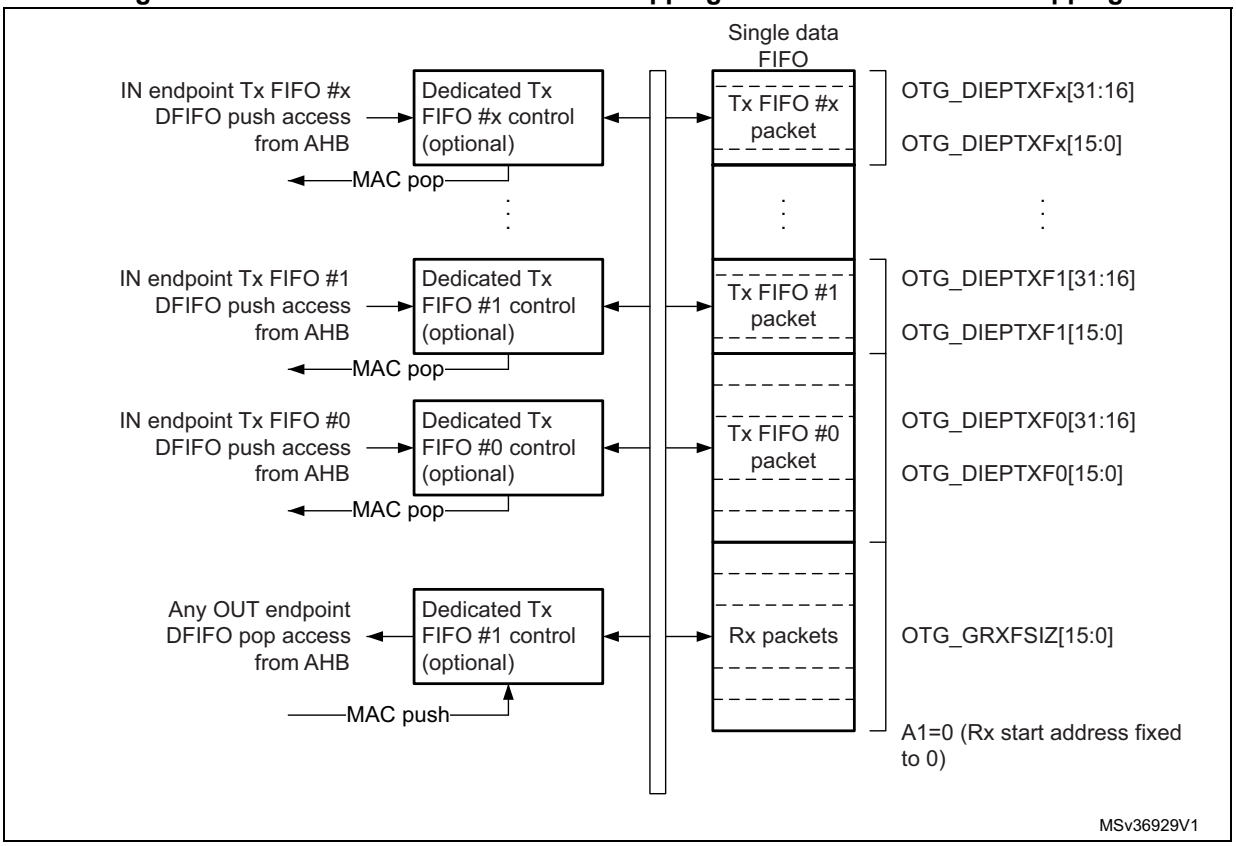


73.11 OTG data FIFOs

The USB system features 4 Kbytes of dedicated RAM with a sophisticated FIFO control mechanism. The packet FIFO controller module in the OTG core organizes RAM space into Tx FIFOs into which the application pushes the data to be temporarily stored before the USB transmission, and into a single Rx FIFO where the data received from the USB are temporarily stored before retrieval (popped) by the application. The number of instructed FIFOs and how these are organized inside the RAM depends on the device's role. In peripheral mode an additional Tx FIFO is instructed for each active IN endpoint. Any FIFO size is software configured to better meet the application requirements.

73.11.1 Peripheral FIFO architecture

Figure 987. Device-mode FIFO address mapping and AHB FIFO access mapping



Peripheral Rx FIFO

The OTG peripheral uses a single receive FIFO that receives the data directed to all OUT endpoints. Received packets are stacked back-to-back until free space is available in the Rx FIFO. The status of the received packet (which contains the OUT endpoint destination number, the byte count, the data PID and the validity of the received data) is also stored by the core on top of the data payload. When no more space is available, host transactions are NACKed and an interrupt is received on the addressed endpoint. The size of the receive FIFO is configured in the receive FIFO size register (OTG_GRXFSIZ).

The single receive FIFO architecture makes it more efficient for the USB peripheral to fill in the receive RAM buffer:

- All OUT endpoints share the same RAM buffer (shared FIFO)
- The OTG core can fill in the receive FIFO up to the limit for any host sequence of OUT tokens

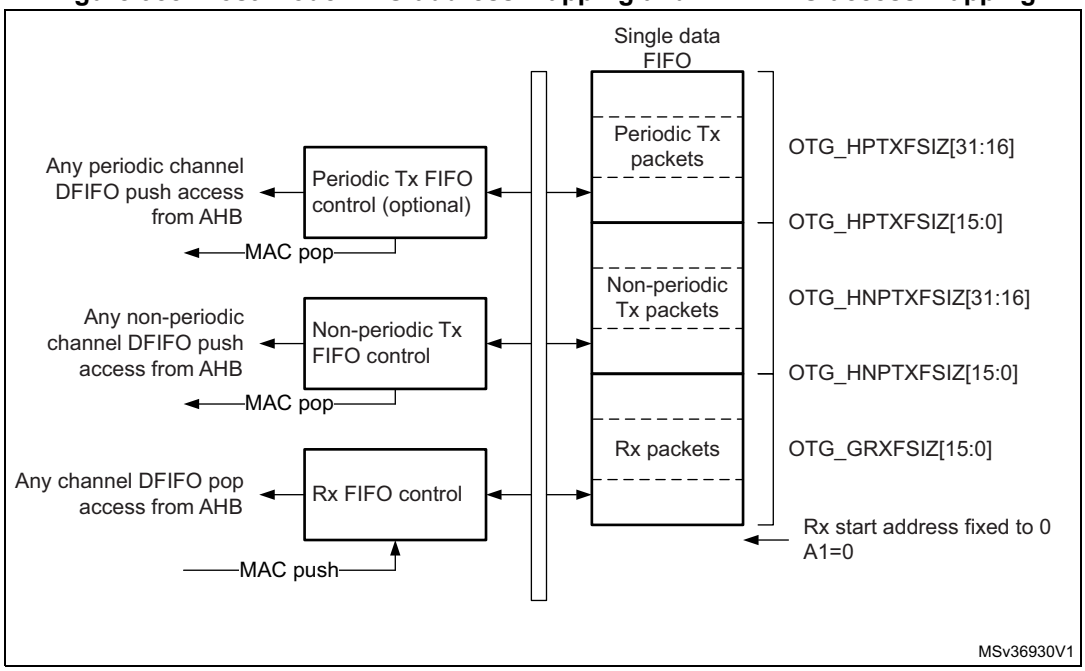
The application keeps receiving the Rx FIFO non-empty interrupt (RXFLVL bit in OTG_GINTSTS) as long as there is at least one packet available for download. It reads the packet information from the receive status read and pop register (OTG_GRXSTSP) and finally pops data off the receive FIFO by reading from the endpoint-related pop address.

Peripheral Tx FIFOs

The core has a dedicated FIFO for each IN endpoint. The application configures FIFO sizes by writing the endpoint 0 transmit FIFO size register (OTG_DIEPTXF0) for IN endpoint0 and the device IN endpoint transmit FIFOx registers (OTG_DIEPTXFx) for IN endpoint-x.

73.11.2 Host FIFO architecture

Figure 988. Host-mode FIFO address mapping and AHB FIFO access mapping



Host Rx FIFO

The host uses one receiver FIFO for all periodic and nonperiodic transactions. The FIFO is used as a receive buffer to hold the received data (payload of the received packet) from the USB until it is transferred to the system memory. Packets received from any remote IN endpoint are stacked back-to-back until free space is available. The status of each received packet with the host channel destination, byte count, data PID and validity of the received data are also stored into the FIFO. The size of the receive FIFO is configured in the receive FIFO size register (OTG_GRXFSIZ).

The single receive FIFO architecture makes it highly efficient for the USB host to fill in the receive data buffer:

- All IN configured host channels share the same RAM buffer (shared FIFO)
- The OTG core can fill in the receive FIFO up to the limit for any sequence of IN tokens driven by the host software

The application receives the Rx FIFO not-empty interrupt as long as there is at least one packet available for download. It reads the packet information from the receive status read and pop register and finally pops the data off the receive FIFO.

Host Tx FIFOs

The host uses one transmit FIFO for all non-periodic (control and bulk) OUT transactions and one transmit FIFO for all periodic (isochronous and interrupt) OUT transactions. FIFOs are used as transmit buffers to hold the data (payload of the transmit packet) to be transmitted over the USB. The size of the periodic (nonperiodic) Tx FIFO is configured in the host periodic (nonperiodic) transmit FIFO size OTG_HPTXFSIZ / OTG_HNPTXFSIZ) register.

The two Tx FIFO implementation derives from the higher priority granted to the periodic type of traffic over the USB frame. At the beginning of each frame, the built-in host scheduler processes the periodic request queue first, followed by the nonperiodic request queue.

The two transmit FIFO architecture provides the USB host with separate optimization for periodic and nonperiodic transmit data buffer management:

- All host channels configured to support periodic (nonperiodic) transactions in the OUT direction share the same RAM buffer (shared FIFOs)
- The OTG core can fill in the periodic (nonperiodic) transmit FIFO up to the limit for any sequence of OUT tokens driven by the host software

The OTG core issues the periodic Tx FIFO empty interrupt (PTXFE bit in OTG_GINTSTS) as long as the periodic Tx FIFO is half or completely empty, depending on the value of the periodic Tx FIFO empty level bit in the AHB configuration register (PTXFELVL bit in OTG_GAHBCFG). The application can push the transmission data in advance as long as free space is available in both the periodic Tx FIFO and the periodic request queue. The host periodic transmit FIFO and queue status register (OTG_HPTXSTS) can be read to know how much space is available in both.

OTG core issues the non periodic Tx FIFO empty interrupt (NPTXFE bit in OTG_GINTSTS) as long as the nonperiodic Tx FIFO is half or completely empty depending on the non periodic Tx FIFO empty level bit in the AHB configuration register (TXFELVL bit in OTG_GAHBCFG). The application can push the transmission data as long as free space is available in both the nonperiodic Tx FIFO and nonperiodic request queue. The host nonperiodic transmit FIFO and queue status register (OTG_HNPTXSTS) can be read to know how much space is available in both.

73.11.3 FIFO RAM allocation

Device mode

Receive FIFO RAM allocation: the application must allocate RAM for SETUP packets:

- 10 locations must be reserved in the receive FIFO to receive SETUP packets on control endpoint. The core does not use these locations, which are reserved for SETUP packets, to write any other data.
- One location is to be allocated for Global OUT NAK.
- Status information is written to the FIFO along with each received packet. Therefore, a minimum space of (largest packet size / 4) + 1 must be allocated to receive packets. If multiple isochronous endpoints are enabled, then at least two (largest packet size / 4) + 1 spaces must be allocated to receive back-to-back packets. Typically, two (largest packet size / 4) + 1 spaces are recommended so that when the previous packet is being transferred to the CPU, the USB can receive the subsequent packet.
- Along with the last packet for each endpoint, transfer complete status information is also pushed to the FIFO. One location for each OUT endpoint is recommended.

Device RxFIFO =

(5 * number of control endpoints + 8) + ((largest USB packet used / 4) + 1 for status information) + (2 * number of OUT endpoints) + 1 for Global NAK

Example: The MPS is 1,024 bytes for a periodic USB packet and 512 bytes for a non-periodic USB packet. There are three OUT endpoints, three IN endpoints, one control endpoint, and three host channels.

Device RxFIFO = (5 * 1 + 8) + ((1,024 / 4) +1) + (2 * 4) + 1 = 279

Transmit FIFO RAM allocation: the minimum RAM space required for each IN endpoint Transmit FIFO is the maximum packet size for that particular IN endpoint.

Note: More space allocated in the transmit IN endpoint FIFO results in better performance on the USB.

Host mode

Receive FIFO RAM allocation:

Status information is written to the FIFO along with each received packet. Therefore, a minimum space of (largest packet size / 4) + 1 must be allocated to receive packets. If multiple isochronous channels are enabled, then at least two (largest packet size / 4) + 1 spaces must be allocated to receive back-to-back packets. Typically, two (largest packet size / 4) + 1 spaces are recommended so that when the previous packet is being transferred to the CPU, the USB can receive the subsequent packet.

Along with the last packet in the host channel, transfer complete status information is also pushed to the FIFO. So one location must be allocated for this.

Host RxFIFO = (largest USB packet used / 4) + 1 for status information + 1 transfer complete

Example: Host RxFIFO = ((1,024 / 4) + 1) + 1 = 258

Transmit FIFO RAM allocation:

The minimum amount of RAM required for the host Non-periodic Transmit FIFO is the largest maximum packet size among all supported non-periodic OUT channels.

Typically, two largest packet sizes worth of space is recommended, so that when the current packet is under transfer to the USB, the CPU can get the next packet.

Non-Periodic TxFIFO = largest non-periodic USB packet used / 4

Example: Non-Periodic TxFIFO = (512 / 4) = 128

The minimum amount of RAM required for host periodic Transmit FIFO is the largest maximum packet size out of all the supported periodic OUT channels. If there is at least one isochronous OUT endpoint, then the space must be at least two times the maximum packet size of that channel.

Host Periodic TxFIFO = largest periodic USB packet used / 4

Example: Host Periodic TxFIFO = (1,024 / 4) = 256

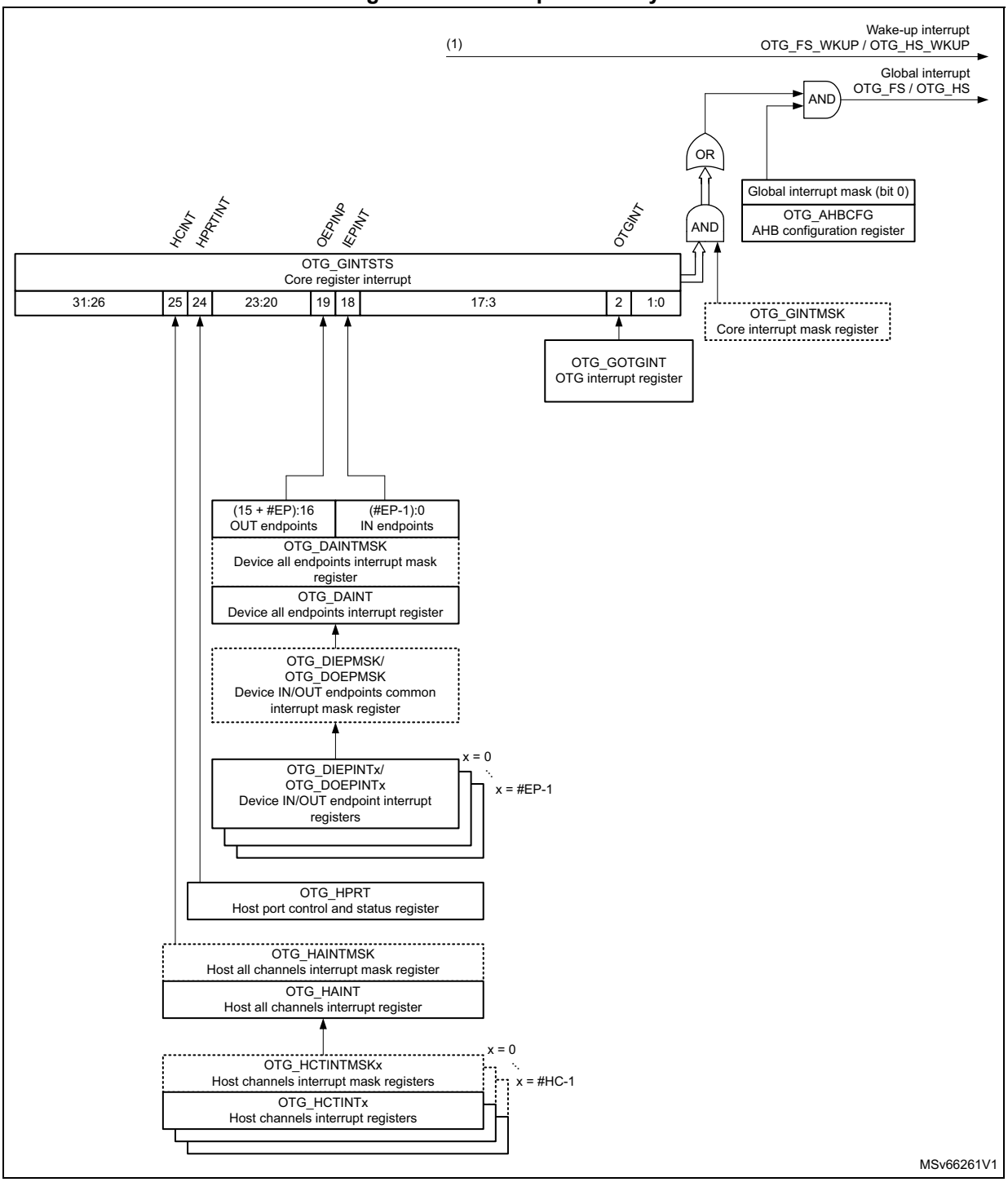
Note: More space allocated in the Transmit Non-periodic FIFO results in better performance on the USB.

73.12 OTG interrupts

When the OTG controller is operating in one mode, either device or host, the application must not access registers from the other mode. If an illegal access occurs, a mode mismatch interrupt is generated and reflected in the core interrupt register (MMIS bit in the OTG_GINTSTS register). When the core switches from one mode to the other, the registers in the new mode of operation must be reprogrammed as they would be after a power-on reset.

Figure 989 shows the interrupt hierarchy.

Figure 989. Interrupt hierarchy



1. OTG_WKUP becomes active (high state) when resume condition occurs during L1 SLEEP or L2 SUSPEND states.

73.13 OTG control and status registers

By reading from and writing to the control and status registers (CSRs) through the AHB slave interface, the application controls the OTG controller. These registers are 32 bits wide, and the addresses are 32-bit block aligned. The OTG registers must be accessed by words (32 bits).

CSRs are classified as follows:

- Core global registers
- Host-mode registers
- Host global registers
- Host port CSRs
- Host channel-specific registers
- Device-mode registers
- Device global registers
- Device endpoint-specific registers
- Power and clock-gating registers
- Data FIFO (DFIFO) access registers

Only the core global, power and clock-gating, data FIFO access, and host port control and status registers can be accessed in both host and device modes. When the OTG controller is operating in one mode, either device or host, the application must not access registers from the other mode. If an illegal access occurs, a mode mismatch interrupt is generated and reflected in the core interrupt register (MMIS bit in the OTG_GINTSTS register). When the core switches from one mode to the other, the registers in the new mode of operation must be reprogrammed as they would be after a power-on reset.

73.13.1 CSR memory map

The host and device mode registers occupy different addresses. All registers are implemented in the AHB clock domain.

Global CSR map

These registers are available in both host and device modes.

Table 740. Core global control and status registers (CSRs)

Acronym	Address offset	Register name
OTG_GOTGCTL	0x000	*Section 73.14.1: OTG control and status register (OTG_GOTGCTL)*
OTG_GOTGINT	0x004	*Section 73.14.2: OTG interrupt register (OTG_GOTGINT)*
OTG_GAHBCFG	0x008	*Section 73.14.3: OTG AHB configuration register (OTG_GAHBCFG)*
OTG_GUSBCFG	0x00C	*Section 73.14.4: OTG USB configuration register (OTG_GUSBCFG)*
OTG_GRSTCTL	0x010	*Section 73.14.5: OTG reset register (OTG_GRSTCTL)*
OTG_GINTSTS	0x014	*Section 73.14.6: OTG core interrupt register [alternate] (OTG_GINTSTS)* *Section 73.14.7: OTG core interrupt register [alternate] (OTG_GINTSTS)*

Table 740. Core global control and status registers (CSRs) (continued)

Acronym	Address offset	Register name
OTG_GINTMSK	0x018	*Section 73.14.8: OTG interrupt mask register [alternate] (OTG_GINTMSK)* *Section 73.14.9: OTG interrupt mask register [alternate] (OTG_GINTMSK)*
OTG_GRXSTSR	0x01C	*Section 73.14.10: OTG receive status debug read register [alternate] (OTG_GRXSTSR)* *Section 73.14.11: OTG receive status debug read register [alternate] (OTG_GRXSTSR)*
OTG_GRXSTSP	0x020	*Section 73.14.12: OTG status read and pop registers (OTG_GRXSTSP)* *Section 73.14.13: OTG status read and pop registers [alternate] (OTG_GRXSTSP)*
OTG_GRXFSIZ	0x024	*Section 73.14.14: OTG receive FIFO size register (OTG_GRXFSIZ)*
OTG_HNPTXFSIZ/ OTG_DIEPTXF0(1)	0x028	*Section 73.14.15: OTG host non-periodic transmit FIFO size register [alternate] (OTG_HNPTXFSIZ)* *Section 73.14.16: Endpoint 0 transmit FIFO size [alternate] (OTG_DIEPTXF0)*
OTG_HNPTXSTS	0x02C	*Section 73.14.17: OTG non-periodic transmit FIFO/queue status register (OTG_HNPTXSTS)*
OTG_GCCFG	0x038	*Section 73.14.18: OTG general core configuration register (OTG_GCCFG)*
OTG_CID	0x03C	*Section 73.14.19: OTG core ID register (OTG_CID)*
OTG_GLPMCFG	0x54	*Section 73.14.20: OTG core LPM configuration register (OTG_GLPMCFG)*
OTG_HPTXFSIZ	0x100	*Section 73.14.21: OTG host periodic transmit FIFO size register (OTG_HPTXFSIZ)*
OTG_DIEPTXFx	0x104 0x108 ... 0x120	*Section 73.14.22: OTG device IN endpoint transmit FIFO x size register (OTG_DIEPTXFx)*

1. The general rule is to use OTG_HNPTXFSIZ for host mode and OTG_DIEPTXF0 for device mode.

Host-mode CSR map

These registers must be programmed every time the core changes to host mode.

Table 741. Host-mode control and status registers (CSRs)

Acronym	Offset address	Register name
OTG_HCFG	0x400	*Section 73.14.24: OTG host configuration register (OTG_HCFG)*
OTG_HFIR	0x404	*Section 73.14.25: OTG host frame interval register (OTG_HFIR)*
OTG_HFNUM	0x408	*Section 73.14.26: OTG host frame number/frame time remaining register (OTG_HFNUM)*
OTG_HPTXSTS	0x410	*Section 73.14.27: OTG_Host periodic transmit FIFO/queue status register (OTG_HPTXSTS)*

Table 741. Host-mode control and status registers (CSRs) (continued)

Acronym	Offset address	Register name
OTG_HAINT	0x414	*Section 73.14.28: OTG host all channels interrupt register (OTG_HAINT)*
OTG_HAINTMSK	0x418	*Section 73.14.29: OTG host all channels interrupt mask register (OTG_HAINTMSK)*
OTG_HPRT	0x440	*Section 73.14.30: OTG host port control and status register (OTG_HPRT)*
OTG_HCCHARx	0x500 0x520 ... 0x6E0	*Section 73.14.31: OTG host channel x characteristics register (OTG_HCCHARx)*
OTG_HCSPLTx	0x504 0x524 .... 0x6E4	*Section 73.14.32: OTG host channel x split control register (OTG_HCSPLTx)*
OTG_HCINTx	0x508 0x528 .... 0x6E8	*Section 73.14.33: OTG host channel x interrupt register (OTG_HCINTx)*
OTG_HCINTMSKx	0x50C 0x52C 0x6EC	*Section 73.14.34: OTG host channel x interrupt mask register (OTG_HCINTMSKx)*
OTG_HCTSIZx	0x510 0x530 .... 0x6F0	*Section 73.14.35: OTG host channel x transfer size register (OTG_HCTSIZx)*
OTG_HCDMAx	0x514 0x534 .... 0x6F4	*Section 73.14.36: OTG host channel x DMA address register(OTG_HCDMAx)*

Device-mode CSR map

These registers must be programmed every time the core changes to device mode.

Table 742. Device-mode control and status registers

Acronym	Offset address	Register name
OTG_DCFG	0x800	*Section 73.14.38: OTG device configuration register (OTG_DCFG)*
OTG_DCTL	0x804	*Section 73.14.39: OTG device control register (OTG_DCTL)*
OTG_DSTS	0x808	*Section 73.14.40: OTG device status register (OTG_DSTS)*

Table 742. Device-mode control and status registers (continued)

Acronym	Offset address	Register name
OTG_DIEPMSK	0x810	*Section 73.14.41: OTG device IN endpoint common interrupt mask register (OTG_DIEPMSK)*
OTG_DOEPMSK	0x814	*Section 73.14.42: OTG device OUT endpoint common interrupt mask register (OTG_DOEPMSK)*
OTG_DAINT	0x818	*Section 73.14.43: OTG device all endpoints interrupt register (OTG_DAINT)*
OTG_DAINTMSK	0x81C	*Section 73.14.44: OTG all endpoints interrupt mask register (OTG_DAINTMSK)*
OTG_DTHRCTL	0x830	*Section 73.14.45: OTG device threshold control register (OTG_DTHRCTL)*
OTG_DIEPEMPMSK	0x834	*Section 73.14.46: OTG device IN endpoint FIFO empty interrupt mask register (OTG_DIEPEMPMSK)*
OTG_DIEPCTLx	0x900 0x920 ... 0xA00	*Section 73.14.47: OTG device IN endpoint x control register [alternate] (OTG_DIEPCTLx)* *Section 73.14.48: OTG device IN endpoint x control register [alternate] (OTG_DIEPCTLx)*
OTG_DIEPINTx	0x908 0x928 ... 0x9E8	*Section 73.14.49: OTG device IN endpoint x interrupt register (OTG_DIEPINTx)*
OTG_DIEPTSIZ0	0x910	*Section 73.14.50: OTG device IN endpoint 0 transfer size register (OTG_DIEPTSIZ0)*
OTG_DIEPDMAx	0x914 0x934 ... 0x9F4	*Section 73.14.51: OTG device IN endpoint x DMA address register (OTG_DIEPDMAx)*
OTG_DTXFSTSx	0x918 0x938 ..... 0x9F8	*Section 73.14.52: OTG device IN endpoint transmit FIFO status register (OTG_DTXFSTSx)*
OTG_DIEPTSIZx	0x930 0x950 ... 0x9F0	*Section 73.14.53: OTG device IN endpoint x transfer size register (OTG_DIEPTSIZx)*
OTG_DOEPCTL0	0xB00	*Section 73.14.54: OTG device control OUT endpoint 0 control register (OTG_DOEPCTL0)*
OTG_DOEPINTx	0xB08 0XB28 ... 0xC08	*Section 73.14.55: OTG device OUT endpoint x interrupt register (OTG_DOEPINTx)*

Table 742. Device-mode control and status registers (continued)

Acronym	Offset address	Register name
OTG_DOEPTSIZ0	0xB10	*Section 73.14.56: OTG device OUT endpoint 0 transfer size register (OTG_DOEPTSIZ0)*
OTG_DOEPDMAx	0xB14 0xB34 ... 0xC14	*Section 73.14.57: OTG device OUT endpoint x DMA address register (OTG_DOEPDMAx)*
OTG_DOEPCTLx	0xB20 0xB40 ... 0xC00	*Section 73.14.58: OTG device OUT endpoint x control register [alternate] (OTG_DOEPCTLx)* *Section 73.14.59: OTG device OUT endpoint x control register [alternate] (OTG_DOEPCTLx)*
OTG_DOEPTSIZx	0xB30 0xB50 .. 0xBF0	*Section 73.14.60: OTG device OUT endpoint x transfer size register (OTG_DOEPTSIZx)*

Data FIFO (DFIFO) access register map

These registers, available in both host and device modes, are used to read or write the FIFO space for a specific endpoint or a channel, in a given direction. If a host channel is of type IN, the FIFO can only be read on the channel. Similarly, if a host channel is of type OUT, the FIFO can only be written on the channel.

Table 743. Data FIFO (DFIFO) access register map

FIFO access register section	Offset address	Access
Device IN endpoint 0/Host OUT Channel 0: DFIFO write access Device OUT endpoint 0/Host IN Channel 0: DFIFO read access	0x1000–0x1FFC	w r
Device IN endpoint 1/Host OUT Channel 1: DFIFO write access Device OUT endpoint 1/Host IN Channel 1: DFIFO read access	0x2000–0x2FFC	w r
...	...	...
Device IN endpoint x(1)/Host OUT Channel x(1): DFIFO write access Device OUT endpoint x(1)/Host IN Channel x(1): DFIFO read access	0xX000–0xXFFC	w r

1. Where x is 8 in device mode and 15 in host mode.

Power and clock gating CSR map

There is a single register for power and clock gating. It is available in both host and device modes.

Table 744. Power and clock gating control and status registers

Acronym	Offset address	Register name
OTG_PCGCCTL	0xE00–0xE04	*Section 73.14.61: OTG power and clock gating control register (OTG_PCGCCTL)*

73.14 OTG registers

These registers are available in both host and device modes, and do not need to be reprogrammed when switching between these modes.

Bit values in the register descriptions are expressed in binary unless otherwise specified.

73.14.1 OTG control and status register (OTG_GOTGCTL)

The OTG_GOTGCTL register controls the behavior and reflects the status of the OTG function of the core.

Address offset: 0x000

Reset value: 0x0001 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CUR MOD	OTG VER	BSVLD	ASVLD	DBCT	CID STS
										r	rw	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	EHEN	Res.	Res.	Res.	Res.	BVALO VAL	BVALO EN	AVALO VAL	AVALO EN	VBVAL OVAL	VBVAL OEN	Res.	Res.
			rw					rw	rw	rw	rw	rw	rw		

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **CURMOD:** Current mode of operation

Indicates the current mode (host or device).

0: Device mode

1: Host mode

Bit 20 **OTGVER:** OTG version

Selects the OTG revision.

0:OTG Version 1.3. OTG1.3 is obsolete for new product development.

1:OTG Version 2.0. In this version the core supports only data line pulsing for SRP.

Bit 19 **BSVLD:** B-session valid

Indicates the device mode transceiver status.

0: B-session is not valid.

1: B-session is valid.

In OTG mode, the user can use this bit to determine if the device is connected or disconnected.

Note: Only accessible in device mode.

Bit 18 **ASVLD:** A-session valid

Indicates the host mode transceiver status.
0: A-session is not valid
1: A-session is valid
Note: Only accessible in host mode.

Bit 17 **DBCT:** Long/short debounce time

Indicates the debounce time of a detected connection.
0: Long debounce time, used for physical connections (100 ms + 2.5 µs)
1: Short debounce time, used for soft connections (2.5 µs)
Note: Only accessible in host mode.

Bit 16 **CIDSTS:** Connector ID status

Indicates the connector ID status on a connect event.
0: The OTG controller is in A-device mode
1: The OTG controller is in B-device mode
Note: Accessible in both device and host modes.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **EHEN:** Embedded host enable

It is used to select between OTG A device state machine and embedded host state machine.
0: OTG A device state machine is selected
1: Embedded host state machine is selected

Bits 11:8 Reserved, must be kept at reset value.

Bit 7 **BVALOVAL:** B-peripheral session valid override value

This bit is used to set override value for Bvalid signal when BVALOEN bit is set.
0: Bvalid value is '0' when BVALOEN = 1
1: Bvalid value is '1' when BVALOEN = 1
Note: Only accessible in device mode.

Bit 6 **BVALOEN:** B-peripheral session valid override enable

This bit is used to enable/disable the software to override the Bvalid signal using the BVALOVAL bit.
0: B-peripheral session valid is '0'
1: Use bit BVALOVAL to indicate B-peripheral session valid
Note: Only accessible in device mode.

Bit 5 **AVALOVAL:** A-peripheral session valid override value

This bit is used to set override value for Avalid signal when AVALOEN bit is set.
0: Avalid value is '0' when AVALOEN = 1
1: Avalid value is '1' when AVALOEN = 1
Note: Only accessible in host mode.

Bit 4 **AVALOEN:** A-peripheral session valid override enable

This bit is used to enable/disable the software to override the Avalid signal using the AVALOVAL bit.
0: A-peripheral session valid is '0'
1: Use bit AVALOVAL to indicate A-peripheral session valid
Note: Only accessible in host mode.

Bit 3 **VBVALOVAL:** V_{BUS} valid override value

This bit is used to set override value for vbusvalid signal when VBVALOEN bit is set.

0: vbusvalid value is '0' when VBVALOEN = 1

1: vbusvalid value is '1' when VBVALOEN = 1

Note: Only accessible in host mode.

Bit 2 **VBVALOEN:** V_{BUS} valid override enable

This bit is used to enable/disable the software to override the vbusvalid signal using the VBVALOVAL bit.

0: VBUS valid is '0'

1: Use bit VBVALOVAL to indicate A-peripheral session valid

Note: Only accessible in host mode.

Bits 1:0 Reserved, must be kept at reset value.

73.14.2 OTG interrupt register (OTG_GOTGINT)

The application reads this register whenever there is an OTG interrupt and clears the bits in this register to clear the OTG interrupt.

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADTO CHG	Res.	Res.
													rc_w1		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEDET	Res.	Res.
													rc_w1		

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **ADTOCHG:** A-device timeout change

The core sets this bit to indicate that the A-device has timed out while waiting for the B-device to connect.

Note: Accessible in both device and host modes.

Bits 17:3 Reserved, must be kept at reset value.

Bit 2 **SEDET:** Session end detected

The core sets this bit to indicate that the level of the voltage on V_{BUS} is no longer valid for a B-Peripheral session when V_{BUS} < 0.8 V.

Note: Accessible in both device and host modes.

Bits 1:0 Reserved, must be kept at reset value.

73.14.3 OTG AHB configuration register (OTG_GAHBCFG)

This register can be used to configure the core after power-on or a change in mode. This register mainly contains AHB system-related configuration parameters. Do not change this register after the initial programming. The application must program this register before starting any transactions on either the AHB or the USB.

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	PTXFE LVL	TXFE LVL	Res.	DMAE N	HBSTLEN[3:0]				GINT MSK
							rw	rw		rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **PTXFELVL:** Periodic Tx FIFO empty level

Indicates when the periodic Tx FIFO empty interrupt bit in the OTG_GINTSTS register (PTXFE bit in OTG_GINTSTS) is triggered.

0: PTXFE (in OTG_GINTSTS) interrupt indicates that the Periodic Tx FIFO is half empty

1: PTXFE (in OTG_GINTSTS) interrupt indicates that the Periodic Tx FIFO is completely empty

Note: Only accessible in host mode.

Bit 7 **TXFELVL:** Tx FIFO empty level

Condition: device mode

This bit indicates when IN endpoint Transmit FIFO empty interrupt (TXFE in OTG_DIEPINTx) is triggered:

0: The TXFE (in OTG_DIEPINTx) interrupt indicates that the IN endpoint Tx FIFO is half empty

1: The TXFE (in OTG_DIEPINTx) interrupt indicates that the IN endpoint Tx FIFO is completely empty

Condition: host mode

This bit indicates when the nonperiodic Tx FIFO empty interrupt (NPTXFE bit in OTG_GINTSTS) is triggered:

0: The NPTXFE (in OTG_GINTSTS) interrupt indicates that the nonperiodic Tx FIFO is half empty

1: The NPTXFE (in OTG_GINTSTS) interrupt indicates that the nonperiodic Tx FIFO is completely empty

Bit 6 Reserved, must be kept at reset value.

Bit 5 **DMAEN:** DMA enabled

0: The core operates in slave mode

1: The core operates in DMA mode

Bits 4:1 **HBSTLEN[3:0]:** Burst length/type

0000 Single: Bus transactions use single 32 bit accesses (not recommended)

0001 INCR: Bus transactions use unspecified length accesses (not recommended, uses the INCR AHB bus command)

0011 INCR4: Bus transactions target 4x 32 bit accesses

0101 INCR8: Bus transactions target 8x 32 bit accesses

0111 INCR16: Bus transactions based on 16x 32 bit accesses

Others: Reserved

Bit 0 **GINTMSK:** Global interrupt mask

The application uses this bit to mask or unmask the interrupt line assertion to itself. Irrespective of this bit setting, the interrupt status registers are updated by the core.

0: Mask the interrupt assertion to the application.

1: Unmask the interrupt assertion to the application.

Note: Accessible in both device and host modes.

73.14.4 OTG USB configuration register (OTG_GUSBCFG)

This register can be used to configure the core after power-on or a changing to host mode or device mode. It contains USB and USB-PHY related configuration parameters. The application must program this register before starting any transactions on either the AHB or the USB. Do not make changes to this register after the initial programming.

Address offset: 0x00C

Reset value: 0x0000 1400

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	FD MOD	FH MOD	Res.	Res.	Res.	Res.	Res.	Res.	TSDPS	Res.	Res.	Res.	Res.	Res.	Res.
	rw	rw							rw						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PHYL PC	Res.	TRDT[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOCAL[2:0]		
rw		rw	rw	rw	rw								rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **FDMOD:** Force device mode

Writing a 1 to this bit, forces the core to device mode irrespective of the OTG_ID input pin.
0: Normal mode
1: Force device mode
After setting the force bit, the application must wait at least 25 ms before the change takes effect.

Note: Accessible in both device and host modes.

Bit 29 **FHMOD:** Force host mode

Writing a 1 to this bit, forces the core to host mode irrespective of the OTG_ID input pin.
0: Normal mode
1: Force host mode
After setting the force bit, the application must wait at least 25 ms before the change takes effect.

Note: Accessible in both device and host modes.

Bits 28:26 Reserved, must be kept at reset value.

Bits 25:23 Reserved, must be kept at reset value.

Bit 22 **TSDPS**: TermSel DLine pulsing selection

This bit selects utmi_termselect to drive the data line pulse during SRP (session request protocol).
0: Data line pulsing using utmi_txvalid (default)
1: Data line pulsing using utmi_termsel

Bits 21:16 Reserved, must be kept at reset value.

Bit 15 **PHYLPC:** PHY Low-power clock select

This bit selects either 480 MHz or 48 MHz (low-power) PHY mode. In FS and LS modes, the PHY can usually operate on a 48 MHz clock to save power.
0: 480 MHz internal PLL clock
1: 48 MHz external clock
In 480 MHz mode, the UTMI interface operates at either 60 or 30 MHz, depending on whether the 8- or 16-bit data width is selected. In 48 MHz mode, the UTMI interface operates at 48 MHz in FS and LS modes.

Bit 14 Reserved, must be kept at reset value.

Bits 13:10 **TRDT[3:0]:** USB turnaround time

These bits allows to set the turnaround time in PHY clocks. They must be configured according to *Table 745: TRDT values*, depending on the application AHB frequency. Higher TRDT values allow stretching the USB response time to IN tokens in order to compensate for longer AHB read access latency to the data FIFO.

Note: Only accessible in device mode.

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 Reserved, must be kept at reset value.

Bit 6 Reserved, must be kept at reset value.

Bit 5 Reserved, must be kept at reset value.

Bit 4 Reserved, must be kept at reset value.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **TOCAL[2:0]:** FS timeout calibration

The number of PHY clocks that the application programs in this field is added to the full-speed interpacket timeout duration in the core to account for any additional delays introduced by the PHY. This can be required, because the delay introduced by the PHY in generating the line state condition can vary from one PHY to another.

The USB standard timeout value for full-speed operation is 16 to 18 (inclusive) bit times. The application must program this field based on the speed of enumeration. The number of bit times added per PHY clock is 0.25 bit times.

Table 745. TRDT values

AHB frequency range (MHz)		TRDT minimum value
Min	Max	
30	-	0x9

73.14.5 OTG reset register (OTG_GRSTCTL)

The application uses this register to reset various hardware features inside the core.

Address offset: 0x010

Reset value: 0x8000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AHB IDL	DMAR EQ	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	TXFNUM[4:0]					TXF FLSH	RXF FLSH	Res.	FCRST	PSRST	CSRST
					rw	rw	rw	rw	rw	rs	rs		rs	rs	rs

Bit 31 **AHBIDL:** AHB master idle

Indicates that the AHB master state machine is in the Idle condition.

Note: Accessible in both device and host modes.

Bit 30 **DMAREQ:** DMA request signal enabled

This bit indicates that the DMA request is in progress. Used for debug.

Bits 29:11 Reserved, must be kept at reset value.

Bits 10:6 **TXFNUM[4:0]:** Tx FIFO number

This is the FIFO number that must be flushed using the Tx FIFO Flush bit. This field must not be changed until the core clears the Tx FIFO Flush bit.

Condition: host mode

00000: Non-periodic Tx FIFO flush

00001: Periodic Tx FIFO flush

10000: Flush all the transmit FIFOs

Condition: device mode

00000: Tx FIFO 0 flush

00001: Tx FIFO 1 flush

00010: Tx FIFO 2 flush

...

01111: Tx FIFO 15 flush

10000: Flush all the transmit FIFOs

Note: Accessible in both device and host modes.

Bit 5 **TXFFLSH:** Tx FIFO flush

This bit selectively flushes a single or all transmit FIFOs, but cannot do so if the core is in the midst of a transaction.

The application must write this bit only after checking that the core is neither writing to the Tx FIFO nor reading from the Tx FIFO. Verify using these registers:

Read—NAK Effective interrupt ensures the core is not reading from the FIFO

Write—AHBIDL bit in OTG_GRSTCTL ensures the core is not writing anything to the FIFO.

Flushing is normally recommended when FIFOs are reconfigured. FIFO flushing is also recommended during device endpoint disable. The application must wait until the core clears this bit before performing any operations. This bit takes eight clocks to clear, using the slower clock of phy_clk or hclk.

Note: Accessible in both device and host modes.

Bit 4 **RXFFLSH:** Rx FIFO flush

The application can flush the entire Rx FIFO using this bit, but must first ensure that the core is not in the middle of a transaction.

The application must only write to this bit after checking that the core is neither reading from the Rx FIFO nor writing to the Rx FIFO.

The application must wait until the bit is cleared before performing any other operations. This bit requires 8 clocks (slowest of PHY or AHB clock) to clear.

Note: Accessible in both device and host modes.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **FCRST:** Host frame counter reset

The application writes this bit to reset the (micro-)frame number counter inside the core. When the (micro-)frame counter is reset, the subsequent SOF sent out by the core has a frame number of 0.

When application writes '1' to the bit, it might not be able to read back the value as it gets cleared by the core in a few clock cycles.

Note: Only accessible in host mode.

Bit 1 **PSRST:** Partial soft reset

Resets the internal state machines but keeps the enumeration info. Can be used to recover some specific PHY errors.

Note: Accessible in both device and host modes.

Bit 0 **CSRST:** Core soft reset

Resets the HCLK and PHY clock domains as follows:

Clears the interrupts and all the CSR register bits except for the following bits:

- GATEHCLK bit in OTG_PCGCCTL
- STPPCLK bit in OTG_PCGCCTL
- FSLSPCS bits in OTG_HCFG
- DSPD bit in OTG_DCFG
- SDIS bit in OTG_DCTL
- OTG_GCCFG register
- FDMOD bit in OTG_GUSBCFG
- FHMOD bit in OTG_GUSBCFG
- PHYLPC bit in OTG_GUSBCFG
- TSDPS bit in OTG_GUSBCFG

All module state machines (except for the AHB slave unit) are reset to the Idle state, and all the transmit FIFOs and the receive FIFO are flushed.

Any transactions on the AHB Master are terminated as soon as possible, after completing the last data phase of an AHB transfer. Any transactions on the USB are terminated immediately.

The application can write to this bit any time it wants to reset the core. This is a self-clearing bit and the core clears this bit after all the necessary logic is reset in the core, which can take several clocks, depending on the current state of the core. Once this bit has been cleared, the software must wait at least 3 PHY clocks before accessing the PHY domain (synchronization delay). The software must also check that bit 31 in this register is set to 1 (AHB Master is Idle) before starting any operation.

Typically, the software reset is used during software development and also when the user dynamically changes the PHY selection bits in the above listed USB configuration registers. When the user changes the PHY, the corresponding clock for the PHY is selected and used in the PHY domain. Once a new clock is selected, the PHY domain has to be reset for proper operation.

Note: Accessible in both device and host modes.

73.14.6 OTG core interrupt register [alternate] (OTG_GINTSTS)

Valid for Host mode, see next section for Device mode.

This register also indicates the current mode. To clear the interrupt status bits of the rc_w1 type, the application must write 1 into the bit.

This register interrupts the application for system-level events in the current mode (device mode or host mode).

The FIFO status interrupts are read-only; once software reads from or writes to the FIFO while servicing these interrupts, FIFO interrupt conditions are cleared automatically.

The application must clear the OTG_GINTSTS register at initialization before unmasking the interrupt bit to avoid any interrupts generated prior to initialization.

Address offset: 0x014

Reset value: 0x0400 0020

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WKUP INT	SRQ INT	DISC INT	CIDS CHG	LPM INT	PTXFE	HCINT	HPRT INT	RST DET	DATAF SUSP	IPXFR	IISOI XFR	OEP INT	IEPINT	Res.	Res.
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	r	r	r	rc_w1	rc_w1	rc_w1	rc_w1	r	r		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EOPF	ISOO DRP	ENUM DNE	USB RST	USB SUSP	ESUSP	Res.	Res.	GO NAK EFF	GI NAK EFF	NPTXF E	RXF LVL	SOF	OTG INT	MMIS	CMOD
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1			r	r	r	r	rc_w1	r	rc_w1	r

Bit 31 **WKUPINT:** Resume/remote wake-up detected interrupt

Wake-up interrupt during suspend(L2) or LPM(L1) state.

– During suspend(L2):

In device mode, this interrupt is asserted when a resume is detected on the USB. In host mode, this interrupt is asserted when a remote wake-up is detected on the USB.

– During LPM(L1):

This interrupt is asserted for either host initiated resume or device initiated remote wake-up on USB.

Note: Accessible in both device and host modes.

Bit 30 **SRQINT:** Session request/new session detected interrupt

In host mode, this interrupt is asserted when a session request is detected from the device. In device mode, this interrupt is asserted when V_{BUS} is in the valid range for a B-peripheral device. Accessible in both device and host modes.

Bit 29 **DISCINT:** Disconnect detected interrupt

Asserted when a device disconnect is detected.

Note: Only accessible in host mode.

Bit 28 **CIDSCHG:** Connector ID status change

The core sets this bit when there is a change in connector ID status.

Note: Accessible in both device and host modes.

Bit 27 **LPMINT:** LPM interrupt

In device mode, this interrupt is asserted when the device receives an LPM transaction and responds with a non-ERRORed response.

In host mode, this interrupt is asserted when the device responds to an LPM transaction with a non-ERRORed response or when the host core has completed LPM transactions for the programmed number of times (RETRYCNT bit in OTG_GLPMCFG).

This field is valid only if the LPMEN bit in OTG_GLPMCFG is set to 1.

Bit 26 **PTXFE:** Periodic Tx FIFO empty

Asserted when the periodic transmit FIFO is either half or completely empty and there is space for at least one entry to be written in the periodic request queue. The half or completely empty status is determined by the periodic Tx FIFO empty level bit in the OTG_GAHBCFG register (PTXFELVL bit in OTG_GAHBCFG).

Note: Only accessible in host mode.

Bit 25 **HCINT:** Host channels interrupt

The core sets this bit to indicate that an interrupt is pending on one of the channels of the core (in host mode). The application must read the OTG_HAINT register to determine the exact number of the channel on which the interrupt occurred, and then read the corresponding OTG_HCINTx register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the OTG_HCINTx register to clear this bit.

Note: Only accessible in host mode.

Bit 24 **HPRTINT:** Host port interrupt

The core sets this bit to indicate a change in port status of one of the OTG controller ports in host mode. The application must read the OTG_HPRT register to determine the exact event that caused this interrupt. The application must clear the appropriate status bit in the OTG_HPRT register to clear this bit.

Note: Only accessible in host mode.

Bit 23 **RSTDET:** Reset detected interrupt

In device mode, this interrupt is asserted when a reset is detected on the USB in partial power-down mode when the device is in suspend.

Note: Only accessible in device mode.

Bit 22 **DATAFSUSP:** Data fetch suspended

This interrupt is valid only in DMA mode. This interrupt indicates that the core has stopped fetching data for IN endpoints due to the unavailability of TxFIFO space or request queue space. This interrupt is used by the application for an endpoint mismatch algorithm. For example, after detecting an endpoint mismatch, the application:

- Sets a global nonperiodic IN NAK handshake
- Disables IN endpoints
- Flushes the FIFO
- Determines the token sequence from the IN token sequence learning queue
- Re-enables the endpoints

Clears the global nonperiodic IN NAK handshake If the global nonperiodic IN NAK is cleared, the core has not yet fetched data for the IN endpoint, and the IN token is received: the core generates an “IN token received when FIFO empty” interrupt. The OTG then sends a NAK response to the host. To avoid this scenario, the application can check the FetSusp interrupt in OTG_GINTSTS, which ensures that the FIFO is full before clearing a global NAK handshake. Alternatively, the application can mask the “IN token received when FIFO empty” interrupt when clearing a global IN NAK handshake.

Bit 21 **IPXFR:** Incomplete periodic transfer

In host mode, the core sets this interrupt bit when there are incomplete periodic transactions still pending, which are scheduled for the current frame.

Bit 20 **IISOIXFR:** Incomplete isochronous IN transfer

The core sets this interrupt to indicate that there is at least one isochronous IN endpoint on which the transfer is not completed in the current frame. This interrupt is asserted along with the End of periodic frame interrupt (EOPF) bit in this register.

Note: Only accessible in device mode.

Bit 19 **OEPINT:** OUT endpoint interrupt

The core sets this bit to indicate that an interrupt is pending on one of the OUT endpoints of the core (in device mode). The application must read the OTG_DAINT register to determine the exact number of the OUT endpoint on which the interrupt occurred, and then read the corresponding OTG_DOEPINTx register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the corresponding OTG_DOEPINTx register to clear this bit.

Note: Only accessible in device mode.

Bit 18 **IEPINT:** IN endpoint interrupt

The core sets this bit to indicate that an interrupt is pending on one of the IN endpoints of the core (in device mode). The application must read the OTG_DAINT register to determine the exact number of the IN endpoint on which the interrupt occurred, and then read the corresponding OTG_DIEPINTx register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the corresponding OTG_DIEPINTx register to clear this bit.

Note: Only accessible in device mode.

Bits 17:16 Reserved, must be kept at reset value.

Bit 15 **EOPF:** End of periodic frame interrupt

Indicates that the period specified in the periodic frame interval field of the OTG_DCFG register (PFIVL bit in OTG_DCFG) has been reached in the current frame.

Note: Only accessible in device mode.

Bit 14 **ISOODRP:** Isochronous OUT packet dropped interrupt

The core sets this bit when it fails to write an isochronous OUT packet into the Rx FIFO because the Rx FIFO does not have enough space to accommodate a maximum size packet for the isochronous OUT endpoint.

Note: Only accessible in device mode.

Bit 13 **ENUMDNE:** Enumeration done

The core sets this bit to indicate that speed enumeration is complete. The application must read the OTG_DSTS register to obtain the enumerated speed.

Note: Only accessible in device mode.

Bit 12 **USBRST:** USB reset

The core sets this bit to indicate that a reset is detected on the USB.

Note: Only accessible in device mode.

Bit 11 **USBSUSP:** USB suspend

The core sets this bit to indicate that a suspend was detected on the USB. The core enters the suspended state when there is no activity on the data lines for an extended period of time.

Note: Only accessible in device mode.

Bit 10 **ESUSP:** Early suspend

The core sets this bit to indicate that an Idle state has been detected on the USB for 3 ms.

Note: Only accessible in device mode.

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **GONAKEFF:** Global OUT NAK effective

Indicates that the Set global OUT NAK bit in the OTG_DCTL register (SGONAK bit in OTG_DCTL), set by the application, has taken effect in the core. This bit can be cleared by writing the Clear global OUT NAK bit in the OTG_DCTL register (CGONAK bit in OTG_DCTL).

Note: Only accessible in device mode.

Bit 6 **GINAKEFF:** Global IN non-periodic NAK effective

Indicates that the Set global non-periodic IN NAK bit in the OTG_DCTL register (SGINAK bit in OTG_DCTL), set by the application, has taken effect in the core. That is, the core has sampled the Global IN NAK bit set by the application. This bit can be cleared by clearing the Clear global non-periodic IN NAK bit in the OTG_DCTL register (CGINAK bit in OTG_DCTL). This interrupt does not necessarily mean that a NAK handshake is sent out on the USB. The STALL bit takes precedence over the NAK bit.

Note: Only accessible in device mode.

Bit 5 **NPTXFE:** Non-periodic Tx FIFO empty

This interrupt is asserted when the non-periodic Tx FIFO is either half or completely empty, and there is space for at least one entry to be written to the non-periodic transmit request queue. The half or completely empty status is determined by the non-periodic Tx FIFO empty level bit in the OTG_GAHBCFG register (TXFELVL bit in OTG_GAHBCFG).

Note: Accessible in host mode only.

Bit 4 **RXFLVL:** Rx FIFO non-empty

Indicates that there is at least one packet pending to be read from the Rx FIFO.

Note: Accessible in both host and device modes.

Bit 3 **SOF:** Start of frame

In host mode, the core sets this bit to indicate that an SOF (FS), or Keep-Alive (LS) is transmitted on the USB. The application must write a 1 to this bit to clear the interrupt.

In device mode, in the core sets this bit to indicate that an SOF token has been received on the USB. The application can read the OTG_DSTS register to get the current frame number. This interrupt is seen only when the core is operating in FS.

Note: This register may return '1' if read immediately after power on reset. If the register bit reads '1' immediately after power on reset it does not indicate that an SOF has been sent (in case of host mode) or SOF has been received (in case of device mode). The read value of this interrupt is valid only after a valid connection between host and device is established. If the bit is set after power on reset the application can clear the bit.

Note: Accessible in both host and device modes.

Bit 2 **OTGINT:** OTG interrupt

The core sets this bit to indicate an OTG protocol event. The application must read the OTG interrupt status (OTG_GOTGINT) register to determine the exact event that caused this interrupt. The application must clear the appropriate status bit in the OTG_GOTGINT register to clear this bit.

Note: Accessible in both host and device modes.

Bit 1 **MMIS:** Mode mismatch interrupt

The core sets this bit when the application is trying to access:

– A host mode register, when the core is operating in device mode

– A device mode register, when the core is operating in host mode

The register access is completed on the AHB with an OKAY response, but is ignored by the core internally and does not affect the operation of the core.

Note: Accessible in both host and device modes.

Bit 0 **CMOD:** Current mode of operation

Indicates the current mode.

0: Device mode

1: Host mode

Note: Accessible in both host and device modes.

73.14.7 OTG core interrupt register [alternate] (OTG_GINTSTS)

Valid for Device mode, see previous section for Host mode.

This register also indicates the current mode. To clear the interrupt status bits of the rc_w1 type, the application must write 1 into the bit.

This register interrupts the application for system-level events in the current mode (device mode or host mode).

The FIFO status interrupts are read-only; once software reads from or writes to the FIFO while servicing these interrupts, FIFO interrupt conditions are cleared automatically.

The application must clear the OTG_GINTSTS register at initialization before unmasking the interrupt bit to avoid any interrupts generated prior to initialization.

Address offset: 0x014

Reset value: 0x0400 0020

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WKUP INT	SRQ INT	DISC INT	CIDS CHG	LPM INT	PTXFE	HCINT	HPRT INT	RST DET	DATAF SUSP	IN COMP ISO OUT	IISOI XFR	OEP INT	IEPINT	Res.	Res.
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	r	r	r	rc_w1	rc_w1	rc_w1	rc_w1	r	r		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EOPF	ISOO DRP	ENUM DNE	USB RST	USB SUSP	ESUSP	Res.	Res.	GO NAK EFF	GI NAK EFF	NPTXF E	RXF LVL	SOF	OTG INT	MMIS	CMOD
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1			r	r	r	r	rc_w1	r	rc_w1	r

Bit 31 **WKUPINT:** Resume/remote wake-up detected interrupt

Wake-up interrupt during suspend(L2) or LPM(L1) state.

– During suspend(L2):

In device mode, this interrupt is asserted when a resume is detected on the USB. In host mode, this interrupt is asserted when a remote wake-up is detected on the USB.

– During LPM(L1):

This interrupt is asserted for either host initiated resume or device initiated remote wake-up on USB.

Note: Accessible in both device and host modes.

Bit 30 **SRQINT:** Session request/new session detected interrupt

In host mode, this interrupt is asserted when a session request is detected from the device. In device mode, this interrupt is asserted when V_{BUS} is in the valid range for a B-peripheral device. Accessible in both device and host modes.

Bit 29 **DISCINT:** Disconnect detected interrupt

Asserted when a device disconnect is detected.

Note: Only accessible in host mode.

Bit 28 **CIDSCHG:** Connector ID status change

The core sets this bit when there is a change in connector ID status.

Note: Accessible in both device and host modes.

Bit 27 **LPMINT:** LPM interrupt

In device mode, this interrupt is asserted when the device receives an LPM transaction and responds with a non-ERRORed response.

In host mode, this interrupt is asserted when the device responds to an LPM transaction with a non-ERRORed response or when the host core has completed LPM transactions for the programmed number of times (RETRYCNT bit in OTG_GLPMCFG).

This field is valid only if the LPMEN bit in OTG_GLPMCFG is set to 1.

Bit 26 **PTXFE:** Periodic Tx FIFO empty

Asserted when the periodic transmit FIFO is either half or completely empty and there is space for at least one entry to be written in the periodic request queue. The half or completely empty status is determined by the periodic Tx FIFO empty level bit in the OTG_GAHBCFG register (PTXFELVL bit in OTG_GAHBCFG).

Note: Only accessible in host mode.

Bit 25 **HCINT:** Host channels interrupt

The core sets this bit to indicate that an interrupt is pending on one of the channels of the core (in host mode). The application must read the OTG_HAINT register to determine the exact number of the channel on which the interrupt occurred, and then read the corresponding OTG_HCINTx register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the OTG_HCINTx register to clear this bit.

Note: Only accessible in host mode.

Bit 24 **HPRTINT:** Host port interrupt

The core sets this bit to indicate a change in port status of one of the OTG controller ports in host mode. The application must read the OTG_HPRT register to determine the exact event that caused this interrupt. The application must clear the appropriate status bit in the OTG_HPRT register to clear this bit.

Note: Only accessible in host mode.

Bit 23 **RSTDET:** Reset detected interrupt

In device mode, this interrupt is asserted when a reset is detected on the USB in partial power-down mode when the device is in suspend.

Note: Only accessible in device mode.

Bit 22 **DATAFSUSP:** Data fetch suspended

This interrupt is valid only in DMA mode. This interrupt indicates that the core has stopped fetching data for IN endpoints due to the unavailability of TxFIFO space or request queue space. This interrupt is used by the application for an endpoint mismatch algorithm. For example, after detecting an endpoint mismatch, the application:

- Sets a global nonperiodic IN NAK handshake
- Disables IN endpoints
- Flushes the FIFO
- Determines the token sequence from the IN token sequence learning queue
- Re-enables the endpoints

Clears the global nonperiodic IN NAK handshake If the global nonperiodic IN NAK is cleared, the core has not yet fetched data for the IN endpoint, and the IN token is received: the core generates an “IN token received when FIFO empty” interrupt. The OTG then sends a NAK response to the host. To avoid this scenario, the application can check the FetSusp interrupt in OTG_GINTSTS, which ensures that the FIFO is full before clearing a global NAK handshake. Alternatively, the application can mask the “IN token received when FIFO empty” interrupt when clearing a global IN NAK handshake.

Bit 21 **INCOMPISOOUT:** Incomplete isochronous OUT transfer

In device mode, the core sets this interrupt to indicate that there is at least one isochronous OUT endpoint on which the transfer is not completed in the current frame. This interrupt is asserted along with the End of periodic frame interrupt (EOPF) bit in this register.

Bit 20 **IISOIXFR:** Incomplete isochronous IN transfer

The core sets this interrupt to indicate that there is at least one isochronous IN endpoint on which the transfer is not completed in the current frame. This interrupt is asserted along with the End of periodic frame interrupt (EOPF) bit in this register.

Note: Only accessible in device mode.

Bit 19 **OEPINT:** OUT endpoint interrupt

The core sets this bit to indicate that an interrupt is pending on one of the OUT endpoints of the core (in device mode). The application must read the OTG_DAINT register to determine the exact number of the OUT endpoint on which the interrupt occurred, and then read the corresponding OTG_DOEPINTx register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the corresponding OTG_DOEPINTx register to clear this bit.

Note: Only accessible in device mode.

Bit 18 **IEPINT:** IN endpoint interrupt

The core sets this bit to indicate that an interrupt is pending on one of the IN endpoints of the core (in device mode). The application must read the OTG_DAINT register to determine the exact number of the IN endpoint on which the interrupt occurred, and then read the corresponding OTG_DIEPINTx register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the corresponding OTG_DIEPINTx register to clear this bit.

Note: Only accessible in device mode.

Bits 17:16 Reserved, must be kept at reset value.

Bit 15 **EOPF:** End of periodic frame interrupt

Indicates that the period specified in the periodic frame interval field of the OTG_DCFG register (PFIVL bit in OTG_DCFG) has been reached in the current frame.

Note: Only accessible in device mode.

Bit 14 **ISOODRP:** Isochronous OUT packet dropped interrupt

The core sets this bit when it fails to write an isochronous OUT packet into the Rx FIFO because the Rx FIFO does not have enough space to accommodate a maximum size packet for the isochronous OUT endpoint.

Note: Only accessible in device mode.

Bit 13 **ENUMDNE:** Enumeration done

The core sets this bit to indicate that speed enumeration is complete. The application must read the OTG_DSTS register to obtain the enumerated speed.

Note: Only accessible in device mode.

Bit 12 **USBRST:** USB reset

The core sets this bit to indicate that a reset is detected on the USB.

Note: Only accessible in device mode.

Bit 11 **USBSUSP:** USB suspend

The core sets this bit to indicate that a suspend was detected on the USB. The core enters the suspended state when there is no activity on the data lines for an extended period of time.

Note: Only accessible in device mode.

Bit 10 **ESUSP:** Early suspend

The core sets this bit to indicate that an Idle state has been detected on the USB for 3 ms.

Note: Only accessible in device mode.

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **GONAKEFF:** Global OUT NAK effective

Indicates that the Set global OUT NAK bit in the OTG_DCTL register (SGONAK bit in OTG_DCTL), set by the application, has taken effect in the core. This bit can be cleared by writing the Clear global OUT NAK bit in the OTG_DCTL register (CGONAK bit in OTG_DCTL).

Note: Only accessible in device mode.

Bit 6 **GINAKEFF:** Global IN non-periodic NAK effective

Indicates that the Set global non-periodic IN NAK bit in the OTG_DCTL register (SGINAK bit in OTG_DCTL), set by the application, has taken effect in the core. That is, the core has sampled the Global IN NAK bit set by the application. This bit can be cleared by clearing the Clear global non-periodic IN NAK bit in the OTG_DCTL register (CGINAK bit in OTG_DCTL). This interrupt does not necessarily mean that a NAK handshake is sent out on the USB. The STALL bit takes precedence over the NAK bit.

Note: Only accessible in device mode.

Bit 5 **NPTXFE:** Non-periodic Tx FIFO empty

This interrupt is asserted when the non-periodic Tx FIFO is either half or completely empty, and there is space for at least one entry to be written to the non-periodic transmit request queue. The half or completely empty status is determined by the non-periodic Tx FIFO empty level bit in the OTG_GAHBCFG register (TXFELVL bit in OTG_GAHBCFG).

Note: Accessible in host mode only.

Bit 4 **RXFLVL:** Rx FIFO non-empty

Indicates that there is at least one packet pending to be read from the Rx FIFO.

Note: Accessible in both host and device modes.

Bit 3 **SOF:** Start of frame

In host mode, the core sets this bit to indicate that an SOF (FS), or Keep-Alive (LS) is transmitted on the USB. The application must write a 1 to this bit to clear the interrupt.

In device mode, in the core sets this bit to indicate that an SOF token has been received on the USB. The application can read the OTG_DSTS register to get the current frame number. This interrupt is seen only when the core is operating in FS.

Note: This register may return '1' if read immediately after power on reset. If the register bit reads '1' immediately after power on reset it does not indicate that an SOF has been sent (in case of host mode) or SOF has been received (in case of device mode). The read value of this interrupt is valid only after a valid connection between host and device is established. If the bit is set after power on reset the application can clear the bit.

Note: Accessible in both host and device modes.

Bit 2 **OTGINT:** OTG interrupt

The core sets this bit to indicate an OTG protocol event. The application must read the OTG interrupt status (OTG_GOTGINT) register to determine the exact event that caused this interrupt. The application must clear the appropriate status bit in the OTG_GOTGINT register to clear this bit.

Note: Accessible in both host and device modes.

Bit 1 **MMIS:** Mode mismatch interrupt

The core sets this bit when the application is trying to access:

– A host mode register, when the core is operating in device mode

– A device mode register, when the core is operating in host mode

The register access is completed on the AHB with an OKAY response, but is ignored by the core internally and does not affect the operation of the core.

Note: Accessible in both host and device modes.

Bit 0 **CMOD:** Current mode of operation

Indicates the current mode.

0: Device mode

1: Host mode

Note: Accessible in both host and device modes.

73.14.8 OTG interrupt mask register [alternate] (OTG_GINTMSK)

Valid for Host mode, see next section for Device mode.

This register works with the core interrupt register to interrupt the application. When an interrupt bit is masked, the interrupt associated with that bit is not generated. However, the core interrupt (OTG_GINTSTS) register bit corresponding to that interrupt is still set.

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WUIM	SRQIM	DISCINT	CIDSCHGM	LPMINTM	PTXFEM	HCIM	PRTIM	Res.	Res.	IPXFRM	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	r			rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NPTXFEM	RXFLVLM	SOFM	OTGINT	MMISM	Res.
										rw	rw	rw	rw	rw	

Bit 31 **WUIM:** Resume/remote wake-up detected interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 30 **SRQIM:** Session request/new session detected interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 29 **DISCINT:** Disconnect detected interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 28 **CIDSCHGM:** Connector ID status change mask
0: Masked interrupt
1: Unmasked interrupt

Bit 27 **LPMINTM:** LPM interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 26 **PTXFEM:** Periodic Tx FIFO empty mask
0: Masked interrupt
1: Unmasked interrupt

Bit 25 **HCIM:** Host channels interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 24 **PRTIM:** Host port interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **IPXFRM:** Incomplete periodic transfer mask
0: Masked interrupt
1: Unmasked interrupt

Bits 20:6 Reserved, must be kept at reset value.

Bit 5 **NPTXFEM:** Non-periodic Tx FIFO empty mask
0: Masked interrupt
1: Unmasked interrupt

Bit 4 **RXFLVLM:** Receive FIFO non-empty mask
0: Masked interrupt
1: Unmasked interrupt

Bit 3 **SOFM:** Start of frame mask
0: Masked interrupt
1: Unmasked interrupt

Bit 2 **OTGINT:** OTG interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 1 **MMISM:** Mode mismatch interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 0 Reserved, must be kept at reset value.

73.14.9 OTG interrupt mask register [alternate] (OTG_GINTMSK)

Valid for Device mode, see previous section for Host mode.

This register works with the core interrupt register to interrupt the application. When an interrupt bit is masked, the interrupt associated with that bit is not generated. However, the core interrupt (OTG_GINTSTS) register bit corresponding to that interrupt is still set.

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
WUIM	SRQIM	Res.	CIDSCHGM	LPMINTM	Res.	Res.	Res.	RSTDETM	FSUSPM	IISOOXFRM	IISOIXFRM	OEPINT	IEPINT	Res.	Res.
rw	rw		rw	rw				rw	rw	rw	rw	rw	rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EOPFM	ISOODRPM	ENUMDNEM	USBRST	USBSUSPM	ESUSPM	Res.	Res.	GONAKEFFM	GINAKEFFM	Res.	RXFLVLM	SOFM	OTGINT	MMISM	Res.
rw	rw	rw	rw	rw	rw			rw	rw		rw	rw	rw	rw	

Bit 31 **WUIM:** Resume/remote wake-up detected interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 30 **SRQIM:** Session request/new session detected interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 29 Reserved, must be kept at reset value.

Bit 28 **CIDSCHGM:** Connector ID status change mask
0: Masked interrupt
1: Unmasked interrupt

Bit 27 **LPMINTM:** LPM interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bits 26:24 Reserved, must be kept at reset value.

Bit 23 **RSTDETM:** Reset detected interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 22 **FSUSPM:** Data fetch suspended mask
0: Masked interrupt
1: Unmasked interrupt

Bit 21 **IISOOXFRM:** Incomplete isochronous OUT transfer mask
0: Masked interrupt
1: Unmasked interrupt

Bit 20 **IISOIXFRM:** Incomplete isochronous IN transfer mask
0: Masked interrupt
1: Unmasked interrupt

Bit 19 **OEPINT:** OUT endpoints interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 18 **IEPINT:** IN endpoints interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bits 17:16 Reserved, must be kept at reset value.

Bit 15 **EOPFM:** End of periodic frame interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 14 **ISOODRPM:** Isochronous OUT packet dropped interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 13 **ENUMDNEM:** Enumeration done mask
0: Masked interrupt
1: Unmasked interrupt

Bit 12 **USBRST:** USB reset mask
0: Masked interrupt
1: Unmasked interrupt

Bit 11 **USBSUSPM:** USB suspend mask
0: Masked interrupt
1: Unmasked interrupt

Bit 10 **ESUSPM:** Early suspend mask
0: Masked interrupt
1: Unmasked interrupt

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **GONAKEFFM:** Global OUT NAK effective mask
0: Masked interrupt
1: Unmasked interrupt

Bit 6 **GINAKEFFM:** Global non-periodic IN NAK effective mask
0: Masked interrupt
1: Unmasked interrupt

Bit 5 Reserved, must be kept at reset value.

Bit 4 **RXFLVLM:** Receive FIFO non-empty mask
0: Masked interrupt
1: Unmasked interrupt

Bit 3 **SOFM:** Start of frame mask
0: Masked interrupt
1: Unmasked interrupt

Bit 2 **OTGINT:** OTG interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 1 **MMISM:** Mode mismatch interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 0 Reserved, must be kept at reset value.

73.14.10 OTG receive status debug read register [alternate] (OTG_GRXSTSR)

This description is for register OTG_GRXSTSR in Device mode.

A read to the receive status debug read register returns the contents of the top of the receive FIFO.

The core ignores the receive status read when the receive FIFO is empty and returns a value of 0x0000 0000.

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	STSPHST	Res.	Res.	FRMNUM[3:0]				PKTSTS[3:0]				DPID[1]
				r			r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DPID[0]	BCNT[10:0]											EPNUM[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **STSPHST:** Status phase start
Indicates the start of the status phase for a control write transfer. This bit is set along with the OUT transfer completed PKTSTS pattern.

Bits 26:25 Reserved, must be kept at reset value.

Bits 24:21 **FRMNUM[3:0]:** Frame number

This is the least significant 4 bits of the frame number in which the packet is received on the USB. This field is supported only when isochronous OUT endpoints are supported.

Bits 20:17 **PKTSTS[3:0]:** Packet status

Indicates the status of the received packet
0001: Global OUT NAK (triggers an interrupt)
0010: OUT data packet received
0011: OUT transfer completed (triggers an interrupt)
0100: SETUP transaction completed (triggers an interrupt)
0110: SETUP data packet received
Others: Reserved

Bits 16:15 **DPID[1:0]:** Data PID

Indicates the data PID of the received OUT data packet
00: DATA0
10: DATA1
01: DATA2
11: MDATA

Bits 14:4 **BCNT[10:0]:** Byte count

Indicates the byte count of the received data packet.

Bits 3:0 **EPNUM[3:0]:** Endpoint number

Indicates the endpoint number to which the current received packet belongs.

73.14.11 OTG receive status debug read register [alternate] (OTG_GRXSTSR)

This description is for register OTG_GRXSTSR in Host mode.

A read to the receive status debug read register returns the contents of the top of the receive FIFO.

The core ignores the receive status read when the receive FIFO is empty and returns a value of 0x0000 0000.

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKTSTS[3:0]				DPID
											r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DPID	BCNT[10:0]											CHNUM[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:17 **PKTSTS[3:0]:** Packet status
Indicates the status of the received packet
0010: IN data packet received
0011: IN transfer completed (triggers an interrupt)
0101: Data toggle error (triggers an interrupt)
0111: Channel halted (triggers an interrupt)
Others: Reserved

Bits 16:15 **DPID[1:0]:** Data PID
Indicates the data PID of the received packet
00: DATA0
10: DATA1
01: DATA2
11: MDATA

Bits 14:4 **BCNT[10:0]:** Byte count
Indicates the byte count of the received IN data packet.

Bits 3:0 **CHNUM[3:0]:** Channel number
Indicates the channel number to which the current received packet belongs.

73.14.12 OTG status read and pop registers (OTG_GRXSTSP)

This description is for register OTG_GRXSTSP in Device mode.

Similarly to OTG_GRXSTSR (receive status debug read register) where a read returns the contents of the top of the receive FIFO, a read to OTG_GRXSTSP (receive status read and pop register) additionally pops the top data entry out of the Rx FIFO.

The core ignores the receive status pop/read when the receive FIFO is empty and returns a value of 0x0000 0000. The application must only pop the receive status FIFO when the receive FIFO non-empty bit of the core interrupt register (RXFLVL bit in OTG_GINTSTS) is asserted.

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	STSPHST	Res.	Res.	FRMNUM[3:0]				PKTSTS[3:0]				DPID[1]
				r			r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DPID[0]	BCNT[10:0]											EPNUM[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **STSPHST:** Status phase start
Indicates the start of the status phase for a control write transfer. This bit is set along with the OUT transfer completed PKTSTS pattern.

Bits 26:25 Reserved, must be kept at reset value.

Bits 24:21 **FRMNUM[3:0]:** Frame number

This is the least significant 4 bits of the frame number in which the packet is received on the USB. This field is supported only when isochronous OUT endpoints are supported.

Bits 20:17 **PKTSTS[3:0]:** Packet status

Indicates the status of the received packet
0001: Global OUT NAK (triggers an interrupt)
0010: OUT data packet received
0011: OUT transfer completed (triggers an interrupt)
0100: SETUP transaction completed (triggers an interrupt)
0110: SETUP data packet received
Others: Reserved

Bits 16:15 **DPID[1:0]:** Data PID

Indicates the data PID of the received OUT data packet
00: DATA0
10: DATA1
01: DATA2
11: MDATA

Bits 14:4 **BCNT[10:0]:** Byte count

Indicates the byte count of the received data packet.

Bits 3:0 **EPNUM[3:0]:** Endpoint number

Indicates the endpoint number to which the current received packet belongs.

73.14.13 OTG status read and pop registers [alternate] (OTG_GRXSTSP)

This description is for register OTG_GRXSTSP in Host mode.

Similarly to OTG_GRXSTSR (receive status debug read register) where a read returns the contents of the top of the receive FIFO, a read to OTG_GRXSTSP (receive status read and pop register) additionally pops the top data entry out of the Rx FIFO.

The core ignores the receive status pop/read when the receive FIFO is empty and returns a value of 0x0000 0000. The application must only pop the receive status FIFO when the receive FIFO non-empty bit of the core interrupt register (RXFLVL bit in OTG_GINTSTS) is asserted.

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKTSTS[3:0]				DPID
											r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DPID	BCNT[10:0]											CHNUM[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:17 **PKTSTS[3:0]:** Packet status
Indicates the status of the received packet
0010: IN data packet received
0011: IN transfer completed (triggers an interrupt)
0101: Data toggle error (triggers an interrupt)
0111: Channel halted (triggers an interrupt)
Others: Reserved

Bits 16:15 **DPID[1:0]:** Data PID
Indicates the data PID of the received packet
00: DATA0
10: DATA1
01: DATA2
11: MDATA

Bits 14:4 **BCNT[10:0]:** Byte count
Indicates the byte count of the received IN data packet.

Bits 3:0 **CHNUM[3:0]:** Channel number
Indicates the channel number to which the current received packet belongs.

73.14.14 OTG receive FIFO size register (OTG_GRXFSIZ)

The application can program the RAM size that must be allocated to the Rx FIFO.

Address offset: 0x024

Reset value: 0x0000 0400

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXFD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **RXFD[15:0]:** Rx FIFO depth
This value is in terms of 32-bit words.
Maximum value is 1024
Programmed values must respect the available FIFO memory allocation and must not exceed the power-on value.

73.14.15 OTG host non-periodic transmit FIFO size register [alternate] (OTG_HNPTXFSIZ)

Valid for Host mode, see next section for Device mode.

Address offset: 0x028

Reset value: 0x0200 0200

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NPTXFD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NPTXFSA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **NPTXFD[15:0]:** Non-periodic Tx FIFO depth
This value is in terms of 32-bit words.
Minimum value is 16
Programmed values must respect the available FIFO memory allocation and must not exceed the power-on value.

Bits 15:0 **NPTXFSA[15:0]:** Non-periodic transmit RAM start address
This field configures the memory start address for non-periodic transmit FIFO RAM.

73.14.16 Endpoint 0 transmit FIFO size [alternate] (OTG_DIEPTXF0)

Valid for Device mode, see previous section for Host mode.

Address offset: 0x028

Reset value: 0x0200 0200

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TX0FD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TX0FSA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **TX0FD[15:0]:** Endpoint 0 Tx FIFO depth
This value is in terms of 32-bit words.
Minimum value is 16
Programmed values must respect the available FIFO memory allocation and must not exceed the power-on value.

Bits 15:0 **TX0FSA[15:0]:** Endpoint 0 transmit RAM start address
This field configures the memory start address for the endpoint 0 transmit FIFO RAM.

73.14.17 OTG non-periodic transmit FIFO/queue status register (OTG_HNPTXSTS)

Note: In device mode, this register is not valid.

This read-only register contains the free space information for the non-periodic Tx FIFO and the non-periodic transmit request queue.

Address offset: 0x02C

Reset value: 0x0008 0400

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	NPTXQTOP[6:0]							NPTQXSAV[7:0]							
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NPTXFSAV[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bits 30:24 **NPTXQTOP[6:0]:** Top of the non-periodic transmit request queue
Entry in the non-periodic Tx request queue that is currently being processed by the MAC.
Bits 30:27: Channel/endpoint number
Bits 26:25:
XXXX00X: IN/OUT token
XXXX01X: Zero-length transmit packet (device IN/host OUT)
XXXX11X: Channel halt command
Bit 24: Terminate (last entry for selected channel/endpoint)

Bits 23:16 **NPTQXSAV[7:0]:** Non-periodic transmit request queue space available
Indicates the amount of free space available in the non-periodic transmit request queue. This queue holds both IN and OUT requests.
0: Non-periodic transmit request queue is full
1: 1 location available
2: locations available
n: n locations available (0 ≤ n ≤ 8)
Others: Reserved

Bits 15:0 **NPTXFSAV[15:0]:** Non-periodic Tx FIFO space available
Indicates the amount of free space available in the non-periodic Tx FIFO.
Values are in terms of 32-bit words.
0: Non-periodic Tx FIFO is full
1: 1 word available
2: 2 words available
n: n words available (where 0 ≤ n ≤ 512)
Others: Reserved

73.14.18 OTG general core configuration register (OTG_GCCFG)

This register is available in host and device modes.

Address offset: 0x038

Reset value: 0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	IDPULLUPDIS	Res.	BCDEN	FORCEHOSTPD	Res.	VBVALOVAL	SDEN	Res.	PDEN	DCDEN	HVDMSRCEN	HCDPDETEN	HCDPEN
			rw		rw	rw		rw	rw		rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBUSVLD	Res.	FSVMINUS	FSVPLUS	CHGDET
											r		r	r	r

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **IDPULLUPDIS:** ID pin pull-up disable
- 0: Enable ID pin pull-up
- 1: Disable ID pin pull-up

Bit 27 Reserved, must be kept at reset value.

Bit 26 **BCDEN:** Force Battery charging (BC) mode

This bit is set by the software to force the Battery charging mode. When enabled, the USB PHY is fully controlled by Battery Charging controls and cannot be used for normal communication.
- 0: Normal USB mode
- 1: Battery Charging mode

Bit 25 **FORCEHOSTPD:** Force host mode pull-downs

If the ID pin functions are enabled, the host mode pull-downs on DP and DM activate automatically. However, whenever that is not the case, yet host mode is required, this bit must be used to force the pull-downs active.
- 0: Do not force host mode pull-downs
- 1: Force host mode pull-downs

Bit 24 Reserved, must be kept at reset value.

Bit 23 **VBVALOVAL:** Software override value of the VBUS B-session detection

Note: In this device, the software override is always active (no hardware detection is available).
- 0: B-session inactive
- 1: B-session active

Bit 22 **SDEN:** Battery charging specification – secondary detection enable
- 0: Secondary detection disabled
- 1: Secondary detection enabled

Bit 21 Reserved, must be kept at reset value.

Bit 20 **PDEN:** Battery charging specification – primary detection enable
- 0: Primary detection disabled
- 1: Primary detection enabled

Bit 19 **DCDEN:** Battery charging specification – data contact detection enable
0: Data Contact Detection disabled
1: Data Contact Detection enabled

Bit 18 **HVDMSRCEN:** Battery charging specification – host CDP port voltage source enable on DM
0: DM voltage source disabled
1: DM Voltage source enabled

Bit 17 **HCDPDETEN:** Battery charging specification – host CDP port voltage detector enable on DP
0: DP voltage detection disabled
1: DP voltage detection enabled

Bit 16 **HCDPEN:** Battery charging specification – host CDP behavior enable
0: Disable CDP behavior
1: Enable CDP behavior

Bits 15:5 Reserved, must be kept at reset value.

Bit 4 **VBUSVLD:** VBUS valid indicator
Indicates if VBUS is above VBUS valid threshold.
0: VBUS is below VBUS valid threshold
1: VBUS is above VBUS valid threshold

Bit 3 Reserved, must be kept at reset value.

Bit 2 **FSVMINUS:** Battery charging specification – single-ended DM indicator
This bit gives the voltage level on DM (also result of the comparison with V_{LGC} threshold as defined in BC v1.2 standard).
0: DM voltage at low level
1: DM voltage at high level

Bit 1 **FSVPLUS:** Battery charging specification – single-ended DP indicator
This bit gives the voltage level on DP (also result of the comparison with V_{LGC} threshold as defined in BC v1.2 standard).
0: DM voltage at low level
1: DM voltage at high level

Bit 0 **CHGDET:** Battery charging specification – charger detection, result of the current mode (primary or secondary).
0: Low value on pin
1: High value on pin

73.14.19 OTG core ID register (OTG_CID)

This is a register containing the Product ID as reset value.

Address offset: 0x03C

Reset value: 0x0000 5000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRODUCT_ID[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRODUCT_ID[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PRODUCT_ID[31:0]:** Product ID field
Application-programmable ID field.

73.14.20 OTG core LPM configuration register (OTG_GLPMCFG)

Address offset: 0x054

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	EN BESL	LPMRCNTSTS[2:0]			SND LPM	LPMRCNT[2:0]			LPMCHIDX[3:0]				L1RSM OK
			rw	r	r	r	rs	rw	rw	rw	rw	rw	rw	rw	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SLP STS	LPMRSP[1:0]		L1DS EN	BESLTHRS[3:0]				L1SS EN	REM WAKE	BESL[3:0]				LPM ACK	LPM EN
r	r	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **ENBESL:** Enable best effort service latency
This bit enables the BESL feature as defined in the LPM errata:
0: The core works as described in the following document:
USB 2.0 Link Power Management Addendum Engineering Change Notice to the USB 2.0 specification, July 16, 2007
1: The core works as described in the LPM Errata:
Errata for USB 2.0 ECN: Link Power Management (LPM) - 7/2007
Note: Only the updated behavior (described in LPM Errata) is considered in this document and so the ENBESL bit must be set to '1' by application SW.

Bits 27:25 **LPMRCNTSTS[2:0]:** LPM retry count status
Number of LPM host retries still remaining to be transmitted for the current LPM sequence.
Note: Accessible only in host mode.

Bit 24 **SNDLPM:** Send LPM transaction
When the application software sets this bit, an LPM transaction containing two tokens, EXT and LPM is sent. The hardware clears this bit once a valid response (STALL, NYET, or ACK) is received from the device or the core has finished transmitting the programmed number of LPM retries.
Note: This bit must be set only when the host is connected to a local port.
Note: Accessible only in host mode.

Bits 23:21 **LPMRCNT[2:0]:** LPM retry count
When the device gives an ERROR response, this is the number of additional LPM retries that the host performs until a valid device response (STALL, NYET, or ACK) is received.
Note: Accessible only in host mode.

Bits 20:17 **LPMCHIDX[3:0]:** LPM Channel Index
The channel number on which the LPM transaction has to be applied while sending an LPM transaction to the local device. Based on the LPM channel index, the core automatically inserts the device address and endpoint number programmed in the corresponding channel into the LPM transaction.
Note: Accessible only in host mode.

Bit 16 **L1RSMOK:** Sleep state resume OK

Indicates that the device or host can start resume from Sleep state. This bit is valid in LPM sleep (L1) state. It is set in sleep mode after a delay of 50 µs ($T_{L1Residency}$).
This bit is reset when SLPSTS is 0.
1: The application or host can start resume from Sleep state
0: The application or host cannot start resume from Sleep state

Bit 15 **SLPSTS:** Port sleep status

Device mode:
This bit is set as long as a Sleep condition is present on the USB bus. The core enters the Sleep state when an ACK response is sent to an LPM transaction and the $T_{L1TokenRetry}$ timer has expired. To stop the PHY clock, the application must set the STPPCLK bit in OTG_PCGCCTL, which asserts the PHY suspend input signal.
The application must rely on SLPSTS and not ACK in LPMRSP to confirm transition into sleep.
The core comes out of sleep:
– When there is any activity on the USB linestate
– When the application writes to the RWUSIG bit in OTG_DCTL or when the application resets or soft-disconnects the device.

Host mode:
The host transitions to Sleep (L1) state as a side-effect of a successful LPM transaction by the core to the local port with ACK response from the device. The read value of this bit reflects the current Sleep status of the port.
The core clears this bit after:
– The core detects a remote L1 wake-up signal,
– The application sets the PRST bit or the PRES bit in the OTG_HPRT register, or
– The application sets the L1Resume/ remote wake-up detected interrupt bit or disconnect detected interrupt bit in the core interrupt register (WKUPINT or DISCINT bit in OTG_GINTSTS, respectively).
0: Core not in L1
1: Core in L1

Bits 14:13 **LPMRSP[1:0]:** LPM response

Device mode:
The response of the core to LPM transaction received is reflected in these two bits.
Host mode:
Handshake response received from local device for LPM transaction
11: ACK
10: NYET
01: STALL
00: ERROR (No handshake response)

Bit 12 **L1DSEN:** L1 deep sleep enable

Enables suspending the PHY in L1 Sleep mode. For maximum power saving during L1 Sleep mode, this bit must be set to '1' by application SW in all the cases.

Bits 11:8 **BESLTHRS[3:0]:** BESL threshold

Device mode:

The core puts the PHY into deep low power mode in L1 when BESL value is greater than or equal to the value defined in this field BESL_Thres[3:0].

Host mode:

The core puts the PHY into deep low power mode in L1. BESLTHRS[3:0] specifies the time for which resume signaling is to be reflected by host ($T_{L1HubDrvResume2}$) on the USB bus when it detects device initiated resume.

BESLTHRS must not be programmed with a value greater than 1100b in host mode, because this exceeds maximum $T_{L1HubDrvResume2}$.

Thres[3:0] host mode resume signaling time (µs):

0000: 75

0001: 100

0010: 150

0011: 250

0100: 350

0101: 450

0110: 950

All other values: reserved

Bit 7 **L1SSEN:** L1 Shallow Sleep enable

Enables suspending the PHY in L1 Sleep mode. For maximum power saving during L1 Sleep mode, this bit must be set to '1' by application SW in all the cases.

Bit 6 **REMWAKE:** bRemoteWake value

Host mode:

The value of remote wake up to be sent in the wIndex field of LPM transaction.

Device mode (read-only):

This field is updated with the received LPM token bRemoteWake bmAttribute when an ACK, NYET, or STALL response is sent to an LPM transaction.

Bits 5:2 **BESL[3:0]:** Best effort service latency

Host mode:

The value of BESL to be sent in an LPM transaction. This value is also used to initiate resume for a duration $T_{L1HubDrvResume1}$ for host initiated resume.

Device mode (read-only):

This field is updated with the received LPM token BESL bmAttribute when an ACK, NYET, or STALL response is sent to an LPM transaction.

BESL[3:0]T_{BESL} (µs)

0000: 125
0001: 150
0010: 200
0011: 300
0100: 400
0101: 500
0110: 1000
0111: 2000
1000: 3000
1001: 4000
1010: 5000
1011: 6000
1100: 7000
1101: 8000
1110: 9000
1111: 10000

Bit 1 **LPMACK:** LPM token acknowledge enable

Handshake response to LPM token preprogrammed by device application software.

1: ACK

Even though ACK is preprogrammed, the core device responds with ACK only on successful LPM transaction. The LPM transaction is successful if:

– No PID/CRC5 errors in either EXT token or LPM token (else ERROR)
– Valid bLinkState = 0001B (L1) received in LPM transaction (else STALL)
– No data pending in transmit queue (else NYET).

0: NYET

The preprogrammed software bit is over-ridden for response to LPM token when:

– The received bLinkState is not L1 (STALL response), or
– An error is detected in either of the LPM token packets because of corruption (ERROR response).

Note: Accessible only in device mode.

Bit 0 **LPMEN:** LPM support enable

The application uses this bit to control the OTG core LPM capabilities.

If the core operates as a non-LPM-capable host, it cannot request the connected device or hub to activate LPM mode.

If the core operates as a non-LPM-capable device, it cannot respond to any LPM transactions.

0: LPM capability is not enabled
1: LPM capability is enabled

73.14.21 OTG host periodic transmit FIFO size register (OTG_HPTXFSIZ)

Address offset: 0x100

Reset value: 0x0400 0800

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PTXFSIZ[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PTXSA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **PTXFSIZ[15:0]:** Host periodic Tx FIFO depth
This value is in terms of 32-bit words.
Minimum value is 16

Bits 15:0 **PTXSA[15:0]:** Host periodic Tx FIFO start address
This field configures the memory start address for periodic transmit FIFO RAM.

73.14.22 OTG device IN endpoint transmit FIFO x size register (OTG_DIEPTXFx)

Address offset: 0x104 + 0x04 * (x - 1), (x = 1 to 8)

Reset value: 0x0200 0400, 0x0200 0600, 0x0200 0800, 0x0200 0A00, 0x0200 0C00, 0x0200 0E00, 0x0200 1000, 0x0200 1200

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
INEPTXFD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INEPTXSA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **INEPTXFD[15:0]:** IN endpoint Tx FIFO depth
This value is in terms of 32-bit words.
Minimum value is 16

Bits 15:0 **INEPTXSA[15:0]:** IN endpoint FIFOx transmit RAM start address
This field contains the memory start address for IN endpoint transmit FIFOx. The address must be aligned with a 32-bit memory location.

73.14.23 Host-mode registers

Bit values in the register descriptions are expressed in binary unless otherwise specified.

Host-mode registers affect the operation of the core in the host mode. Host mode registers must not be accessed in device mode, as the results are undefined. Host mode registers can be categorized as follows:

73.14.24 OTG host configuration register (OTG_HCFG)

This register configures the core after power-on. Do not make changes to this register after initializing the host.

Address offset: 0x400

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSLSS	FSLSPCS[1:0]	
													r	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **FSLSS:** FS- and LS-only support

The application uses this bit to control the core enumeration speed. Using this bit, the application can make the core enumerate as an FS host, even if the connected device supports HS traffic. Do not make changes to this field after initial programming.

Bits 1:0 **FSLSPCS[1:0]:** FS/LS PHY clock select

Condition: FS host mode
01: PHY clock is running at 48 MHz
Others: Reserved
Condition: LS host mode
00: Reserved
01: Select 48 MHz PHY clock frequency
10: Select 6 MHz PHY clock frequency
11: Reserved

Note: The FSLSPCS must be set on a connection event according to the speed of the connected device (after changing this bit, a software reset must be performed).

73.14.25 OTG host frame interval register (OTG_HFIR)

This register stores the frame interval information for the current speed to which the OTG controller has enumerated.

Address offset: 0x404

Reset value: 0x0000 EA60

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RLD CTRL
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FRIVL[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **RLDCTRL:** Reload control

This bit allows dynamic reloading of the HFIR register during run time.

0: The HFIR cannot be reloaded dynamically

1: The HFIR can be dynamically reloaded during run time.

This bit needs to be programmed during initial configuration and its value must not be changed during run time.

Caution: RLDCTRL = 0 is not recommended.

Bits 15:0 **FRIVL[15:0]:** Frame interval

The value that the application programs to this field, specifies the interval between two consecutive micro-SOFs (HS) or Keep-Alive tokens (LS). This field contains the number of PHY clocks that constitute the required frame interval. The application can write a value to this register only after the port enable bit of the host port control and status register (PENA bit in OTG_HPRT) has been set. If no value is programmed, the core calculates the value based on the PHY clock specified in the FS/LS PHY clock select field of the host configuration register (FSLSPCS in OTG_HCFG). Do not change the value of this field after the initial configuration, unless the RLDCTRL bit is set. In such case, the FRIVL is reloaded with each SOF event.

– Frame interval = 125 µs × (FRIVL - 1) in high speed operation

– Frame interval = 1 ms × (FRIVL - 1) in low/full speed operation

73.14.26 OTG host frame number/frame time remaining register (OTG_HFNUM)

This register indicates the current frame number. It also indicates the time remaining (in terms of the number of PHY clocks) in the current frame.

Address offset: 0x408

Reset value: 0x0000 3FFF

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FTREM[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FRNUM[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **FTREM[15:0]:** Frame time remaining

Indicates the amount of time remaining in the current frame, in terms of PHY clocks. This field decrements on each PHY clock. When it reaches zero, this field is reloaded with the value in the Frame interval register and a new SOF is transmitted on the USB.

Bits 15:0 **FRNUM[15:0]:** Frame number

This field increments when a new SOF is transmitted on the USB, and is cleared to 0 when it reaches 0x3FFF.

73.14.27 OTG_Host periodic transmit FIFO/queue status register (OTG_HPTXSTS)

This read-only register contains the free space information for the periodic Tx FIFO and the periodic transmit request queue.

Address offset: 0x410

Reset value: 0x0008 0100

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PTXQTOP[7:0]								PTXQSAV[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PTXFSAVL[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **PTXQTOP[7:0]:** Top of the periodic transmit request queue

This indicates the entry in the periodic Tx request queue that is currently being processed by the MAC.

This register is used for debugging.

Bit 31: Odd/Even frame

0XXXXXXX: send in even frame

1XXXXXXX: send in odd frame

Bits 30:27: Channel/endpoint number

Bits 26:25: Type

XXXXX00X: IN/OUT

XXXXX01X: Zero-length packet

XXXXX11X: Disable channel command

Bit 24: Terminate (last entry for the selected channel/endpoint)

Bits 23:16 **PTXQSAV[7:0]:** Periodic transmit request queue space available

Indicates the number of free locations available to be written in the periodic transmit request queue. This queue holds both IN and OUT requests.

0: Periodic transmit request queue is full

1: 1 location available

2: 2 locations available

n: n locations available (0 ≤ n ≤ 8)

Others: Reserved

Bits 15:0 **PTXFSAVL[15:0]:** Periodic transmit data FIFO space available

Indicates the number of free locations available to be written to in the periodic Tx FIFO.

Values are in terms of 32-bit words

0: Periodic Tx FIFO is full

1: 1 word available

2: 2 words available

n: n words available (where 0 ≤ n ≤ PTXFD)

Others: Reserved

73.14.28 OTG host all channels interrupt register (OTG_HAINT)

When a significant event occurs on a channel, the host all channels interrupt register interrupts the application using the host channels interrupt bit of the core interrupt register (HCINT bit in OTG_GINTSTS). This is shown in *Figure 989*. There is one interrupt bit per channel, up to a maximum of 16 bits. Bits in this register are set and cleared when the application sets and clears bits in the corresponding host channel-x interrupt register.

Address offset: 0x414

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HAINT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **HAINT[15:0]:** Channel interrupts
One bit per channel: Bit 0 for Channel 0, bit 15 for Channel 15

73.14.29 OTG host all channels interrupt mask register (OTG_HAINTMSK)

The host all channel interrupt mask register works with the host all channel interrupt register to interrupt the application when an event occurs on a channel. There is one interrupt mask bit per channel, up to a maximum of 16 bits.

Address offset: 0x418

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HAINTM[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **HAINTM[15:0]:** Channel interrupt mask
0: Masked interrupt
1: Unmasked interrupt
One bit per channel: Bit 0 for channel 0, bit 15 for channel 15

73.14.30 OTG host port control and status register (OTG_HPRT)

This register is available only in host mode. Currently, the OTG host supports only one port.

A single register holds USB port-related information such as USB reset, enable, suspend, resume, connect status, and test mode for each port. It is shown in *Figure 989*. The rc_w1 bits in this register can trigger an interrupt to the application through the host port interrupt bit of the core interrupt register (HPRTINT bit in OTG_GINTSTS). On a port interrupt, the application must read this register and clear the bit that caused the interrupt. For the rc_w1 bits, the application must write a 1 to the bit to clear the interrupt.

Address offset: 0x440

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSPD[1:0]		PTCTL[3]
													r	r	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PTCTL[2:0]			PPWR	PLSTS[1:0]		Res.	PRST	PSUSP	PRES	POC CHNG	POCA	PEN CHNG	PENA	PCDET	PCSTS
rw	rw	rw	rw	r	r		rw	rs	rw	rc_w1	r	rc_w1	rc_w1	rc_w1	r

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:17 **PSPD[1:0]:** Port speed

Indicates the speed of the device attached to this port.
01: Full speed
10: Low speed
11: Reserved
00: High speed

Bits 16:13 **PTCTL[3:0]:** Port test control

The application writes a nonzero value to this field to put the port into a Test mode, and the corresponding pattern is signaled on the port.
0000: Test mode disabled
0001: Test_J mode
0010: Test_K mode
0011: Test_SE0_NAK mode
0100: Test_Packet mode
0101: Test_Force_Enable
Others: Reserved

Bit 12 **PPWR:** Port power

The application uses this field to control power to this port, and the core clears this bit on an overcurrent condition. Note that this bit does not directly activate the voltage on the connector VBUS.
0: Power off
1: Power on

Bits 11:10 **PLSTS[1:0]:** Port line status

Indicates the current logic level USB data lines
Bit 10: Logic level of OTGx_HSDP
Bit 11: Logic level of OTGx_HSDM

Bit 9 Reserved, must be kept at reset value.

Bit 8 **PRST:** Port reset

When the application sets this bit, a reset sequence is started on this port. The application must time the reset period and clear this bit after the reset sequence is complete.

0: Port not in reset

1: Port in reset

The application must leave this bit set for a minimum duration of at least 10 ms to start a reset on the port. The application can leave it set for another 10 ms in addition to the required minimum duration, before clearing the bit, even though there is no maximum limit set by the USB standard.

High speed: 50 ms

Full speed/Low speed: 10 ms

Bit 7 **PSUSP:** Port suspend

The application sets this bit to put this port in suspend mode. The core only stops sending SOFs when this is set. To stop the PHY clock, the application must set the port clock stop bit, which asserts the suspend input pin of the PHY.

The read value of this bit reflects the current suspend status of the port. This bit is cleared by the core after a remote wake-up signal is detected or the application sets the port reset bit or port resume bit in this register or the resume/remote wake-up detected interrupt bit or disconnect detected interrupt bit in the core interrupt register (WKUPINT or DISCINT in OTG_GINTSTS, respectively).

0: Port not in suspend mode

1: Port in suspend mode

Bit 6 **PRES:** Port resume

The application sets this bit to drive resume signaling on the port. The core continues to drive the resume signal until the application clears this bit.

If the core detects a USB remote wake-up sequence, as indicated by the port resume/remote wake-up detected interrupt bit of the core interrupt register (WKUPINT bit in OTG_GINTSTS), the core starts driving resume signaling without application intervention and clears this bit when it detects a disconnect condition. The read value of this bit indicates whether the core is currently driving resume signaling.

0: No resume driven

1: Resume driven

When LPM is enabled and the core is in L1 state, the behavior of this bit is as follow:

1. The application sets this bit to drive resume signaling on the port.

2. The core continues to drive the resume signal until a predetermined time specified in BESLTHRS[3:0] field of OTG_GLPMCFG register.

3. If the core detects a USB remote wake-up sequence, as indicated by the port L1Resume/Remote L1wake-up detected interrupt bit of the core interrupt register (WKUPINT in OTG_GINTSTS), the core starts driving resume signaling without application intervention and clears this bit at the end of resume.This bit can be set or cleared by both the core and the application. This bit is cleared by the core even if there is no device connected to the host.

Bit 5 **POCCHNG:** Port overcurrent change

The core sets this bit when the status of the port overcurrent active bit (bit 4) in this register changes.

Bit 4 **POCA:** Port overcurrent active

Indicates the overcurrent condition of the port.

0: No overcurrent condition

1: Overcurrent condition

Bit 3 **PENCHNG:** Port enable/disable change

The core sets this bit when the status of the port enable bit 2 in this register changes.

Bit 2 **PENA:** Port enable

A port is enabled only by the core after a reset sequence, and is disabled by an overcurrent condition, a disconnect condition, or by the application clearing this bit. The application cannot set this bit by a register write. It can only clear it to disable the port. This bit does not trigger any interrupt to the application.

0: Port disabled

1: Port enabled

Bit 1 **PCDET:** Port connect detected

The core sets this bit when a device connection is detected to trigger an interrupt to the application using the host port interrupt bit in the core interrupt register (HPRTINT bit in OTG_GINTSTS). The application must write a 1 to this bit to clear the interrupt.

Bit 0 **PCSTS:** Port connect status

0: No device is attached to the port

1: A device is attached to the port

73.14.31 OTG host channel x characteristics register (OTG_HCCHARx)

Address offset: 0x500 + 0x20 * x, (x = 0 to 15)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CHENA	CHDIS	ODD FRM	DAD[6:0]							MCNT[1:0]		EPTYP[1:0]		LSDEV	Res.
rs	rs	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EPDIR	EPNUM[3:0]				MPSIZ[10:0]										
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **CHENA:** Channel enable

This field is set by the application and cleared by the OTG host.

0: Channel disabled

1: Channel enabled

Bit 30 **CHDIS:** Channel disable

The application sets this bit to stop transmitting/receiving data on a channel, even before the transfer for that channel is complete. The application must wait for the Channel disabled interrupt before treating the channel as disabled.

Bit 29 **ODDFRM:** Odd frame

This field is set (reset) by the application to indicate that the OTG host must perform a transfer in an odd frame. This field is applicable for only periodic (isochronous and interrupt) transactions.

0: Even frame

1: Odd frame

Bits 28:22 **DAD[6:0]:** Device address

This field selects the specific device serving as the data source or sink.

Bits 21:20 **MCNT[1:0]:** Multicount

This field indicates to the host the number of transactions that must be executed per frame for this periodic endpoint. For non-periodic transfers, this field is not used

00: Reserved. This field yields undefined results

01: 1 transaction

10: 2 transactions per frame to be issued for this endpoint

11: 3 transactions per frame to be issued for this endpoint

Note: This field must be set to at least 01.

Bits 19:18 **EPTYP[1:0]:** Endpoint type

Indicates the transfer type selected.

00: Control

01: Isochronous

10: Bulk

11: Interrupt

Bit 17 **LSDEV:** Low-speed device

This field is set by the application to indicate that this channel is communicating to a low-speed device.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **EPDIR:** Endpoint direction

Indicates whether the transaction is IN or OUT.

0: OUT

1: IN

Bits 14:11 **EPNUM[3:0]:** Endpoint number

Indicates the endpoint number on the device serving as the data source or sink.

Bits 10:0 **MPSIZ[10:0]:** Maximum packet size

Indicates the maximum packet size of the associated endpoint.

73.14.32 OTG host channel x split control register (OTG_HCSPLTx)

Address offset: 0x504 + 0x20 * x, (x = 0 to 15)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SPLIT EN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMP LSPLT
rw															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XACTPOS[1:0]		HUBADDR[6:0]							PRTADDR[6:0]						
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **SPLITEN:** Split enable

The application sets this bit to indicate that this channel is enabled to perform split transactions.

Bits 30:17 Reserved, must be kept at reset value.

Bit 16 **COMPLSPLT:** Do complete split

The application sets this bit to request the OTG host to perform a complete split transaction.

Bits 15:14 **XACTPOS[1:0]:** Transaction position

This field is used to determine whether to send all, first, middle, or last payloads with each OUT transaction.

11: All. This is the entire data payload of this transaction (which is less than or equal to 188 bytes)

10: Begin. This is the first data payload of this transaction (which is larger than 188 bytes)

00: Mid. This is the middle payload of this transaction (which is larger than 188 bytes)

01: End. This is the last payload of this transaction (which is larger than 188 bytes)

Bits 13:7 **HUBADDR[6:0]:** Hub address

This field holds the device address of the transaction translator hub.

Bits 6:0 **PRTADDR[6:0]:** Port address

This field is the port number of the recipient transaction translator.

73.14.33 OTG host channel x interrupt register (OTG_HCINTx)

This register indicates the status of a channel with respect to USB- and AHB-related events. It is shown in *Figure 989*. The application must read this register when the host channels interrupt bit in the core interrupt register (HCINT bit in OTG_GINTSTS) is set. Before the application can read this register, it must first read the host all channels interrupt (OTG_HAINT) register to get the exact channel number for the host channel-x interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_HAINT and OTG_GINTSTS registers.

Address offset: 0x508 + 0x20 * x, (x = 0 to 15)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	DTERR	FRMOR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHBERR	CHH	XFRC
					rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **DTERR:** Data toggle error.

Bit 9 **FRMOR:** Frame overrun.

Bit 8 **BBERR:** Babble error.

Bit 7 **TXERR:** Transaction error.

Indicates one of the following errors occurred on the USB.

CRC check failure

Timeout

Bit stuff error

False EOP

Bit 6 **NYET:** Not yet ready response received interrupt.

Bit 5 **ACK:** ACK response received/transmitted interrupt.

Bit 4 **NAK:** NAK response received interrupt.

Bit 3 **STALL:** STALL response received interrupt.

Bit 2 **AHBERR:** AHB error

This error is generated only in Internal DMA mode when an AHB error occurs during an AHB read/write operation. The application can read the corresponding DMA channel address register to get the error address.

Bit 1 **CHH:** Channel halted

This indicates that the transfer completed:

– because of any USB transaction error

– in response to disable request by the application

– normally

Bit 0 **XFRC:** Transfer completed.

Transfer completed normally without any errors.

73.14.34 OTG host channel x interrupt mask register (OTG_HCINTMSKx)

This register reflects the mask for each channel status described in the previous section.

Address offset: 0x50C + 0x20 * x, (x = 0 to 15)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	AHBERRM	CHHM	XFRCM
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bit 10 **DTERRM:** Data toggle error mask.

0: Masked interrupt

1: Unmasked interrupt

Bit 9 **FRMORM:** Frame overrun mask.

0: Masked interrupt

1: Unmasked interrupt

Bit 8 **BBERRM:** Babble error mask.

0: Masked interrupt

1: Unmasked interrupt

Bit 7 **TXERRM:** Transaction error mask.

0: Masked interrupt

1: Unmasked interrupt

Bit 6 **NYET:** response received interrupt mask.

0: Masked interrupt

1: Unmasked interrupt

Bit 5 **ACKM:** ACK response received/transmitted interrupt mask.
0: Masked interrupt
1: Unmasked interrupt

Bit 4 **NAKM:** NAK response received interrupt mask.
0: Masked interrupt
1: Unmasked interrupt

Bit 3 **STALLM:** STALL response received interrupt mask.
0: Masked interrupt
1: Unmasked interrupt

Bit 2 **AHBERRM:** AHB error.
0: Masked interrupt
1: Unmasked interrupt

Bit 1 **CHHM:** Channel halted mask
0: Masked interrupt
1: Unmasked interrupt

Bit 0 **XFRCM:** Transfer completed mask
0: Masked interrupt
1: Unmasked interrupt

73.14.35 OTG host channel x transfer size register (OTG_HCTSIZx)

Address offset: 0x510 + 0x20 * x, (x = 0 to 15)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DO PNG	DPID[1:0]		PKTCNT[9:0]										XFRSIZ[18:16]		
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XFRSIZ[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **DOPNG:** Do Ping
This bit is used only for OUT transfers. Setting this field to 1 directs the host to do PING protocol.
Note: Do not set this bit for IN transfers. If this bit is set for IN transfers, it disables the channel.

Bits 30:29 **DPID[1:0]:** Data PID
The application programs this field with the type of PID to use for the initial transaction. The host maintains this field for the rest of the transfer.
00: DATA0
01: DATA2
10: DATA1
11: SETUP (control) / MDATA (non-control)

Bits 28:19 **PKTCNT[9:0]:** Packet count

This field is programmed by the application with the expected number of packets to be transmitted (OUT) or received (IN).

The host decrements this count on every successful transmission or reception of an OUT/IN packet. Once this count reaches zero, the application is interrupted to indicate normal completion.

Bits 18:0 **XFRSIZ[18:0]:** Transfer size

For an OUT, this field is the number of data bytes the host sends during the transfer.

For an IN, this field is the buffer size that the application has reserved for the transfer. The application is expected to program this field as an integer multiple of the maximum packet size for IN transactions (periodic and non-periodic).

73.14.36 OTG host channel x DMA address register(OTG_HCDMAx)

Address offset: 0x514 + 0x20 * x, (x = 0 to 15)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMAADDR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAADDR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DMAADDR[31:0]:** DMA address

This field holds the start address in the external memory from which the data for the endpoint must be fetched or to which it must be stored. This register is incremented on every AHB transaction.

73.14.37 Device-mode registers

These registers must be programmed every time the core changes to device mode

73.14.38 OTG device configuration register (OTG_DCFG)

This register configures the core in device mode after power-on or after certain control commands or enumeration. Do not make changes to this register after initial programming.

Address offset: 0x800

Reset value: 0x0220 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	PERSCHIVL[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
						rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ERRATIM	Res.	Res.	PFIVL[1:0]		DAD[6:0]							Res.	NZLSOHSK	DSPD[1:0]	
rw			rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:24 **PERSCHIVL[1:0]:** Periodic schedule interval

This field specifies the amount of time the Internal DMA engine must allocate for fetching periodic IN endpoint data. Based on the number of periodic endpoints, this value must be specified as 25, 50 or 75% of the (micro) frame.

- When any periodic endpoints are active, the internal DMA engine allocates the specified amount of time in fetching periodic IN endpoint data
- When no periodic endpoint is active, then the internal DMA engine services nonperiodic endpoints, ignoring this field
- After the specified time within a (micro) frame, the DMA switches to fetching nonperiodic endpoints

00: 25% of (micro)frame
01: 50% of (micro)frame
10: 75% of (micro)frame
11: Reserved

Bits 23:16 Reserved, must be kept at reset value.

Bit 15 **ERRATIM:** Erratic error interrupt mask

1: Mask early suspend interrupt on erratic error
0: Early suspend interrupt is generated on erratic error

Bit 14 Reserved, must be kept at reset value.

Bit 13 Reserved, must be kept at reset value.

Bits 12:11 **PFIVL[1:0]:** Periodic frame interval

Indicates the time within a frame at which the application must be notified using the end of periodic frame interrupt. This can be used to determine if all the isochronous traffic for that frame is complete.

00: 80% of the frame interval
01: 85% of the frame interval
10: 90% of the frame interval
11: 95% of the frame interval

Bits 10:4 **DAD[6:0]:** Device address

The application must program this field after every SetAddress control command.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **NZLSOHSK:** Non-zero-length status OUT handshake

The application can use this field to select the handshake the core sends on receiving a nonzero-length data packet during the OUT transaction of a control transfer's status stage.

1: Send a STALL handshake on a nonzero-length status OUT transaction and do not send the received OUT packet to the application.
0: Send the received OUT packet to the application (zero-length or nonzero-length) and send a handshake based on the NAK and STALL bits for the endpoint in the device endpoint control register.

Bits 1:0 **DSPD[1:0]:** Device speed

Indicates the speed at which the application requires the core to enumerate, or the maximum speed the application can support. However, the actual bus speed is determined only after the chirp sequence is completed, and is based on the speed of the USB host to which the core is connected.

00: High speed

01: Full speed

10: Reserved

11: Reserved

73.14.39 OTG device control register (OTG_DCTL)

Address offset: 0x804

Reset value: 0x0000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DSBESLRJCT	Res.	Res.
													rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	POPRGDNE	CGONAK	SGONAK	CGINAK	SGINAK	TCTL[2:0]			GONSTS	GINSTS	SDIS	RWUSIG
				rw	w	w	w	w	rw	rw	rw	r	r	rw	rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **DSBESLRJCT:** Deep sleep BESL reject

Core rejects LPM request with BESL value greater than BESL threshold programmed. NYET response is sent for LPM tokens with BESL value greater than BESL threshold. By default, the deep sleep BESL reject feature is disabled.

Bits 17:12 Reserved, must be kept at reset value.

Bit 11 **POPRGDNE:** Power-on programming done

The application uses this bit to indicate that register programming is completed after a wake-up from power down mode.

Bit 10 **CGONAK:** Clear global OUT NAK

Writing 1 to this field clears the Global OUT NAK.

Bit 9 **SGONAK:** Set global OUT NAK

Writing 1 to this field sets the Global OUT NAK.
The application uses this bit to send a NAK handshake on all OUT endpoints.
The application must set the this bit only after making sure that the Global OUT NAK effective bit in the core interrupt register (GONAKEFF bit in OTG_GINTSTS) is cleared.

Bit 8 **CGINAK:** Clear global IN NAK

Writing 1 to this field clears the Global IN NAK.

Bit 7 **SGINAK:** Set global IN NAK

Writing 1 to this field sets the Global non-periodic IN NAK.The application uses this bit to send a NAK handshake on all non-periodic IN endpoints.
The application must set this bit only after making sure that the Global IN NAK effective bit in the core interrupt register (GINAKEFF bit in OTG_GINTSTS) is cleared.

Bits 6:4 **TCTL[2:0]:** Test control
000: Test mode disabled
001: Test_J mode
010: Test_K mode
011: Test_SE0_NAK mode
100: Test_Packet mode
101: Test_Force_Enable
Others: Reserved

Bit 3 **GONSTS:** Global OUT NAK status
0: A handshake is sent based on the FIFO status and the NAK and STALL bit settings.
1: No data is written to the Rx FIFO, irrespective of space availability. Sends a NAK handshake on all packets, except on SETUP transactions. All isochronous OUT packets are dropped.

Bit 2 **GINSTS:** Global IN NAK status
0: A handshake is sent out based on the data availability in the transmit FIFO.
1: A NAK handshake is sent out on all non-periodic IN endpoints, irrespective of the data availability in the transmit FIFO.

Bit 1 **SDIS:** Soft disconnect
The application uses this bit to signal the USB OTG core to perform a soft disconnect. As long as this bit is set, the host does not see that the device is connected, and the device does not receive signals on the USB. The core stays in the disconnected state until the application clears this bit.
0: Normal operation. When this bit is cleared after a soft disconnect, the core generates a device connect event to the USB host. When the device is reconnected, the USB host restarts device enumeration.
1: The core generates a device disconnect event to the USB host.

Bit 0 **RWUSIG:** Remote wake-up signaling
When the application sets this bit, the core initiates remote signaling to wake up the USB host. The application must set this bit to instruct the core to exit the suspend state. As specified in the USB 2.0 specification, the application must clear this bit 1 ms to 15 ms after setting it.
If LPM is enabled and the core is in the L1 (sleep) state, when the application sets this bit, the core initiates L1 remote signaling to wake up the USB host. The application must set this bit to instruct the core to exit the sleep state. As specified in the LPM specification, the hardware automatically clears this bit 50 µs ($T_{L1DevDrvResume}$) after being set by the application. The application must not set this bit when bRemoteWake from the previous LPM transaction is zero (refer to REMWAKE bit in GLPMCFG register).

Table 746 contains the minimum duration (according to device state) for which the Soft disconnect (SDIS) bit must be set for the USB host to detect a device disconnect. To accommodate clock jitter, it is recommended that the application add some extra delay to the specified minimum duration.

Table 746. Minimum duration for soft disconnect

Operating speed	Device state	Minimum duration
Full speed	Suspended	1 ms + 2.5 µs
Full speed	Idle	2.5 µs
Full speed	Not Idle or suspended (Performing transactions)	2.5 µs
High speed	Not Idle or suspended (Performing transactions)	125 µs

73.14.40 OTG device status register (OTG_DSTS)

This register indicates the status of the core with respect to USB-related events. It must be read on interrupts from the device all interrupts (OTG_DAINT) register.

Address offset: 0x808

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEVLNSTS[1:0]		FNSOF[13:8]					
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FNSOF[7:0]								Res.	Res.	Res.	Res.	EERR	ENUMSPD[1:0]		SUSP STS
r	r	r	r	r	r	r	r					r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:22 **DEVLNSTS[1:0]:** Device line status

Indicates the current logic level USB data lines.
Bit [23]: Logic level of D+
Bit [22]: Logic level of D-

Bits 21:8 **FNSOF[13:0]:** Frame number of the received SOF

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **EERR:** Erratic error

The core sets this bit to report any erratic errors.
Due to erratic errors, the OTG controller goes into suspended state and an interrupt is generated to the application with Early suspend bit of the OTG_GINTSTS register (ESUSP bit in OTG_GINTSTS). If the early suspend is asserted due to an erratic error, the application can only perform a soft disconnect recover.

Bits 2:1 **ENUMSPD[1:0]:** Enumerated speed

Indicates the speed at which the OTG controller has come up after speed detection through a chirp sequence.
00: High Speed
01: Full Speed
11: Reserved
Others: reserved

Bit 0 **SUSPSTS:** Suspend status

In device mode, this bit is set as long as a suspend condition is detected on the USB. The core enters the suspended state when there is no activity on the USB data lines for a period of 3 ms. The core comes out of the suspend:

– When there is an activity on the USB data lines
– When the application writes to the remote wake-up signaling bit in the OTG_DCTL register (RWUSIG bit in OTG_DCTL).

73.14.41 OTG device IN endpoint common interrupt mask register (OTG_DIEPMSK)

This register works with each of the OTG_DIEPINTx registers for all endpoints to generate an interrupt per IN endpoint. The IN endpoint interrupt for a specific status in the OTG_DIEPINTx register can be masked by writing to the corresponding bit in this register. Status bits are masked by default.

Address offset: 0x810

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	NAKM	Res.	Res.	Res.	Res.	TXFURM	Res.	INEPNEM	INEPNMM	ITTXFEMSK	TOM	AHBERRM	EPDM	XFRCM
		rw					rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **NAKM:** NAK interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bits 12:10 Reserved, must be kept at reset value.

Bit 9 Reserved, must be kept at reset value.

Bit 8 **TXFURM:** FIFO underrun mask
0: Masked interrupt
1: Unmasked interrupt

Bit 7 Reserved, must be kept at reset value.

Bit 6 **INEPNEM:** IN endpoint NAK effective mask
0: Masked interrupt
1: Unmasked interrupt

Bit 5 **INEPNMM:** IN token received with EP mismatch mask
0: Masked interrupt
1: Unmasked interrupt

Bit 4 **ITTXFEMSK:** IN token received when Tx FIFO empty mask
0: Masked interrupt
1: Unmasked interrupt

Bit 3 **TOM:** Timeout condition mask (Non-isochronous endpoints)
0: Masked interrupt
1: Unmasked interrupt

Bit 2 **AHBERRM:** AHB error mask
0: Masked interrupt
1: Unmasked interrupt

Bit 1 **EPDM:** Endpoint disabled interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 0 **XFRCM:** Transfer completed interrupt mask
0: Masked interrupt
1: Unmasked interrupt

73.14.42 OTG device OUT endpoint common interrupt mask register (OTG_DOEPMSK)

This register works with each of the OTG_DOEPINTx registers for all endpoints to generate an interrupt per OUT endpoint. The OUT endpoint interrupt for a specific status in the OTG_DOEPINTx register can be masked by writing into the corresponding bit in this register. Status bits are masked by default.

Address offset: 0x814

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	NYET MSK	NAK MSK	BERR M	Res.	Res.	Res.	OUT PKT ERRM	Res.	B2B STUPM	STS PHSR XM	OTEPD M	STUPM	AHB ERRM	EPDM	XFRC M
	rw	rw	rw				rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **NYETMSK:** NYET interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 13 **NAKMSK:** NAK interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 12 **BERRM**: Babble error interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 Reserved, must be kept at reset value.

Bit 8 **OUTPKTERRM:** Out packet error mask
0: Masked interrupt
1: Unmasked interrupt

Bit 7 Reserved, must be kept at reset value.

Bit 6 **B2BSTUPM:** Back-to-back SETUP packets received mask
Applies to control OUT endpoints only.
0: Masked interrupt
1: Unmasked interrupt

Bit 5 **STSPHSRXM**: Status phase received for control write mask
0: Masked interrupt
1: Unmasked interrupt

Bit 4 **OTEPDM:** OUT token received when endpoint disabled mask. Applies to control OUT endpoints only.
0: Masked interrupt
1: Unmasked interrupt

Bit 3 **STUPM:** SETUP phase done mask. Applies to control endpoints only.
0: Masked interrupt
1: Unmasked interrupt

Bit 2 **AHBERRM:** AHB error mask
0: Masked interrupt
1: Unmasked interrupt

Bit 1 **EPDM:** Endpoint disabled interrupt mask
0: Masked interrupt
1: Unmasked interrupt

Bit 0 **XFRCM:** Transfer completed interrupt mask
0: Masked interrupt
1: Unmasked interrupt

73.14.43 OTG device all endpoints interrupt register (OTG_DAINT)

When a significant event occurs on an endpoint, a OTG_DAINT register interrupts the application using the device OUT endpoints interrupt bit or device IN endpoints interrupt bit of the OTG_GINTSTS register (OEPINT or IEPINT in OTG_GINTSTS, respectively). There is one interrupt bit per endpoint, up to a maximum of 16 bits for OUT endpoints and 16 bits for IN endpoints. For a bidirectional endpoint, the corresponding IN and OUT interrupt bits are used. Bits in this register are set and cleared when the application sets and clears bits in the corresponding device endpoint-x interrupt register (OTG_DIEPINTx/OTG_DOEPINTx).

Address offset: 0x818

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OEPINT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IEPINT[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **OEPINT[15:0]:** OUT endpoint interrupt bits
One bit per OUT endpoint:
Bit 16 for OUT endpoint 0, bit 19 for OUT endpoint 3.

Bits 15:0 **IEPINT[15:0]:** IN endpoint interrupt bits
One bit per IN endpoint:
Bit 0 for IN endpoint 0, bit 3 for endpoint 3.

73.14.44 OTG all endpoints interrupt mask register (OTG_DAINTMSK)

The OTG_DAINTMSK register works with the device endpoint interrupt register to interrupt the application when an event occurs on a device endpoint. However, the OTG_DAINT register bit corresponding to that interrupt is still set.

Address offset: 0x81C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OEPM[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IEPM[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **OEPM[15:0]:** OUT EP interrupt mask bits
One per OUT endpoint:
Bit 16 for OUT EP 0, bit 19 for OUT EP 3
0: Masked interrupt
1: Unmasked interrupt

Bits 15:0 **IEPM[15:0]:** IN EP interrupt mask bits
One bit per IN endpoint:
Bit 0 for IN EP 0, bit 3 for IN EP 3
0: Masked interrupt
1: Unmasked interrupt

73.14.45 OTG device threshold control register (OTG_DTHRCTL)

Address offset: 0x0830

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	ARPEN	Res.	RXTHRLEN[8:0]									RXTHREN
				rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	TXTHRLEN[8:0]									ISOTHREN	NONISOTHREN
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **ARPEN:** Arbiter parking enable
This bit controls internal DMA arbiter parking for IN endpoints. When thresholding is enabled and this bit is set to one, then the arbiter parks on the IN endpoint for which there is a token received on the USB. This is done to avoid getting into underrun conditions. By default parking is enabled.

Bit 26 Reserved, must be kept at reset value.

Bits 25:17 **RXTHRLEN[8:0]:** Receive threshold length
This field specifies the receive thresholding size in 32-bit words. This field also specifies the amount of data received on the USB before the core can start transmitting on the AHB. The threshold length has to be at least eight 32-bit words. The recommended value for RXTHRLEN is to be the same as the programmed AHB burst length (HBSTLEN bit in OTG_GAHBCFG).

Bit 16 **RXTHREN:** Receive threshold enable
When this bit is set, the core enables thresholding in the receive direction.

Bits 15:11 Reserved, must be kept at reset value.

Bits 10:2 **TXTHRLEN[8:0]:** Transmit threshold length
This field specifies the transmit thresholding size in 32-bit words. This field specifies the amount of data in bytes to be in the corresponding endpoint transmit FIFO, before the core can start transmitting on the USB. The threshold length has to be at least eight 32-bit words. This field controls both isochronous and nonisochronous IN endpoint thresholds. The recommended value for TXTHRLEN is to be the same as the programmed AHB burst length (HBSTLEN bit in OTG_GAHBCFG).

Bit 1 **ISOTHREN:** ISO IN endpoint threshold enable
When this bit is set, the core enables thresholding for isochronous IN endpoints.

Bit 0 **NONISOTHREN:** Nonisochronous IN endpoints threshold enable
When this bit is set, the core enables thresholding for nonisochronous IN endpoints.

73.14.46 OTG device IN endpoint FIFO empty interrupt mask register (OTG_DIEPEMPMSK)

This register is used to control the IN endpoint FIFO empty interrupt generation (TXFE_OTG_DIEPINTx).

Address offset: 0x834

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INEPTXFEM[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **INEPTXFEM[15:0]:** IN EP Tx FIFO empty interrupt mask bits
These bits act as mask bits for OTG_DIEPINTx.
TXFE interrupt one bit per IN endpoint:
Bit 0 for IN endpoint 0, bit 3 for IN endpoint 3
0: Masked interrupt
1: Unmasked interrupt

73.14.47 OTG device IN endpoint x control register [alternate] (OTG_DIEPCTLx)

Valid for INT/BULK endpoints, see next section for ISO endpoints.

The application uses this register to control the behavior of each logical endpoint other than endpoint 0.

Address offset: 0x900 + 0x20 * x, (x = 0 to 8)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EPENA	EPDIS	SD1 PID	SD0 PID	SNAK	CNAK	TXFNUM[3:0]				STALL	Res.	EPTYP[1:0]		NAK STS	DPID
rs	rs	w	w	w	w	rw	rw	rw	rw	rw		rw	rw	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBA EP	Res.	Res.	Res.	Res.	MPSIZ[10:0]										
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **EPENA:** Endpoint enable

The application sets this bit to start transmitting data on an endpoint.
The core clears this bit before setting any of the following interrupts on this endpoint:
– SETUP phase done
– Endpoint disabled
– Transfer completed

Bit 30 **EPDIS:** Endpoint disable

The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the endpoint disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the endpoint disabled interrupt. The application must set this bit only if endpoint enable is already set for this endpoint.

Bit 29 **SD1PID:** Set DATA1 PID

Writing to this field sets the endpoint data PID (DPID) field in this register to DATA1.

Bit 28 **SD0PID:** Set DATA0 PID

Applies to interrupt/bulk IN endpoints only.
Writing to this field sets the endpoint data PID (DPID) field in this register to DATA0.

Bit 27 **SNAK:** Set NAK

A write to this bit sets the NAK bit for the endpoint.
Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit for OUT endpoints on a transfer completed interrupt, or after a SETUP is received on the endpoint.

Bit 26 **CNAK:** Clear NAK

A write to this bit clears the NAK bit for the endpoint.

Bits 25:22 **TXFNUM[3:0]:** Tx FIFO number

These bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number.
This field is valid only for IN endpoints.

Bit 21 **STALL:** STALL handshake

Applies to non-control, non-isochronous IN endpoints only (access type is rw).

The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.

Bit 20 Reserved, must be kept at reset value.

Bits 19:18 **EPTYP[1:0]:** Endpoint type

This is the transfer type supported by this logical endpoint.

00: Control

01: Isochronous

10: Bulk

11: Interrupt

Bit 17 **NAKSTS:** NAK status

It indicates the following:

0: The core is transmitting non-NAK handshakes based on the FIFO status.

1: The core is transmitting NAK handshakes on this endpoint.

When either the application or the core sets this bit:

For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there are data available in the Tx FIFO.

For isochronous IN endpoints: The core sends out a zero-length data packet, even if there are data available in the Tx FIFO.

Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 16 **DPID:** Endpoint data PID

Applies to interrupt/bulk IN endpoints only.

Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The application uses the SD0PID and SD1PID register fields to program either DATA0 or DATA1 PID.

0: DATA0

1: DATA1

Bit 15 **USBAEP:** USB active endpoint

Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:0 **MPSIZ[10:0]:** Maximum packet size

The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.

73.14.48 OTG device IN endpoint x control register [alternate] (OTG_DIEPCTLx)

Valid for ISO endpoints, see previous section for INT/BULK endpoints.

The application uses this register to control the behavior of each logical endpoint other than endpoint 0.

Address offset: 0x900 + 0x20 * x, (x = 0 to 8)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EPENA	EPDIS	SODD FRM	SEVN FRM	SNAK	CNAK	TXFNUM[3:0]				STALL	Res.	EPTYP[1:0]		NAK STS	EO NUM
rs	rs	w	w	w	w	rw	rw	rw	rw	rw		rw	rw	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBA EP	Res.	Res.	Res.	Res.	MPSIZ[10:0]										
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **EPENA:** Endpoint enable

The application sets this bit to start transmitting data on an endpoint.
The core clears this bit before setting any of the following interrupts on this endpoint:
– SETUP phase done
– Endpoint disabled
– Transfer completed

Bit 30 **EPDIS:** Endpoint disable

The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the endpoint disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the endpoint disabled interrupt. The application must set this bit only if endpoint enable is already set for this endpoint.

Bit 29 **SODDFRM:** Set odd frame

Applies to isochronous IN and OUT endpoints only.
Writing to this field sets the Even/Odd frame (EONUM) field to odd frame.

Bit 28 **SEVNFRM:** Set even frame

Applies to isochronous IN endpoints only.
Writing to this field sets the Even/Odd frame (EONUM) field to even frame.

Bit 27 **SNAK:** Set NAK

A write to this bit sets the NAK bit for the endpoint.
Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit for OUT endpoints on a transfer completed interrupt, or after a SETUP is received on the endpoint.

Bit 26 **CNAK:** Clear NAK

A write to this bit clears the NAK bit for the endpoint.

Bits 25:22 **TXFNUM[3:0]:** Tx FIFO number

These bits specify the FIFO number associated with this endpoint. Each active IN endpoint must be programmed to a separate FIFO number.
This field is valid only for IN endpoints.

Bit 21 **STALL:** STALL handshake

Applies to non-control, non-isochronous IN endpoints only (access type is rw).

The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.

Bit 20 Reserved, must be kept at reset value.

Bits 19:18 **EPTYP[1:0]:** Endpoint type

This is the transfer type supported by this logical endpoint.

00: Control

01: Isochronous

10: Bulk

11: Interrupt

Bit 17 **NAKSTS:** NAK status

It indicates the following:

0: The core is transmitting non-NAK handshakes based on the FIFO status.

1: The core is transmitting NAK handshakes on this endpoint.

When either the application or the core sets this bit:

For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there are data available in the Tx FIFO.

For isochronous IN endpoints: The core sends out a zero-length data packet, even if there are data available in the Tx FIFO.

Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 16 **EONUM:** Even/odd frame

Applies to isochronous IN endpoints only.

Indicates the frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd frame number in which it intends to transmit/receive isochronous data for this endpoint using the SEVNFRM and SODDFRM fields in this register.

0: Even frame

1: Odd frame

Bit 15 **USBAEP:** USB active endpoint

Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:0 **MPSIZ[10:0]:** Maximum packet size

The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.

73.14.49 OTG device IN endpoint x interrupt register (OTG_DIEPINTx)

This register indicates the status of an endpoint with respect to USB- and AHB-related events. It is shown in *Figure 989*. The application must read this register when the IN endpoints interrupt bit of the core interrupt register (IEPINT in OTG_GINTSTS) is set. Before the application can read this register, it must first read the device all endpoints interrupt (OTG_DAINT) register to get the exact endpoint number for the device endpoint-x interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_DAINT and OTG_GINTSTS registers.

Address offset: 0x908 + 0x20 * x, (x = 0 to 8)

Reset value: 0x0000 0080

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	NAK	Res.	PKTDRPSTS	Res.	Res.	TXFIFOUDRN	TXFE	INEPNE	INEPNM	ITTXFE	TOC	AHBERR	EPDISD	XFRC
		rc_w1		rc_w1			rc_w1	r	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **NAK:** NAK input

The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to unavailability of data in the Tx FIFO.

Bit 12 Reserved, must be kept at reset value.

Bit 11 **PKTDRPSTS:** Packet dropped status

This bit indicates to the application that an ISOC OUT packet has been dropped. This bit does not have an associated mask bit and does not generate an interrupt.

Bit 10 Reserved, must be kept at reset value.

Bit 9 Reserved, must be kept at reset value.

Bit 8 **TXFIFOUDRN:** Transmit Fifo Underrun (TxfifoUndrn)

The core generates this interrupt when it detects a transmit FIFO underrun condition for this endpoint. Dependency: This interrupt is valid only when Thresholding is enabled

Bit 7 **TXFE:** Transmit FIFO empty

This interrupt is asserted when the Tx FIFO for this endpoint is either half or completely empty. The half or completely empty status is determined by the Tx FIFO Empty Level bit in the OTG_GAHBCFG register (TXFELVL bit in OTG_GAHBCFG).

Bit 6 **INEPNE:** IN endpoint NAK effective

This bit can be cleared when the application clears the IN endpoint NAK by writing to the CNAK bit in OTG_DIEPCTLx.

This interrupt indicates that the core has sampled the NAK bit set (either by the application or by the core). The interrupt indicates that the IN endpoint NAK bit set by the application has taken effect in the core.

This interrupt does not guarantee that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit.

Bit 5 **INEPNM**: IN token received with EP mismatch

Indicates that the data in the top of the non-periodic TxFIFO belongs to an endpoint other than the one for which the IN token was received. This interrupt is asserted on the endpoint for which the IN token was received.

Bit 4 **ITTXFE:** IN token received when Tx FIFO is empty

Indicates that an IN token was received when the associated Tx FIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint for which the IN token was received.

Bit 3 **TOC:** Timeout condition

Indicates that the core has detected a timeout condition on the USB for the last IN token on this endpoint.

Bit 2 **AHBERR**: AHB error

This is generated only in internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.

Bit 1 **EPDISD:** Endpoint disabled interrupt

This bit indicates that the endpoint is disabled per the application's request.

Bit 0 **XFRC:** Transfer completed interrupt

This field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint.

73.14.50 OTG device IN endpoint 0 transfer size register (OTG_DIEPTSIZ0)

The application must modify this register before enabling endpoint 0. Once endpoint 0 is enabled using the endpoint enable bit in the device control endpoint 0 control registers (EPENA in OTG_DIEPCTL0), the core modifies this register. The application can only read this register once the core has cleared the endpoint enable bit.

Nonzero endpoints use the registers for endpoints 1–3.

Address offset: 0x910

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKTCNT[1:0]		Res.	Res.	Res.
											rw	rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	XFRSIZ[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:19 **PKTCNT[1:0]:** Packet count

Indicates the total number of USB packets that constitute the transfer size amount of data for endpoint 0.

This field is decremented every time a packet (maximum size or short packet) is read from the Tx FIFO.

Bits 18:7 Reserved, must be kept at reset value.

Bits 6:0 **XFRSIZ[6:0]:** Transfer size

Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.

The core decrements this field every time a packet from the external memory is written to the Tx FIFO.

73.14.51 OTG device IN endpoint x DMA address register (OTG_DIEPDMAx)

Address offset: 0x914 + 0x20 * x, (x = 0 to 8)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMAADDR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAADDR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DMAADDR[31:0]:** DMA Address

This field holds the start address in the external memory from which the data for the endpoint must be fetched. This register is incremented on every AHB transaction.

73.14.52 OTG device IN endpoint transmit FIFO status register (OTG_DTXFSTSx)

This read-only register contains the free space information for the device IN endpoint Tx FIFO.

Address offset: 0x918 + 0x20 * x, (x = 0 to 8)

Reset value: 0x0000 0200

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
INEPTFSAV[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **INEPTFSAV[15:0]:** IN endpoint Tx FIFO space available

Indicates the amount of free space available in the endpoint Tx FIFO.

Values are in terms of 32-bit words:

0x0: Endpoint Tx FIFO is full

0x1: 1 word available

0x2: 2 words available

0xn: n words available

Others: Reserved

73.14.53 OTG device IN endpoint x transfer size register (OTG_DIEPTSIZx)

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using the endpoint enable bit in the OTG_DIEPCTLx registers (EPENA bit in OTG_DIEPCTLx), the core modifies this register. The application can only read this register once the core has cleared the endpoint enable bit.

Address offset: 0x910 + 0x20 * x, (x = 1 to 8)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	MCNT[1:0]		PKTCNT[9:0]										XFRSIZ[18:16]		
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XFRSIZ[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:29 **MCNT[1:0]:** Multi count

For periodic IN endpoints, this field indicates the number of packets that must be transmitted per frame on the USB. The core uses this field to calculate the data PID for isochronous IN endpoints.

01: 1 packet

10: 2 packets

11: 3 packets

Bits 28:19 **PKTCNT[9:0]:** Packet count

Indicates the total number of USB packets that constitute the transfer size amount of data for this endpoint.

This field is decremented every time a packet (maximum size or short packet) is read from the Tx FIFO.

Bits 18:0 **XFRSIZ[18:0]:** Transfer size

This field contains the transfer size in bytes for the current endpoint. The core only interrupts the application after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.

The core decrements this field every time a packet from the external memory is written to the Tx FIFO.

73.14.54 OTG device control OUT endpoint 0 control register (OTG_DOEPCTL0)

This section describes the OTG_DOEPCTL0 register. Nonzero control endpoints use registers for endpoints 1–8.

Address offset: 0xB00

Reset value: 0x0000 8000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EPENA	EPDIS	Res.	Res.	SNAK	CNAK	Res.	Res.	Res.	Res.	STALL	SNPM	EPTYP[1:0]		NAK STS	Res.
w	r			w	w					rs	rw	r	r	r	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBA EP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MPSIZ[1:0]	
r														r	r

Bit 31 **EPENA:** Endpoint enable

The application sets this bit to start transmitting data on endpoint 0.

The core clears this bit before setting any of the following interrupts on this endpoint:

– SETUP phase done

– Endpoint disabled

– Transfer completed

Bit 30 **EPDIS:** Endpoint disable

The application cannot disable control OUT endpoint 0.

Bits 29:28 Reserved, must be kept at reset value.

Bit 27 **SNAK:** Set NAK

A write to this bit sets the NAK bit for the endpoint.

Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit on a transfer completed interrupt, or after a SETUP is received on the endpoint.

Bit 26 **CNAK:** Clear NAK

A write to this bit clears the NAK bit for the endpoint.

Bits 25:22 Reserved, must be kept at reset value.

Bit 21 **STALL:** STALL handshake

The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 20 **SNPM:** Snoop mode

This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.

Bits 19:18 **EPTYP[1:0]:** Endpoint type

Hardcoded to 00 for control.

Bit 17 **NAKSTS:** NAK status

Indicates the following:

0: The core is transmitting non-NAK handshakes based on the FIFO status.

1: The core is transmitting NAK handshakes on this endpoint.

When either the application or the core sets this bit, the core stops receiving data, even if there is space in the Rx FIFO to accommodate the incoming packet. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 16 Reserved, must be kept at reset value.

Bit 15 **USBAEP:** USB active endpoint

This bit is always set to 1, indicating that a control endpoint 0 is always active in all configurations and interfaces.

Bits 14:2 Reserved, must be kept at reset value.

Bits 1:0 **MPSIZ[1:0]:** Maximum packet size

The maximum packet size for control OUT endpoint 0 is the same as what is programmed in control IN endpoint 0.
00: 64 bytes
01: 32 bytes
10: 16 bytes
11: 8 bytes

73.14.55 OTG device OUT endpoint x interrupt register (OTG_DOEPINTx)

This register indicates the status of an endpoint with respect to USB- and AHB-related events. It is shown in *Figure 989*. The application must read this register when the OUT endpoints interrupt bit of the OTG_GINTSTS register (OEPINT bit in OTG_GINTSTS) is set. Before the application can read this register, it must first read the OTG_DAINT register to get the exact endpoint number for the OTG_DOEPINTx register. The application must clear the appropriate bit in this register to clear the corresponding bits in the OTG_DAINT and OTG_GINTSTS registers.

Address offset: 0xB08 + 0x20 * x, (x = 0 to 8)

Reset value: 0x0000 0080

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STPK TRX	NYET	NAK	BERR	Res.	Res.	Res.	OUT PKT ERR	Res.	B2B STUP	STSPH SRX	OTEP DIS	STUP	AHB ERR	EP DISD	XFRC
rc_w1	rc_w1	rc_w1	rc_w1				rc_w1		rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **STPKTRX:** Setup packet received

Applicable for control OUT endpoints in only in the Buffer DMA Mode. Set by the OTG, this bit indicates that this buffer holds 8 bytes of setup data. There is only one setup packet per buffer. On receiving a setup packet, the OTG closes the buffer and disables the corresponding endpoint after SETUP_COMPLETE status is seen in the Rx FIFO. OTG puts a SETUP_COMPLETE status into the Rx FIFO when it sees the first IN or OUT token after the SETUP packet for that particular endpoint. The application must then re-enable the endpoint to receive any OUT data for the control transfer and reprogram the buffer start address. Because of the above behavior, OTG can receive any number of back to back setup packets and one buffer for every setup packet is used.

Bit 14 **NYET:** NYET interrupt

This interrupt is generated when a NYET response is transmitted for a non isochronous OUT endpoint.

Bit 13 **NAK:** NAK input

The core generates this interrupt when a NAK is transmitted or received by the device. In case of isochronous IN endpoints the interrupt gets generated when a zero length packet is transmitted due to unavailability of data in the Tx FIFO.

Bit 12 **BERR:** Babble error interrupt
The core generates this interrupt when babble is received for the endpoint.

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 Reserved, must be kept at reset value.

Bit 8 **OUTPKTERR**: OUT packet error
This interrupt is asserted when the core detects an overflow or a CRC error for an OUT packet. This interrupt is valid only when thresholding is enabled.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **B2BSTUP:** Back-to-back SETUP packets received
Applies to control OUT endpoint only.
This bit indicates that the core has received more than three back-to-back SETUP packets for this particular endpoint.

Bit 5 **STSPHSRX**: Status phase received for control write
This interrupt is valid only for control OUT endpoints. This interrupt is generated only after OTG has transferred all the data that the host has sent during the data phase of a control write transfer, to the system memory buffer. The interrupt indicates to the application that the host has switched from data phase to the status phase of a control write transfer. The application can use this interrupt to ACK or STALL the status phase, after it has decoded the data phase.

Bit 4 **OTEPDIS:** OUT token received when endpoint disabled
Applies only to control OUT endpoints.
Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint for which the OUT token was received.

Bit 3 **STUP:** SETUP phase done
Applies to control OUT endpoint only.Indicates that the SETUP phase for the control endpoint is complete and no more back-to-back SETUP packets were received for the current control transfer. On this interrupt, the application can decode the received SETUP data packet.

Bit 2 **AHBERR**: AHB error
This is generated only in internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.

Bit 1 **EPDISD:** Endpoint disabled interrupt
This bit indicates that the endpoint is disabled per the application’s request.

Bit 0 **XFRC:** Transfer completed interrupt
This field indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint.

73.14.56 OTG device OUT endpoint 0 transfer size register (OTG_DOEPTSIZ0)

The application must modify this register before enabling endpoint 0. Once endpoint 0 is enabled using the endpoint enable bit in the OTG_DOEPCTL0 registers (EPENA bit in OTG_DOEPCTL0), the core modifies this register. The application can only read this register once the core has cleared the endpoint enable bit.

Nonzero endpoints use the registers for endpoints 1–8.

Address offset: 0xB10

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	STUPCNT[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKTCNT	Res.	Res.	Res.
	rw	rw										rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	XFRSIZ[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:29 **STUPCNT[1:0]:** SETUP packet count

This field specifies the number of back-to-back SETUP data packets the endpoint can receive.
01: 1 packet
10: 2 packets
11: 3 packets

Bits 28:20 Reserved, must be kept at reset value.

Bit 19 **PKTCNT:** Packet count

This field is decremented to zero after a packet is written into the Rx FIFO.

Bits 18:7 Reserved, must be kept at reset value.

Bits 6:0 **XFRSIZ[6:0]:** Transfer size

Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.
The core decrements this field every time a packet is read from the Rx FIFO and written to the external memory.

73.14.57 OTG device OUT endpoint x DMA address register (OTG_DOEPDMAx)

Address offset: 0xB14 + 0x20 * x, (x = 0 to 8)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DMAADDR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DMAADDR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DMAADDR[31:0]:** DMA Address

This field holds the start address in the external memory from which the data for the endpoint must be fetched. This register is incremented on every AHB transaction.

73.14.58 OTG device OUT endpoint x control register [alternate] (OTG_DOEPCTLx)

Valid for INT/BULK endpoints, see next section for ISO endpoints.

The application uses this register to control the behavior of each logical endpoint other than endpoint 0.

Address offset: 0xB00 + 0x20 * x, (x = 1 to 8)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EPENA	EPDIS	SD1 PID	SD0 PID	SNAK	CNAK	Res.	Res.	Res.	Res.	STALL	SNPM	EPTYP[1:0]		NAK STS	DPID
rs	rs	w	w	w	w					rw	rw	rw	rw	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBA EP	Res.	Res.	Res.	Res.	MPSIZ[10:0]										
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **EPENA:** Endpoint enable

Applies to IN and OUT endpoints.

The application sets this bit to start transmitting data on an endpoint.

The core clears this bit before setting any of the following interrupts on this endpoint:

– SETUP phase done

– Endpoint disabled

– Transfer completed

Bit 30 **EPDIS:** Endpoint disable

The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the endpoint disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the endpoint disabled interrupt. The application must set this bit only if endpoint enable is already set for this endpoint.

Bit 29 **SD1PID:** Set DATA1 PID
Writing to this field sets the endpoint data PID (DPID) field in this register to DATA1.

Bit 28 **SD0PID:** Set DATA0 PID
Applies to interrupt/bulk OUT endpoints only.
Writing to this field sets the endpoint data PID (DPID) field in this register to DATA0.

Bit 27 **SNAK:** Set NAK
A write to this bit sets the NAK bit for the endpoint.
Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit for OUT endpoints on a transfer completed interrupt, or after a SETUP is received on the endpoint.

Bit 26 **CNAK:** Clear NAK
A write to this bit clears the NAK bit for the endpoint.

Bits 25:22 Reserved, must be kept at reset value.

Bit 21 **STALL:** STALL handshake
Applies to non-control, non-isochronous OUT endpoints only (access type is rw).
The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.
Applies to control endpoints only (access type is rs).
The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 20 **SNPM:** Snoop mode
This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.

Bits 19:18 **EPTYP[1:0]:** Endpoint type
This is the transfer type supported by this logical endpoint.
00: Control
01: Isochronous
10: Bulk
11: Interrupt

Bit 17 **NAKSTS:** NAK status
Indicates the following:
0: The core is transmitting non-NAK handshakes based on the FIFO status.
1: The core is transmitting NAK handshakes on this endpoint.
When either the application or the core sets this bit:
The core stops receiving any data on an OUT endpoint, even if there is space in the Rx FIFO to accommodate the incoming packet.
Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 16 **DPID:** Endpoint data PID

Applies to interrupt/bulk OUT endpoints only.

Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. The application uses the SD0PID and SD1PID register fields to program either DATA0 or DATA1 PID.

0: DATA0

1: DATA1

Bit 15 **USBAEP:** USB active endpoint

Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:0 **MPSIZ[10:0]:** Maximum packet size

The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.

73.14.59 OTG device OUT endpoint x control register [alternate] (OTG_DOEPCTLx)

Valid for ISO endpoints, see previous section for INT/BULK endpoints.

The application uses this register to control the behavior of each logical endpoint other than endpoint 0.

Address offset: 0xB00 + 0x20 * x, (x = 1 to 8)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EPENA	EPDIS	SODD FRM	SEVN FRM	SNAK	CNAK	Res.	Res.	Res.	Res.	STALL	SNPM	EPTYP[1:0]		NAK STS	EO NUM
rs	rs	w	w	w	w					rw	rw	rw	rw	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USBA EP	Res.	Res.	Res.	Res.	MPSIZ[10:0]										
rw					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **EPENA:** Endpoint enable

Applies to IN and OUT endpoints.

The application sets this bit to start transmitting data on an endpoint.

The core clears this bit before setting any of the following interrupts on this endpoint:

– SETUP phase done

– Endpoint disabled

– Transfer completed

Bit 30 **EPDIS:** Endpoint disable

The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the endpoint disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the endpoint disabled interrupt. The application must set this bit only if endpoint enable is already set for this endpoint.

Bit 29 **SODDFRM:** Set odd frame

Applies to isochronous IN and OUT endpoints only.
Writing to this field sets the Even/Odd frame (EONUM) field to odd frame.

Bit 28 **SEVNFRM:** Set even frame

Applies to isochronous OUT endpoints only.
Writing to this field sets the Even/Odd frame (EONUM) field to even frame.

Bit 27 **SNAK:** Set NAK

A write to this bit sets the NAK bit for the endpoint.
Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit for OUT endpoints on a transfer completed interrupt, or after a SETUP is received on the endpoint.

Bit 26 **CNAK:** Clear NAK

A write to this bit clears the NAK bit for the endpoint.

Bits 25:22 Reserved, must be kept at reset value.

Bit 21 **STALL:** STALL handshake

Applies to non-control, non-isochronous OUT endpoints only (access type is rw).
The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core.
Applies to control endpoints only (access type is rs).
The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 20 **SNPM:** Snoop mode

This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.

Bits 19:18 **EPTYP[1:0]:** Endpoint type

This is the transfer type supported by this logical endpoint.
00: Control
01: Isochronous
10: Bulk
11: Interrupt

Bit 17 **NAKSTS:** NAK status

Indicates the following:
0: The core is transmitting non-NAK handshakes based on the FIFO status.
1: The core is transmitting NAK handshakes on this endpoint.
When either the application or the core sets this bit:
The core stops receiving any data on an OUT endpoint, even if there is space in the Rx FIFO to accommodate the incoming packet.
Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.

Bit 16 **EONUM:** Even/odd frame

Applies to isochronous OUT endpoints only.

Indicates the frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd frame number in which it intends to transmit/receive isochronous data for this endpoint using the SEVNFRM and SODDFRM fields in this register.

0: Even frame

1: Odd frame

Bit 15 **USBAEP:** USB active endpoint

Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:0 **MPSIZ[10:0]:** Maximum packet size

The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.

73.14.60 OTG device OUT endpoint x transfer size register (OTG_DOEPTSIZx)

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using endpoint enable bit of the OTG_DOEPCTLx registers (EPENA bit in OTG_DOEPCTLx), the core modifies this register. The application can only read this register once the core has cleared the endpoint enable bit.

Address offset: 0xB10 + 0x20 * x, (x = 1 to 8)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	RXDPID[1:0]		PKTCNT[9:0]										XFRSIZ[18:16]		
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XFRSIZ[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:29 **RXDPID[1:0]:**

Condition: isochronous OUT endpoints
Received data PID
This is the data PID received in the last packet for this endpoint.
00: DATA0
01: DATA2
10: DATA1
11: MDATA
Condition: control OUT endpoints
STUPCNT[1:0]: SETUP packet count
This field specifies the number of back-to-back SETUP data packets the endpoint can receive.
01: 1 packet
10: 2 packets
11: 3 packets

Bits 28:19 **PKTCNT[9:0]:** Packet count

Indicates the total number of USB packets that constitute the transfer size amount of data for this endpoint.
This field is decremented every time a packet (maximum size or short packet) is written to the Rx FIFO.

Bits 18:0 **XFRSIZ[18:0]:** Transfer size

This field contains the transfer size in bytes for the current endpoint. The core only interrupts the application after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet.
The core decrements this field every time a packet is read from the Rx FIFO and written to the external memory.

73.14.61 OTG power and clock gating control register (OTG_PCGCCTL)

This register is available in host and device modes.

Address offset: 0xE00

Reset value: 0x200B 8000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUSP	PHY SLEEP	ENL1 GTG	PHY SUSP	Res.	Res.	GATE HCLK	STPP CLK
								r	r	rw	r			rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **SUSP:** Deep Sleep

This bit indicates that the PHY is in Deep Sleep when in L1 state.

Bit 6 **PHYSLEEP:** PHY in Sleep

This bit indicates that the PHY is in the Sleep state.

Bit 5 **ENL1GTG:** Enable sleep clock gating

When this bit is set, core internal clock gating is enabled in Sleep state if the core cannot assert utmi_l1_suspend_n. When this bit is not set, the PHY clock is not gated in Sleep state.

Bit 4 **PHYSUSP:** PHY suspended

Indicates that the PHY has been suspended. This bit is updated once the PHY is suspended after the application has set the STPPCLK bit.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **GATEHCLK:** Gate HCLK

The application sets this bit to gate HCLK to modules other than the AHB Slave and Master and wake-up logic when the USB is suspended or the session is not valid. The application clears this bit when the USB is resumed or a new session starts.

Bit 0 **STPPCLK:** Stop PHY clock

The application sets this bit to stop the PHY clock when the USB is suspended, the session is not valid, or the device is disconnected. The application clears this bit when the USB is resumed or a new session starts.

73.14.62 OTG power and clock gating control register 1 (OTG_PCGCCTL1)

This register is available in host and device modes.

Address offset: 0xE04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RAM GATE EN	CNTGATECLK [1:0]		GATE EN
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **RAMGATEEN:** Enable RAM clock gating

Enable gating of the FIFO RAM.

Bits 2:1 **CNTGATECLK[1:0]:** Counter for clock gating

Indicates to the controller how many PHY Clock cycles and AHB Clock cycles of 'IDLE' (no activity) the controller waits for before Gating the respective PHY and AHB clocks internal to the controller.

00: 64 clocks

01: 128 clocks

10: Reserved

11: Reserved

Bit 0 **GATEEN:** Enable active clock gating

The application programs GATEEN to enable Active Clock Gating feature for the PHY and AHB clocks.

73.14.63 OTG register map

The table below gives the USB OTG register map and reset values.

Table 747. OTG register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	OTG_ GOTGCTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CURMOD	OTGVER	BSVLD	ASVLD	DBCT	CIDSTS	Res.	Res.	Res.	EHEN	Res.	Res.	Res.	Res.	BVALOVAL	BVALOEN	AVALOVAL	AVALOEN	VBVALOVA	VBVALOEN	Res.	Res.
	Reset value											0	0	0	0	0	1				0					0	0	0	0	0	0		
0x004	OTG_ GOTGINT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADTOCHG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEDET	Res.	Res.
	Reset value														0								-								0		
0x008	OTG_ GAHBCFG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PTXFELVL	TXFELVL	Res.	DMAEN	HBSTLEN				GINTMSK
	Reset value																								0	0		0	0	0	0	0	0
0x00C	OTG_ GUSBCFG	Res.	FDMOD	FHMOD	Res.	Res.	Res.	Res.	Res.	Res.	TSDPS	Res.	Res.	Res.	Res.	Res.	Res.	PHYLPC	Res.	TRDT				Res.	Res.	Res.	Res.	Res.	Res.	Res.	TOCAL		
	Reset value		0	0							0							0		0	1	0	1								0	0	0
0x010	OTG_GRSTCTL	AHBIDL	DMAREQ	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXFNUM					TXFFLSH	RXFFLSH	Res.	FCRST	PSRST	CSRST
	Reset value	1	0																				0	0	0	0	0	0	0		0	0	0
0x014	OTG_GINTSTS (Device mode)	WKUPINT	SRQINT	DISCINT	CIDSCHG	LPMINT	PTXFE	HCINT	HPRTINT	RSTDET	DATAFSUSP	IPXFR	IISOIXFR	OEPINT	IEPINT	Res.	Res.	EOPF	ISOODRP	ENUMDNE	USBRST	USBSUSP	ESUSP	Res.	Res.	GONAKEFF	GINAKEFF	NPTXFE	RXFLVL	SOF	OTGINT	MMIS	CMOD
	Reset value	0	0	0	0	0	1	0	0	0	0	0	0	0	0			0	0	0	0	0	0			0	0	1	0	0	0	0	0
	OTG_GINTSTS (Host mode)	WKUPINT	SRQINT	DISCINT	CIDSCHG	LPMINT	PTXFE	HCINT	HPRTINT	RSTDET	DATAFSUSP	INCOMPISOOUT	IISOIXFR	OEPINT	IEPINT	Res.	Res.	EOPF	ISOODRP	ENUMDNE	USBRST	USBSUSP	ESUSP	Res.	Res.	GONAKEFF	GINAKEFF	NPTXFE	RXFLVL	SOF	OTGINT	MMIS	CMOD
	Reset value	0	0	0	0	0	1	0	0	0	0	0	0	0	0			0	0	0	0	0	0			0	0	1	0	0	0	0	0

Table 747. OTG register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x018	OTG_GINTMSK (Device mode)	WUIM	SRQIM	DISCINT	CIDSCHGM	LPMINTM	PTXFEM	HCIM	PRTIM	RSTDETM	FSUSPM	IPXFRM	IISOIXFRM	OEPINT	IEPINT	Res.	Res.	EOPFM	ISOODRPM	ENUMDNEM	USBRST	USBSUSPM	ESUSPM	Res.	Res.	GONAKEFFM	GINAKEFFM	NPTXFEM	RXFLVLM	SOFM	OTGINT	MMISM	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0	0	
	OTG_GINTMSK (Host mode)	WUIM	SRQIM	DISCINT	CIDSCHGM	LPMINTM	PTXFEM	HCIM	PRTIM	RSTDETM	FSUSPM	IISOOXFRM	IISOIXFRM	OEPINT	IEPINT	Res.	Res.	EOPFM	ISOODRPM	ENUMDNEM	USBRST	USBSUSPM	ESUSPM	Res.	Res.	GONAKEFFM	GINAKEFFM	NPTXFEM	RXFLVLM	SOFM	OTGINT	MMISM	Res.
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0	0			0	0	0	0	0	0	0	
0x01C	OTG_GRXSTSR (Device mode)	Res.	Res.	Res.	Res.	STSPHST	Res.	Res.	FRMNUM				PKTSTS				DPID		BCNT											EPNUM			
	Reset value					0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	OTG_GRXSTSR (Host mode)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKTSTS				DPID		BCNT											CHNUM			
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	OTG_GRXSTSP (Device mode)	Res.	Res.	Res.	Res.	STSPHST	Res.	Res.	FRMNUM				PKTSTS				DPID		BCNT											EPNUM			
	Reset value					0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	OTG_ GRXSTSP (Host mode)	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKTSTS				DPID		BCNT											CHNUM			
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	OTG_ GRXFSIZ	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXFD															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x028	OTG_ HNPTXFSIZ (Device mode)	NPTXFD																NPTXFSA															
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
	OTG_ DIEPTXF0 (Host mode)	TX0FD																TX0FSA															
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x02C	OTG_ HNPTXSTS	Res.	NPTXQTOP							NPTQXSAV								NPTXFSAV															
	Reset value		0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0x038	OTG_GCCFG	Res.	Res.	Res.	IDPULLUPDIS	Res.	FORCEHOSTPD	Res.	FORCEHOSTPD	SDEN	Res.	PDEN	DCDEN	HVDMSRCEN	HCDPDETEN	HCDPEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VBUSVLD	Res.	FSVMINUS	FSVPLUS	CHGDET
	Reset value				0		0	0		0	0		0	0	0	0	0												X		X	X	X

Table 747. OTG register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x03C	**OTG_CID**	PRODUCT_ID																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0x054	OTG_GLPMCFG	Res.	Res.	Res.	ENBESL	LPMR CNTSTS			SNDLPM	LPM RCNT			LPMCHIDX				L1RSMOK	SLPSTS	LPM RSP		L1DSEN	BESLTHRS				L1SSEN	REMWAKE	BESL				LPMACK	LPMEN
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x100	OTG_HPTXFSIZ	PTXFSIZ																PTXSA															
	Reset value	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0x104	OTG_ DIEPTXF1	INEPTXFD																INEPTXSA															
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
. . . .	. . . .	.																															
0x120	OTG_DIEPTXF7	INEPTXFD																INEPTXSA															
	Reset value	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
0x400	OTG_HCFG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSLSS	FSLS PCS	
	Reset value																														0	0	0
0x404	OTG_HFIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RLDCTRL	FRIVL															
	Reset value																0	1	1	1	0	1	0	1	0	0	1	1	0	0	0	0	0
0x408	OTG_HFNUM	FTREM																FRNUM															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x410	OTG_HPTXSTS	PTXQTOP								PTXQSAV								PTXFSAVL															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0x414	OTG_HAINT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HAINT															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x418	OTG_ HAINTMSK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HAINTM															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x440	OTG_HPRT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSP D	PTCTL					PPWR	PLSTS		Res.	PRST	PSUSP	PRES	POCCHNG	POCA	PENCHNG	PENA	PCDET	PCSTS
	Reset value														0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0
0x500	OTG_ HCCHAR0	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP	LSDEV	Res.	EPDIR	EPNUM				MPSIZ											
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 747. OTG register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x504	OTG_ HCSPLT0	SPLITEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMPLSPLT	XACTPOS		HUBADDR							PRTADDR						
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x508	OTG_HCINT0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTERR	FRMOR	BBERR	TXERR	Res.	ACK	NAK	STALL	Res.	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0
0x508	OTG_HCINT0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTERR	FRMOR	BBERR	TXERR	NYET	ACK	NAK	STALL	AHBERR	CHH	XFRC
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x50C	OTG_ HCINTMSK0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	Res.	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0		0	0
0x510	OTG_HCTSIZ0	DOPNG	DPID		PKTCNT										XFRSIZ																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x514	OTG_HCDMA0	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
. . . .	. . . .	.Iterating preceding block of registers starting at offset 0x500..																															
0x6E0	OTG_ HCCHAR15	CHENA	CHDIS	ODDFRM	DAD							MCNT		EPTYP		LSDEV	Res.	EPDIR	EPNUM				MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x6E4	OTG_ HCSPLT15	SPLITEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMPLSPLT	XACTPOS		HUBADDR							PRTADDR						
	Reset value	0															0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x6E8	OTG_HCINT15	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTERR	FRMOR	BBERR	TXERR	Res.	ACK	NAK	STALL	Res.	CHH	XFRC
	Reset value																						0	0	0	0		0	0	0		0	0
0x6EC	OTG_ HCINTMSK15	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DTERRM	FRMORM	BBERRM	TXERRM	NYET	ACKM	NAKM	STALLM	Res.	CHHM	XFRCM
	Reset value																						0	0	0	0	0	0	0	0		0	0

Table 747. OTG register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x6F0	OTG_ HCTSIZ15	DOPNG	DPID		PKTCNT										XFRSIZ																		
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x6F4	OTG_HCDMA15	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x800	OTG_DCFG	Res.	Res.	Res.	Res.	Res.	Res.	PERSCHIVL		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ERRATIM	XCVRDLY	Res.	PFIVL		DAD							Res.	NZLSOHSK	DSPD	
	Reset value							0	0									0	0		0	0	0	0	0	0	0	0		0	0	0
0x804	OTG_DCTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DSBESLRJCT	Res.	Res.	Res.	Res.	Res.	Res.	POPRGDNE	CGONAK	SGONAK	CGINAK	SGINAK	TCTL			GONSTS	GINSTS	SDIS	RWUSIG
	Reset value														0							0	0	0	0	0	0	0	0	0	1	0
0x808	OTG_DSTS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEVLNSTS		FNSOF														Res.	Res.	Res.	Res.	EERR	ENUMSPD		SUSPSTS
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0
0x810	OTG_DIEPMSK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NAKM	Res.	Res.	Res.	Res.	TXFURM	Res.	INEPNEM	INEPNMM	ITTXFEMSK	TOM	AHBERRM	EPDM	XFRCM
	Reset value																			0					0		0	0	0	0	0	0	0
0x814	OTG_DOEPMSK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NYETMSK	NAKMSK	BERRM	Res.	Res.	Res.	OUTPKTERRM	Res.	B2BSTUPM	Res.	OTEPDM	STUPM	AHBERRM	EPDM	XFRCM
	Reset value																		0	0	0				0		0		0	0	0	0	0
0x818	OTG_DAINT	OEPINT																IEPINT															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x81C	OTG_ DAINTMSK	OEPM																IEPM															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x830	OTG_ DTHRCTL	Res.	Res.	Res.	Res.	ARPEN	Res.	RXTHRLEN							RXTHREN	Res.	Res.	Res.	Res.	TXTHRLEN												ISOTHREN	NONISOTHREN
	Reset value					0									0							0	0	0	0	0	0	0	0	0	0	0	0
0x834	OTG_DIE PEMPMSK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INEPTXFEM															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 747. OTG register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x900	OTG_ DIEPCTL0 (INT/BULK)	EPENA	EPDIS	SD1PID	SD1P0D	SNAK	CNAK	TXFNUM				STALL	Res.	EPTYP		NAKSTS	DPID	USBAEP	Res.	Res.	Res.	Res.	MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
	OTG_ DIEPCTL0 (ISO)	EPENA	EPDIS	SODDFRM	SEVNFRM	SNAK	CNAK	TXFNUM				STALL	Res.	EPTYP		NAKSTS	EONUM	USBAEP	Res.	Res.	Res.	Res.	MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0x908	OTG_ DIEPINT0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NAK	Res.	PKTDRPSTS	Res.	Res.	TXFIFOUDRN	TXFE	INEPNE	INEPNM	ITTXFE	TOC	AHBERR	EPDISD	XFRC
	Reset value																			0		0			0	1	0	0	0	0	0	0	0
0x910	OTG_ DIEPTSIZ0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKT CNT		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	XFRSIZ						
	Reset value												0	0													0	0	0	0	0	0	0
0x914	OTG_ DIEPDMA0	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x918	OTG_ DTXFSTS0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INEPTFSAV															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
. . . .	. . . .	Iterating preceding block of registers starting at offset 0x900.																															
0x9E0	OTG_ DIEPCTL7 (INT/BULK)	EPENA	EPDIS	SD1PID	SD1P0D	SNAK	CNAK	TXFNUM				STALL	Res.	EPTYP		NAKSTS	DPID	USBAEP	Res.	Res.	Res.	Res.	MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
	OTG_ DIEPCTL7 (ISO)	EPENA	EPDIS	SODDFRM	SEVNFRM	SNAK	CNAK	TXFNUM				STALL	Res.	EPTYP		NAKSTS	EONUM	USBAEP	Res.	Res.	Res.	Res.	MPSIZ										
	Reset value	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0x9E8	OTG_ DIEPINT7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NAK	Res.	PKTDRPSTS	Res.	Res.	TXFIFOUDRN	TXFE	INEPNE	INEPNM	ITTXFE	TOC	AHBERR	EPDISD	XFRC
	Reset value																			0		0			0	1	0	0	0	0	0	0	0
0x9F0	OTG_ DIEPTSIZ7	Res.	MCNT		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x9F8	OTG_ DTXFSTS7	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INEPTFSAV															
	Reset value																	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

Table 747. OTG register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xB00	OTG_ DOEPCTL0	EPENA	EPDIS	Res.	Res.	SNAK	CNAK	Res.	Res.	Res.	Res.	STALL	SNPM	EPTYP		NAKSTS	Res.	USBAEP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MPSIZ	
	Reset value	0	0			0	0					0	0	0	0	0		1														0	0
0xB08	OTG_ DOEPINT0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STPKTRX	NYET	NAK	BERR	Res.	Res.	Res.	OUTPKTERR	Res.	B2BSTUP	STSPHSRX	OTEPDIS	STUP	AHBERR	EPDISD	XFRC
	Reset value																	0	0	0	0				0		0	0	0	0	0	0	0
0xB10	OTG_ DOEPTSIZ0	Res.	STUP CNT		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKTCNT		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	XFRSIZ						
	Reset value		0	0										0													0	0	0	0	0	0	0
0xB14	OTG_ DOEPDMA0	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB20	OTG_ DOEPCTL1 (INT/BULK)	EPENA	EPDIS	SD1PID	SD0PID	SNAK	CNAK	Res.	Res.	Res.	Res.	STALL	SNPM	EP TYP		NAKSTS	DPID	USBAEP	Res.	Res.	Res.	Res.	MPSIZ										
	Reset value	0	0	0	0	0	0					0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
	OTG_ DOEPCTL1 (ISO)	EPENA	EPDIS	SODDFRM	SEVNFRM	SNAK	CNAK	Res.	Res.	Res.	Res.	STALL	SNPM	EP TYP		NAKSTS	EONUM	USBAEP	Res.	Res.	Res.	Res.	MPSIZ										
	Reset value	0	0	0	0	0	0					0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0xB28	OTG_ DOEPINT1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STPKTRX	NYET	NAK	BERR	Res.	Res.	Res.	OUTPKTERR	Res.	B2BSTUP	STSPHSRX	OTEPDIS	STUP	AHBERR	EPDISD	XFRC
	Reset value																	0	0	0	0				0		0	0	0	0	0	0	0
0xB30	OTG_ DOEPTSIZ1	Res.	RXDPID/ STUPCNT		PKTCNT										XFRSIZ																		
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xB34	OTG_ DOEPDMA1	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
. . . .	. . . .	Iterating preceding block of registers starting at offset 0xB20.																															

Table 747. OTG register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xC00	OTG_ DOEPCTL8 (INT/BULK)	EPENA	EPDIS	SD1PID	SD0PID	SNAK	CNAK	Res.	Res.	Res.	Res.	STALL	SNPM	EP TYP		NAKSTS	DPID	USBAEP	Res.	Res.	Res.	Res.	MPSIZ										
	Reset value	0	0	0	0	0	0					0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
	OTG_ DOEPCTL8 (ISO)	EPENA	EPDIS	SODDFRM	SEVNFRM	SNAK	CNAK	Res.	Res.	Res.	Res.	STALL	SNPM	EP TYP		NAKSTS	EONUM	USBAEP	Res.	Res.	Res.	Res.	MPSIZ										
	Reset value	0	0	0	0	0	0					0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0
0xC08	OTG_ DOEPINT8	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STPKTRX	NYET	NAK	BERR	Res.	Res.	Res.	OUTPKTERR	Res.	B2BSTUP	STSPHSRX	OTEPDIS	STUP	AHBERR	EPDISD	XFRC
	Reset value																	0	0	0	0		-		0		0	0	0	0	0	0	0
0xC10	OTG_ DOEPTSIZ8	Res.	RXDPID/ STUPCNT		PKTCNT											XFRSIZ																	
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xC14	OTG_ DOEPDMA8	DMAADDR																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xE00	OTG_ PCGCCTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUSP	PHYSLEEP	ENL1GTG	PHYSUSP	Res.	Res.	GATEHCLK	STPPCLK
	Reset value																									0	0	0	0			0	0
0xE04	OTG_ PCGCCTL1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RAMGATEEN	CNTGATECLK		GATEEN
	Reset value																													0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

73.15 OTG programming model

73.15.1 Core initialization

The application must perform the core initialization sequence. If the cable is connected during power-up, the current mode of operation bit in the OTG_GINTSTS (CMOD bit in OTG_GINTSTS) reflects the mode. The OTG controller enters host mode when an “A” plug is connected or device mode when a “B” plug is connected.

This section explains the initialization of the OTG controller after power-on. The application must follow the initialization sequence irrespective of host or device mode operation. All core global registers are initialized according to the core's configuration:

1. Program the following fields in the OTG_GAHBCFG register:
   – Global interrupt mask bit GINTMSK = 1
   – Rx FIFO non-empty (RXFLVL bit in OTG_GINTSTS)
   – Periodic Tx FIFO empty level
2. Program the following fields in the OTG_GUSBCFG register:
   – OTG timeout calibration field
   – USB turnaround time field
3. The software must unmask the following bits in the OTG_GINTMSK register:
   OTG interrupt mask
   Mode mismatch interrupt mask
4. The software can read the CMOD bit in OTG_GINTSTS to determine whether the OTG controller is operating in host or device mode.

73.15.2 Host initialization

To initialize the core as host, the application must perform the following steps:

1. Program the HPRTINT in the OTG_GINTMSK register to unmask
2. Program the OTG_HCFG register to select full-speed host
3. Program the PPWR bit in OTG_HPRT to 1. The port is now considered powered however this does not actually drive V_{BUS} on the USB, so a further action must be taken to control the external circuitry in the system so as to enable V_{BUS} generation.
4. Wait for the PCDET interrupt in OTG_HPRT0. This indicates that a device is connecting to the port.
5. Program the PRST bit in OTG_HPRT to 1. This starts the reset process.
6. Wait at least 10 ms for the reset process to complete.
7. Program the PRST bit in OTG_HPRT to 0.
8. Wait for the PENCHNG interrupt in OTG_HPRT.
9. Read the PSPD bit in OTG_HPRT to get the enumerated speed.
10. Program the HFIR register with a value corresponding to the selected PHY clock 1
11. Program the FSLSPCS field in the OTG_HCFG register following the speed of the device detected in step 9. If FSLSPCS has been changed a port reset must be performed.
12. Program the OTG_GRXFSIZ register to select the size of the receive FIFO.
13. Program the OTG_HNPTXFSIZ register to select the size and the start address of the Non-periodic transmit FIFO for non-periodic transactions.
14. Program the OTG_HPTXFSIZ register to select the size and start address of the periodic transmit FIFO for periodic transactions.

To communicate with devices, the system software must initialize and enable at least one channel.

73.15.3 Device initialization

The application must perform the following steps to initialize the core as a device on power-up or after a mode change from host to device.

1. Program the following fields in the OTG_DCFG register:
   - Device speed
   - Non-zero-length status OUT handshake
   - Periodic Frame Interval
2. Program the Device threshold control register. This is required only if you are using DMA mode and you are planning to enable thresholding.
3. Clear the DCTL.SDIS bit. The core issues a connect after this bit is cleared.
4. Program the OTG_GINTMSK register to unmask the following interrupts:
   - USB reset
   - Enumeration done
   - Early suspend
   - USB suspend
   - SOF
5. Wait for the USBRST interrupt in OTG_GINTSTS. It indicates that a reset has been detected on the USB that lasts for about 10 ms on receiving this interrupt.
6. Wait for the ENUMDNE interrupt in OTG_GINTSTS. This interrupt indicates the end of reset on the USB. On receiving this interrupt, the application must read the OTG_DSTS register to determine the enumeration speed and perform the steps listed in *Endpoint initialization on enumeration completion*.

At this point, the device is ready to accept SOF packets and perform control transfers on control endpoint 0.

73.15.4 DMA mode

The OTG host uses the AHB master interface to fetch the transmit packet data (AHB to USB) and receive the data update (USB to AHB). The AHB master uses the programmed DMA address (OTG_HCDMAx register in host mode and OTG_DIEPDMAx/OTG_DOEPDMAx register in peripheral mode) to access the data buffers.

73.15.5 Host programming model

Channel initialization

The application must initialize one or more channels before it can communicate with connected devices. To initialize and enable a channel, the application must perform the following steps:

1. Program the OTG_GINTMSK register to unmask the following:
2. Channel interrupt
   – Non-periodic transmit FIFO empty for OUT transactions (applicable when operating in pipelined transaction-level with the packet count field programmed with more than one).
   – Non-periodic transmit FIFO half-empty for OUT transactions (applicable when operating in pipelined transaction-level with the packet count field programmed with more than one).
3. Program the OTG_HAINTMSK register to unmask the selected channels' interrupts.
4. Program the OTG_HCINTMSK register to unmask the transaction-related interrupts of interest given in the host channel interrupt register.
5. Program the selected channel's OTG_HCTSIZx register with the total transfer size, in bytes, and the expected number of packets, including short packets. The application must program the PID field with the initial data PID (to be used on the first OUT transaction or to be expected from the first IN transaction).
6. Program the OTG_HCCHARx register of the selected channel with the device's endpoint characteristics, such as type, speed, direction, and so forth. (The channel can be enabled by setting the channel enable bit to 1 only when the application is ready to transmit or receive any packet).
7. Program the selected channels in the OTG_HCSPLTx register(s) with the hub and port addresses (split transactions only).
8. Program the selected channels in the OTG_HCDMAx register(s) with the buffer start address (DMA transactions only).

Halting a channel

The application can disable any channel by programming the OTG_HCCHARx register with the CHDIS and CHENA bits set to 1. This enables the OTG host to flush the posted requests (if any) and generates a channel halted interrupt. The application must wait for the CHH interrupt in OTG_HCINTx before reallocating the channel for other transactions. The OTG host does not interrupt the transaction that has already been started on the USB.

To disable a channel in DMA mode operation, the application does not need to check for space in the request queue. The OTG host checks for space to write the disable

request on the disabled channel's turn during arbitration. Meanwhile, all posted requests are dropped from the request queue when the CHDIS bit in OTG_HCCHARx is set to 1.

Before disabling a channel, the application must ensure that there is at least one free space available in the non-periodic request queue (when disabling a non-periodic channel) or the periodic request queue (when disabling a periodic channel). The application can simply flush the posted requests when the request queue is full (before disabling the channel), by programming the OTG_HCCHARx register with the CHDIS bit set to 1 which automatically clears the CHENA bit to 0.

The application is expected to disable a channel on any of the following conditions:

1. When an STALL, TXERR, BBERR or DTERR interrupt in OTG_HCINTx is received for an IN or OUT channel. The application must be able to receive other interrupts (DTERR, Nak, data, TXERR) for the same channel before receiving the halt.
2. When an XFRC interrupt in OTG_HCINTx is received during a non periodic IN transfer or high-bandwidth interrupt IN transfer
3. When a DISCINT (disconnect device) interrupt in OTG_GINTSTS is received. (The application is expected to disable all enabled channels).
4. When the application aborts a transfer before normal completion.

Ping protocol

When the OTG host operates in high speed, the application must initiate the ping protocol when communicating with high-speed bulk or control (data and status stage) OUT endpoints. The application must initiate the ping protocol when it receives a NAK/NYET/TXERR interrupt. When the OTG host receives one of the above responses, it does not continue any transaction for a specific endpoint, drops all posted or fetched OUT requests (from the request queue), and flushes the corresponding data (from the transmit FIFO).This is valid in slave mode only. In Slave mode, the application can send a ping token either by setting the DOPING bit in OTG_HCTSIZx before enabling the channel or by just writing the OTG_HCTSIZx register with the DOPING bit set when the channel is already enabled. This enables the OTG host to write a ping request entry to the request queue. The application must wait for the response to the ping token (a NAK, ACK, or TXERR interrupt) before continuing the transaction or sending another ping token. The application can continue the data transaction only after receiving an ACK from the OUT endpoint for the requested ping. In DMA mode operation, the application does not need to set the DOPING bit in OTG_HCTSIZx for a NAK/NYET response in case of bulk/control OUT. The OTG host automatically sets the DOPING bit in OTG_HCTSIZx, and issues the ping tokens for bulk/control OUT. The OTG host continues sending ping tokens until it receives an ACK, and then switches automatically to the data transaction.

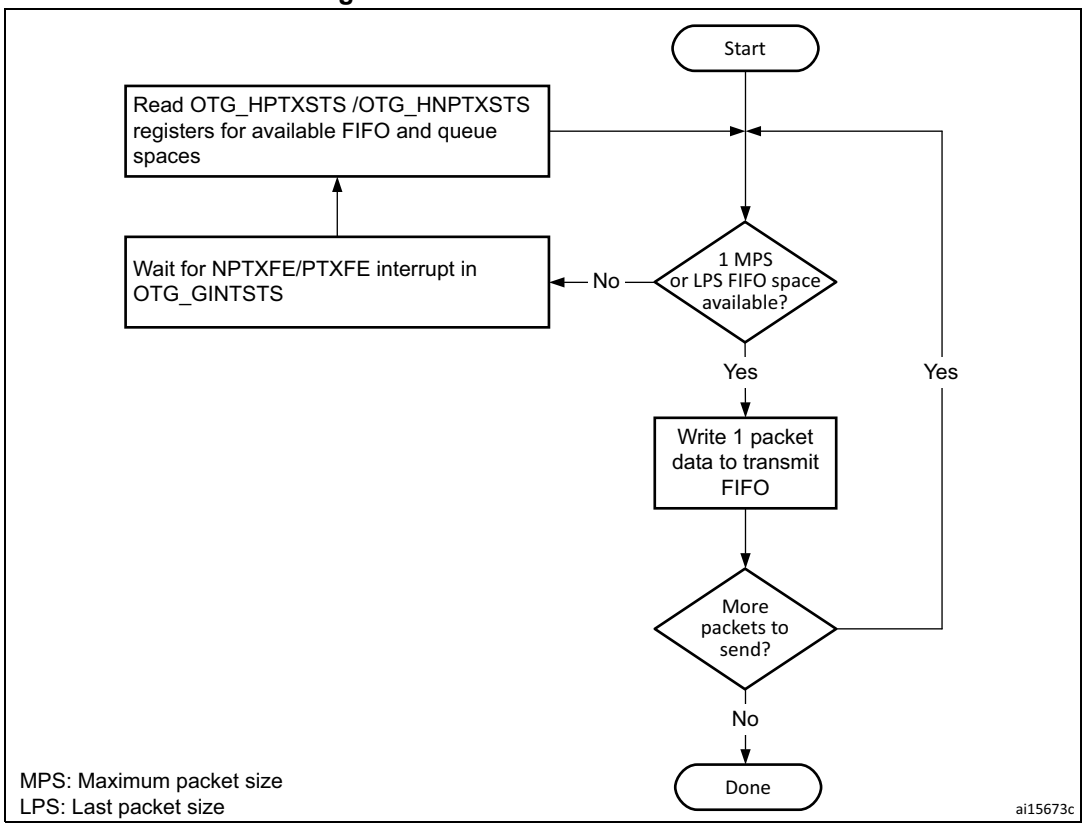
Operational model

The application must initialize a channel before communicating to the connected device. This section explains the sequence of operation to be performed for different types of USB transactions.

- **Writing the transmit FIFO**

  The OTG host automatically writes an entry (OUT request) to the periodic/non-periodic request queue, along with the last 32-bit word write of a packet. The application must ensure that at least one free space is available in the periodic/non-periodic request queue before starting to write to the transmit FIFO. The application must always write to the transmit FIFO in 32-bit words. If the packet size is non-32-bit word aligned, the application must use padding. The OTG host determines the actual packet size based on the programmed maximum packet size and transfer size.

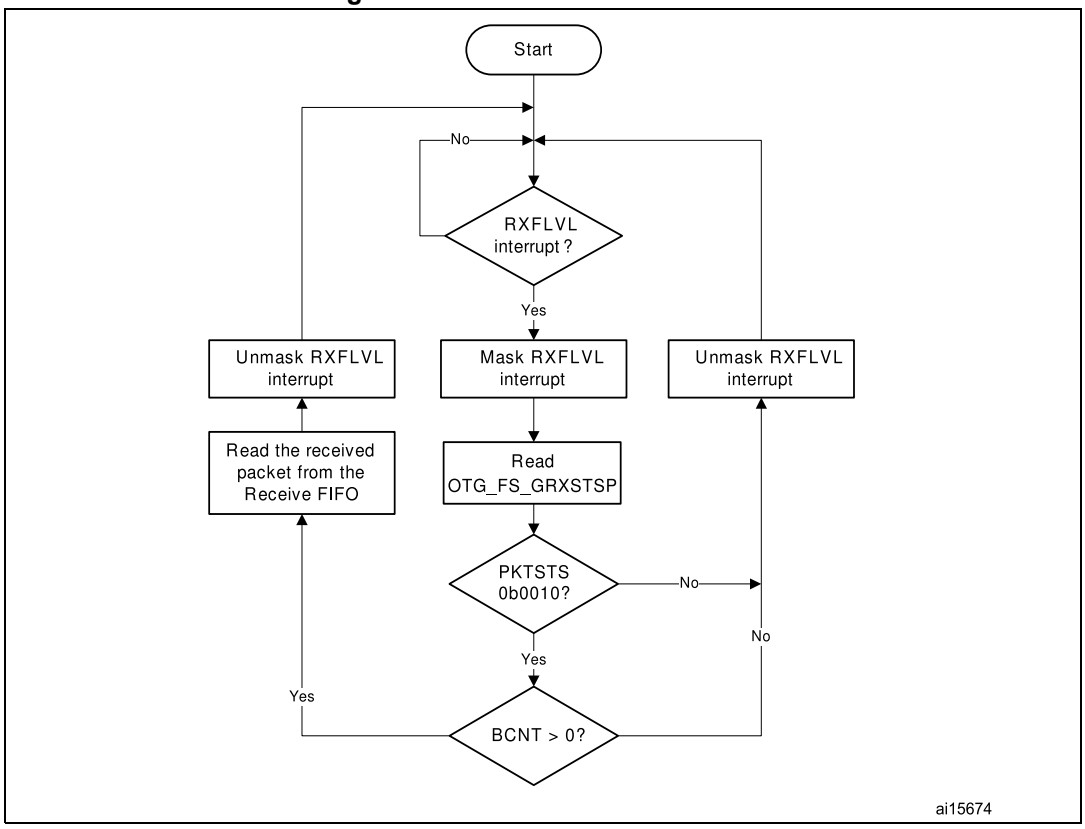
Figure 990. Transmit FIFO write task



- **Reading the receive FIFO**

  The application must ignore all packet statuses other than IN data packet (bx0010).

Figure 991. Receive FIFO read task



- **Bulk and control OUT/SETUP transactions**

  A typical bulk or control OUT/SETUP pipelined transaction-level operation is shown in *Figure 992*. See channel 1 (ch_1). Two bulk OUT packets are transmitted. A control SETUP transaction operates in the same way but has only one packet. The assumptions are:

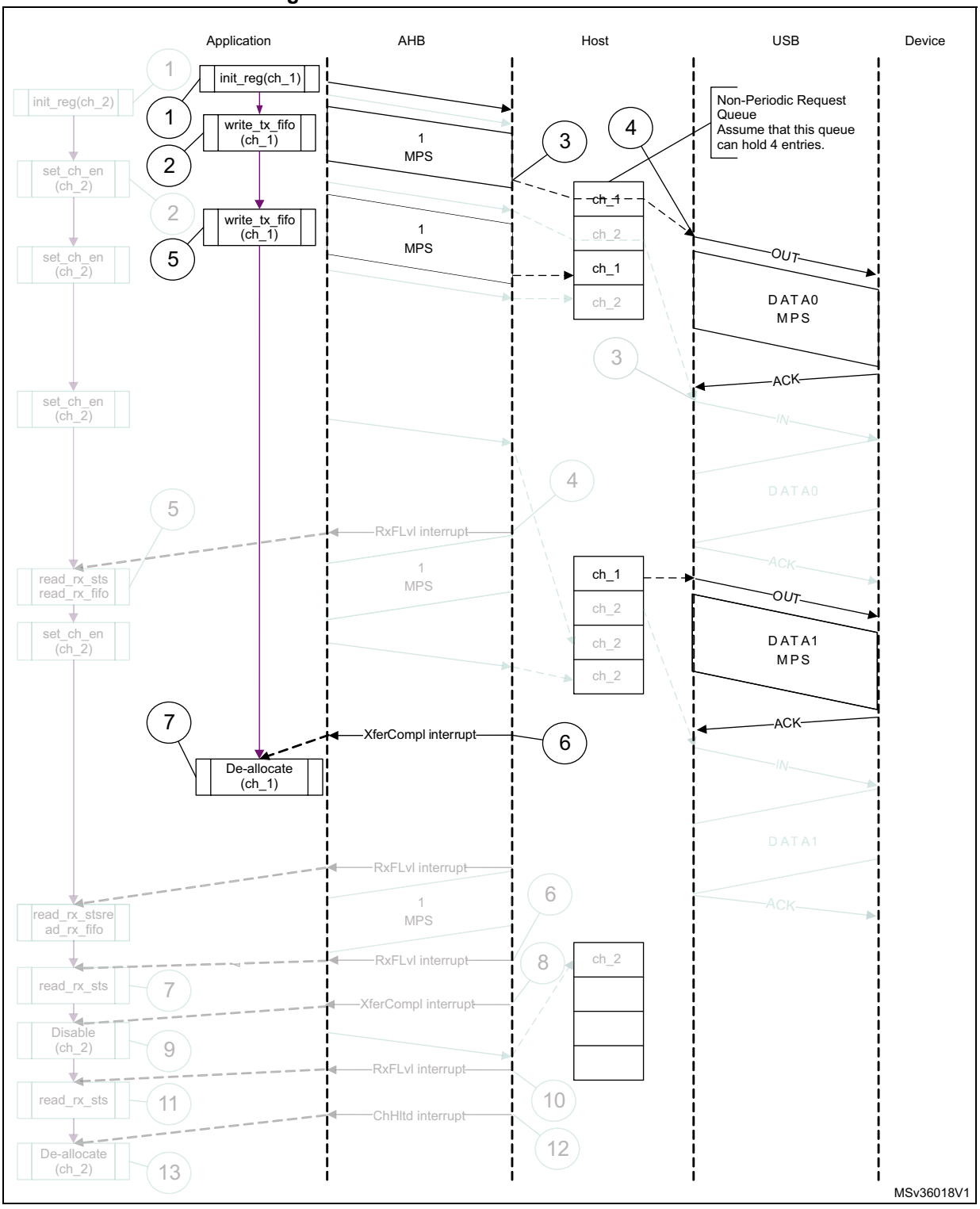
  – The application is attempting to send two maximum-packet-size packets (transfer size = 1, 024 bytes).
  – The non-periodic transmit FIFO can hold two packets (1 Kbyte for HS).
  – The non-periodic request queue depth = 4.

- **Normal bulk and control OUT/SETUP operations**

  The sequence of operations in (channel 1) is as follows:

1. Initialize channel 1
2. Write the first packet for channel 1
3. Along with the last word write, the core writes an entry to the non-periodic request queue
4. As soon as the non-periodic queue becomes non-empty, the core attempts to send an OUT token in the current frame
5. Write the second (last) packet for channel 1
6. The core generates the XFRC interrupt as soon as the last transaction is completed successfully
7. In response to the XFRC interrupt, de-allocate the channel for other transfers
8. Handling non-ACK responses

Figure 992. Normal bulk/control OUT/SETUP



1. The grayed elements are not relevant in the context of this figure.

The channel-specific interrupt service routine for bulk and control OUT/SETUP transactions is shown in the following code samples.

- **Interrupt service routine for bulk/control OUT/SETUP and bulk/control IN transactions**

  a) Bulk/control OUT/SETUP

```
Unmask (NAK/TXERR/STALL/XFRC)
if (XFRC)
   {
   Reset Error Count
   Mask ACK
   De-allocate Channel
   }
else if (STALL)
   {
   Transfer Done = 1
   Unmask CHH
   Disable Channel
   }
else if (NAK or TXERR )
   {
   Rewind Buffer Pointers
   Unmask CHH
   Disable Channel
   if (TXERR)
      {
      Increment Error Count
      Unmask ACK
      }
   else
      {
      Reset Error Count
      }
      }
else if (CHH)
   {
   Mask CHH
   if (Transfer Done or (Error_count == 3))
      {
      De-allocate Channel
      }
   else
      {
      Re-initialize Channel
      }
   }
```

```
   else if (ACK)
      {
      Reset Error Count
      Mask ACK
      }
```

The application is expected to write the data packets into the transmit FIFO when the space is available in the transmit FIFO and the request queue. The application can make use of the NPTXFE interrupt in OTG_GINTSTS to find the transmit FIFO space.

b) Bulk/control IN

```
Unmask (TXERR/XFRC/BBERR/STALL/DTERR)
if (XFRC)
   {
   Reset Error Count
   Unmask CHH
   Disable Channel
   Reset Error Count
   Mask ACK
   }
else if (TXERR or BBERR or STALL)
   {
   Unmask CHH
   Disable Channel
   if (TXERR)
      {
      Increment Error Count
      Unmask ACK
      }
   }
else if (CHH)
   {
   Mask CHH
   if (Transfer Done or (Error_count == 3))
      {
      De-allocate Channel
      }
   else
      {
      Re-initialize Channel
      }
   }
else if (ACK)
   {
   Reset Error Count
   Mask ACK
   }
```

```
else if (DTERR)
   {
   Reset Error Count
   }
```

The application is expected to write the requests as and when the request queue space is available and until the XFRC interrupt is received.

- **Bulk and control IN transactions**

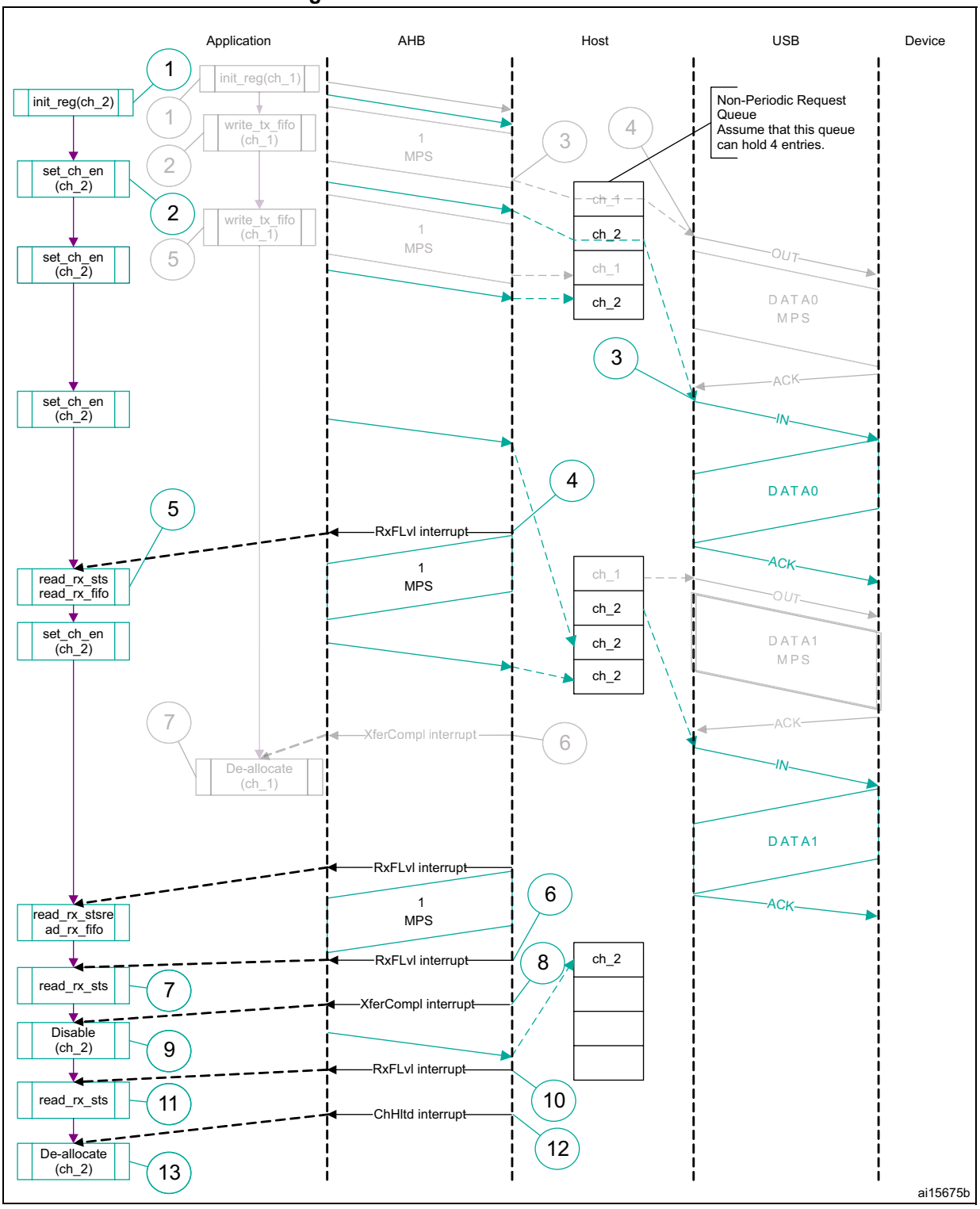
  A typical bulk or control IN pipelined transaction-level operation is shown in *Figure 993*. See channel 2 (ch_2). The assumptions are:

  – The application is attempting to receive two maximum-packet-size packets (transfer size = 1 024 bytes).

  – The receive FIFO can contain at least one maximum-packet-size packet and two status words per packet (520 bytes for HS).

  – The non-periodic request queue depth = 4.

Figure 993. Bulk/control IN transactions



1. The grayed elements are not relevant in the context of this figure.

The sequence of operations is as follows:

1. Initialize channel 2.
2. Set the CHENA bit in OTG_HCCHAR2 to write an IN request to the non-periodic request queue.
3. The core attempts to send an IN token after completing the current OUT transaction.
4. The core generates an RXFLVL interrupt as soon as the received packet is written to the receive FIFO.
5. In response to the RXFLVL interrupt, mask the RXFLVL interrupt and read the received packet status to determine the number of bytes received, then read the receive FIFO accordingly. Following this, unmask the RXFLVL interrupt.
6. The core generates the RXFLVL interrupt for the transfer completion status entry in the receive FIFO.
7. The application must read and ignore the receive packet status when the receive packet status is not an IN data packet (PKTSTS in OTG_GRXSTSR ≠ 0b0010).
8. The core generates the XFRC interrupt as soon as the receive packet status is read.
9. In response to the XFRC interrupt, disable the channel and stop writing the OTG_HCCHAR2 register for further requests. The core writes a channel disable request to the non-periodic request queue as soon as the OTG_HCCHAR2 register is written.
10. The core generates the RXFLVL interrupt as soon as the halt status is written to the receive FIFO.
11. Read and ignore the receive packet status.
12. The core generates a CHH interrupt as soon as the halt status is popped from the receive FIFO.
13. In response to the CHH interrupt, de-allocate the channel for other transfers.
14. Handling non-ACK responses

- **Control transactions**

  Setup, data, and status stages of a control transfer must be performed as three separate transfers. setup-, data- or status-stage OUT transactions are performed similarly to the bulk OUT transactions explained previously. Data- or status-stage IN transactions are performed similarly to the bulk IN transactions explained previously. For all three stages, the application is expected to set the EPTYP field in

OTG_HCCHAR1 to control. During the setup stage, the application is expected to set the PID field in OTG_HCTSIZ1 to SETUP.

- **Interrupt OUT transactions**

  A typical interrupt OUT operation is shown in *Figure 994*. The assumptions are:

  – The application is attempting to send one packet in every frame (up to 1 maximum packet size), starting with the odd frame (transfer size = 1 024 bytes)

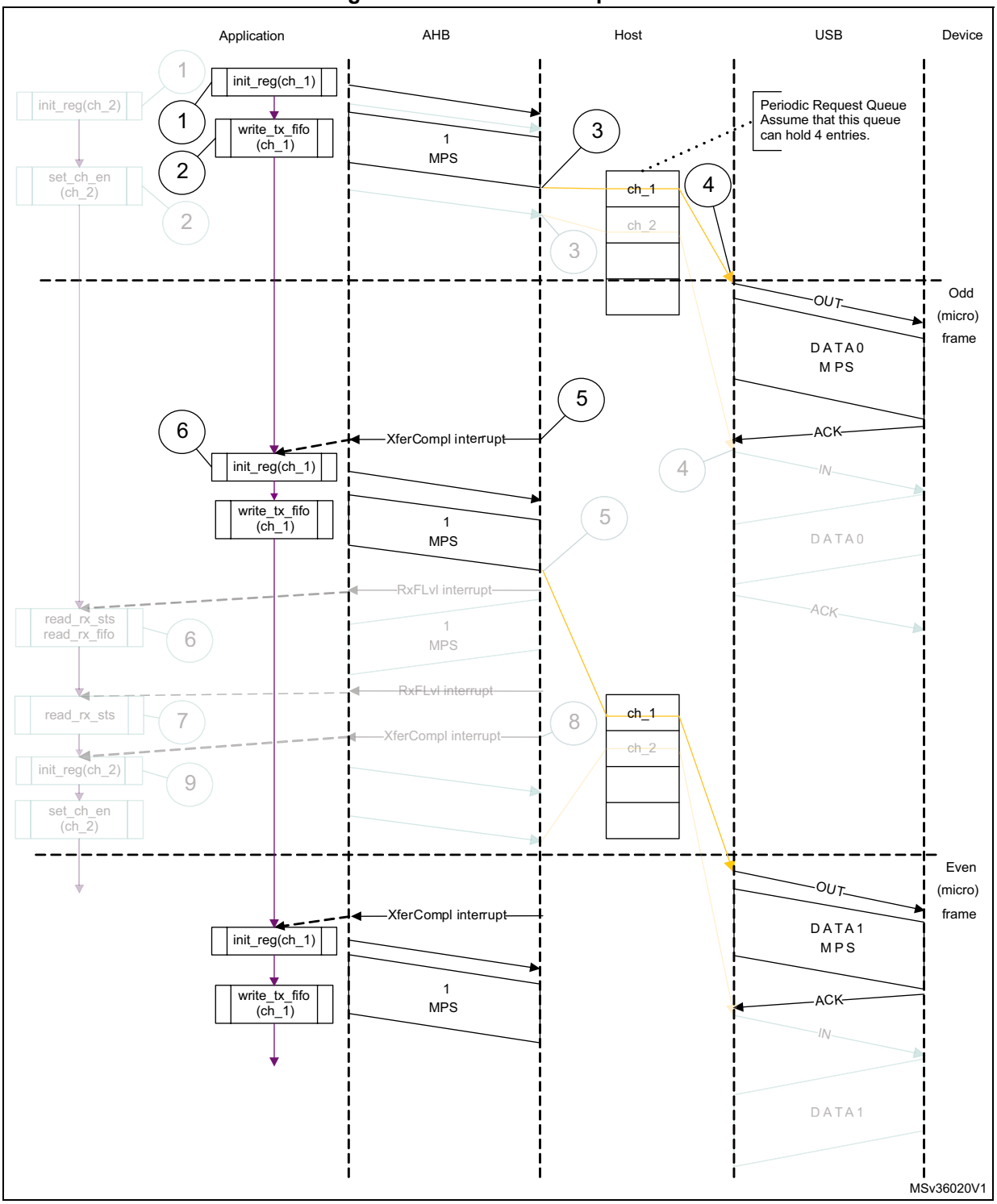
  – The periodic transmit FIFO can hold one packet (1 Kbyte)

  – Periodic request queue depth = 4

  The sequence of operations is as follows:

1. Initialize and enable channel 1. The application must set the ODDFRM bit in OTG_HCCHAR1.
2. Write the first packet for channel 1.
3. Along with the last word write of each packet, the OTG host writes an entry to the periodic request queue.
4. The OTG host attempts to send an OUT token in the next (odd) frame.
5. The OTG host generates an XFRC interrupt as soon as the last packet is transmitted successfully.
6. In response to the XFRC interrupt, reinitialize the channel for the next transfer.

Figure 994. Normal interrupt OUT



1. The grayed elements are not relevant in the context of this figure.

- **Interrupt service routine for interrupt OUT/IN transactions**

  a) Interrupt OUT

`Unmask (NAK/TXERR/STALL/XFRC/FRMOR)`

```
if (XFRC)
   {
   Reset Error Count
   Mask ACK
   De-allocate Channel
   }
else
   if (STALL or FRMOR)
      {
      Mask ACK
      Unmask CHH
      Disable Channel
      if (STALL)
         {
         Transfer Done = 1
         }
      }
   else
      if (NAK or TXERR)
         {
         Rewind Buffer Pointers
         Reset Error Count
         Mask ACK
         Unmask CHH
         Disable Channel
         }
      else
         if (CHH)
         {
         Mask CHH
         if (Transfer Done or (Error_count == 3))
            {
            De-allocate Channel
            }
         else
            {
            Re-initialize Channel (in next b_interval - 1 Frame)
            }
         }
      else
         if (ACK)
         {
         Reset Error Count
         Mask ACK
         }
```

```
The application uses the NPTXFE interrupt in OTG_GINTSTS to find the
transmit FIFO space.
Interrupt IN
Unmask (NAK/TXERR/XFRC/BBERR/STALL/FRMOR/DTERR)
if (XFRC)
   {
   Reset Error Count
   Mask ACK
   if (OTG_HCTSIZx.PKTCNT == 0)
      {
      De-allocate Channel
      }
   else
      {
      Transfer Done = 1
      Unmask CHH
      Disable Channel
      }
   }
else
   if (STALL or FRMOR or NAK or DTERR or BBERR)
      {
      Mask ACK
      Unmask CHH
      Disable Channel
      if (STALL or BBERR)
         {
         Reset Error Count
         Transfer Done = 1
         }
      else
         if (!FRMOR)
         {
         Reset Error Count
         }
      }
else
   if (TXERR)
      {
      Increment Error Count
      Unmask ACK
      Unmask CHH
      Disable Channel
      }
else
```

```
   if (CHH)
      {
      Mask CHH
      if (Transfer Done or (Error_count == 3))
         {
         De-allocate Channel
         }
      else
         Re-initialize Channel (in next b_interval - 1 /Frame)
      }
   }
else
   if (ACK)
      {
      Reset Error Count
      Mask ACK
      }
```

- **Interrupt IN transactions**

  The assumptions are:

  – The application is attempting to receive one packet (up to 1 maximum packet size) in every frame, starting with odd (transfer size = 1 024 bytes).

  – The receive FIFO can hold at least one maximum-packet-size packet and two status words per packet (1 031 bytes).

  – Periodic request queue depth = 4.

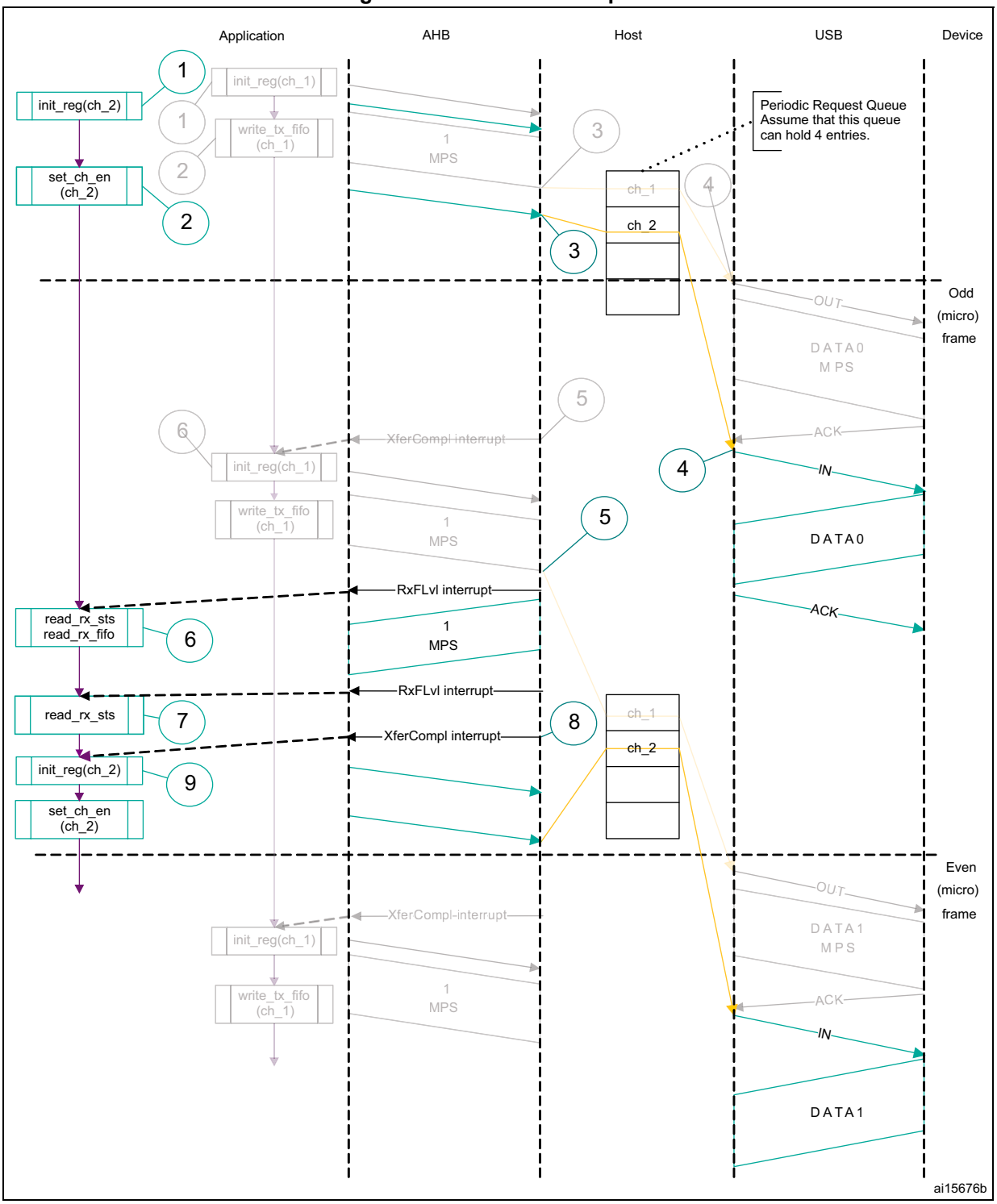
- **Normal interrupt IN operation**

  The sequence of operations is as follows:

1. Initialize channel 2. The application must set the ODDFRM bit in OTG_HCCHAR2.
2. Set the CHENA bit in OTG_HCCHAR2 to write an IN request to the periodic request queue.
3. The OTG host writes an IN request to the periodic request queue for each OTG_HCCHAR2 register write with the CHENA bit set.
4. The OTG host attempts to send an IN token in the next (odd) frame.
5. As soon as the IN packet is received and written to the receive FIFO, the OTG host generates an RXFLVL interrupt.
6. In response to the RXFLVL interrupt, read the received packet status to determine the number of bytes received, then read the receive FIFO accordingly. The application must mask the RXFLVL interrupt before reading the receive FIFO, and unmask after reading the entire packet.
7. The core generates the RXFLVL interrupt for the transfer completion status entry in the receive FIFO. The application must read and ignore the receive packet status when the receive packet status is not an IN data packet (PKTSTS in GRXSTSR ≠ 0b0010).
8. The core generates an XFRC interrupt as soon as the receive packet status is read.
9. In response to the XFRC interrupt, read the PKTCNT field in OTG_HCTSIZ2. If the PKTCNT bit in OTG_HCTSIZ2 is not equal to 0, disable the channel before re-

initializing the channel for the next transfer, if any). If PKTCNT bit in OTG_HCTSIZ2 = 0, reinitialize the channel for the next transfer. This time, the application must reset the ODDFRM bit in OTG_HCCHAR2.

Figure 995. Normal interrupt IN



1. The grayed elements are not relevant in the context of this figure.

- **Isochronous OUT transactions**

  A typical isochronous OUT operation is shown in *Figure 996*. The assumptions are:

  – The application is attempting to send one packet every frame (up to 1 maximum

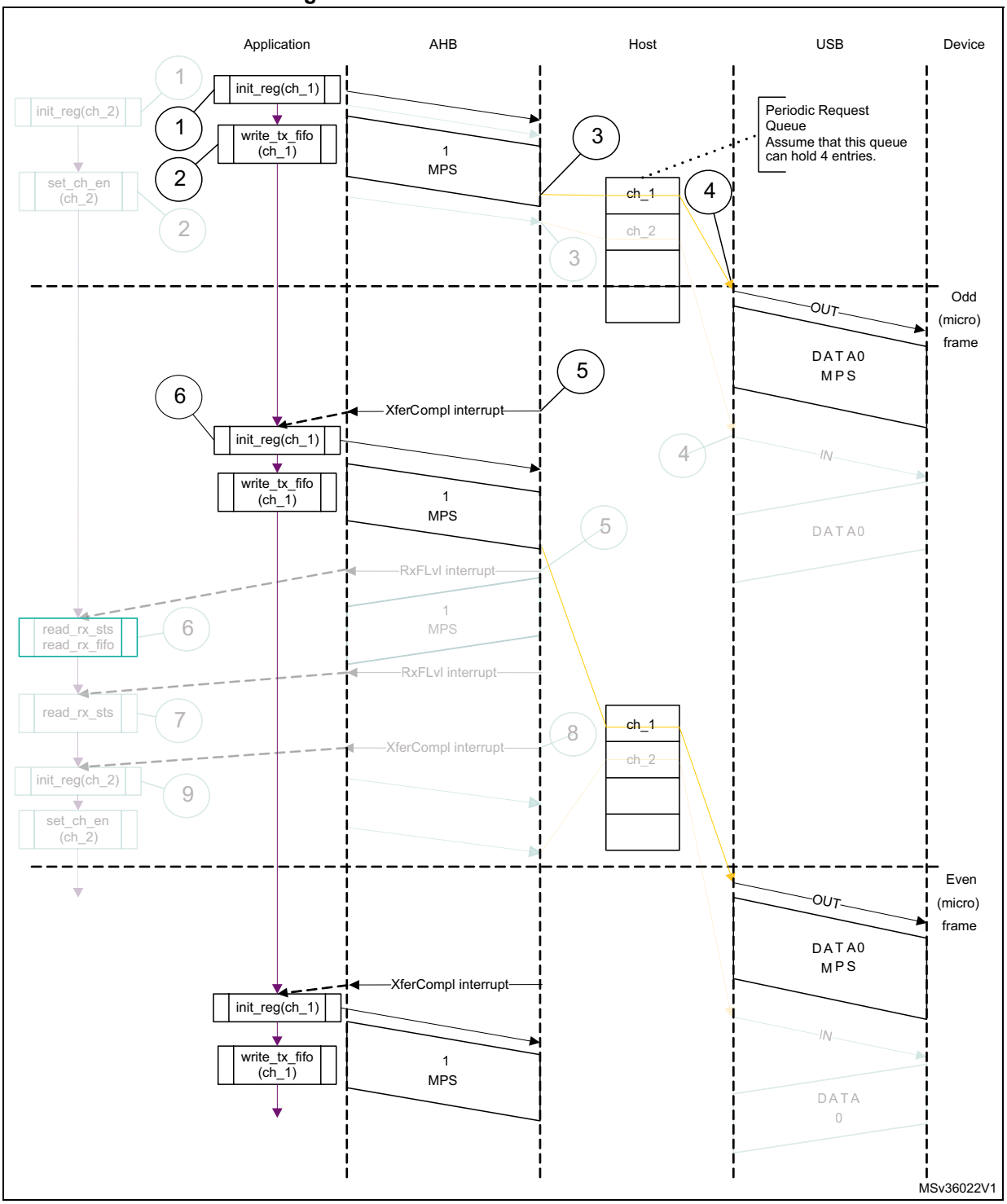
packet size), starting with an odd frame. (transfer size = 1 024 bytes).

– The periodic transmit FIFO can hold one packet (1 Kbyte).
– Periodic request queue depth = 4.

The sequence of operations is as follows:

1. Initialize and enable channel 1. The application must set the ODDFRM bit in OTG_HCCHAR1.
2. Write the first packet for channel 1.
3. Along with the last word write of each packet, the OTG host writes an entry to the periodic request queue.
4. The OTG host attempts to send the OUT token in the next frame (odd).
5. The OTG host generates the XFRC interrupt as soon as the last packet is transmitted successfully.
6. In response to the XFRC interrupt, reinitialize the channel for the next transfer.
7. Handling non-ACK responses

Figure 996. Isochronous OUT transactions



1. The grayed elements are not relevant in the context of this figure.

- **Interrupt service routine for isochronous OUT/IN transactions**

  Code sample: isochronous OUT

```
Unmask (FRMOR/XFRC)
if (XFRC)
```

```
   {
   De-allocate Channel
   }
else
   if (FRMOR)
      {
      Unmask CHH
      Disable Channel
      }
   else
   if (CHH)
      {
      Mask CHH
      De-allocate Channel
      }
Code sample: Isochronous IN
Unmask (TXERR/XFRC/FRMOR/BBERR)
if (XFRC or FRMOR)
   {
   if (XFRC and (OTG_HCTSIZx.PKTCNT == 0))
      {
      Reset Error Count
      De-allocate Channel
      }
   else
      {
      Unmask CHH
      Disable Channel
      }
   }
   else
      if (TXERR or BBERR)
         {
         Increment Error Count
         Unmask CHH
         Disable Channel
         }
      else
         if (CHH)
            {
            Mask CHH
            if (Transfer Done or (Error_count == 3))
               {
               De-allocate Channel
               }
```

```
else
   {
   Re-initialize Channel
   }
}
```

- **Isochronous IN transactions**

  The assumptions are:

  – The application is attempting to receive one packet (up to 1 maximum packet size) in every frame starting with the next odd frame (transfer size = 1 024 bytes).

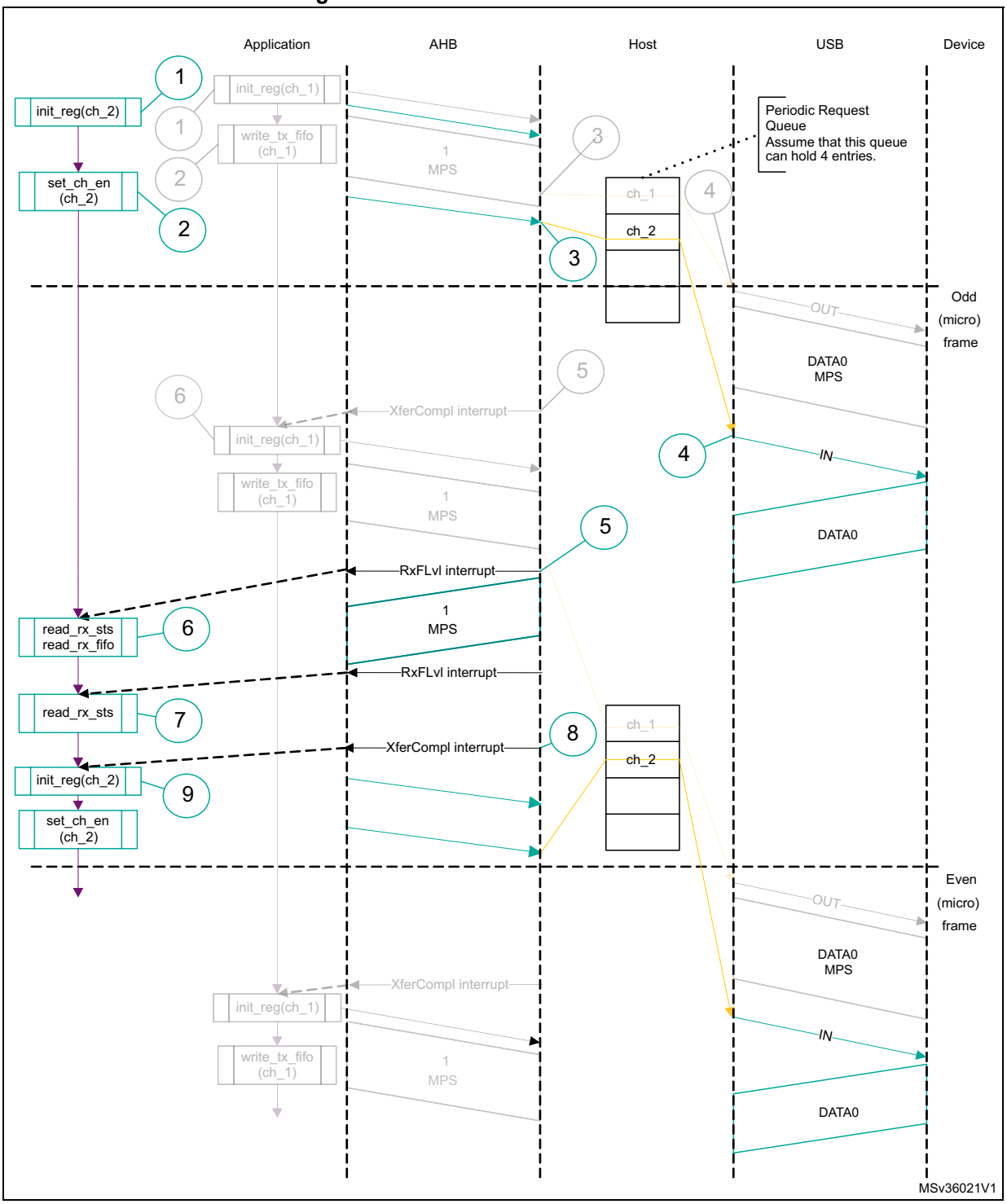
  – The receive FIFO can hold at least one maximum-packet-size packet and two status word per packet (1 031 bytes).

  – Periodic request queue depth = 4.

  The sequence of operations is as follows:

1. Initialize channel 2. The application must set the ODDFRM bit in OTG_HCCHAR2.
2. Set the CHENA bit in OTG_HCCHAR2 to write an IN request to the periodic request queue.
3. The OTG host writes an IN request to the periodic request queue for each OTG_HCCHAR2 register write with the CHENA bit set.
4. The OTG host attempts to send an IN token in the next odd frame.
5. As soon as the IN packet is received and written to the receive FIFO, the OTG host generates an RXFLVL interrupt.
6. In response to the RXFLVL interrupt, read the received packet status to determine the number of bytes received, then read the receive FIFO accordingly. The application must mask the RXFLVL interrupt before reading the receive FIFO, and unmask it after reading the entire packet.
7. The core generates an RXFLVL interrupt for the transfer completion status entry in the receive FIFO. This time, the application must read and ignore the receive packet status when the receive packet status is not an IN data packet (PKTSTS bit in OTG_GRXSTSR ≠ 0b0010).
8. The core generates an XFRC interrupt as soon as the receive packet status is read.
9. In response to the XFRC interrupt, read the PKTCNT field in OTG_HCTSIZ2. If PKTCNT ≠ 0 in OTG_HCTSIZ2, disable the channel before re-initializing the channel for the next transfer, if any. If PKTCNT = 0 in OTG_HCTSIZ2, reinitialize the channel for the next transfer. This time, the application must reset the ODDFRM bit in OTG_HCCHAR2.

Figure 997. Isochronous IN transactions



1. The grayed elements are not relevant in the context of this figure.

- **Selecting the queue depth**

  Choose the periodic and non-periodic request queue depths carefully to match the number of periodic/non-periodic endpoints accessed.

  The non-periodic request queue depth affects the performance of non-periodic

transfers. The deeper the queue (along with sufficient FIFO size), the more often the core is able to pipeline non-periodic transfers. If the queue size is small, the core is able to put in new requests only when the queue space is freed up.

The core's periodic request queue depth is critical to perform periodic transfers as scheduled. Select the periodic queue depth, based on the number of periodic transfers scheduled in a microframe. If the periodic request queue depth is smaller than the periodic transfers scheduled in a microframe, a frame overrun condition occurs.

- **Handling babble conditions**

  OTG controller handles two cases of babble: packet babble and port babble. Packet babble occurs if the device sends more data than the maximum packet size for the channel. Port babble occurs if the core continues to receive data from the device at EOF2 (the end of frame 2, which is very close to SOF).

  When OTG controller detects a packet babble, it stops writing data into the Rx buffer and waits for the end of packet (EOP). When it detects an EOP, it flushes already written data in the Rx buffer and generates a Babble interrupt to the application.

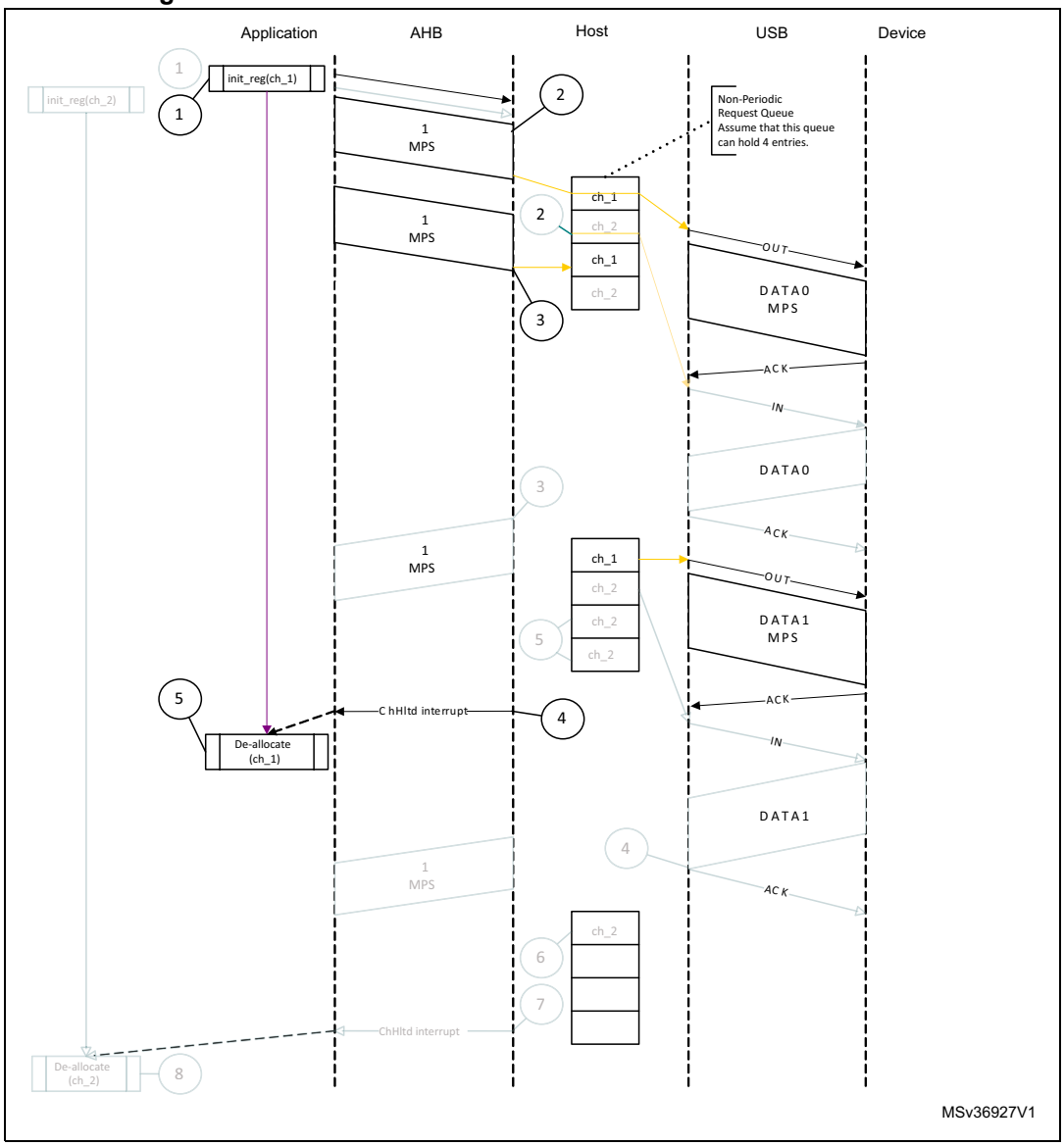
  When OTG controller detects a port babble, it flushes the Rx FIFO and disables the port. The core then generates a port disabled interrupt (HPRTINT in OTG_GINTSTS, PENCHNG in OTG_HPRT). On receiving this interrupt, the application must determine that this is not due to an overcurrent condition (another cause of the port disabled interrupt) by checking POCA in OTG_HPRT, then perform a soft reset. The core does not send any more tokens after it has detected a port babble condition.

- **Bulk and control OUT/SETUP transactions in DMA mode**

  The sequence of operations is as follows:

1. Initialize and enable channel 1 as explained in *Section : Channel initialization*.
2. The OTG host starts fetching the first packet as soon as the channel is enabled. For internal DMA mode, the OTG host uses the programmed DMA address to fetch the packet.
3. After fetching the last 32-bit word of the second (last) packet, the OTG host masks channel 1 internally for further arbitration.
4. The OTG host generates a CHH interrupt as soon as the last packet is sent.
5. In response to the CHH interrupt, de-allocate the channel for other transfers.

Figure 998. Normal bulk/control OUT/SETUP transactions - DMA



- **NAK and NYET handling with internal DMA:**

1. The OTG host sends a bulk OUT transaction.
2. The device responds with NAK or NYET.
3. If the application has unmasked NAK or NYET, the core generates the corresponding interrupt(s) to the application. The application is not required to service these interrupts, since the core takes care of rewinding the buffer pointers and re-initializing the Channel without application intervention.
4. The core automatically issues a ping token.
5. When the device returns an ACK, the core continues with the transfer. Optionally, the application can utilize these interrupts, in which case the NAK or NYET interrupt is masked by the application.

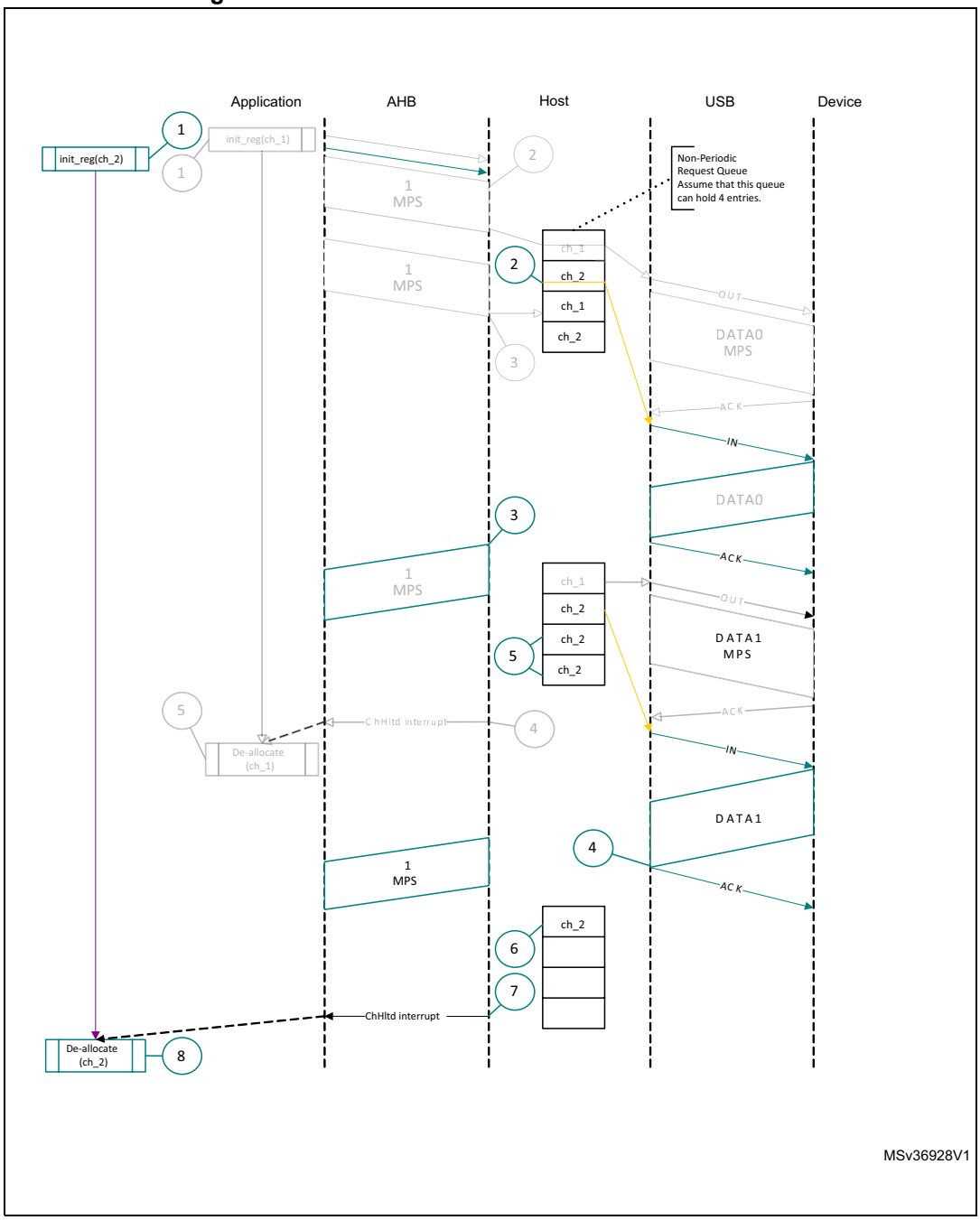
The core does not generate a separate interrupt when NAK or NYET is received by the host functionality.

- **Bulk and control IN transactions in DMA mode**

  The sequence of operations is as follows:

1. Initialize and enable the used channel (channel x) as explained in *Section : Channel initialization*.
2. The OTG host writes an IN request to the request queue as soon as the channel receives the grant from the arbiter (arbitration is performed in a round-robin fashion).
3. The OTG host starts writing the received data to the system memory as soon as the last byte is received with no errors.
4. When the last packet is received, the OTG host sets an internal flag to remove any extra IN requests from the request queue.
5. The OTG host flushes the extra requests.
6. The final request to disable channel x is written to the request queue. At this point, channel 2 is internally masked for further arbitration.
7. The OTG host generates the CHH interrupt as soon as the disable request comes to the top of the queue.
8. In response to the CHH interrupt, de-allocate the channel for other transfers.

Figure 999. Normal bulk/control IN transaction - DMA

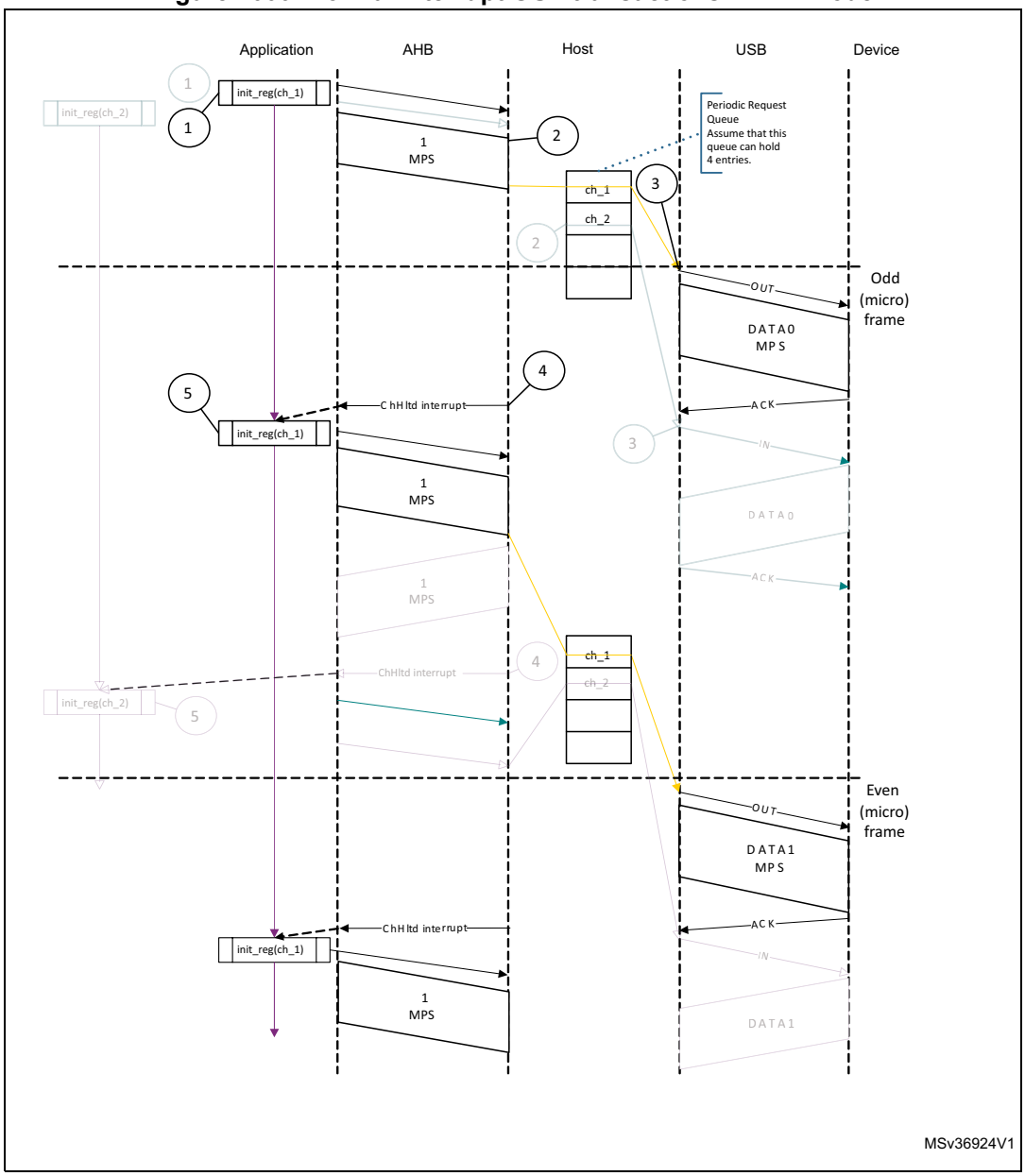


- **Interrupt OUT transactions in DMA mode**

1. Initialize and enable channel x as explained in *Section : Channel initialization*.
2. The OTG host starts fetching the first packet as soon the channel is enabled and writes the OUT request along with the last 32-bit word fetch. In high-bandwidth transfers, the OTG host continues fetching the next packet (up to the value specified in the MC field) before switching to the next channel.
3. The OTG host attempts to send the OUT token at the beginning of the next odd frame/micro-frame.

4. After successfully transmitting the packet, the OTG host generates a CHH interrupt.
5. In response to the CHH interrupt, reinitialize the channel for the next transfer.

Figure 1000. Normal interrupt OUT transactions - DMA mode



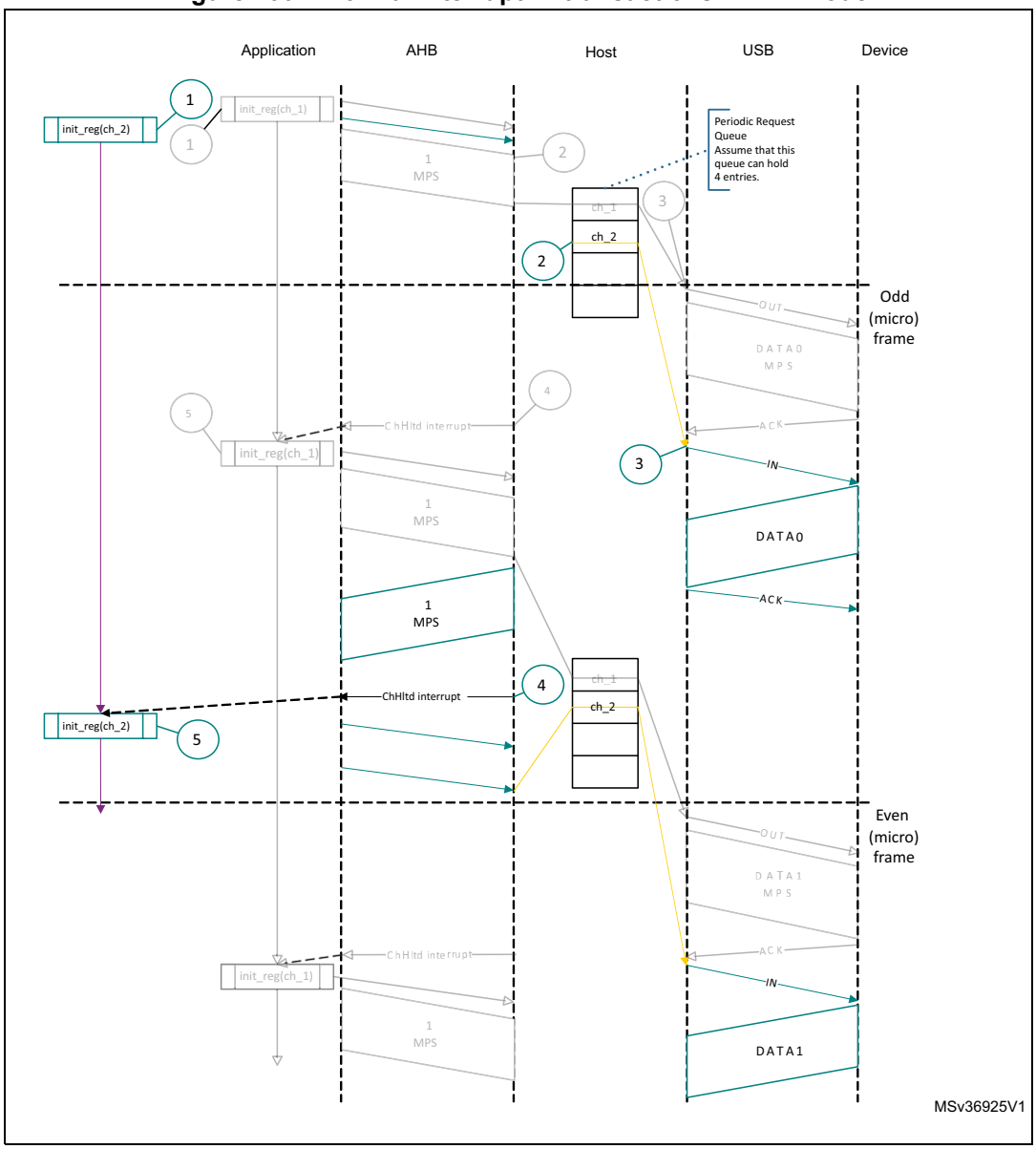
- **Interrupt IN transactions in DMA mode**

  The sequence of operations (channelx) is as follows:

1. Initialize and enable channel x as explained in *Section : Channel initialization*.
2. The OTG host writes an IN request to the request queue as soon as the channel x gets the grant from the arbiter (round-robin with fairness). In high-bandwidth transfers, the OTG host writes consecutive writes up to MC times.
3. The OTG host attempts to send an IN token at the beginning of the next (odd) frame/micro-frame.

4. As soon the packet is received and written to the receive FIFO, the OTG host generates a CHH interrupt.
5. In response to the CHH interrupt, reinitialize the channel for the next transfer.

Figure 1001. Normal interrupt IN transactions - DMA mode



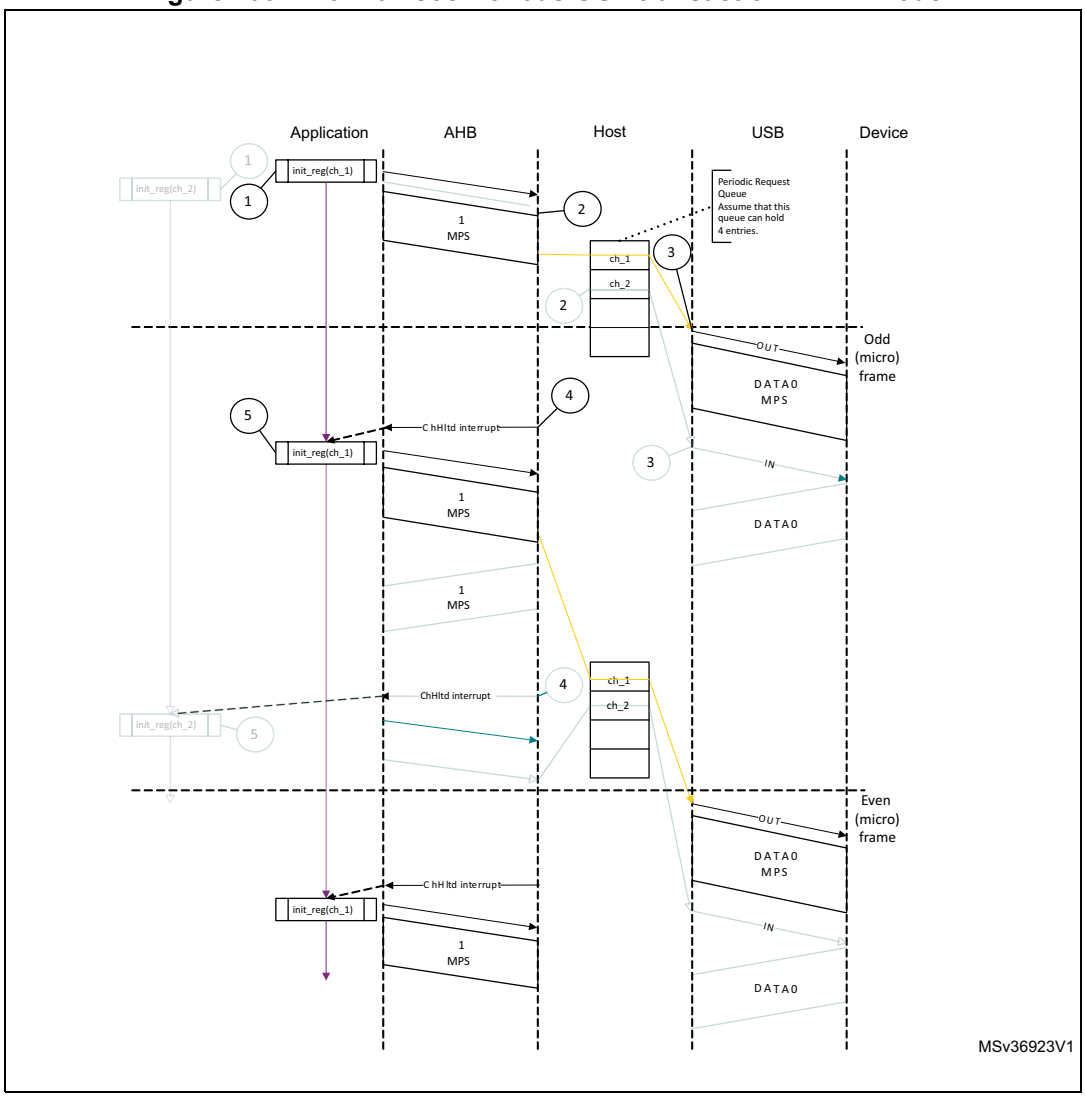
- **Isochronous OUT transactions in DMA mode**

1. Initialize and enable channel x as explained in *Section : Channel initialization*.
2. The OTG host starts fetching the first packet as soon as the channel is enabled, and writes the OUT request along with the last 32-bit word fetch. In high-bandwidth

transfers, the OTG host continues fetching the next packet (up to the value specified in the MC field) before switching to the next channel.

3. The OTG host attempts to send an OUT token at the beginning of the next (odd) frame/micro-frame.
4. After successfully transmitting the packet, the OTG host generates a CHH interrupt.
5. In response to the CHH interrupt, reinitialize the channel for the next transfer.

Figure 1002. Normal isochronous OUT transaction - DMA mode



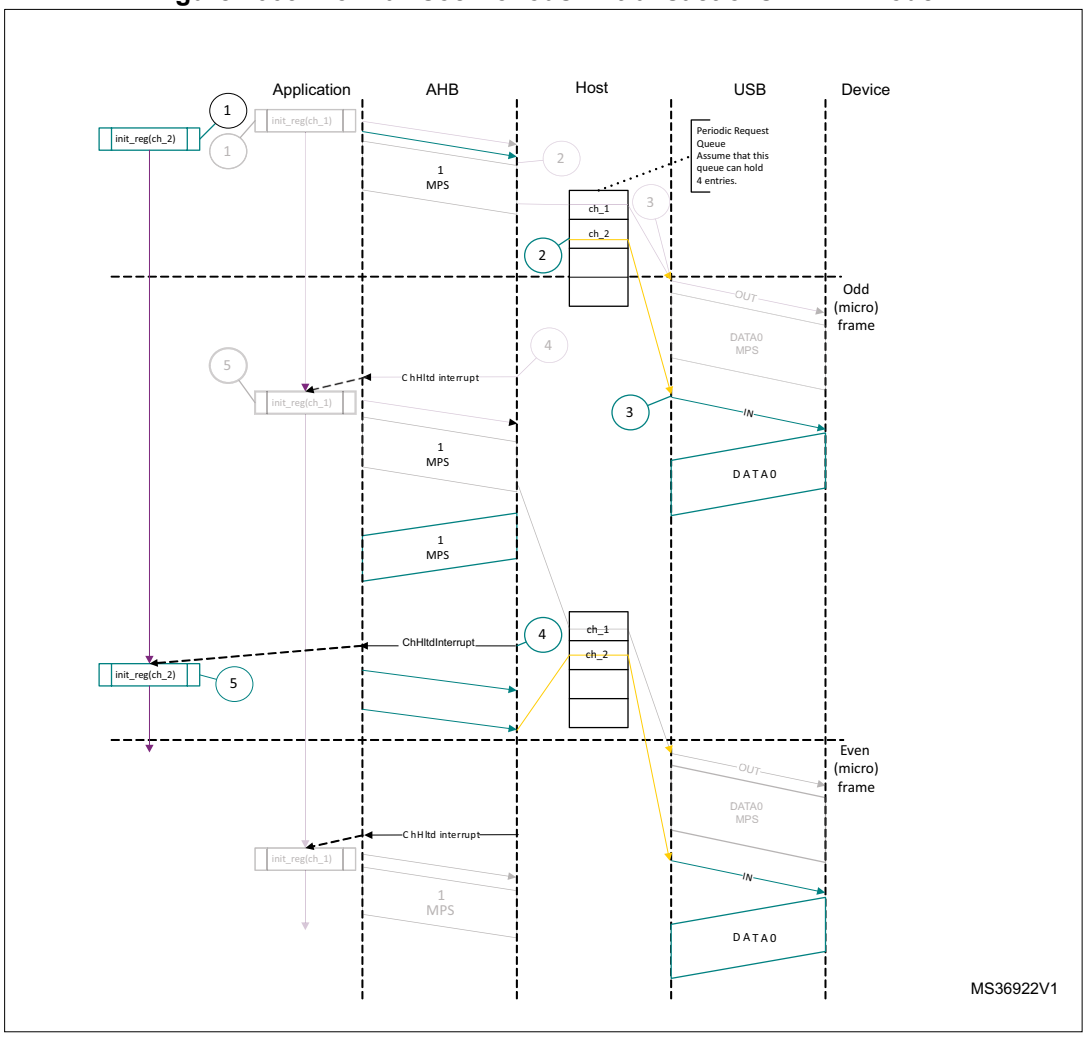
- **Isochronous IN transactions in DMA mode**

The sequence of operations ((channel x) is as follows:

1. Initialize and enable channel x as explained in *Section : Channel initialization*.
2. The OTG host writes an IN request to the request queue as soon as the channel x gets the grant from the arbiter (round-robin with fairness). In high-bandwidth transfers, the OTG host performs consecutive write operations up to MC times.
3. The OTG host attempts to send an IN token at the beginning of the next (odd) frame/micro-frame.

4. As soon the packet is received and written to the receive FIFO, the OTG host generates a CHH interrupt.
5. In response to the CHH interrupt, reinitialize the channel for the next transfer.

Figure 1003. Normal isochronous IN transactions - DMA mode



- **Bulk and control OUT/SETUP split transactions in DMA mode**

  The sequence of operations in (channel x) is as follows:

1. Initialize and enable channel x for start split as explained in *Section : Channel initialization*.
2. The OTG host starts fetching the first packet as soon the channel is enabled and writes the OUT request along with the last 32-bit word fetch.
3. After successfully transmitting start split, the OTG host generates the CHH interrupt.
4. In response to the CHH interrupt, set the COMPLSPLT bit in OTG_HCSPLT1 to send the complete split.
5. After successfully transmitting complete split, the OTG host generates the CHH interrupt.

6. In response to the CHH interrupt, de-allocate the channel.

- **Bulk/control IN split transactions in DMA mode**

The sequence of operations (channel x) is as follows:

1. Initialize and enable channel x as explained in *Section : Channel initialization*.
2. The OTG host writes the start split request to the nonperiodic request after getting the grant from the arbiter. The OTG host masks the channel x internally for the arbitration after writing the request.
3. As soon as the IN token is transmitted, the OTG host generates the CHH interrupt.
4. In response to the CHH interrupt, set the COMPLSPLT bit in OTG_HCSPLT2 and re-enable the channel to send the complete split token. This unmasks channel x for arbitration.
5. The OTG host writes the complete split request to the nonperiodic request after receiving the grant from the arbiter.
6. The OTG host starts writing the packet to the system memory after receiving the packet successfully.
7. As soon as the received packet is written to the system memory, the OTG host generates a CHH interrupt.
8. In response to the CHH interrupt, de-allocate the channel.

- **Interrupt OUT split transactions in DMA mode**

The sequence of operations in (channel x) is as follows:

1. Initialize and enable channel 1 for start split as explained in *Section : Channel initialization*. The application must set the ODDFRM bit in OTG_HCCHAR1.
2. The OTG host starts reading the packet.
3. The OTG host attempts to send the start split transaction.
4. After successfully transmitting the start split, the OTG host generates the CHH interrupt.
5. In response to the CHH interrupt, set the COMPLSPLT bit in OTG_HCSPLT1 to send the complete split.
6. After successfully completing the complete split transaction, the OTG host generates the CHH interrupt.
7. In response to CHH interrupt, de-allocate the channel.

- **Interrupt IN split transactions in DMA mode**

The sequence of operations in (channel x) is as follows:

1. Initialize and enable channel x for start split as explained in *Section : Channel initialization*.
2. The OTG host writes an IN request to the request queue as soon as channel x receives the grant from the arbiter.
3. The OTG host attempts to send the start split IN token at the beginning of the next odd micro-frame.
4. The OTG host generates the CHH interrupt after successfully transmitting the start split IN token.
5. In response to the CHH interrupt, set the COMPLSPLT bit in OTG_HCSPLT2 to send the complete split.

6. As soon as the packet is received successfully, the OTG host starts writing the data to the system memory.
7. The OTG host generates the CHH interrupt after transferring the received data to the system memory.
8. In response to the CHH interrupt, de-allocate or reinitialize the channel for the next start split.

- **Isochronous OUT split transactions in DMA mode**

The sequence of operations (channel x) is as follows:

1. Initialize and enable channel x for start split (begin) as explained in *Section : Channel initialization*. The application must set the ODDFRM bit in OTG_HCCHAR1. Program the MPS field.
2. The OTG host starts reading the packet.
3. After successfully transmitting the start split (begin), the OTG host generates the CHH interrupt.
4. In response to the CHH interrupt, reinitialize the registers to send the start split (end).
5. After successfully transmitting the start split (end), the OTG host generates a CHH interrupt.
6. In response to the CHH interrupt, de-allocate the channel.

- **Isochronous IN split transactions in DMA mode**

The sequence of operations (channel x) is as follows:

1. Initialize and enable channel x for start split as explained in *Section : Channel initialization*.
2. The OTG host writes an IN request to the request queue as soon as channel x receives the grant from the arbiter.
3. The OTG host attempts to send the start split IN token at the beginning of the next odd micro-frame.
4. The OTG host generates the CHH interrupt after successfully transmitting the start split IN token.
5. In response to the CHH interrupt, set the COMPLSPLT bit in OTG_HCSPLT2 to send the complete split.
6. As soon as the packet is received successfully, the OTG host starts writing the data to the system memory.

   The OTG host generates the CHH interrupt after transferring the received data to the system memory. In response to the CHH interrupt, de-allocate the channel or reinitialize the channel for the next start split.

73.15.6 Device programming model

Endpoint initialization on USB reset

1. Set the NAK bit for all OUT endpoints
   – SNAK = 1 in OTG_DOEPCTLx (for all OUT endpoints)
2. Unmask the following interrupt bits
   – INEP0 = 1 in OTG_DAINTMSK (control 0 IN endpoint)
   – OUTEP0 = 1 in OTG_DAINTMSK (control 0 OUT endpoint)
   – STUPM = 1 in OTG_DOEPMSK
   – XFRCM = 1 in OTG_DOEPMSK
   – XFRCM = 1 in OTG_DIEPMSK
   – TOM = 1 in OTG_DIEPMSK
3. Set up the data FIFO RAM for each of the FIFOs
   – Program the OTG_GRXFSIZ register, to be able to receive control OUT data and setup data. If thresholding is not enabled, at a minimum, this must be equal to 1 max packet size of control endpoint 0 + 2 words (for the status of the control OUT data packet) + 10 words (for setup packets).
   – Program the OTG_DIEPTXF0 register (depending on the FIFO number chosen) to be able to transmit control IN data. At a minimum, this must be equal to 1 max packet size of control endpoint 0.
4. Program the following fields in the endpoint-specific registers for control OUT endpoint 0 to receive a SETUP packet
   – STUPCNT = 3 in OTG_DOEPTSIZ0 (to receive up to 3 back-to-back SETUP packets)
5. For USB OTG in DMA mode, the OTG_DOEPDMA0 register must have a valid memory address to store any SETUP packets received.

At this point, all initialization required to receive SETUP packets is done.

Endpoint initialization on enumeration completion

1. On the Enumeration Done interrupt (ENUMDNE in OTG_GINTSTS), read the OTG_DSTS register to determine the enumeration speed.
2. Program the MPSIZ field in OTG_DIEPCTL0 to set the maximum packet size. This step configures control endpoint 0. The maximum packet size for a control endpoint depends on the enumeration speed.
3. For USB OTG in DMA mode, program the OTG_DOEPCTL0 register to enable control OUT endpoint 0, to receive a SETUP packet.

At this point, the device is ready to receive SOF packets and is configured to perform control transfers on control endpoint 0.

Endpoint initialization on SetAddress command

This section describes what the application must do when it receives a SetAddress command in a SETUP packet.

1. Program the OTG_DCFG register with the device address received in the SetAddress command
2. Program the core to send out a status IN packet

Endpoint initialization on SetConfiguration/SetInterface command

This section describes what the application must do when it receives a SetConfiguration or SetInterface command in a SETUP packet.

1. When a SetConfiguration command is received, the application must program the endpoint registers to configure them with the characteristics of the valid endpoints in the new configuration.
2. When a SetInterface command is received, the application must program the endpoint registers of the endpoints affected by this command.
3. Some endpoints that were active in the prior configuration or alternate setting are not valid in the new configuration or alternate setting. These invalid endpoints must be deactivated.
4. Unmask the interrupt for each active endpoint and mask the interrupts for all inactive endpoints in the OTG_DAINTMSK register.
5. Set up the data FIFO RAM for each FIFO.
6. After all required endpoints are configured; the application must program the core to send a status IN packet.

At this point, the device core is configured to receive and transmit any type of data packet.

Endpoint activation

This section describes the steps required to activate a device endpoint or to configure an existing device endpoint to a new type.

1. Program the characteristics of the required endpoint into the following fields of the OTG_DIEPCTLx register (for IN or bidirectional endpoints) or the OTG_DOEPCTLx register (for OUT or bidirectional endpoints).
   - Maximum packet size
   - USB active endpoint = 1
   - Endpoint start data toggle (for interrupt and bulk endpoints)
   - Endpoint type
   - Tx FIFO number
2. Once the endpoint is activated, the core starts decoding the tokens addressed to that endpoint and sends out a valid handshake for each valid token received for the endpoint.

Endpoint deactivation

This section describes the steps required to deactivate an existing endpoint.

1. In the endpoint to be deactivated, clear the USB active endpoint bit in the OTG_DIEPCTLx register (for IN or bidirectional endpoints) or the OTG_DOEPCTLx register (for OUT or bidirectional endpoints).
2. Once the endpoint is deactivated, the core ignores tokens addressed to that endpoint, which results in a timeout on the USB.

Note: The application must meet the following conditions to set up the device core to handle traffic:
NPTXFEM and RXFLVLM in the OTG_GINTMSK register must be cleared.

Operational model

SETUP and OUT data transfers:

This section describes the internal data flow and application-level operations during data OUT transfers and SETUP transactions.

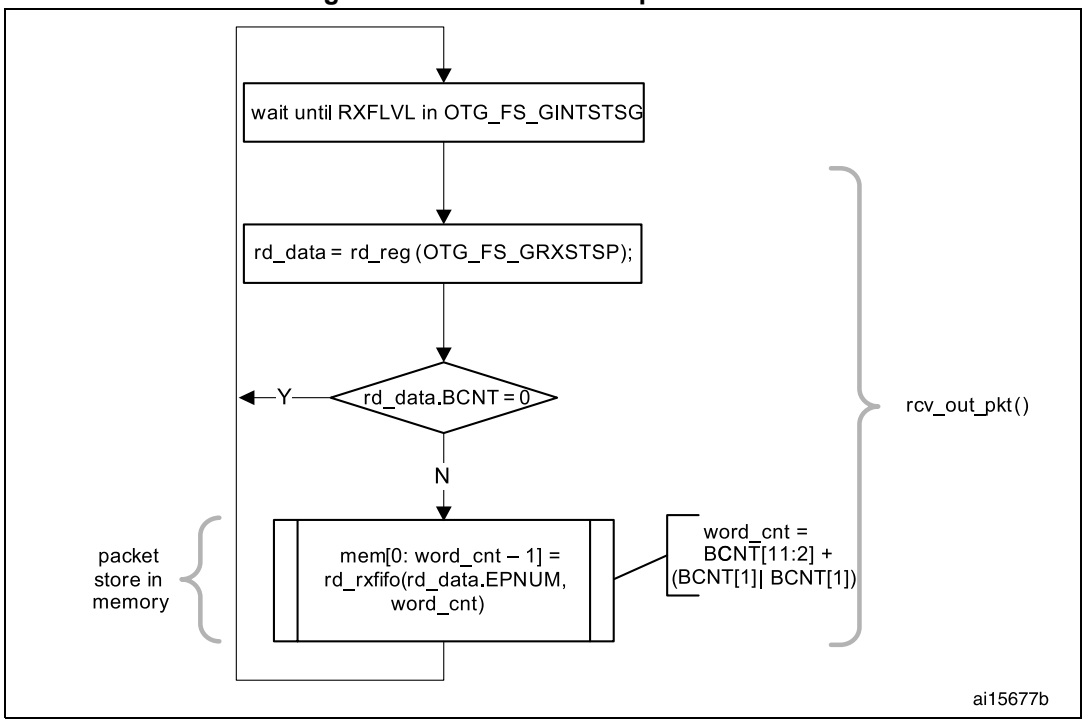
- **Packet read**

This section describes how to read packets (OUT data and SETUP packets) from the receive FIFO.

1. On catching an RXFLVL interrupt (OTG_GINTSTS register), the application must read the receive status pop register (OTG_GRXSTSP).
2. The application can mask the RXFLVL interrupt (in OTG_GINTSTS) by writing to RXFLVLM = 0 (in OTG_GINTMSK), until it has read the packet from the receive FIFO.
3. If the received packet's byte count is not 0, the byte count amount of data is popped from the receive data FIFO and stored in memory. If the received packet byte count is 0, no data is popped from the receive data FIFO.
4. The receive status readout of the packet of FIFO indicates one of the following:
   a) Global OUT NAK pattern:
   PKTSTS = Global OUT NAK, BCNT = 0x000, EPNUM = (0x0),
   DPID = (0b00).
   These data indicate that the global OUT NAK bit has taken effect.
   b) SETUP packet pattern:
   PKTSTS = SETUP, BCNT = 0x008, EPNUM = Control EP Num,
   DPID = DATA0. These data indicate that a SETUP packet for the specified endpoint is now available for reading from the receive FIFO.
   c) Setup stage done pattern:
   PKTSTS = Setup Stage Done, BCNT = 0x0, EPNUM = Control EP Num,
   DPID = (0b00).
   These data indicate that the setup stage for the specified endpoint has completed and the data stage has started. After this entry is popped from the receive FIFO, the core asserts a setup interrupt on the specified control OUT endpoint.
   d) Data OUT packet pattern:
   PKTSTS = DataOUT, BCNT = size of the received data OUT packet (0 ≤ BCNT ≤ 1 024), EPNUM = EPNUM on which the packet was received, DPID = Actual Data PID.
   e) Data transfer completed pattern:
   PKTSTS = Data OUT transfer done, BCNT = 0x0, EPNUM = OUT EP Num on which the data transfer is complete, DPID = (0b00).
   These data indicate that an OUT data transfer for the specified OUT endpoint has completed. After this entry is popped from the receive FIFO, the core asserts a transfer completed interrupt on the specified OUT endpoint.
5. After the data payload is popped from the receive FIFO, the RXFLVL interrupt (OTG_GINTSTS) must be unmasked.
6. Steps 1–5 are repeated every time the application detects assertion of the interrupt line due to RXFLVL in OTG_GINTSTS. Reading an empty receive FIFO can result in undefined core behavior.

Figure 1004 provides a flowchart of the above procedure.

Figure 1004. Receive FIFO packet read



SETUP transactions

This section describes how the core handles SETUP packets and the application's sequence for handling SETUP transactions.

- **Application requirements**

1. To receive a SETUP packet, the STUPCNT field (OTG_DOEPTSIZx) in a control OUT endpoint must be programmed to a non-zero value. When the application programs the STUPCNT field to a non-zero value, the core receives SETUP packets and writes them to the receive FIFO, irrespective of the NAK status and EPENA bit setting in OTG_DOEPCTLx. The STUPCNT field is decremented every time the control endpoint receives a SETUP packet. If the STUPCNT field is not programmed to a proper value before receiving a SETUP packet, the core still receives the SETUP packet and decrements the STUPCNT field, but the application may not be able to determine the correct number of SETUP packets received in the setup stage of a control transfer.
   – STUPCNT = 3 in OTG_DOEPTSIZx
2. The application must always allocate some extra space in the receive data FIFO, to be able to receive up to three SETUP packets on a control endpoint.
   – The space to be reserved is 10 words. Three words are required for the first SETUP packet, 1 word is required for the setup stage done word and 6 words are required to store two extra SETUP packets among all control endpoints.
   – 3 words per SETUP packet are required to store 8 bytes of SETUP data and 4 bytes of SETUP status (setup packet pattern). The core reserves this space in the

receive data FIFO to write SETUP data only, and never uses this space for data packets.

3. The application must read the 2 words of the SETUP packet from the receive FIFO.
4. The application must read and discard the setup stage done word from the receive FIFO.

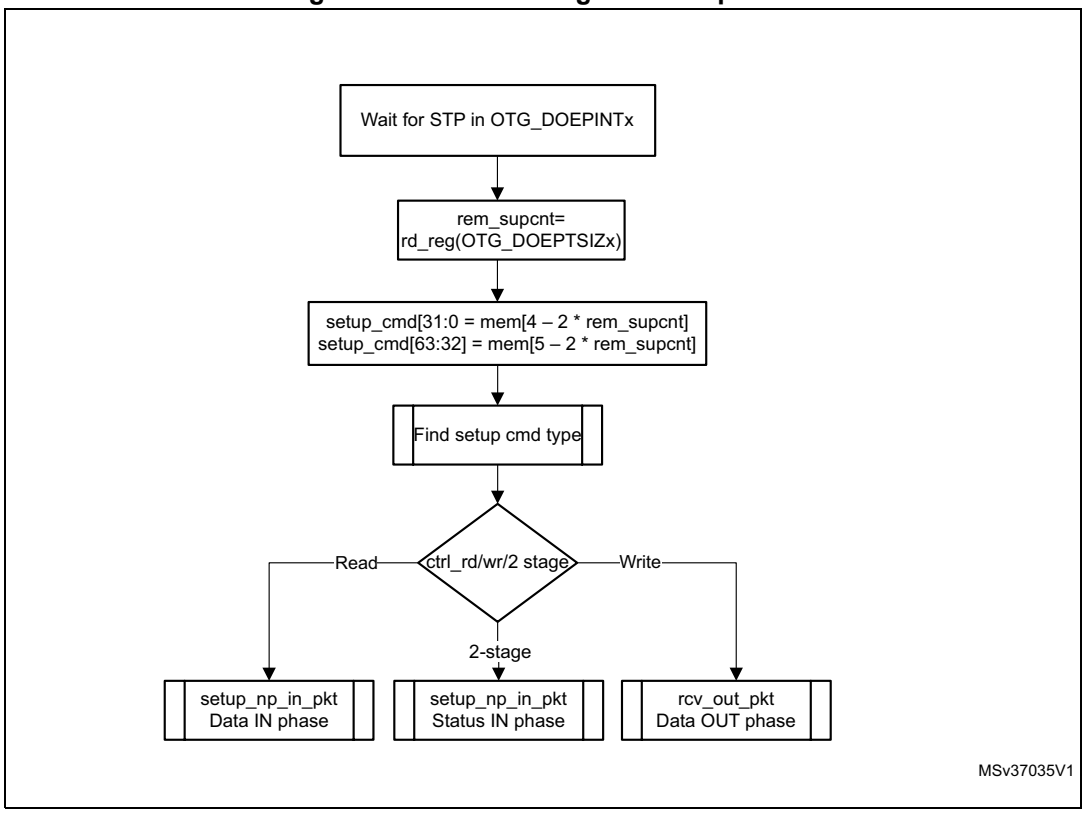
- **Internal data flow**

1. When a SETUP packet is received, the core writes the received data to the receive FIFO, without checking for available space in the receive FIFO and irrespective of the endpoint's NAK and STALL bit settings.
   – The core internally sets the IN NAK and OUT NAK bits for the control IN/OUT endpoints on which the SETUP packet was received.
2. For every SETUP packet received on the USB, 3 words of data are written to the receive FIFO, and the STUPCNT field is decremented by 1.
   – The first word contains control information used internally by the core
   – The second word contains the first 4 bytes of the SETUP command
   – The third word contains the last 4 bytes of the SETUP command
3. When the setup stage changes to a data IN/OUT stage, the core writes an entry (setup stage done word) to the receive FIFO, indicating the completion of the setup stage.
4. On the AHB side, SETUP packets are emptied by the application.
5. When the application pops the setup stage done word from the receive FIFO, the core interrupts the application with an STUP interrupt (OTG_DOEPINTx), indicating it can process the received SETUP packet.
6. The core clears the endpoint enable bit for control OUT endpoints.

- **Application programming sequence**

1. Program the OTG_DOEPTSIZx register.
   – STUPCNT = 3
2. Wait for the RXFLVL interrupt (OTG_GINTSTS) and empty the data packets from the receive FIFO.
3. Assertion of the STUP interrupt (OTG_DOEPINTx) marks a successful completion of the SETUP data transfer.
   – On this interrupt, the application must read the OTG_DOEPTSIZx register to determine the number of SETUP packets received and process the last received SETUP packet.

Figure 1005. Processing a SETUP packet



- **Handling more than three back-to-back SETUP packets**

Per the USB 2.0 specification, normally, during a SETUP packet error, a host does not send more than three back-to-back SETUP packets to the same endpoint. However, the USB 2.0 specification does not limit the number of back-to-back SETUP packets a host can send to the same endpoint. When this condition occurs, the OTG controller generates an interrupt (B2BSTUP in OTG_DOEPINTx).

- **Setting the global OUT NAK**

Internal data flow:

1. When the application sets the Global OUT NAK (SGONAK bit in OTG_DCTL), the core stops writing data, except SETUP packets, to the receive FIFO. Irrespective of the space availability in the receive FIFO, non-isochronous OUT tokens receive a NAK handshake response, and the core ignores isochronous OUT data packets
2. The core writes the Global OUT NAK pattern to the receive FIFO. The application must reserve enough receive FIFO space to write this data pattern.
3. When the application pops the Global OUT NAK pattern word from the receive FIFO, the core sets the GONAKEFF interrupt (OTG_GINTSTS).
4. Once the application detects this interrupt, it can assume that the core is in Global OUT NAK mode. The application can clear this interrupt by clearing the SGONAK bit in OTG_DCTL.

Application programming sequence:

1. To stop receiving any kind of data in the receive FIFO, the application must set the Global OUT NAK bit by programming the following field:
   – SGONAK = 1 in OTG_DCTL
2. Wait for the assertion of the GONAKEFF interrupt in OTG_GINTSTS. When asserted, this interrupt indicates that the core has stopped receiving any type of data except SETUP packets.
3. The application can receive valid OUT packets after it has set SGONAK in OTG_DCTL and before the core asserts the GONAKEFF interrupt (OTG_GINTSTS).
4. The application can temporarily mask this interrupt by writing to the GONAKEFFM bit in the OTG_GINTMSK register.
   – GONAKEFFM = 0 in the OTG_GINTMSK register
5. Whenever the application is ready to exit the Global OUT NAK mode, it must clear the SGONAK bit in OTG_DCTL. This also clears the GONAKEFF interrupt (OTG_GINTSTS).
   – CGONAK = 1 in OTG_DCTL
6. If the application has masked this interrupt earlier, it must be unmasked as follows:
   – GONAKEFFM = 1 in OTG_GINTMSK

- **Disabling an OUT endpoint**

The application must use this sequence to disable an OUT endpoint that it has enabled.

Application programming sequence:

1. Before disabling any OUT endpoint, the application must enable Global OUT NAK mode in the core.
   – SGONAK = 1 in OTG_DCTL
2. Wait for the GONAKEFF interrupt (OTG_GINTSTS)
3. Disable the required OUT endpoint by programming the following fields:
   – EPDIS = 1 in OTG_DOEPCTLx
   – SNAK = 1 in OTG_DOEPCTLx
4. Wait for the EPDISD interrupt (OTG_DOEPINTx), which indicates that the OUT endpoint is completely disabled. When the EPDISD interrupt is asserted, the core also clears the following bits:
   – EPDIS = 0 in OTG_DOEPCTLx
   – EPENA = 0 in OTG_DOEPCTLx
5. The application must clear the Global OUT NAK bit to start receiving data from other non-disabled OUT endpoints.
   – SGONAK = 0 in OTG_DCTL

- **Transfer Stop Programming for OUT endpoints**

The application must use the following programing sequence to stop any transfers (because of an interrupt from the host, typically a reset).

Sequence of operations:

1. Enable all OUT endpoints by setting
   – EPENA = 1 in all OTG_DOEPCTLx registers.
2. Flush the RxFIFO as follows
   – Poll OTG_GRSTCTL.AHBIDL until it is 1. This indicates that AHB master is idle.
   – Perform read modify write operation on OTG_GRSTCTL.RXFFLSH =1
   – Poll OTG_GRSTCTL.RXFFLSH until it is 0, but also using a timeout of less than 10 milli-seconds (corresponds to minimum reset signaling duration). If 0 is seen before the timeout, then the RxFIFO flush is successful. If at the moment the timeout occurs, there is still a 1, (this may be due to a packet on EP0 coming from the host) then go back (once only) to the previous step (“Perform read modify write operation”).
3. Before disabling any OUT endpoint, the application must enable Global OUT NAK mode in the core, according to the instructions in “*Setting the global OUT NAK*”. This ensures that data in the RxFIFO is sent to the application successfully. Set SGONAK = 1 in OTG_DCTL
4. Wait for the GONAKEFF interrupt (OTG_GINTSTS)
5. Disable all active OUT endpoints by programming the following register bits:
   – EPDIS = 1 in registers OTG_DOEPCTLx
   – SNAK = 1 in registers OTG_DOEPCTLx
6. Wait for the EPDIS interrupt in OTG_DOEPINTx for each OUT endpoint programmed in the previous step. The EPDIS interrupt in OTG_DOEPINTx indicates that the corresponding OUT endpoint is completely disabled. When the EPDIS interrupt is asserted, the following bits are cleared:
   – EPENA = 0 in registers OTG_DOEPCTLx
   – EPDIS = 0 in registers OTG_DOEPCTLx
   – SNAK = 0 in registers OTG_DOEPCTLx

- **Generic non-isochronous OUT data transfers**

This section describes a regular non-isochronous OUT data transfer (control, bulk, or interrupt).

Application requirements:

1. Before setting up an OUT transfer, the application must allocate a buffer in the memory to accommodate all data to be received as part of the OUT transfer.
2. For OUT transfers, the transfer size field in the endpoint’s transfer size register must be a multiple of the maximum packet size of the endpoint, adjusted to the word boundary.
   – transfer size[EPNUM] = $n \times$ (MPSIZ[EPNUM] + 4 – (MPSIZ[EPNUM] mod 4))
   – packet count[EPNUM] = n
   – $n > 0$
3. On any OUT endpoint interrupt, the application must read the endpoint’s transfer size register to calculate the size of the payload in the memory. The received payload size can be less than the programmed transfer size.
   – Payload size in memory = application programmed initial transfer size – core updated final transfer size
   – Number of USB packets in which this payload was received = application programmed initial packet count – core updated final packet count

Internal data flow:

1. The application must set the transfer size and packet count fields in the endpoint-specific registers, clear the NAK bit, and enable the endpoint to receive the data.
2. Once the NAK bit is cleared, the core starts receiving data and writes it to the receive FIFO, as long as there is space in the receive FIFO. For every data packet received on the USB, the data packet and its status are written to the receive FIFO. Every packet (maximum packet size or short packet) written to the receive FIFO decrements the packet count field for that endpoint by 1.
   - OUT data packets received with bad data CRC are flushed from the receive FIFO automatically.
   - After sending an ACK for the packet on the USB, the core discards non-isochronous OUT data packets that the host, which cannot detect the ACK, re-sends. The application does not detect multiple back-to-back data OUT packets on the same endpoint with the same data PID. In this case the packet count is not decremented.
   - If there is no space in the receive FIFO, isochronous or non-isochronous data packets are ignored and not written to the receive FIFO. Additionally, non-isochronous OUT tokens receive a NAK handshake reply.
   - In all the above three cases, the packet count is not decremented because no data are written to the receive FIFO.
3. When the packet count becomes 0 or when a short packet is received on the endpoint, the NAK bit for that endpoint is set. Once the NAK bit is set, the isochronous or non-isochronous data packets are ignored and not written to the receive FIFO, and non-isochronous OUT tokens receive a NAK handshake reply.
4. After the data are written to the receive FIFO, the application reads the data from the receive FIFO and writes it to external memory, one packet at a time per endpoint.
5. At the end of every packet write on the AHB to external memory, the transfer size for the endpoint is decremented by the size of the written packet.
6. The OUT data transfer completed pattern for an OUT endpoint is written to the receive FIFO on one of the following conditions:
   - The transfer size is 0 and the packet count is 0
   - The last OUT data packet written to the receive FIFO is a short packet (0 ≤ packet size < maximum packet size)
7. When either the application pops this entry (OUT data transfer completed), a transfer completed interrupt is generated for the endpoint and the endpoint enable is cleared.

Application programming sequence:

1. Program the OTG_DOEPTSIZx register for the transfer size and the corresponding packet count.
2. Program the OTG_DOEPCTLx register with the endpoint characteristics, and set the EPENA and CNAK bits.
   – EPENA = 1 in OTG_DOEPCTLx
   – CNAK = 1 in OTG_DOEPCTLx
3. Wait for the RXFLVL interrupt (in OTG_GINTSTS) and empty the data packets from the receive FIFO.
   – This step can be repeated many times, depending on the transfer size.
4. Asserting the XFRC interrupt (OTG_DOEPINTx) marks a successful completion of the non-isochronous OUT data transfer.
5. Read the OTG_DOEPTSIZx register to determine the size of the received data payload.

- **Generic isochronous OUT data transfer**

This section describes a regular isochronous OUT data transfer.

Application requirements:

1. All the application requirements for non-isochronous OUT data transfers also apply to isochronous OUT data transfers.
2. For isochronous OUT data transfers, the transfer size and packet count fields must always be set to the number of maximum-packet-size packets that can be received in a single frame and no more. Isochronous OUT data transfers cannot span more than 1 frame.
3. The application must read all isochronous OUT data packets from the receive FIFO (data and status) before the end of the periodic frame (EOPF interrupt in OTG_GINTSTS).
4. To receive data in the following frame, an isochronous OUT endpoint must be enabled after the EOPF (OTG_GINTSTS) and before the SOF (OTG_GINTSTS).

Internal data flow:

1. The internal data flow for isochronous OUT endpoints is the same as that for non-isochronous OUT endpoints, but for a few differences.
2. When an isochronous OUT endpoint is enabled by setting the endpoint enable and clearing the NAK bits, the Even/Odd frame bit must also be set appropriately. The core receives data on an isochronous OUT endpoint in a particular frame only if the following condition is met:
   – EONUM (in OTG_DOEPCTLx) = FNSOF[0] (in OTG_DSTS)
3. When the application completely reads an isochronous OUT data packet (data and status) from the receive FIFO, the core updates the RXDPID field in OTG_DOEPTSIZx with the data PID of the last isochronous OUT data packet read from the receive FIFO.

Application programming sequence:

1. Program the OTG_DOEPTSIZx register for the transfer size and the corresponding packet count
2. Program the OTG_DOEPCTLx register with the endpoint characteristics and set the endpoint enable, ClearNAK, and Even/Odd frame bits.
   – EPENA = 1
   – CNAK = 1
   – EONUM = (0: Even/1: Odd)
3. Wait for the RXFLVL interrupt (in OTG_GINTSTS) and empty the data packets from the receive FIFO
   – This step can be repeated many times, depending on the transfer size.
4. The assertion of the XFRC interrupt (in OTG_DOEPINTx) marks the completion of the isochronous OUT data transfer. This interrupt does not necessarily mean that the data in memory are good.
5. This interrupt cannot always be detected for isochronous OUT transfers. Instead, the application can detect the INCOMPISOOUT interrupt in OTG_GINTSTS.
6. Read the OTG_DOEPTSIZx register to determine the size of the received transfer and to determine the validity of the data received in the frame. The application must treat the data received in memory as valid only if one of the following conditions is met:
   – RXDPID = DATA0 (in OTG_DOEPTSIZx) and the number of USB packets in which this payload was received = 1
   – RXDPID = DATA1 (in OTG_DOEPTSIZx) and the number of USB packets in which this payload was received = 2

     RXDPID = D2 (in OTG_DOEPTSIZx) and the number of USB packets in which this payload was received = 3The number of USB packets in which this payload was received =
     Application programmed initial packet count – core updated final packet count

   The application can discard invalid data packets.

- **Incomplete isochronous OUT data transfers**

This section describes the application programming sequence when isochronous OUT data packets are dropped inside the core.

Internal data flow:

1. For isochronous OUT endpoints, the XFRC interrupt (in OTG_DOEPINTx) may not always be asserted. If the core drops isochronous OUT data packets, the application may fail to detect the XFRC interrupt (OTG_DOEPINTx) under the following circumstances:
   – When the receive FIFO cannot accommodate the complete ISO OUT data packet, the core drops the received ISO OUT data
   – When the isochronous OUT data packet is received with CRC errors
   – When the isochronous OUT token received by the core is corrupted
   – When the application is very slow in reading the data from the receive FIFO
2. When the core detects an end of periodic frame before transfer completion to all isochronous OUT endpoints, it asserts the incomplete isochronous OUT data interrupt (INCOMPISOOUT in OTG_GINTSTS), indicating that an XFRC interrupt (in OTG_DOEPINTx) is not asserted on at least one of the isochronous OUT endpoints. At

this point, the endpoint with the incomplete transfer remains enabled, but no active transfers remain in progress on this endpoint on the USB.

Application programming sequence:

1. Asserting the INCOMPISOOUT interrupt (OTG_GINTSTS) indicates that in the current frame, at least one isochronous OUT endpoint has an incomplete transfer.
2. If this occurs because isochronous OUT data is not completely emptied from the endpoint, the application must ensure that the application empties all isochronous OUT data (data and status) from the receive FIFO before proceeding.
   – When all data are emptied from the receive FIFO, the application can detect the XFRC interrupt (OTG_DOEPINTx). In this case, the application must re-enable the endpoint to receive isochronous OUT data in the next frame.
3. When it receives an INCOMPISOOUT interrupt (in OTG_GINTSTS), the application must read the control registers of all isochronous OUT endpoints (OTG_DOEPCTLx) to determine which endpoints had an incomplete transfer in the current microframe. An endpoint transfer is incomplete if both the following conditions are met:
   – EONUM bit (in OTG_DOEPCTLx) = FNSOF[0] (in OTG_DSTS)
   – EPENA = 1 (in OTG_DOEPCTLx)
4. The previous step must be performed before the SOF interrupt (in OTG_GINTSTS) is detected, to ensure that the current frame number is not changed.
5. For isochronous OUT endpoints with incomplete transfers, the application must discard the data in the memory and disable the endpoint by setting the EPDIS bit in OTG_DOEPCTLx.
6. Wait for the EPDISD interrupt (in OTG_DOEPINTx) and enable the endpoint to receive new data in the next frame.
   – Because the core can take some time to disable the endpoint, the application may not be able to receive the data in the next frame after receiving bad isochronous data.

- **Stalling a non-isochronous OUT endpoint**

This section describes how the application can stall a non-isochronous endpoint.

1. Put the core in the Global OUT NAK mode.
2. Disable the required endpoint
   – When disabling the endpoint, instead of setting the SNAK bit in OTG_DOEPCTL, set STALL = 1 (in OTG_DOEPCTL).

   The STALL bit always takes precedence over the NAK bit.
3. When the application is ready to end the STALL handshake for the endpoint, the STALL bit (in OTG_DOEPCTLx) must be cleared.
4. If the application is setting or clearing a STALL for an endpoint due to a SetFeature.Endpoint Halt or ClearFeature.Endpoint Halt command, the STALL bit must be set or cleared before the application sets up the status stage transfer on the control endpoint.

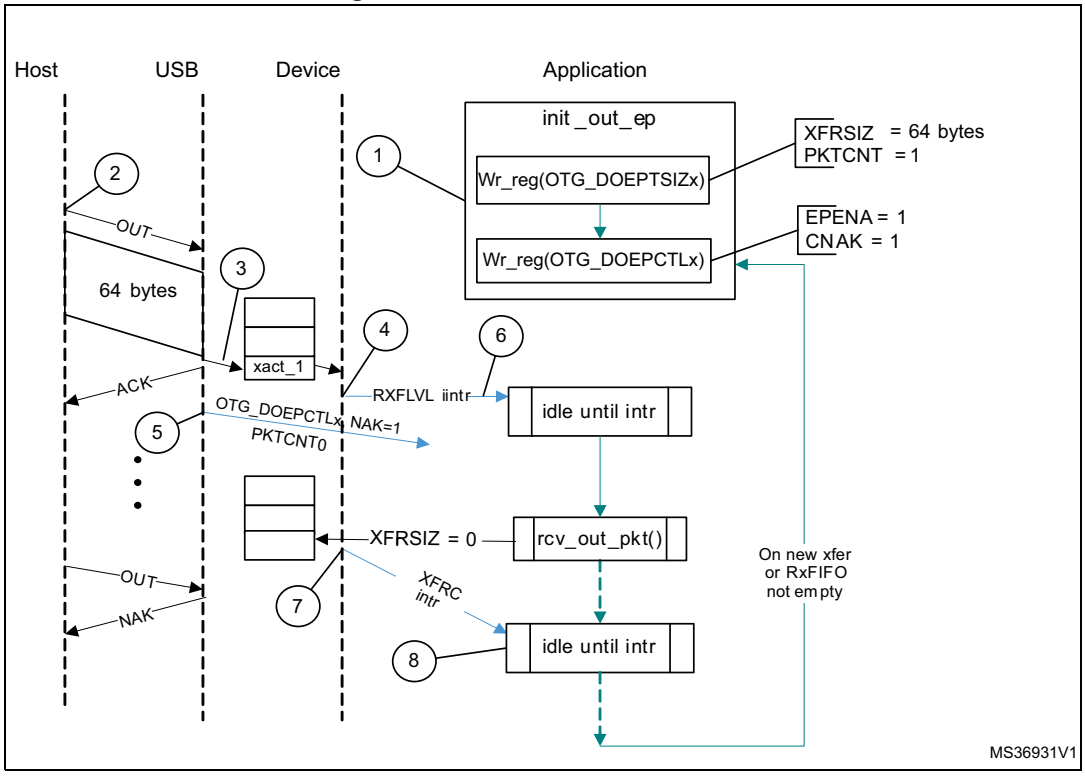
Examples

This section describes and depicts some fundamental transfer types and scenarios.

- Bulk OUT transaction

Figure 1006 depicts the reception of a single Bulk OUT data packet from the USB to the AHB and describes the events involved in the process.

Figure 1006. Bulk OUT transaction



After a SetConfiguration/SetInterface command, the application initializes all OUT endpoints by setting CNAK = 1 and EPENA = 1 (in OTG_DOEPCTLx), and setting a suitable XFRSIZ and PKTCNT in the OTG_DOEPTSIZx register.

1. host attempts to send data (OUT token) to an endpoint.
2. When the core receives the OUT token on the USB, it stores the packet in the Rx FIFO because space is available there.
3. After writing the complete packet in the Rx FIFO, the core then asserts the RXFLVL interrupt (in OTG_GINTSTS).
4. On receiving the PKTCNT number of USB packets, the core internally sets the NAK bit for this endpoint to prevent it from receiving any more packets.
5. The application processes the interrupt and reads the data from the Rx FIFO.
6. When the application has read all the data (equivalent to XFRSIZ), the core generates an XFRC interrupt (in OTG_DOEPINTx).
7. The application processes the interrupt and uses the setting of the XFRC interrupt bit (in OTG_DOEPINTx) to determine that the intended transfer is complete.

IN data transfers

- **Packet write**

This section describes how the application writes data packets to the endpoint FIFO when dedicated transmit FIFOs are enabled.

1. The application can either choose the polling or the interrupt mode.
   – In polling mode, the application monitors the status of the endpoint transmit data FIFO by reading the OTG_DTXFSTSx register, to determine if there is enough space in the data FIFO.
   – In interrupt mode, the application waits for the TXFE interrupt (in OTG_DIEPINTx) and then reads the OTG_DTXFSTSx register, to determine if there is enough space in the data FIFO.
   – To write a single non-zero length data packet, there must be space to write the entire packet in the data FIFO.
   – To write zero length packet, the application must not look at the FIFO space.
2. Using one of the above mentioned methods, when the application determines that there is enough space to write a transmit packet, the application must first write into the endpoint control register, before writing the data into the data FIFO. Typically, the application, must do a read modify write on the OTG_DIEPCTLx register to avoid modifying the contents of the register, except for setting the endpoint enable bit.

The application can write multiple packets for the same endpoint into the transmit FIFO, if space is available. For periodic IN endpoints, the application must write packets only for one microframe. It can write packets for the next periodic transaction only after getting transfer complete for the previous transaction.

- **Setting IN endpoint NAK**

Internal data flow:

1. When the application sets the IN NAK for a particular endpoint, the core stops transmitting data on the endpoint, irrespective of data availability in the endpoint's transmit FIFO.
2. Non-isochronous IN tokens receive a NAK handshake reply
   – Isochronous IN tokens receive a zero-data-length packet reply
3. The core asserts the INEPNE (IN endpoint NAK effective) interrupt in OTG_DIEPINTx in response to the SNAK bit in OTG_DIEPCTLx.
4. Once this interrupt is seen by the application, the application can assume that the endpoint is in IN NAK mode. This interrupt can be cleared by the application by setting the CNAK bit in OTG_DIEPCTLx.

Application programming sequence:

1. To stop transmitting any data on a particular IN endpoint, the application must set the IN NAK bit. To set this bit, the following field must be programmed.
   – SNAK = 1 in OTG_DIEPCTLx
2. Wait for assertion of the INEPNE interrupt in OTG_DIEPINTx. This interrupt indicates that the core has stopped transmitting data on the endpoint.
3. The core can transmit valid IN data on the endpoint after the application has set the NAK bit, but before the assertion of the NAK Effective interrupt.
4. The application can mask this interrupt temporarily by writing to the INEPNEM bit in OTG_DIEPMSK.
   – INEPNEM = 0 in OTG_DIEPMSK
5. To exit endpoint NAK mode, the application must clear the NAK status bit (NAKSTS) in OTG_DIEPCTLx. This also clears the INEPNE interrupt (in OTG_DIEPINTx).

– CNAK = 1 in OTG_DIEPCTLx

6. If the application masked this interrupt earlier, it must be unmasked as follows:
   – INEPNEM = 1 in OTG_DIEPMSK

- **IN endpoint disable**

Use the following sequence to disable a specific IN endpoint that has been previously enabled.

Application programming sequence:

1. The application must stop writing data on the AHB for the IN endpoint to be disabled.
2. The application must set the endpoint in NAK mode.
   – SNAK = 1 in OTG_DIEPCTLx
3. Wait for the INEPNE interrupt in OTG_DIEPINTx.
4. Set the following bits in the OTG_DIEPCTLx register for the endpoint that must be disabled.
   – EPDIS = 1 in OTG_DIEPCTLx
   – SNAK = 1 in OTG_DIEPCTLx
5. Assertion of the EPDISD interrupt in OTG_DIEPINTx indicates that the core has completely disabled the specified endpoint. Along with the assertion of the interrupt, the core also clears the following bits:
   – EPENA = 0 in OTG_DIEPCTLx
   – EPDIS = 0 in OTG_DIEPCTLx
6. The application must read the OTG_DIEPTSIZx register for the periodic IN EP, to calculate how much data on the endpoint were transmitted on the USB.
7. The application must flush the data in the endpoint transmit FIFO, by setting the following fields in the OTG_GRSTCTL register:
   – TXFNUM (in OTG_GRSTCTL) = Endpoint transmit FIFO number
   – TXFFLSH in (OTG_GRSTCTL) = 1

The application must poll the OTG_GRSTCTL register, until the TXFFLSH bit is cleared by the core, which indicates the end of flush operation. To transmit new data on this endpoint, the application can re-enable the endpoint at a later point.

- **Transfer Stop Programming for IN endpoints**

The application must use the following programing sequence to stop any transfers (because of an interrupt from the host, typically a reset).

Sequence of operations:

1. Disable the IN endpoint by setting:
   – EPDIS = 1 in all OTG_DIEPCTLx registers
2. Wait for the EPDIS interrupt in OTG_DIEPINTx, which indicates that the IN endpoint is completely disabled. When the EPDIS interrupt is asserted the following bits are cleared:
   – EPDIS = 0 in OTG_DIEPCTLx
   – EPENA = 0 in OTG_DIEPCTLx
3. Flush the TxFIFO by programming the following bits:
   – TXFFLSH = 1 in OTG_GRSTCTL
   – TXFNUM = “FIFO number specific to endpoint” in OTG_GRSTCTL
4. The application can start polling till TXFFLSH in OTG_GRSTCTL is cleared. When this bit is cleared, it ensures that there is no data left in the Tx FIFO.

- **Generic non-periodic IN data transfers**

Application requirements:

1. Before setting up an IN transfer, the application must ensure that all data to be transmitted as part of the IN transfer are part of a single buffer.
2. For IN transfers, the transfer size field in the endpoint transfer size register denotes a payload that constitutes multiple maximum-packet-size packets and a single short packet. This short packet is transmitted at the end of the transfer.
   – To transmit a few maximum-packet-size packets and a short packet at the end of the transfer:
     Transfer size[EPNUM] = *x* × MPSIZ[EPNUM] + sp
     If (sp > 0), then packet count[EPNUM] = x + 1.
     Otherwise, packet count[EPNUM] = *x*
   – To transmit a single zero-length data packet:
     Transfer size[EPNUM] = 0
     Packet count[EPNUM] = 1
   – To transmit a few maximum-packet-size packets and a zero-length data packet at the end of the transfer, the application must split the transfer into two parts. The first sends maximum-packet-size data packets and the second sends the zero-length data packet alone.
     First transfer: transfer size[EPNUM] = *x* × MPSIZ[epnum]; packet count = *n*;
     Second transfer: transfer size[EPNUM] = 0; packet count = 1;
3. Once an endpoint is enabled for data transfers, the core updates the transfer size register. At the end of the IN transfer, the application must read the transfer size register to determine how much data posted in the transmit FIFO have already been sent on the USB.
4. Data fetched into transmit FIFO = Application-programmed initial transfer size – core-updated final transfer size
   – Data transmitted on USB = (application-programmed initial packet count – core updated final packet count) × MPSIZ[EPNUM]
   – Data yet to be transmitted on USB = (Application-programmed initial transfer size – data transmitted on USB)

Internal data flow:

1. The application must set the transfer size and packet count fields in the endpoint-specific registers and enable the endpoint to transmit the data.
2. The application must also write the required data to the transmit FIFO for the endpoint.
3. Every time a packet is written into the transmit FIFO by the application, the transfer size for that endpoint is decremented by the packet size. The data is fetched from the memory by the application, until the transfer size for the endpoint becomes 0. After writing the data into the FIFO, the “number of packets in FIFO” count is incremented (this is a 3-bit count, internally maintained by the core for each IN endpoint transmit FIFO. The maximum number of packets maintained by the core at any time in an IN endpoint FIFO is eight). For zero-length packets, a separate flag is set for each FIFO, without any data in the FIFO.
4. Once the data are written to the transmit FIFO, the core reads them out upon receiving an IN token. For every non-isochronous IN data packet transmitted with an ACK handshake, the packet count for the endpoint is decremented by one, until the packet count is zero. The packet count is not decremented on a timeout.
5. For zero length packets (indicated by an internal zero length flag), the core sends out a zero-length packet for the IN token and decrements the packet count field.
6. If there are no data in the FIFO for a received IN token and the packet count field for that endpoint is zero, the core generates an “IN token received when Tx FIFO is empty” (ITTXFE) interrupt for the endpoint, provided that the endpoint NAK bit is not set. The core responds with a NAK handshake for non-isochronous endpoints on the USB.
7. The core internally rewinds the FIFO pointers and no timeout interrupt is generated.
8. When the transfer size is 0 and the packet count is 0, the transfer complete (XFRC) interrupt for the endpoint is generated and the endpoint enable is cleared.

Application programming sequence:

1. Program the OTG_DIEPTSIZx register with the transfer size and corresponding packet count.
2. Program the OTG_DIEPCTLx register with the endpoint characteristics and set the CNAK and EPENA (endpoint enable) bits.
3. When transmitting non-zero length data packet, the application must poll the OTG_DTXFSTSx register (where x is the FIFO number associated with that endpoint) to determine whether there is enough space in the data FIFO. The application can optionally use TXFE (in OTG_DIEPINTx) before writing the data.

- **Generic periodic IN data transfers**

This section describes a typical periodic IN data transfer.

Application requirements:

1. Application requirements 1, 2, 3, and 4 of *Generic non-periodic IN data transfers* also apply to periodic IN data transfers, except for a slight modification of requirement 2.
   – The application can only transmit multiples of maximum-packet-size data packets or multiples of maximum-packet-size packets, plus a short packet at the end. To

transmit a few maximum-packet-size packets and a short packet at the end of the transfer, the following conditions must be met:

transfer size[EPNUM] = x × MPSIZ[EPNUM] + sp
(where *x* is an integer ≥ 0, and 0 ≤ sp < MPSIZ[EPNUM])

If (sp > 0), packet count[EPNUM] = *x* + 1
Otherwise, packet count[EPNUM] = *x*;

MCNT[EPNUM] = packet count[EPNUM]

– The application cannot transmit a zero-length data packet at the end of a transfer. It can transmit a single zero-length data packet by itself. To transmit a single zero-length data packet:
– transfer size[EPNUM] = 0

packet count[EPNUM] = 1

MCNT[EPNUM] = packet count[EPNUM]

2. The application can only schedule data transfers one frame at a time.
   – (MCNT – 1) × MPSIZ ≤ XFERSIZ ≤ MCNT × MPSIZ
   – PKTCNT = MCNT (in OTG_DIEPTSIZx)
   – If XFERSIZ < MCNT × MPSIZ, the last data packet of the transfer is a short packet.
   – Note that: MCNT is in OTG_DIEPTSIZx, MPSIZ is in OTG_DIEPCTLx, PKTCNT is in OTG_DIEPTSIZx and XFERSIZ is in OTG_DIEPTSIZx
3. The complete data to be transmitted in the frame must be written into the transmit FIFO by the application, before the IN token is received. Even when 1 word of the data to be transmitted per frame is missing in the transmit FIFO when the IN token is received, the core behaves as when the FIFO is empty. When the transmit FIFO is empty:
   – A zero data length packet would be transmitted on the USB for isochronous IN endpoints
   – A NAK handshake would be transmitted on the USB for interrupt IN endpoints

Internal data flow:

1. The application must set the transfer size and packet count fields in the endpoint-specific registers and enable the endpoint to transmit the data.
2. The application must also write the required data to the associated transmit FIFO for the endpoint.
3. Every time the application writes a packet to the transmit FIFO, the transfer size for that endpoint is decremented by the packet size. The data are fetched from application memory until the transfer size for the endpoint becomes 0.
4. When an IN token is received for a periodic endpoint, the core transmits the data in the FIFO, if available. If the complete data payload (complete packet, in dedicated FIFO

mode) for the frame is not present in the FIFO, then the core generates an IN token received when Tx FIFO empty interrupt for the endpoint.
   – A zero-length data packet is transmitted on the USB for isochronous IN endpoints
   – A NAK handshake is transmitted on the USB for interrupt IN endpoints
5. The packet count for the endpoint is decremented by 1 under the following conditions:
   – For isochronous endpoints, when a zero- or non-zero-length data packet is transmitted
   – For interrupt endpoints, when an ACK handshake is transmitted
   – When the transfer size and packet count are both 0, the transfer completed interrupt for the endpoint is generated and the endpoint enable is cleared.
6. At the “Periodic frame Interval” (controlled by PFIVL in OTG_DCFG), when the core finds non-empty any of the isochronous IN endpoint FIFOs scheduled for the current frame non-empty, the core generates an IISOIXFR interrupt in OTG_GINTSTS.

Application programming sequence:

1. Program the OTG_DIEPCTLx register with the endpoint characteristics and set the CNAK and EPENA bits.
2. Write the data to be transmitted in the next frame to the transmit FIFO.
3. Asserting the ITTXFE interrupt (in OTG_DIEPINTx) indicates that the application has not yet written all data to be transmitted to the transmit FIFO.
4. If the interrupt endpoint is already enabled when this interrupt is detected, ignore the interrupt. If it is not enabled, enable the endpoint so that the data can be transmitted on the next IN token attempt.
5. Asserting the XFRC interrupt (in OTG_DIEPINTx) with no ITTXFE interrupt in OTG_DIEPINTx indicates the successful completion of an isochronous IN transfer. A read to the OTG_DIEPTSIZx register must give transfer size = 0 and packet count = 0, indicating all data were transmitted on the USB.
6. Asserting the XFRC interrupt (in OTG_DIEPINTx), with or without the ITTXFE interrupt (in OTG_DIEPINTx), indicates the successful completion of an interrupt IN transfer. A read to the OTG_DIEPTSIZx register must give transfer size = 0 and packet count = 0, indicating all data were transmitted on the USB.
7. Asserting the incomplete isochronous IN transfer (IISOIXFR) interrupt in OTG_GINTSTS with none of the aforementioned interrupts indicates the core did not receive at least 1 periodic IN token in the current frame.

- **Incomplete isochronous IN data transfers**

This section describes what the application must do on an incomplete isochronous IN data transfer.

Internal data flow:

1. An isochronous IN transfer is treated as incomplete in one of the following conditions:
   a) The core receives a corrupted isochronous IN token on at least one isochronous IN endpoint. In this case, the application detects an incomplete isochronous IN transfer interrupt (IISOIXFR in OTG_GINTSTS).
   b) The application is slow to write the complete data payload to the transmit FIFO and an IN token is received before the complete data payload is written to the FIFO. In this case, the application detects an IN token received when Tx FIFO empty interrupt in OTG_DIEPINTx. The application can ignore this interrupt, as it

eventually results in an incomplete isochronous IN transfer interrupt (IISOIXFR in OTG_GINTSTS) at the end of periodic frame.

The core transmits a zero-length data packet on the USB in response to the received IN token.

2. The application must stop writing the data payload to the transmit FIFO as soon as possible.
3. The application must set the NAK bit and the disable bit for the endpoint.
4. The core disables the endpoint, clears the disable bit, and asserts the endpoint disable interrupt for the endpoint.

Application programming sequence:

1. The application can ignore the IN token received when Tx FIFO empty interrupt in OTG_DIEPINTx on any isochronous IN endpoint, as it eventually results in an incomplete isochronous IN transfer interrupt (in OTG_GINTSTS).
2. Assertion of the incomplete isochronous IN transfer interrupt (in OTG_GINTSTS) indicates an incomplete isochronous IN transfer on at least one of the isochronous IN endpoints.
3. The application must read the endpoint control register for all isochronous IN endpoints to detect endpoints with incomplete IN data transfers.
4. The application must stop writing data to the Periodic Transmit FIFOs associated with these endpoints on the AHB.
5. Program the following fields in the OTG_DIEPCTLx register to disable the endpoint:
   – SNAK = 1 in OTG_DIEPCTLx
   – EPDIS = 1 in OTG_DIEPCTLx
6. The assertion of the endpoint disabled interrupt in OTG_DIEPINTx indicates that the core has disabled the endpoint.
   – At this point, the application must flush the data in the associated transmit FIFO or overwrite the existing data in the FIFO by enabling the endpoint for a new transfer in the next microframe. To flush the data, the application must use the OTG_GRSTCTL register.

- **Stalling non-isochronous IN endpoints**

This section describes how the application can stall a non-isochronous endpoint.

Application programming sequence:

1. Disable the IN endpoint to be stalled. Set the STALL bit as well.
2. EPDIS = 1 in OTG_DIEPCTLx, when the endpoint is already enabled
   – STALL = 1 in OTG_DIEPCTLx
   – The STALL bit always takes precedence over the NAK bit
3. Assertion of the endpoint disabled interrupt (in OTG_DIEPINTx) indicates to the application that the core has disabled the specified endpoint.
4. The application must flush the non-periodic or periodic transmit FIFO, depending on the endpoint type. In case of a non-periodic endpoint, the application must re-enable the other non-periodic endpoints that do not need to be stalled, to transmit data.
5. Whenever the application is ready to end the STALL handshake for the endpoint, the STALL bit must be cleared in OTG_DIEPCTLx.
6. If the application sets or clears a STALL bit for an endpoint due to a SetFeature.Endpoint Halt command or ClearFeature.Endpoint Halt command, the STALL bit must be set or cleared before the application sets up the status stage transfer on the control endpoint.

Special case: stalling the control OUT endpoint

The core must stall IN/OUT tokens if, during the data stage of a control transfer, the host sends more IN/OUT tokens than are specified in the SETUP packet. In this case, the application must enable the ITTXFE interrupt in OTG_DIEPINTx and the OTEPDIS interrupt in OTG_DOEPINTx during the data stage of the control transfer, after the core has transferred the amount of data specified in the SETUP packet. Then, when the application receives this interrupt, it must set the STALL bit in the corresponding endpoint control register, and clear this interrupt.

73.15.7 Worst case response time

When the OTG controller acts as a device, there is a worst case response time for any tokens that follow an isochronous OUT. This worst case response time depends on the AHB clock frequency.

The core registers are in the AHB domain, and the core does not accept another token before updating these register values. The worst case is for any token following an isochronous OUT, because for an isochronous transaction, there is no handshake and the next token may come sooner. This worst case value is 7 PHY clocks when the AHB clock is the same as the PHY clock. When the AHB clock is faster, this value is smaller.

If this worst case condition occurs, the core responds to bulk/interrupt tokens with a NAK and drops isochronous and SETUP tokens. The host interprets this as a timeout condition for SETUP and retries the SETUP packet. For isochronous transfers, the Incomplete isochronous IN transfer interrupt (IISOIXFR) and Incomplete isochronous OUT transfer interrupt (IISOOXFR) inform the application that isochronous IN/OUT packets were dropped.

Choosing the value of TRDT in OTG_GUSBCFG

The value in TRDT (OTG_GUSBCFG) is the time it takes for the MAC, in terms of PHY clocks after it has received an IN token, to get the FIFO status, and thus the first data from the PFC block. This time involves the synchronization delay between the PHY and AHB clocks. The worst case delay for this is when the AHB clock is the same as the PHY clock. In this case, the delay is 5 clocks.

Once the MAC receives an IN token, this information (token received) is synchronized to the AHB clock by the PFC (the PFC runs on the AHB clock). The PFC then reads the data from the SPRAM and writes them into the dual clock source buffer. The MAC then reads the data out of the source buffer (4 deep).

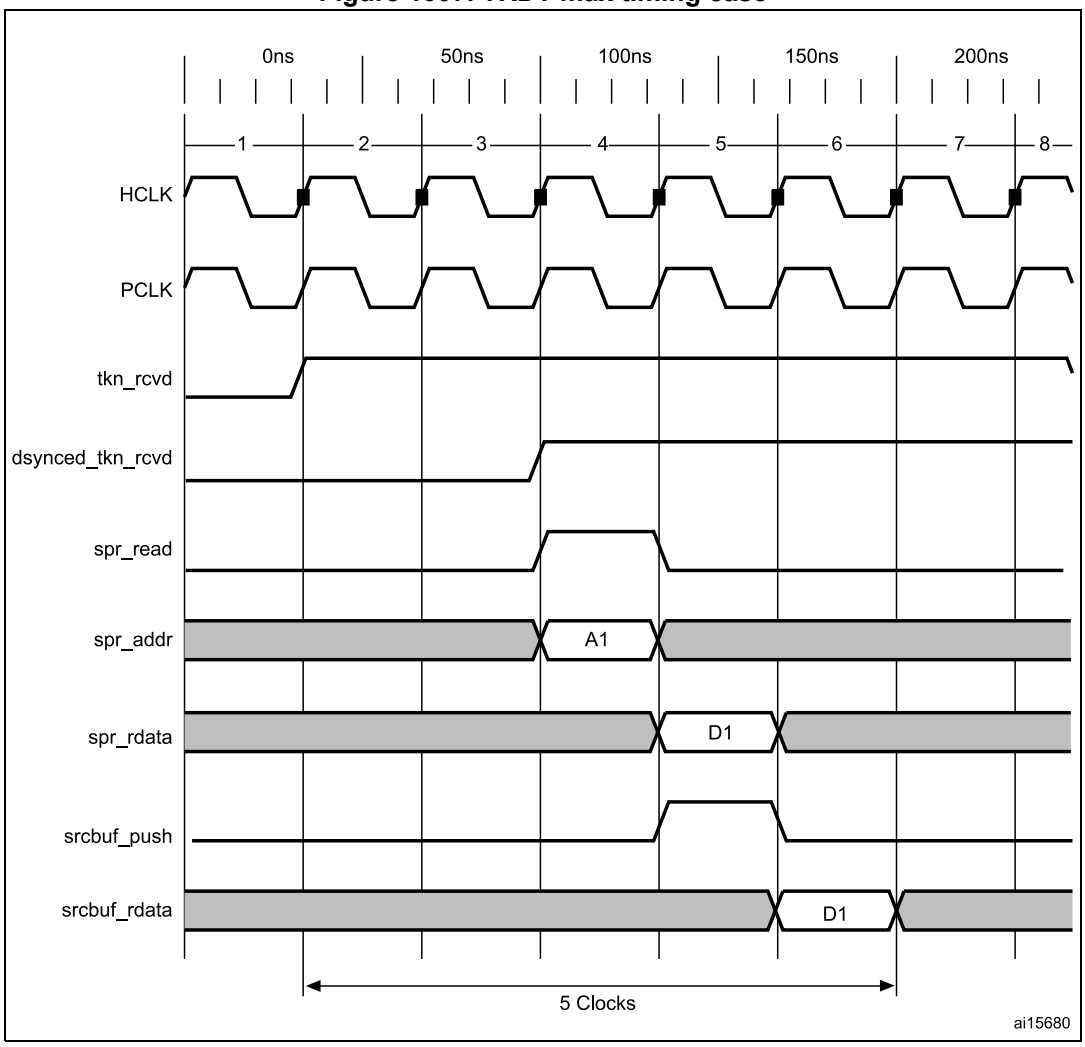
If the AHB is running at a higher frequency than the PHY, the application can use a smaller value for TRDT (in OTG_GUSBCFG).

Figure 1007 has the following signals:

- tkn_rcvd: Token received information from MAC to PFC
- dynced_tkn_rcvd: Doubled sync tkn_rcvd, from PCLK to HCLK domain
- spr_read: Read to SPRAM
- spr_addr: Address to SPRAM
- spr_rdata: Read data from SPRAM
- srcbuf_push: Push to the source buffer
- srcbuf_rdata: Read data from the source buffer. Data seen by MAC

To calculate the value of TRDT, refer to *Table 745: TRDT values*.

Figure 1007. TRDT max timing case



73.15.8 OTG programming model

The OTG controller is an OTG device. When the core is connected to an “A” plug, it is referred to as an A-device. When the core is connected to a “B” plug it is referred to as a B-device.

74 USB HS PHY controller (USBPHYC)

There are two near-identical instances of USBPHYC

- USBPHYC1 (associated with OTG1)
- USBPHYC2 (associated with OTG2)
  - includes one additional control bit concerning hardware debug (HDP)

For a more complete system view of the USB controllers and PHYs, refer to the block diagram of the main OTG controller.

74.1 Introduction

This controller handles general and miscellaneous control of the OTG PHYs.

Note that the main UTMI+ interface for protocol including battery charging features are all controlled by the OTG.

74.2 USBPHYC main features

The USBPHYC allows

- PLL configuration
- Trimming of the electrical parameters (should this be required)

74.3 USBPHYC implementation

Table 748. USBPHYC implementation

USBPHYC feature	USBPHYC1	USBPHYC2
HDP selection control (SELOTGDBG(1)) included	-	X

1. This bit is reserved in USBPHYC1.

74.4 USBPHYC functional description

74.4.1 USBPHYC block diagram

USBPHYC is a small satellite controller allowing full control of a HS PHY block. Refer to the block diagram of the main OTG controller.

74.4.2 USBPHYC reset and clocks

USBPHYC instances can be reset from the RCC using bits

- OTGPHY1RST
- OTGPHY2RST

USBPHYC uses a single AHB clock.

74.4.3 USBPHYC programmable parameters

In addition to the selection of the OTGPHY reference clock frequency, some register bits are used to control the trimming of the electrical parameters of the OTGPHY.

74.4.4 USBPHYC trimming of electrical parameters

Some parameters inside the OTGPHYC can be adjusted by control bits in USBPHYC_TRIM1CR and USBPHYC_TRIM2CR registers. Those adjustments must be programmed while the OTGPHY is under reset.

The adjustable parameters are: the internal PLL, the disconnect threshold, the squelch threshold, the data detection threshold, the transmitter high-speed crossover point, the FS/LS source impedance, the HS DC voltage level, the HS transmitter rise/fall times, the driver source impedance, the HS transmitter pre-emphasis current, the HS transmitter pre-emphasis duration.

The default values in USBPHYC_TRIM1CR and USBPHYC_TRIM2CR registers correspond to the recommended values.

74.5 USBPHYC registers

74.5.1 USBPHYC control register (USBPHYC_CR)

Address offset: 0x0

Reset value: 0x0001 0015

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SELOTGDBG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DRVVBUS0	OTGDISABLE0
rw														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSEL[2:0]			Res.	CMN	AUTORSMENB1	RETENABLEN1
									rw	rw	rw		rw	rw	rw

Bit 31 **SELOTGDBG**: Select OTG debug

Selects the set of USB signals that are routed to the hardware debug port (HDP)

0: Debug signals from OTGPHY1 are routed to the HDP (default)

1: Debug signals from OTGPHY2 are routed to the HDP

Note: This bit is only available in USBPHYC2.

Bits 30:18 Reserved, must be kept at reset value.

Bit 17 **DRVVBUS0**: Drive V_{BUS} 0

Control bit for the V_{BUS} valid comparator

0: The V_{BUS} valid comparator is OFF in Suspend and Sleep modes (default)

1: The V_{BUS} valid comparator is ON in Suspend and Sleep modes

Bit 16 **OTGDISABLE0**: OTG disable 0

Disable control bit for the V_{BUS} valid comparator

0: The V_{BUS} valid comparator is ON in normal mode. The state of the V_{BUS} valid comparator in Suspend and Sleep modes depends on DRVVBUS0 setting. The session valid comparator is ON.

1: The V_{BUS} valid comparator is powered down. The session valid comparator is ON (default)

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:4 **FSEL[2:0]**: Frequency selection

PHY refclk speed setting:

000: 19.2 MHz

001: 20 MHz (default)

010: 24 MHz

Others: reserved

Bit 3 Reserved, must be kept at reset value.

Bit 2 **CMN**: Controls the power down of analog blocks during Suspend and Sleep.

0: PHY clocks are kept active during Suspend and Sleep mode (with the exception of PHYCLK). PLL is on. Requires to keep the PHY reference clock on.

1: PHY clocks are off during Suspend and Sleep mode. PLL is off. (default)

Bit 1 **AUTORSMENB1**: Auto-resume mode enable

Enables auto-resume logic in so that the PHY automatically responds to a remote wake-up without initial involvement of the Host controller.

0: Auto-resume disabled (default)

1: A pulse on this bit prior to enabling the retention mode enables the auto-resume mode.

Bit 0 **RETENABLEN1**: Retention mode enable (active low)

0: Retention mode enabled

1: Retention mode disabled (default)

74.5.2 USBPHYC trimming 1 register (USBPHYC_TRIM1CR)

Address offset: 0x4

Reset value: 0x299E D6F0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TXRESTUNE[1:0]		TXRISETUNE[1:0]		TXVREFTUNE[3:0]				TXFSLSTUNE[3:0]				TXHSXVTUNE[1:0]		OTGTUNE[2]
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OTGTUNE[1:0]		VDATREFTUNE[1:0]		SQRXTUNE[2:0]			COMPDISTUNE[2:0]			PLLPTUNE[3:0]				PLLITUNE[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:29 **TXRESTUNE[1:0]**: USB source impedance adjustment

In some applications there can be significant serial resistance on the DP and DM paths between the transceiver and the cable. This bitfield adjusts the driver source impedance to compensate for added serial resistance on the USB.

Note: Any setting other than the default one can result in source impedance variations across process, voltage and temperature that do not meet USB2.0 specification limits.

11: Source impedance is decreased by approximately 5.6 Ω

10: Source impedance is decreased by approximately 3.6 Ω

01: Nominal

00: Source impedance is increased by approximately 1.3 Ω

This is a strapping option that must be set prior to USBPHYC reset. It must remain static during normal operation.

Bits 28:27 **TXRISETUNE[1:0]**: HS transmitter rise/fall time adjustment

It adjusts the rise/fall times of the high-speed waveform.

11: -8.1%

10: -7.2%

01: Nominal

00: +5.4%

This is a strapping option that must be set prior to USBPHYC reset. It must remain static during normal operation.

Bits 26:23 **TXVREFTUNE[3:0]**: HS DC Voltage Level Adjustment

It adjusts the high speed DC level voltage.

1111: +24%
1110: +22%
1101: +20%
1100: +18%
1011: +16%
1010: +14%
1001: +12%
1000: +10%
0111: +8%
0110: +6%
0101: +4%
0100: +2%
0011: Nominal
0010: -2%
0001: -4%
0000: -6%

When the PHY is in a Host role, any adjustment of the HS transmit voltage level tune bits (TXVREFTUNE[3:0]) might also necessitate a change to the HS disconnect detection threshold (COMPDISTUNE[2:0]).

This is a strapping option that must be set prior to USBPHYC reset. It must remain static during normal operation.

Bits 22:19 **TXFSLSTUNE[3:0]**: FS/LS source impedance adjustment

It adjusts the low- and full-speed single-ended source impedance while driving high. The following adjustment values are based on nominal process, voltage and temperature.

1111: -10.7%
1110: Reserved
1101: Reserved
1100: Reserved
1011: Reserved
1010: Reserved
1001: Reserved
1000: Reserved
0111: -5.6%
0110: Reserved
0101: Reserved
0100: Reserved
0011: Nominal
0010: Reserved
0001: +6.5%
0000: +14%

This is a strapping option that must be set prior to USBPHYC reset. It must remain static during normal operation.

Bits 18:17 **TXHSXVTUNE[1:0]**: Transmitter high-speed crossover adjustment

It adjusts the voltage at which DP and DM signals crossover while transmitting in high-speed mode.

11: Nominal

10: +15 mV

01: -15 mV

00: Reserved

This is a strapping option that must be set prior to USBPHYC reset. It must remain static during normal operation.

Bits 16:14 **OTGTUNE[2:0]**: V_{BUS} valid threshold adjustment

It adjusts the voltage level for the V_{BUS} valid threshold.

111: +12%

110: +9%

101: +6%

100: +3%

011: Nominal

010: -3%

001: -6%

000: -9%

This is a strapping option that must be set prior to USBPHYC reset. It must remain static during normal operation.

Bits 13:12 **VDATREFTUNE[1:0]**: Data detect voltage adjustment

It adjusts the threshold voltage level (V_{DAT_REF}) used to detect data during charger type detection.

11: -20%

10: -10%

01: Nominal

00: +10%

This is a strapping option that must be set prior to USBPHYC reset. It must remain static during normal operation.

Bits 11:9 **SQRXTUNE[2:0]**: Squelch threshold adjustment

It adjusts the voltage level for the threshold used to detect valid high speed data.

111: -20%

110: -15%

101: -10%

100: -5%

011: Nominal

010: +5%

001: +10%

000: +15%

This is a strapping option that must be set prior to USBPHYC reset. It must remain static during normal operation.

Bits 8:6 **COMPDISTUNE[2:0]**: Disconnect threshold adjustment

It adjusts the voltage level for the threshold used to detect a disconnect event at the Host.

111: +12%
110: +10%
101: +7%
100: +3%
011: Nominal
010: -4%
001: -6%
000: -9%

When the PHY is in a Host role, any adjustment of the HS transmit voltage level tune bits (TXVREFTUNE[3:0]) might also necessitate a change to the HS disconnect detection threshold (COMPDISTUNE[2:0]).

This is a strapping option that must be set prior to USBPHYC reset. It must remain static during normal operation.

Bits 5:2 **PLLPTUNE[3:0]**: PLL proportional path tune

1111: 5.5×
1110: 5×
1101: 4.5×
1100: 4×
1011: 4.5×
1010: 4×
1001: 3.5×
1000: 3×
0111: 3.5×
0110: 3×
0101: 2.5×
0100: 2×
0011: 2.5×
0010: 2×
0001: 1.5×
0000: 1×

This is a strapping option that must be set prior to USBPHYC reset. It must remain static during normal operation.

Bits 1:0 **PLLITUNE[1:0]**: PLL integral path tune

11: 2.0×
10: 1.5×
01: 1.0×
00: 0.5×

This is a strapping option that must be set prior to USBPHYC reset. It must remain static during normal operation.

74.5.3 USBPHYC trimming 2 register (USBPHYC_TRIM2CR)

Address offset: 0x8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXPREEMPPULSETUNE	TXPREEMPAMPTUNE[1:0]	
													rw	rw	rw

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **TXPREEMPPULSETUNE**: HS transmitter pre-emphasis duration control

This bit controls the duration for which the HS pre-emphasis current is sourced onto DM or DP. The HS transmitter pre-emphasis duration is defined in terms of unit amounts. One unit of pre-emphasis duration is approximately 580 ps and is defined as 1× pre-emphasis duration. This bit is valid only if TXPREEMPAMPTUNE[0] or TXPREEMPAMPTUNE[1] is set to 1.

1: 1×, short pre-emphasis current duration

0: 2×, long pre-emphasis current duration

This is a strapping option that must be set prior to USBPHYC reset. It must remain static during normal operation.

Bits 1:0 **TXPREEMPAMPTUNE[1:0]**: HS transmitter pre-emphasis current control

This bitfield controls the amount of current sourced to DP and DM after a J-to-K or K-to-J transition. The HS transmitter pre-emphasis current is defined in terms of unit amounts. One unit amount is approximately 600 µA and is defined as 1× pre-emphasis current.

11: HS transmitter pre-emphasis circuit sources 3× pre-emphasis current

10: HS transmitter pre-emphasis circuit sources 2× pre-emphasis current

01: HS transmitter pre-emphasis circuit sources 1× pre-emphasis current

00: HS transmitter pre-emphasis is disabled

This is a strapping option that must be set prior to USBPHYC reset. It must remain static during normal operation.

74.5.4 USBPHYC register map

Table 749. USBPHYC register map and reset values

Offset	Register name / Reset value	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	**USBPHYC1_CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DRVVBUS0	OTGDISABLE0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSEL[2:0]			Res.	CMN	AUTORSMENB1	RETENABLEN1
	Reset value															0	1										0	0	1		1	0	1
0x000	**USBPHYC2_CR**	SELOTGDBG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DRVVBUS0	OTGDISABLE0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FSEL[2:0]			Res.	CMN	AUTORSMENB1	RETENABLEN1
	Reset value	0														0	1										0	0	1		1	0	1
0x004	**USBPHYC_TRIM1CR**	Res.	TXRESTUNE[1:0]		TXRISETUNE[1:0]		TXVREFTUNE[3:0]				TXFSLSTUNE[3:0]				TXHSXVTUNE[1:0]		OTGTUNE[2:0]			VDATREFTUNE[1;0]		SQRXTUNE[2:0]			COMPDISTUNE[2:0]			PLLPTUNE[3:0]				PLLITUNE[1:0]	
	Reset value		0	1	0	1	0	0	1	1	0	0	1	1	1	1	0	1	1	0	1	0	1	1	0	1	1	1	1	0	0	0	0
0x008	**USBPHYC_TRIM2CR**	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXPREEMPPULSETUNE	TXPREEMPAMPTUNE[1:0]	
	Reset value																														0	0	0

Refer to *Section 2.3* for the register boundary addresses.

75 USB Type-C®/USB Power Delivery interface (UCPD)

75.1 UCPD introduction

The USB Type-C/USB Power Delivery interface complies with:

- Universal Serial Bus Type-C Cable and Connector Specification: release 2.4, Oct 2024
- Universal Serial Bus Power Delivery specifications:
  - revision 2.0, version 1.3, January 12, 2017
  - revision 3.2, version 1.1, October 2024

It integrates the physical layer of the Power Delivery (PD) specification, with CC signaling method (no VBUS), for operation with Type-C cables.

75.2 UCPD main features

- Compliance with USB Type-C specification release 2.4
- Compliance with USB Power Delivery specifications revision 2.0 and 3.2
  - Enabling advanced applications such as PPS (programmable power supply)
- Stop mode low-power operation support
- Built-in analog PHY
  - USB Type-C pull-up (Rp, all values) and pull-down (Rd) resistors
  - USB Power Delivery message transmission and reception
- Digital controller
  - BMC (bi-phase mark coding) encode and decode
  - 4b5b encode and decode
  - USB Type-C level detection with de-bounce, generating interrupts
  - FRS signaling
  - DMA-compatible byte-level interface for USB Power Delivery payload, generating interrupts
  - USB Power Delivery clock pre-scaler / dividers
  - CRC generation/checking
  - Support of ordered sets, with a programmable ordered set mask at receive
  - Clock recovery from incoming Rx stream

75.3 UCPD implementation

The devices have one UCPD controller to support one USB Type-C port.

In the following table, “X” denotes *supported*, and “-” *not supported*.

Table 750. UCPD implementation

UCPD feature	UCPD1
UCPDx_FRSTX1/2 as alternate function pins	X
Fully automatic trimming	X(1)
Discrete-component PHY support	-

1. No software trimming required.

75.4 UCPD functional description

The UCPD peripheral provides hardware support for the USB Power Delivery control interface specification, using I/Os specifically designed for that purpose.

The built-in PHY directly detects Type-C voltage levels, supports Power Delivery BIST carrier mode 2 (Tx only), and BIST test data (Tx and Rx).

For Power Delivery FRS Tx signaling, the device can be configured to control, through UCPD_FRSTX1/2 pins (alternate functions), external NMOS transistors that ensure low-resistance pull-down on CC lines.

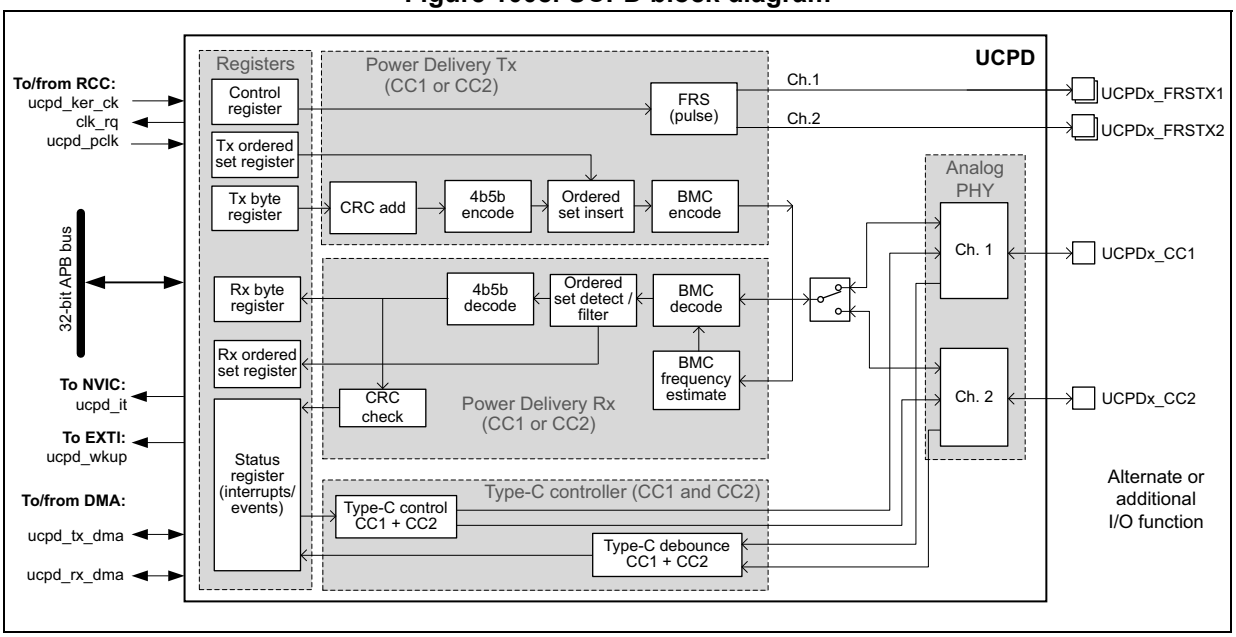
The UCPD transmitter BMC (bi-phase mark) encodes and transmits data: preamble, SOP, payload data from protocol layer (after 4b5b-encoding), CRC, and EOP on the Type-C connector CC lines. It automatically inserts inter-frame gap and executes “Hard Reset”.

The UCPD receiver detects SOP, BMC-decodes the incoming stream, recovers the preamble, 4b5b-decodes payload data, detects EOP, and checks CRC. It automatically detects five K-code SOP and two Reset ordered sets, plus two software-defined patterns (allows for only three out of four K-codes being correctly received, as defined by the standard).

In Stop mode, the peripheral maintains the ability to detect incoming USB Power Delivery messages, which allows low-power operation.

75.4.1 UCPD block diagram

Figure 1008. UCPD block diagram



The following table lists external signals (alternate or additional I/O functions).

Table 751. UCPD signals on pins

Pin name	Signal type	Description
UCPDx_FRSTX1	Output	USB Type-C fast role swap (FRS) signaling, applicable to DRPs only. The signal (active high) drives an external NMOS transistor that pulls down the CC1 line.
UCPDx_FRSTX2	Output	USB Type-C fast role swap (FRS) signaling, applicable to DRPs only. The signal (active high) drives an external NMOS transistor that pulls down the CC2 line.
UCPDx_CC1	Input/output	USB Type-C configuration control line 1, to be routed to the USB Type-C connector CC1 terminal.
UCPDx_CC2	Input/output	USB Type-C configuration control line 2, to be routed to the USB Type-C connector CC2 terminal.

The following table lists key internal signals.

Table 752. UCPD internal signals

Internal signal name	Signal type	Description
ucpd_pclk	Input	APB clock for registers
ucpd_ker_ck	Input	Kernel clock
ucpd_tx_dma	Input/Output	Rx DMA acknowledge / request
ucpd_rx_dma	Input/Output	Tx DMA acknowledge / request
ucpd_it	Output	Interrupt request (all interrupts OR-ed) connected to NVIC

Table 752. UCPD internal signals (continued)

Internal signal name	Signal type	Description
ucpd_wkup	Output	Wake-up request connected to EXTI
clk_rq	Output	Clock request connected to RCC

75.4.2 UCPD reset and clocks

The peripheral has a single reset signal (APB bus reset).

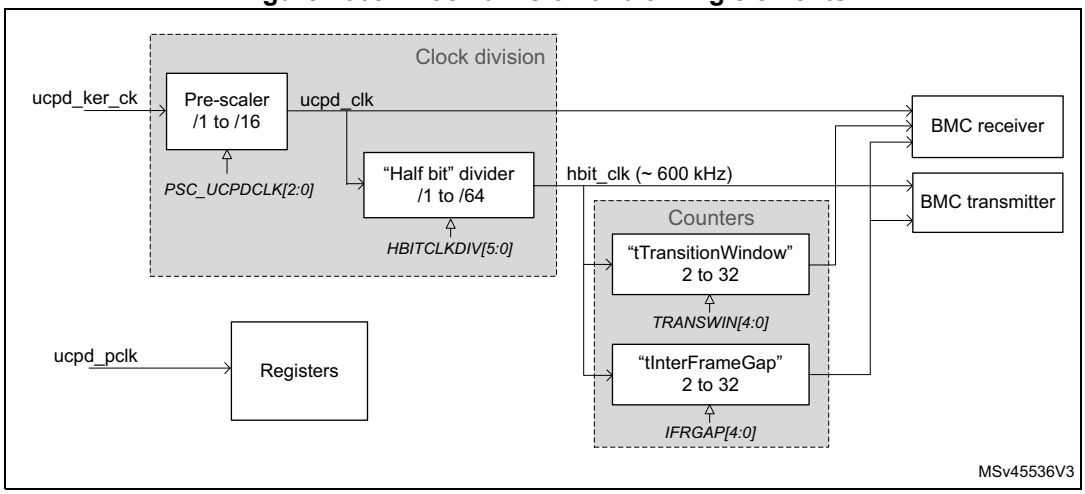
The register section is clocked with the APB clock (ucpd_pclk).

The main functional part of the transmitter is clocked with ucpd_clk clock, pre-scaled from the ucpd_ker_ck clock according to the PSC_UCPDCLK[2:0] bitfield of the UCPD_CFGR1 register. The main functional part of the receiver is clocked with the ucpd_rx_clk recovered from the incoming bitstream.

The receiver is designed to work in the clock frequency range from 6 to 18 MHz. However, the optimum performance is ensured in the range from 6 to 12 MHz.

The following diagram shows the clocking and timing elements of the UCPD peripheral.

Figure 1009. Clock division and timing elements



Refer to the USB PD specification in order to set appropriate delays. For *tTransitionWindow* and especially for *tInterFrameGap*, the clock frequency uncertainty must be taken into account so as to respect specified timings in all cases.

75.4.3 Physical layer protocol

The physical layer covers the signaling underlying the USB Power Delivery specification.

On the transmitter side its main function is to form packets according to the defined packet format including generally:

- preamble
- start of packet (SOP, ordered set)
- payload header
- payload data
- cyclic redundancy check (CRC) information
- end of packet (EOP)

Before going on the CC line, the data stream is BMC-encoded, respecting specified timing restrictions.

On the receive side, the principle task is to:

- extract start of packet (SOP, ordered set) information
- extract payload header
- extract payload data
- receive and check CRC
- receive end of packet (EOP)

The receive is basically a reverse of the transmit process, thus starting with BMC data stream decoding.

Symbol encoding

Apart from the preamble all symbols are encoded with a 4b5b scheme according to the specification shown in the following table.

Table 753. 4b5b symbol encoding table

Name	4b	5b	Symbol description
0	0000	11110	hex data 0
1	0001	01001	hex data 1
2	0010	10100	hex data 2
3	0011	10101	hex data 3
4	0100	01010	hex data 4
5	0101	01011	hex data 5
6	0110	01110	hex data 6
7	0111	01111	hex data 7
8	1000	10010	hex data 8
9	1001	10011	hex data 9
A	1010	10110	hex data A
B	1011	10111	hex data B
C	1100	11010	hex data C
D	1101	11011	hex data D
E	1110	11100	hex data E

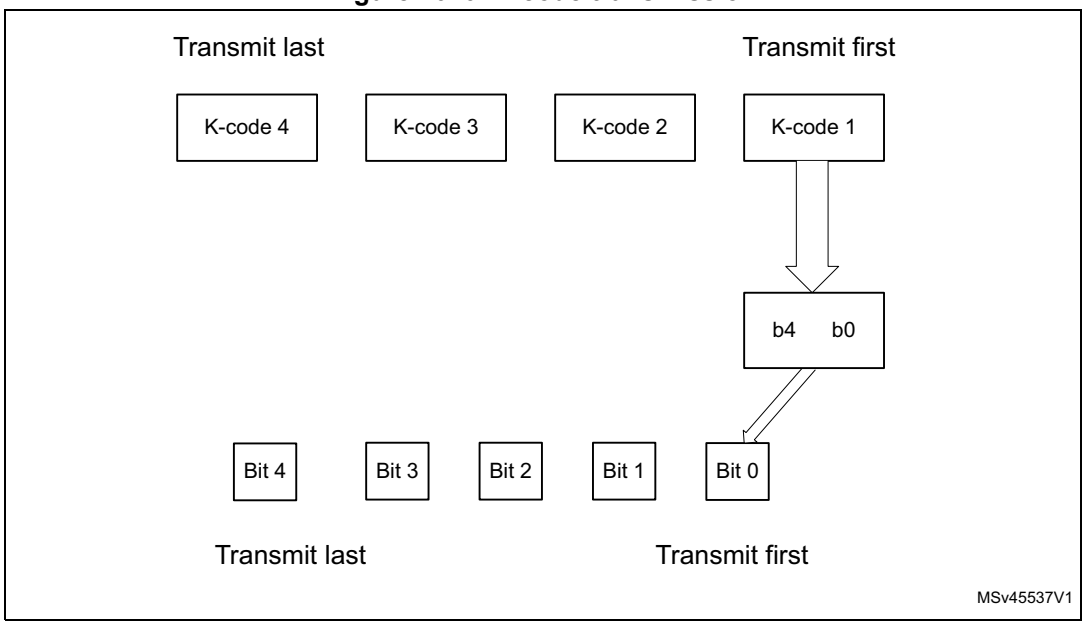
Table 753. 4b5b symbol encoding table (continued)

Name	4b	5b	Symbol description
F	1111	11101	hex data F
Sync-1	K-code	11000	Startsynch #1
Sync-2	K-code	10001	Startsynch #2
RST-1	K-code	00111	Hard Reset #1
RST-2	K-code	11001	Hard Reset #2
EOP	K-code	01101	EOP
Reserved	Error	00000	Do Not Use
Reserved	Error	00001	Do Not Use
Reserved	Error	00010	Do Not Use
Reserved	Error	00011	Do Not Use
Reserved	Error	00100	Do Not Use
Reserved	Error	00101	Do Not Use
Sync-3	K-code	00110	Startsynch #3
Reserved	Error	01000	Do Not Use
Reserved	Error	01100	Do Not Use
Reserved	Error	10000	Do Not Use
Reserved	Error	11111	Do Not Use

Ordered sets

An ordered set consists of four K-codes as shown in the following figure.

Figure 1010. K-code transmission



The following table lists the defined ordered sets, including all possible SOP* sequences.

At the physical layer, the Hard Reset has higher priority than the other ordered sets so it can interrupt an ongoing Tx message.

Table 754. Ordered sets

Ordered set name	K-code #1	K-code #2	K-code #3	K-code #4
SOP	Sync-1	Sync-1	Sync-1	Sync-2
SOP’	Sync-1	Sync-1	Sync-3	Sync-3
SOP’’	Sync-1	Sync-3	Sync-1	Sync-3
Hard Reset	RST-1	RST-1	RST-1	RST-2
Cable Reset	RST-1	Sync-1	RST-1	Sync-3
SOP’_Debug	Sync-1	RST-2	RST-2	Sync-3
SOP’’_Debug	Sync-1	RST-2	Sync-3	Sync-2

On reception, the physical layer must accept ordered sets with any combination of three correct K-codes out of four, as shown in the following table:

Table 755. Validation of ordered sets

Status	1st code	2nd code	3rd code	4th code
Valid	Corrupt	K-code	K-code	K-code
Valid	K-code	Corrupt	K-code	K-code
Valid	K-code	K-code	Corrupt	K-code
Valid	K-code	K-code	K-code	Corrupt
Valid (perfect)	K-code	K-code	K-code	K-code
Not valid (example)	K-code	Corrupt	K-code	Corrupt

Bit ordering at transmission

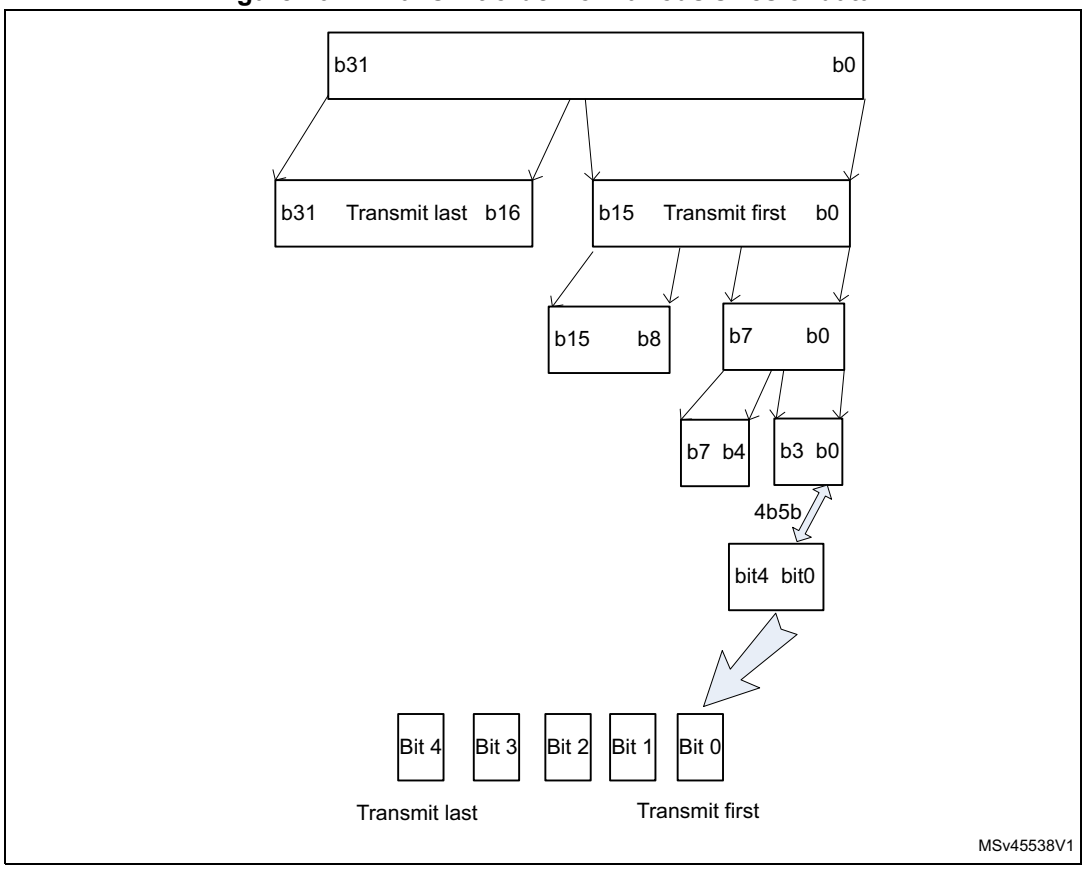
Allowed transmission data units / data sizes are in the following table.

Table 756. Data size

Data unit	Non-encoded	Encoded
Byte	8-bits	10-bits
Word	16-bits	20-bits
DWord	32-bits	40-bits

The bit transmission order is shown in the following figure.

Figure 1011. Transmit order for various sizes of data

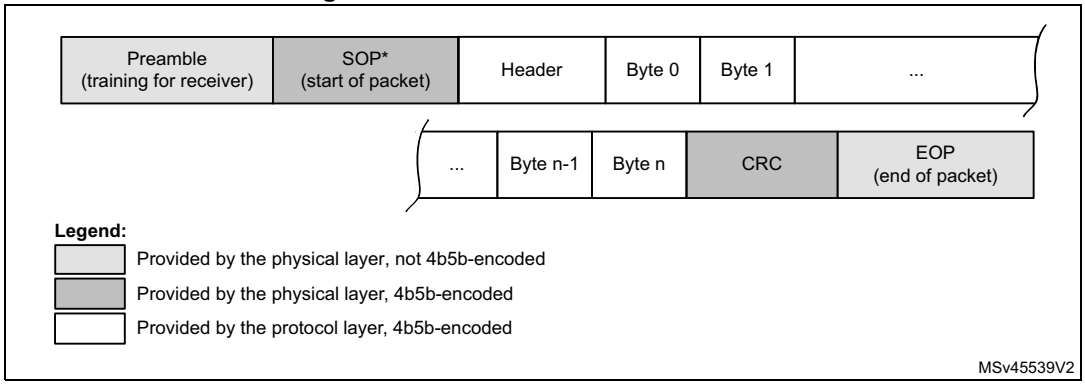


Packet format

Messages other than Hard Reset and Cable Reset

The packet format is shown in the following figure, with information on 4b5b encode and data source.

Figure 1012. Packet format



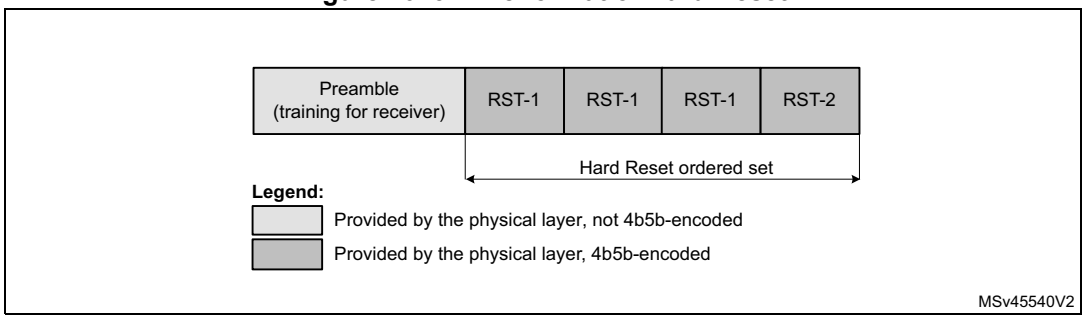
Hard Reset

The physical layer handles the Hard Reset signaling differently than the other types of message as it has higher priority to be able to interrupt an ongoing transfer.

The physical layer specification implies the following sequence in the case of an ongoing Tx message:

1. Terminate the message by sending an EOP K-code and discard the rest of the message.
2. Wait for *tInterFrameGap* time.
3. If the CC line is not idle, wait until it goes idle.
4. Send the preamble followed by the four K-codes of Hard Reset signaling.
5. Disable the CC channel (stop sending and receiving), reset the physical layer and inform the protocol layer that the physical layer is reset.
6. Re-enable the channel when requested by the protocol layer.

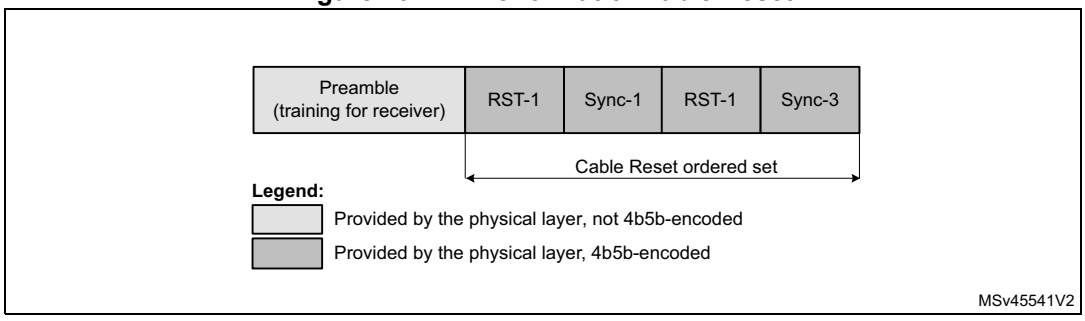
Figure 1013. Line format of Hard Reset



Cable Reset

Cable Reset shown in the following figure is similar in format to Hard Reset, but unlike Hard Reset it does not require a specific high-priority treatment.

Figure 1014. Line format of Cable Reset



Collision avoidance

The physical layer respects the *tInterFrameGap* delay between end of last-transmitted bit of a Tx message, and the first bit of a following message.

It also checks the idle state of the CC line before starting transmission. The CC line is considered idle if it shows less than three (*nTransitionCount*) transitions within

tTransitionWindow (12 to 20 µs). The Power Delivery specification revision 3.1 also requires to manage the Rp value (source) and monitor Type-C voltage level for these Rp modifications (at the sink).

Physical layer signaling schemes

The bit are signaled with bi-phase mark coding (BMC).

BIST

Depending on the BIST action required by the protocol layer, either of the following can be run:

- a Tx BIST pattern test, achieved by writing TXMODE and TXSEND
- an Rx BIST pattern test, achieved by writing RXMODE to the correct value for RXBIST.

The two possible patterns supported in UCPD (corresponding to “BMC” mode) are:

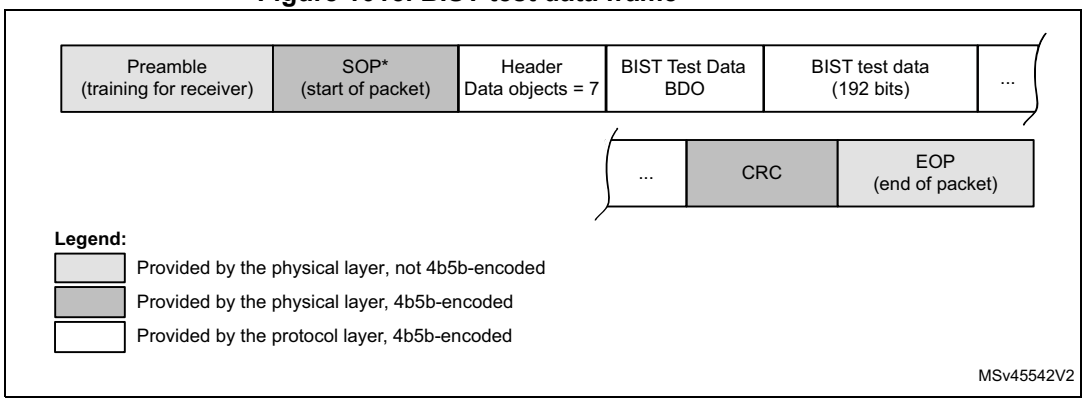
- BIST Test Data (192 bit pattern), applies to Tx and Rx. In the case of Rx, the message is received (but discarded rather than passing to the protocol layer, which must nevertheless still generate a GoodCRC Tx message in acknowledgment).
- BIST Carrier Mode 2 (single pattern, infinite length message), applies to Tx only. As opposed to Tx, the receiver in this mode simply ignores the CC line during this state.

BIST test data pattern

The test data pattern is not viewed as a special case in UCPD.

The BIST test data packet frame format is shown in the following figure.

Figure 1015. BIST test data frame



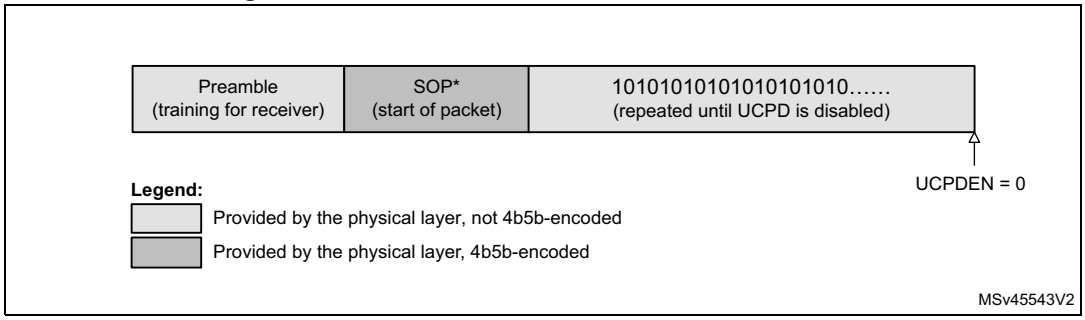
This is a fixed length test data pattern. In reality the only aspect that marks its difference from the general packet format already shown in *Figure 1012: Packet format* is the contents of the Header. As UCPD receives the Tx Header contents via programming (it is simply viewed as part of the payload), it is only this programming (and not the block’s behavior) that differentiates the general packet from the BIST Test Data packet.

BIST Carrier Mode 2

When required, this BIST test mode sends an alternating pattern of 1010 that is continually repeated. As this mode is intended for signal analysis it is stable condition with (in V1.0 of the USB PD specification) no defined length. Starting from V1.1 of the USB PD specification, the protocol layer defines a counter that indicates when to exit this mode.

The way to quit the infinite 1010 sequence (according to requirements of the USB PD specification) is to disable the UCPD peripheral via the UCPDEN bit.

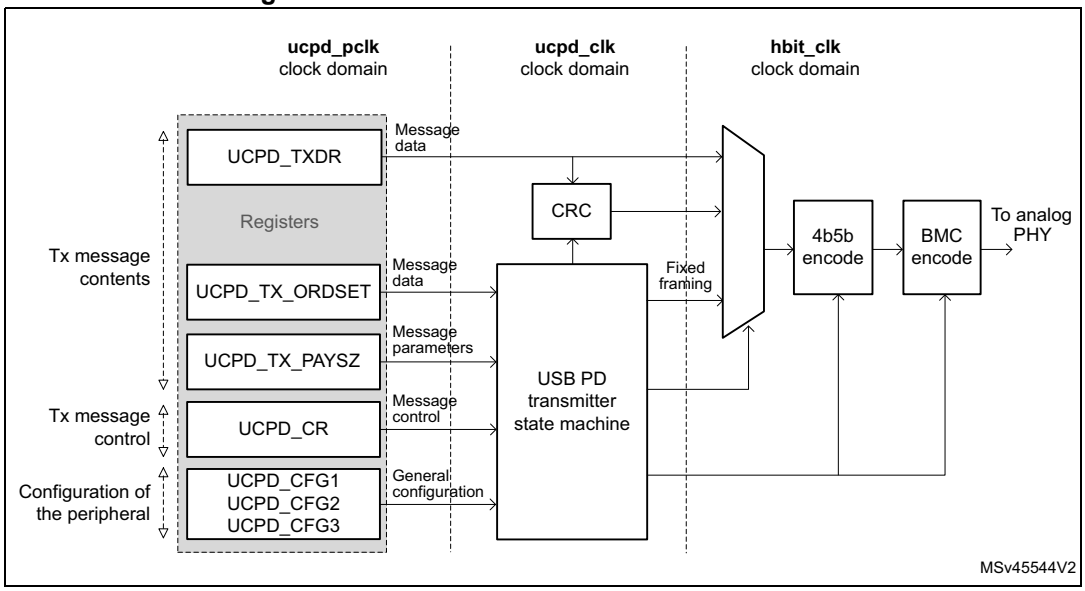
Figure 1016. BIST Carrier Mode 2 frame



75.4.4 UCPD BMC transmitter

The BMC transmitter comprises 4b5b encoding, CRC generation, and BMC encode, as shown in the following figure. Its output goes to the analog PHY through a channel switch.

Figure 1017. UCPD BMC transmitter architecture



BMC encoder

The bi-phase mark coding method is defined in the *IEC 60958-1 Digital Audio Interface Part:1 General Edition 3.0 2008-09 www.iec.ch* specification.

The half-bit clock hbit_clk is derived from ucpd_clk through a simple divider controlled by the HBITCLKDIV[5:0] bitfield of the UCPD_CFGR1 register. This ensures the same duration of high and low half-bit periods (if neglecting a small difference due to different rising and falling edge duration and due to jitter), and the same bit duration (if neglecting jitter).

Transmitter timing and collision avoidance

Hardware support of collision avoidance is made as a function of the half bit time for the transmitter. Two counters are implemented:

– *tInterFrameGap*: via IFRGAP (pre-defined value, can be altered)
– *tTransitionWindow*: via TRANSWIN (pre-defined value, can be altered)

These two counters once set correctly generates the interframe gap.

Hard Reset in transmitter

In order to facilitate generation of a Hard Reset, a special code of TXMODE field is used. No other fields need to be written.

On writing the correct code, the hardware forces Hard Reset Tx under the correct (optimal) timings with respect to an ongoing Tx message, which (if still in progress) is cleanly terminated by truncating the current sequence and directly appending an EOP K-code sequence. No specific interrupt is generated relating to this truncation event.

Transmitter behavior in the case of errors

The under-run condition (TXUND interrupt) may happen by accident and in this case, the UCPD is starved of (the correct) Tx payload and is not able to complete the Tx message correctly. This is a serious error (for this to happen the software fails to respond in time). As a result the hardware ensures the CRC is incorrect at the end of the message, thus guaranteeing the message to be discarded at the receiver.

75.4.5 UCPD BMC receiver

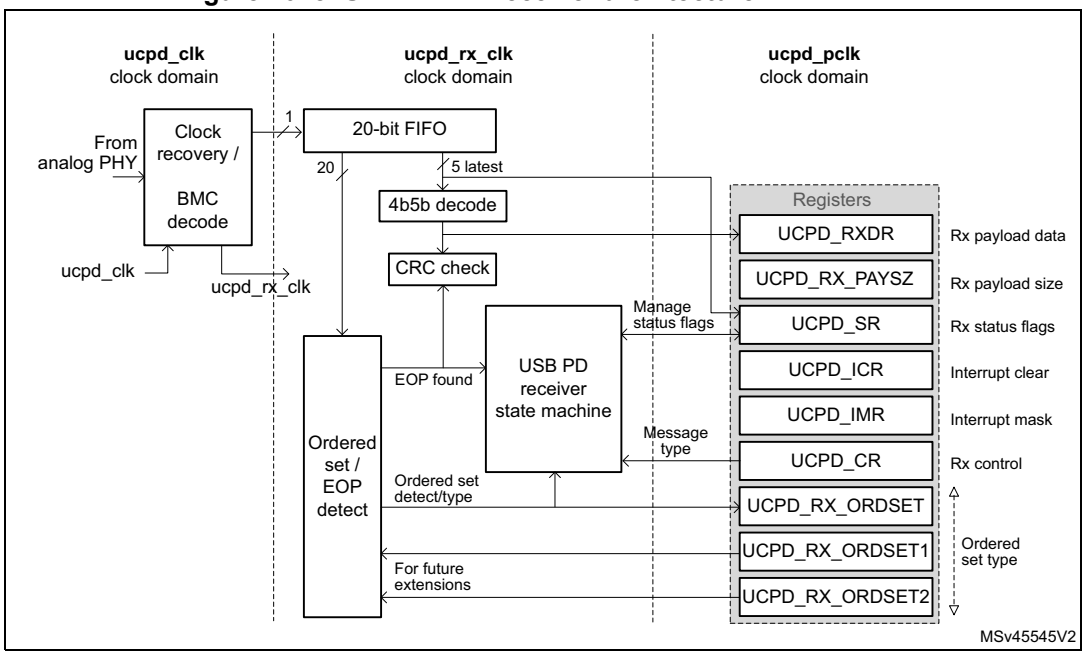
The UCPD BMC receiver performs:

- Clock recovery
- Preamble detection / timing derivation
- BMC decoding
- 4b5b decoding
- K-code ordered set recognition
- CRC checking
- SOP detection
- EOP detection

The receiver is activated as soon as the UCPD peripheral is enabled (via UCPDEN), but it waits for an idle CC line state before attempting to receive a message.

The following figure shows the UCPD BMC receiver high-level architecture.

Figure 1018. UCPD BMC receiver architecture



CRC checker

The received bits are fed into a CRC checker which evolves a 32-bit state during the received the payload bitstream. At the end the 32 bits of the CRC also fed into the logic

The EOP detection (5 bits) halts the process and a check is performed for the fixed residual state which confirms correct reception of the payload (in fact the residual is 0xC704DD78).

At this point the UCPD raises interrupt RXMSGEND. If the CRC was not correct then RXERR is set true and the receive data must be discarded.

Under normal operation, this interrupt would previously have been acknowledged and thus cleared. If this is not the case, a different interrupt RXOVR is generated in place of RXMSGEND.

Ordered set detection

This function detects the different ordered sets each consisting of four 5-bit K-codes.

Once we are in the preamble we opens a sliding window detection of the ordered set (4 words of 5 bits).

The ordered sets detected include all SOP* codes (SOP, SOP', and SOP''), but also Hard Reset, Cable Reset, SOP'_Debug, SOP''_Debug, and two extensions defined by registers UCPD_RX_ORDEXT1 and UCPD_RX_ORDEXT2.

EOP detection and Hard Reset exception handling

EOP is a fixed 5-bit K-code marking the end of a message.

The way in which a transmitter is required to send a Hard Reset (if a previous message transmit is still in progress) is that this previous message is truncated early with an EOP.

If Hard Reset were ignored, then the EOP detection can be done only at the expected time. However, due to the Hard Reset issue, the EOP detector must be active while an Rx

message is arriving. When an “early EOP” is detected, the truncated Rx message is immediately discarded.

Truncated or corrupted message exception

Once the ordered set has been detected, depending on the message, there may be data bytes to be received which is completed with a CRC and EOP. If at any point during these phases an error condition happens:

- the line becomes static for a time significantly longer than one “UI” period (the exact threshold for this condition is not critical but the exception must occur before three UIs), or
- the message goes to the end but it is not recognized (for example EOP is corrupted).

In both cases, the receiver quits the current message, raising RXMSGEND and RXERR flags.

Short preamble or incomplete ordered set exception

In the exceptional case of the receiver seeing less that half of the expected preamble, the frequency estimation allowing correct BMC-decode becomes impossible. Even if the full preamble is seen, allowing frequency estimation, but the ordered set is not fully received before the line becomes static, the receiver state machine does not start.

In both of these cases, the clock-recovery/BMC decoder re-starts, checking initially for an IDLE condition, followed by a preamble, and then estimating frequency.

75.4.6 UCPD Type-C pull-ups (Rp) and pull-downs (Rd)

UCPD offers simple control of these resistors via ANAMODE and ANASUBMODE[1:0]. In case only one of the CC lines is to be used, it is possible to optimize power consumption by disabling control on the other line, through the CCENABLE[1:0] bitfield.

When the MCU is unpowered, it does not signal "dead battery" (which must therefore be handled with external components).

After power arrives and the MCU boots, the desired behavior (for example source) must be programmed into ANAMODE and ANASUBMODE[1:0].

Use of Standby low-power mode is possible for sinks in the unattached state.

75.4.7 UCPD Type-C voltage monitoring and de-bouncing

For correct operation of the Type-C state machine and for detecting the cable orientation, the CC1/2 lines must be monitored for voltage level, while ignoring fast events such as peaks.

Thresholds between voltage levels on the CC1/2 lines are determined through PHY threshold detector settings.

The TYPEC_VSTATE_CC1/2[1:0] bitfields reflect the CC1/2 line levels processed with a hardware de-bouncing filter that suppresses high-speed line events such as peaks. The PHYCCSEL bit selects the line, CC1 or CC2, to be used for Power Delivery signaling.

For minimizing the power consumption, it is recommended to use the polling method, with the Type-C detectors only turned on for the instant of polling, rather than keeping the Type-C detectors permanently on and wake the device up from Stop mode upon CC1/2 line events.

75.4.8 UCPD fast role swap (FRS)

FRS signaling

The FRS condition (a pulse of a specific length), is generated upon setting the FRSTX bit.

For the duration of FRS condition, the currently active I/O configured as UCPD_FRSTX1 (or 2) (alternate function) controls, with high level, the gate of an external NMOS transistor that pulls the active CC line down.

75.4.9 UCPD DMA Interface

DMA is implemented in the UCPD and when it is enabled the byte-level interrupts to handle UCPD1_TXDR and UCPD1_RXDR registers (Tx and Rx data register, each one byte) are no longer needed.

By enabling bits TXDMAEN and/or RXDMAEN, DMA can be activated independently for Tx and/or Rx functionality.

75.4.10 Wake-up from Stop mode

For power consumption optimization, it is useful to use Stop mode and wait for events on CC lines to wake the MCU up.

In order for this to work, it must be first enabled by writing a 1 to WUPEN.

The events causing the wake-up can be:

- Events on the BMC receiver (RXORDDET, RXHRSTDET), hardware enable PHYRXEN
- Events on the Type-C detectors (TYPECEVT1, TYPECEVT2), hardware enables CC1TCDIS, CC2TCDIS

75.5 UCPD programming sequences

The normal sequence of use of the UCPD unit is:

1. Configure UCPD.
2. Enable UCPD.
3. Concurrently:
   – On demand from protocol layer, send Tx message
   – Intercept (poll or wait for interrupt) relevant Rx messages and recover detail to hand off to protocol layer

Repeat the last point infinitely.

75.5.1 Initialization phase

Use the following sequence for a clean startup:

1. Prepare all initial clock divider values, by writing the UCPD_CFG register.
2. Enable the unit, by setting the UCPDEN bit.

75.5.2 Type-C state machine handling

For the general application cases of source, sink, or dual-role port (the last alternating the source and the sink), the software must implement a corresponding USB Type-C state machine. The basic coding is in the following table.

Table 757. Coding for ANAMODE, ANASUBMODE and link with TYPEC_VSTATE_CCx

ANAMODE	ANASUBMODE[1:0]	Notes	TYPEC_VSTATE_CCx[1:0]			
			00	01	10	11
0: Source	00: Disabled	Disabled	N/A			
	01: Default USB Rp	-	vRa[Def]	vRd[Def]	vOPEN[Def]	N/A
	10: 1.5A Rp	-	vRa[1.5]	vRd[1.5]	vOPEN[1.5]	
	11: 3.0A Rp	-	vRa[3.0]	vRd[3.0]	vOPEN[3.0]	
1: Sink	xx	-	vRa	vRd-USB	vRd-1.5	vRd-3.0

The CCENABLE[1:0] bitfield can disable pull-up/pull-downs on one of the CC lines.

Note: The Type-C state machine depends not only on CC line levels, but also on VBUS presence detection (sink mode) and, when in source mode, determines VCONN generation and VBUS state (ON/OFF/+voltage level); discharge). UCPD does not directly control VBUS generation circuitry nor VCONN load switch (enabling VCONN supply generator to be connected to the CC line), but the application needs these inputs and controls, to function correctly.

General programming sequence (with UCPD configured then enabled)

1. Set ANAMODE and ANASUBMODE[1:0] based on the current position in USB Type-C state machine (and also the current advertisement in the case of a source). This turns on the appropriate pull-ups/pull-downs on the CC lines, and defines the voltage levels that the TYPEC_VSTATE fields represent. Note that before programming, the PHY is effectively off.
2. Read TYPEC_VSTATE_CC1/2 to determine the initial Type-C state (for example whether the local source is connected to a remote sink).
3. In the case of no connection, wait for a connection event.
4. Assuming a connection is detected and assuming a local Power Delivery functionality is implemented, start sending/receiving Power Delivery messages.
5. When a new interrupt/event occurs on PHYEVT1/2 indicating a change in stable voltage, re-evaluate the implications and give this input to the Type-C state machine.

Case of a source that needs to change between one of the three possible Rp values (Default-USB / 1.5A / 3.0A) and the sink connected to it:

- [Source] Simply reprogram ANASUBMODE[1:0]
- [Sink behavior from that time] PHYEVT1/2 occurs and the TYPEC_VSTATE1/2 changes accordingly

Programming for a dual-role port (DRP) toggling from source to sink:

- Simply re-program ANAMODE and ANASUBMODE[1:0] to start the new behavior

Detailed programming sequence (example):

Table 758. Type-C sequence (source: 3A); cable/sink connected (Rd on CC1; Ra on CC2)

Type-C state	ANAMODE; ANASUBMODE[1:0]	CCENABLE	PHYCCSEL	RDCH	CC[x] VCONNEN(1)	Event => go to next line	Comments
Unattached. SRC	0:Source; 11:Rp3A0	11:both enabled	0 (don't care)	0: [Normal]	00: [neither]	PHYEVT 1: [VRd-3A0]	Wait for sink attach detect ; seen on CC1 [EVT1]
Attachwait. SRC			0 [Rd on CC1]			PHYEVT 2: [VRa]	Attachwait started (100-200 ms) ; now also see the Ra => requesting VCONN
Attached. SRC [VCONN => CC2]	0:Source; 11:Rp3A0 [SinkTxOK]	01: CC2 disable (possible and recommended due to external VCONN switch)			10: [CC2 active]	Timer (100 ms) and no PHYEVTx	Local CC2 disconnected from PHY (VCONN switch connects VCONN source to CC2 externally; Continue to monitor PHYEVT1
	0:Source; 10:Rp1A5 [SinkTxNG]					SW timers (SinkTxNG)	Source wants to initiate message sequence (SinkTxNG condition set first)
	0:Source; 11:Rp3A0 [SinkTxOK]						Source finished message sequence (SinkTxOK condition afterwards)
						PHYEVT 1: [VOpen-3A0]	Wait for Sink disconnected (Vopen on CC1)
Unattached wait. SRC		11:both enabled	0 (do not care)	1: [discharge]	00: [neither]	> 0.8V detection	Discharge VCONN [CC2] actively [Rdch]; to < 0.8V
Unattached. SRC	0:Source; 11:Rp3A0			0: [Normal]			[Details as first line of table]

1. Two GPIOs to enable VCONN through external load switch components

75.5.3 USB PD transmit

On reception of a message from the protocol layer (that is, to be sent), prepare Tx message contents by writing the UCPD_TX_ORDSET and UCPD_TX_PAYSZ registers.

The message transmission is triggered by setting the TXSEND bit, with an appropriate value of the TXMODE bitfield.

When the data byte is transmitted, the TXIS flag is raised to request a new data write to the UCPD_TXDR register.

This re-iterates until the entire payload of data is transmitted.

Upon sending the CRC packet, the TXMSGSENT flag is set to indicate the completion of the message transmission.

Hard Reset transmission

As soon as it is known that a Hard Reset needs to be transmitted, setting the TXHRST bit of the UCPD_CR register forces the internal state machine to generate the correct sequence. The value of UCPD_TX_ORDSET does not require update in this precise case (the correct code for Hard Reset is sent by UCPD).

The USB Power Delivery specification requires that in the case of an ongoing message transmission, the Hard Reset takes precedence. In this case, for example, UCPD truncates the payload of the current message, appending EOP to the end. No notification is available via the registers (for example through the TXMSGSEND flag). This is justified by the fact that the Hard Reset takes precedence over any previous activity (for which it is therefore no longer important to know if it is completed).

Use of DMA for transmission

DMA (Direct Memory Access) can be enabled for transmission by setting the TXDMAEN bit in the UCPD_CR register.

For each message:

- Prepare the whole message in memory (starting with two header bytes)
- Program the DMA operation with a length corresponding to the two header bytes plus a number of data bytes corresponding to the number of data words multiplied by four
- Write TXSEND to initiate the message transfer
- If TXMSGDISC then go back to previous line (TXSEND)
- Wait for DMA transfer complete interrupt (that is, when last Tx byte written to UCPD)
- Double-check subsequent TXMSGSENT interrupt appears

75.5.4 USB PD receive

Notification of start of the receive message sequence is triggered by an interrupt on UCPD_SR (bit RXORDDET).

The information is recovered by reading:

- UCPD_RX_SOP (on interrupt RXORDDET)
- UCPD_RXDR (on interrupt RXNE, repeats for each byte)
- UCPD_RXPAYSZ (on interrupt RXMSGEND)

The data previously read from UCPD_RXDR above must be discarded at this point if the RXERR flag is set.

If the CRC is valid, the received data is transferred to the protocol layer.

For debug purposes, it may be desirable to track statistics of the number of incorrect K-codes received (this is done only when 3/4 K-codes were valid as defined in the specification). This is facilitated through:

- RXSOP3OF4 bit indicating the presence of at least one invalid K-code
- RXSOPKINVALID bitfield identifying the order of invalid K-code in the ordered set

Use of DMA for reception

DMA (Direct Memory Access) can be enabled for reception by setting the RXDMAEN bit in the UCPD_CR register.

Whenever a Rx message is expected:

- Program a DMA receive operation (and spare buffer) a little longer than the maximum possible message (length depends on extended message support).
- After receiving RXORDDET, DMA operation starts working in the background.
- On reception of RXMSGEND interrupt, read RXPAYSZ.
- Double-check RXPAYSZ vs. the number of DMA Rx bytes (must correspond but DMA read of RXDR is slightly after RXDR gets last byte).
- Process the DMA Rx buffer.
- Prepare next Rx DMA buffer as soon as possible in order to be ready.

75.6 UCPD low-power modes

A summary of low-power modes is shown below in *Table 759: Effect of low power modes on the UCPD*.

Table 759. Effect of low power modes on the UCPD

Mode	Description
Sleep	No effect
Stop	Detection of events (Type-C, BMC Rx) remains operational and can wake up the MCU.
Standby	UCPD is not operating, and cannot wake up the MCU. Pull-downs remain active if configured.
Unpowered	Not functional.

The UCPD is able to wake up the MCU from Stop mode when it recognizes a relevant event, either:

- Type-C event relating to a change in the voltage range seen on either of the CC lines, visible in TYPEC_VSTATE_CCx
- Power delivery receive message with an ordered set matching those filtered according to RXORDSETEN[8:0], visible by reading RXORDSET

Wake-up from Stop mode is enabled by setting the WUPEN bit in the UCPD_CFG2 register.

At UCPD level three types of event requiring kernel clock activity may occur during Stop mode:

- Activity on the analog PHY voltage threshold detectors which can later be confirmed to be a stable change between voltage ranges defined in the Type-C specification
- Activity on Power Delivery BMC receiver (coming from the selected CC line) which can potentially generate an Rx message event (that is, RXORDSET) later

It order to function correctly with the RCC, the clock request signal is activated (conditional on WUPEN) when there is asynchronous activity on:

- Type-C voltage threshold detectors (coming from either CC line)
- Power Delivery receiver signal (from the selected CC line)

75.7 UCPD interrupts

The table below lists the UCPD event flags, with the associated flag clear bits and interrupt enable bits.

Table 760. UCPD interrupt requests

Interrupt event	Event flag	Event flag/Interrupt clearing method	Interrupt enable control bit
Type C voltage level change on CC2	TYPECEVT2	Set TYPECEVT2CF	TYPECEVT2IE
Type C voltage level change on CC1	TYPECEVT1	Set TYPECEVT1CF	TYPECEVT1IE
Rx message received	RXMSGEND	Set RXMSGENDCF	RXMSGENDIE
Rx data overflow	RXOVR	Set RXOVRCF	RXOVR
Rx Hard Reset detected	RXHRSTDET	Set RXHRSTDETCF	RXHRSTDETIE
Rx ordered set (4 K-codes) detected	RXORDDET	Set RXORDDETCF	RXORDDETIE
Receive data register not empty	RXNE	Read data in UCPD_RXDR	RXNEIE
Tx data underrun	TXUND	Set TXUNDCF	TXUNDIE
Hard Reset sent	HRSTSENT	Set HRSTSENTCF	HRSTSENTIE
Hard Reset discarded	HRSTDISC	Set HRSTDISCCF	HRSTDISCIE
Transmit message aborted	TXMSGABT	Set TXMSGABTCF	TXMSGABTIE
Transmit message sent	TXMSGSENT	Set TXMSGSENTCF	TXMSGSENTIE
Transmit message discarded	TXMSGDISC	Set TXMSGDISCCF	TXMSGDISCIE
Transmit data required	TXIS	Write data to the UCPD_TXDR register	TXISIE

When an interrupt from the UCPD is received, then the software has to check what is the source of the interrupt by reading the UCPD_SR register.

Depending on which bit is at 1, the ISR must handle that condition and clear the bit by a write to the appropriate bit of the UCPD_ICR register.

75.8 UCPD registers

75.8.1 UCPD configuration register 1 (UCPD_CFGR1)

Address offset: 0x000

Reset value: 0x0000 0000

General configuration of the peripheral. Writing to this register is only effective when UCPD is disabled (UCPDEN = 0).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UCPDEN	RXDMAEN	TXDMAEN	RXORDSETEN[8:0]									PSC_UCPDCLK[2:0]			Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRANSWIN[4:0]					IFRGAP[4:0]					HBITCLKDIV[5:0]					
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **UCPDEN**: UCPD peripheral enable

General enable of the UCPD peripheral.

0: Disable

1: Enable

Upon disabling, the peripheral instantly quits any ongoing activity and all control bits and bitfields default to their reset values. They must be set to their desired values each time the peripheral transits from disabled to enabled state.

Bit 30 **RXDMAEN**: Reception DMA mode enable

When set, the bit enables DMA mode for reception.

0: Disable

1: Enable

Bit 29 **TXDMAEN**: Transmission DMA mode enable

When set, the bit enables DMA mode for transmission.

0: Disable

1: Enable

Bits 28:20 **RXORDSETEN[8:0]**: Receiver ordered set enable

The bitfield determines the types of ordered sets that the receiver must detect. When set/cleared, each bit enables/disables a specific function:

0bXXXXXXXX1: SOP detect enabled

0bXXXXXXX1X: SOP' detect enabled

0bXXXXXX1XX: SOP'' detect enabled

0bXXXXX1XXX: Hard Reset detect enabled

0bXXXX1XXXX: Cable Detect reset enabled

0bXXX1XXXXX: SOP'_Debug enabled

0bXX1XXXXXX: SOP''_Debug enabled

0bX1XXXXXXX: SOP extension#1 enabled

0b1XXXXXXXX: SOP extension#2 enabled

Bits 19:17 **PSC_UCPDCLK[2:0]**: Pre-scaler division ratio for generating ucpd_clk

The bitfield determines the division ratio of a kernel clock pre-scaler producing UCPD peripheral clock (ucpd_clk).

0x0: 1 (bypass)
0x1: 2
0x2: 4
0x3: 8
0x4: 16

It is recommended to use the pre-scaler so as to set the ucpd_clk frequency in the range from 6 to 9 MHz.

Bit 16 Reserved, must be kept at reset value.

Bits 15:11 **TRANSWIN[4:0]**: Transition window duration

The bitfield determines the division ratio (the bitfield value minus one) of a hbit_clk divider producing *tTransitionWindow* interval.

0x00: Not supported
0x01: 2
0x09: 10 (recommended)
0x1F: 32

Set a value that produces an interval of 12 to 20 us, taking into account the ucpd_clk frequency and the HBITCLKDIV[5:0] bitfield setting.

Bits 10:6 **IFRGAP[4:0]**: Division ratio for producing inter-frame gap timer clock

The bitfield determines the division ratio (the bitfield value minus one) of a ucpd_clk divider producing inter-frame gap timer clock (*tInterFrameGap*).

0x00: Not supported
0x01: 2
0x0D: 14
0x0E: 15
0x0F: 16
0x1F: 32

The division ratio 15 is to apply for Tx clock at the USB PD 2.0 specification nominal value. The division ratios below 15 are to apply for Tx clock below nominal, and the division ratios above 15 for Tx clock above nominal.

Bits 5:0 **HBITCLKDIV[5:0]**: Division ratio for producing half-bit clock

The bitfield determines the division ratio (the bitfield value plus one) of a ucpd_clk divider producing half-bit clock (hbit_clk).

0x00: 1 (bypass)
0x1A: 27
0x3F: 64

75.8.2 UCPD configuration register 2 (UCPD_CFGR2)

Address offset: 0x004

Reset value: 0x0000 0000

Configuration of the UCPD Rx signal filtering. Writing to this register is only effective when UCPD is disabled (UCPDEN = 0).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUPEN	FORCECLK	RXFILT2N3	RXFILTDIS
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **WUPEN**: Wake-up from Stop mode enable
Setting the bit enables the UCPD_ASYNC_INT signal.
0: Disable
1: Enable

Bit 2 **FORCECLK**: Force ClkReq clock request
0: Do not force clock request
1: Force clock request

Bit 1 **RXFILT2N3**: BMC decoder Rx pre-filter sampling method
Number of consistent consecutive samples before confirming a new value.
0: 3 samples
1: 2 samples

Bit 0 **RXFILTDIS**: BMC decoder Rx pre-filter enable
0: Enable
1: Disable
The sampling clock is that of the receiver (that is, after pre-scaler).

75.8.3 UCPD control register (UCPD_CR)

Address offset: 0x00C

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is enabled (UCPDEN = 1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2TCDIS	CC1TCDIS	Res.	RDCH	FRSTX	Res.
										rw	rw		rw	rs	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	CCENABLE[1:0]		ANAMODE	ANASUBMODE[1:0]		PHYCCSEL	PHYRXEN	RXMODE	TXHRST	TXSEND	TXMODE[1:0]	
				rw	rw	rw	rw	rw	rw	rw	rw	rs	rs	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **CC2TCDIS**: CC2 Type-C detector disable

The bit disables the Type-C detector on the CC2 line.

0: Enable

1: Disable

When enabled, the Type-C detector for CC2 is configured through ANAMODE and ANASUBMODE[1:0].

Bit 20 **CC1TCDIS**: CC1 Type-C detector disable

The bit disables the Type-C detector on the CC1 line.

0: Enable

1: Disable

When enabled, the Type-C detector for CC1 is configured through ANAMODE and ANASUBMODE[1:0].

Bit 19 Reserved, must be kept at reset value.

Bit 18 **RDCH**: Rdch condition drive

The bit drives Rdch condition on the CC line selected through the PHYCCSEL bit (thus associated with VCONN), by remaining set during the source-only *UnattachedWait.SRC* state, to respect the Type-C state. Refer to *"USB Type-C ECN for Source VCONN Discharge"*. The CCENABLE[1:0] bitfield must be set accordingly, too.

0: No effect

1: Rdch condition drive

Bit 17 **FRSTX**: FRS Tx signaling enable.

Setting the bit enables FRS Tx signaling.

0: No effect

1: Enable

The bit is cleared by hardware after a delay respecting the USB Power Delivery specification Revision 3.1.

Bit 16 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bit 14 Reserved, must be kept at reset value.

Bit 13 Reserved, must be kept at reset value.

Bit 12 Reserved, must be kept at reset value.

Bits 11:10 **CCENABLE[1:0]**: CC line enable

This bitfield enables CC1 and CC2 line analog PHYs (pull-ups and pull-downs) according to ANAMODE and ANASUBMODE[1:0] setting.

0x0: Disable both PHYs

0x1: Enable CC1 PHY

0x2: Enable CC2 PHY

0x3: Enable CC1 and CC2 PHY

A single line PHY can be enabled when, for example, the other line is driven by VCONN via an external VCONN switch. Enabling both PHYs is the normal usage for sink/source.

Bit 9 **ANAMODE**: Analog PHY operating mode

0: Source

1: Sink

The use of CC1 and CC2 depends on CCENABLE. Refer to *Table 757: Coding for ANAMODE, ANASUBMODE and link with TYPEC_VSTATE_CCx* for the effect of this bitfield in conjunction with ANASUBMODE[1:0].

Bits 8:7 **ANASUBMODE[1:0]**: Analog PHY sub-mode
Refer to *Table 757: Coding for ANAMODE, ANASUBMODE and link with TYPEC_VSTATE_CCx* for the effect of this bitfield.

Bit 6 **PHYCCSEL**: CC1/CC2 line selector for USB Power Delivery signaling
0: Use CC1 IO for Power Delivery communication
1: Use CC2 IO for Power Delivery communication
The selection depends on the cable orientation as discovered at attach.

Bit 5 **PHYRXEN**: USB Power Delivery receiver enable
0: Disable
1: Enable
Both CC1 and CC2 receivers are disabled when the bit is cleared. Only the CC receiver selected via the PHYCCSEL bit is enabled when the bit is set.

Bit 4 **RXMODE**: Receiver mode
Determines the mode of the receiver.
0: Normal receive mode
1: BIST receive mode (BIST test data mode)
When the bit is set, RXORDSET behaves normally, RXDR no longer receives bytes yet the CRC checking still proceeds as for a normal message. As this mode prevents reception of the header (containing MessageID), software has to auto-increment a received MessageID counter for inclusion in the GoodCRC acknowledge that must still be transmitted during this test.

Bit 3 **TXHRST**: Command to send a Tx Hard Reset
0: No effect
1: Start Tx Hard Reset message
The bit is cleared by hardware as soon as the message transmission begins or is discarded.

Bit 2 **TXSEND**: Command to send a Tx packet
0: No effect
1: Start Tx packet transmission
The bit is cleared by hardware as soon as the packet transmission begins or is discarded.

Bits 1:0 **TXMODE[1:0]**: Type of Tx packet
Writing the bitfield triggers the action as follows, depending on the value:
0x0: Transmission of Tx packet previously defined in other registers
0x1: Cable Reset sequence
0x2: BIST test sequence (BIST Carrier Mode 2)
Others: invalid
From V1.1 of the USB PD specification, there is a counter defined for the duration of the BIST Carrier Mode 2. To quit this mode correctly (after the "tBISTContMode" delay), disable the peripheral (UCPDEN = 0).

75.8.4 UCPD interrupt mask register (UCPD_IMR)

Address offset: 0x010

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is enabled (UCPDEN = 1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TYPECEVT2IE	TYPECEVT1IE	Res.	RXMSGENDIE	RXOVRIE	RXHRSTDETIE	RXORDDETIE	RXNEIE	Res.	TXUNDIE	HRSTSENTIE	HRSTDISCIE	TXMSGABTIE	TXMSGSENTIE	TXMSGDISCIE	TXISIE
rw	rw		rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **TYPECEVT2IE**: TYPECEVT2 interrupt enable
0: Disable
1: Enable

Bit 14 **TYPECEVT1IE**: TYPECEVT1 interrupt enable

Bit 13 Reserved, must be kept at reset value.

Bit 12 **RXMSGENDIE**: RXMSGEND interrupt enable
0: Disable
1: Enable

Bit 11 **RXOVRIE**: RXOVR interrupt enable
0: Disable
1: Enable

Bit 10 **RXHRSTDETIE**: RXHRSTDET interrupt enable
0: Disable
1: Enable

Bit 9 **RXORDDETIE**: RXORDDET interrupt enable
0: Disable
1: Enable

Bit 8 **RXNEIE**: RXNE interrupt enable
0: Disable
1: Enable

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TXUNDIE**: TXUND interrupt enable
0: Disable
1: Enable

Bit 5 **HRSTSENTIE**: HRSTSENT interrupt enable
0: Disable
1: Enable

Bit 4 **HRSTDISCIE**: HRSTDISC interrupt enable
0: Disable
1: Enable

Bit 3 **TXMSGABTIE**: TXMSGABT interrupt enable
0: Disable
1: Enable

Bit 2 **TXMSGSENTIE**: TXMSGSENT interrupt enable
0: Disable
1: Enable

Bit 1 **TXMSGDISCIE**: TXMSGDISC interrupt enable
0: Disable
1: Enable

Bit 0 **TXISIE**: TXIS interrupt enable
0: Disable
1: Enable

75.8.5 UCPD status register (UCPD_SR)

Address offset: 0x014

Reset value: 0x0000 0000

The flags (single-bit status bitfields) are associated with interrupt request. Interrupt is generated if enabled by the corresponding bit of the UCPD_IMR register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TYPEC_VSTATE_CC2[1:0]		TYPEC_VSTATE_CC1[1:0]	
												r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TYPEC EVT2	TYPEC EVT1	RXERR	RXMS GEND	RXOV R	RXHRS TDET	RXOR DDET	RXNE	Res.	TXUND	HRSTS ENT	HRSTD ISC	TXMS GABT	TXMS GSENT	TXMS GDISC	TXIS
r	r	r	r	r	r	r	r		r	r	r	r	r	r	r

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:18 **TYPEC_VSTATE_CC2[1:0]**: CC2 line voltage level
The status bitfield indicates the voltage level on the CC2 line in its steady state.
0x0: Lowest
0x1: Low
0x2: High
0x3: Highest
The voltage variation on the CC2 line during USB PD messages due to the BMC PHY modulation does not impact the bitfield value.

Bits 17:16 **TYPEC_VSTATE_CC1[1:0]**:
The status bitfield indicates the voltage level on the CC1 line in its steady state.
0x0: Lowest
0x1: Low
0x2: High
0x3: Highest
The voltage variation on the CC1 line during USB PD messages due to the BMC PHY modulation does not impact the bitfield value.

Bit 15 **TYPECEVT2**: Type-C voltage level event on CC2 line

The flag indicates a change of the TYPEC_VSTATE_CC2[1:0] bitfield value, which corresponds to a new Type-C event. It is cleared by setting the TYPECEVT2CF bit.

0: No new event

1: A new Type-C event

Bit 14 **TYPECEVT1**: Type-C voltage level event on CC1 line

The flag indicates a change of the TYPEC_VSTATE_CC1[1:0] bitfield value, which corresponds to a new Type-C event. It is cleared by setting the TYPECEVT2CF bit.

0: No new event

1: A new Type-C event

Bit 13 **RXERR**: Receive message error

The flag indicates errors of the last Rx message declared (via RXMSGEND), such as incorrect CRC or truncated message (a line becoming static before EOP is met). It is asserted whenever the RXMSGEND flag is set.

0: No error detected

1: Error(s) detected

Bit 12 **RXMSGEND**: Rx message received

The flag indicates whether a message (except Hard Reset message) has been received, regardless the CRC value. The flag is cleared by setting the RXMSGENDCF bit.

0: No new Rx message received

1: A new Rx message received

The RXERR flag set when the RXMSGEND flag goes high indicates errors in the last-received message.

Bit 11 **RXOVR**: Rx data overflow detection

The flag indicates Rx data buffer overflow. It is cleared by setting the RXOVRCF bit.

0: No overflow

1: Overflow

The buffer overflow can occur if the received data are not read fast enough.

Bit 10 **RXHRSTDET**: Rx Hard Reset receipt detection

The flag indicates the receipt of valid Hard Reset message. It is cleared by setting the RXHRSTDETCF bit.

0: Hard Reset not received

1: Hard Reset received

Bit 9 **RXORDDET**: Rx ordered set (4 K-codes) detection

The flag indicates the detection of an ordered set. The relevant information is stored in the RXORDSET[2:0] bitfield of the UCPD_RX_ORDSET register. It is cleared by setting the RXORDDETCF bit.

0: No ordered set detected

1: A new ordered set detected

Bit 8 **RXNE**: Receive data register not empty detection

The flag indicates that the UCPD_RXDR register is not empty. It is automatically cleared upon reading UCPD_RXDR.

0: Rx data register empty

1: Rx data register not empty

Bit 7 Reserved, must be kept at reset value.

Bit 6 **TXUND**: Tx data underrun detection

The flag indicates that the Tx data register (UCPD_TXDR) was not written in time for a transmit message to execute normally. It is cleared by setting the TXUNDCF bit.

0: No Tx data underrun detected

1: Tx data underrun detected

Bit 5 **HRSTSENT**: Hard Reset message sent

The flag indicates that the Hard Reset message is sent. The flag is cleared by setting the HRSTSENTCF bit.

0: No Hard Reset message sent

1: Hard Reset message sent

Bit 4 **HRSTDISC**: Hard Reset discarded

The flag indicates that the Hard Reset message is discarded. The flag is cleared by setting the HRSTDISCCF bit.

0: No Hard Reset discarded

1: Hard Reset discarded

Bit 3 **TXMSGABT**: Transmit message abort

The flag indicates that a Tx message is aborted due to a subsequent Hard Reset message send request taking priority during transmit. It is cleared by setting the TXMSGABTCF bit.

0: No transmit message abort

1: Transmit message abort

Bit 2 **TXMSGSENT**: Message transmission completed

The flag indicates the completion of packet transmission. It is cleared by setting the TXMSGSENTCF bit.

0: No Tx message completed

1: Tx message completed

In the event of a message transmission interrupted by a Hard Reset, the flag is not raised.

Bit 1 **TXMSGDISC**: Message transmission discarded

The flag indicates that a message transmission was dropped. The flag is cleared by setting the TXMSGDISCCF bit.

0: No Tx message discarded

1: Tx message discarded

Transmission of a message can be dropped if there is a concurrent receive in progress or at excessive noise on the line. After a Tx message is discarded, the flag is only raised when the CC line becomes idle.

Bit 0 **TXIS**: Transmit interrupt status

The flag indicates that the UCPD_TXDR register is empty and new data write is required (as the amount of data sent has not reached the payload size defined in the TXPAYSZ bitfield). The flag is cleared with the data write into the UCPD_TXDR register.

0: New Tx data write not required

1: New Tx data write required

75.8.6 UCPD interrupt clear register (UCPD_ICR)

Address offset: 0x018

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is enabled (UCPDEN = 1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TYPECEVT2CF	TYPECEVT1CF	Res.	RXMSGENDCF	RXOVRCF	RXHRSTDETCF	RXORDDETCF	Res.	Res.	TXUNDCF	HRSTSENTCF	HRSTDISCCF	TXMSGABTCF	TXMSGSENTCF	TXMSGDISCCF	Res.
w	w		w	w	w	w			w	w	w	w	w	w	

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **TYPECEVT2CF**: Type-C CC2 line event flag (TYPECEVT2) clear
Setting the bit clears the TYPECEVT2 flag in the UCPD_SR register

Bit 14 **TYPECEVT1CF**: Type-C CC1 event flag (TYPECEVT1) clear
Setting the bit clears the TYPECEVT1 flag in the UCPD_SR register

Bit 13 Reserved, must be kept at reset value.

Bit 12 **RXMSGENDCF**: Rx message received flag (RXMSGEND) clear
Setting the bit clears the RXMSGEND flag in the UCPD_SR register.

Bit 11 **RXOVRCF**: Rx overflow flag (RXOVR) clear
Setting the bit clears the RXOVR flag in the UCPD_SR register.

Bit 10 **RXHRSTDETCF**: Rx Hard Reset detect flag (RXHRSTDET) clear
Setting the bit clears the RXHRSTDET flag in the UCPD_SR register.

Bit 9 **RXORDDETCF**: Rx ordered set detect flag (RXORDDET) clear
Setting the bit clears the RXORDDET flag in the UCPD_SR register.

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **TXUNDCF**: Tx underflow flag (TXUND) clear
Setting the bit clears the TXUND flag in the UCPD_SR register.

Bit 5 **HRSTSENTCF**: Hard reset send flag (HRSTSENT) clear
Setting the bit clears the HRSTSENT flag in the UCPD_SR register.

Bit 4 **HRSTDISCCF**: Hard reset discard flag (HRSTDISC) clear
Setting the bit clears the HRSTDISC flag in the UCPD_SR register.

Bit 3 **TXMSGABTCF**: Tx message abort flag (TXMSGABT) clear
Setting the bit clears the TXMSGABT flag in the UCPD_SR register.

Bit 2 **TXMSGSENTCF**: Tx message send flag (TXMSGSENT) clear
Setting the bit clears the TXMSGSENT flag in the UCPD_SR register.

Bit 1 **TXMSGDISCCF**: Tx message discard flag (TXMSGDISC) clear
Setting the bit clears the TXMSGDISC flag in the UCPD_SR register.

Bit 0 Reserved, must be kept at reset value.

75.8.7 UCPD Tx ordered set type register (UCPD_TX_ORDSETR)

Address offset: 0x01C

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is enabled (UCPDEN = 1) and no packet transmission is in progress (TXSEND and TXHRST bits are both low).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXORDSET[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXORDSET[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **TXORDSET[19:0]**: Ordered set to transmit

The bitfield determines a full 20-bit sequence to transmit, consisting of four K-codes, each of five bits, defining the packet to transmit. The bit 0 (bit 0 of K-code1) is the first, the bit 19 (bit 4 of K-code4) the last.

75.8.8 UCPD Tx payload size register (UCPD_TX_PAYSZR)

Address offset: 0x020

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is enabled (UCPDEN = 1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TXPAYSZ[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:0 **TXPAYSZ[9:0]**: Payload size yet to transmit

The bitfield is modified by software and by hardware. It contains the number of bytes of a payload (including header but excluding CRC) yet to transmit: each time a data byte is written into the UCPD_TXDR register, the bitfield value decrements and the TXIS bit is set, except when the bitfield value reaches zero. The enumerated values are standard payload sizes before the start of transmission.

0x2: 2 bytes - the size of Control message from the protocol layer
0x6: 6 bytes - the shortest Data message allowed from the protocol layer)
0x1E: 30 bytes - the longest non-extended Data message allowed from the protocol layer
0x106: 262 bytes - the longest possible extended message
0x3FF: 1024 bytes - the longest possible payload (for future expansion)

75.8.9 UCPD Tx data register (UCPD_TXDR)

Address offset: 0x024

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is enabled (UCPDEN = 1).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXDATA[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TXDATA[7:0]**: Data byte to transmit

75.8.10 UCPD Rx ordered set register (UCPD_RX_ORDSETR)

Address offset: 0x028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXSOPKINVALID[2:0]			RXSOP 3OF4	RXORDSET[2:0]		
									r	r	r	r	r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 **RXSOPKINVALID[2:0]**:

The bitfield is for debug purposes only.
0x0: No K-code corrupted
0x1: First K-code corrupted
0x2: Second K-code corrupted
0x3: Third K-code corrupted
0x4: Fourth K-code corrupted
Others: Invalid

Bit 3 **RXSOP3OF4**:

The bit indicates the number of correct K-codes. For debug purposes only.
0: 4 correct K-codes out of 4
1: 3 correct K-codes out of 4

Bits 2:0 **RXORDSET[2:0]**: Rx ordered set code detected
0x0: SOP code detected in receiver
0x1: SOP' code detected in receiver
0x2: SOP'' code detected in receiver
0x3: SOP'_Debug detected in receiver
0x4: SOP''_Debug detected in receiver
0x5: Cable Reset detected in receiver
0x6: SOP extension#1 detected in receiver
0x7: SOP extension#2 detected in receiver

75.8.11 UCPD Rx payload size register (UCPD_RX_PAYSZR)

Address offset: 0x02C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	RXPAYSZ[9:0]									
						r	r	r	r	r	r	r	r	r	r

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:0 **RXPAYSZ[9:0]**: Rx payload size received
This bitfield contains the number of bytes of a payload (including header but excluding CRC) received: each time a new data byte is received in the UCPD_RXDR register, the bitfield value increments and the RXMSGEND flag is set (and an interrupt generated if enabled).
0x2: 2 bytes - the size of Control message from the protocol layer
0x6: 6 bytes - the shortest Data message allowed from the protocol layer)
0x1E: 30 bytes - the longest non-extended Data message allowed from the protocol layer
0x106: 262 bytes - the longest possible extended message
0x3FF: 1024 bytes - the longest possible payload (for future expansion)
The bitfield may return a spurious value when a byte reception is ongoing (the RXMSGEND flag is low).

75.8.12 UCPD receive data register (UCPD_RXDR)

Address offset: 0x030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXDATA[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **RXDATA[7:0]**: Data byte received

75.8.13 UCPD Rx ordered set extension register 1 (UCPD_RX_ORDEXTR1)

Address offset: 0x034

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is disabled (UCPDEN = 0).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXSOPX1[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXSOPX1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **RXSOPX1[19:0]**: Ordered set 1 received

The bitfield contains a full 20-bit sequence received, consisting of four K-codes, each of five bits. The bit 0 (bit 0 of K-code1) is receive first, the bit 19 (bit 4 of K-code4) last.

75.8.14 UCPD Rx ordered set extension register 2 (UCPD_RX_ORDEXTR2)

Address offset: 0x038

Reset value: 0x0000 0000

Writing to this register is only effective when the peripheral is disabled (UCPDEN = 0).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXSOPX2[19:16]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXSOPX2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **RXSOPX2[19:0]**: Ordered set 2 received

The bitfield contains a full 20-bit sequence received, consisting of four K-codes, each of five bits. The bit 0 (bit 0 of K-code1) is receive first, the bit 19 (bit 4 of K-code4) last.

75.8.15 UCPD register map

Table 761. UCPD register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	UCPD_CFGR1	UCPDEN	RXDMAEN	TXDMAEN	RXORDSETEN[8:0]									PSC_UCPDCLK[2:0]			Res.	TRANSWIN[4:0]					IFRGAP[4:0]					HBITCLKDIV[5:0]						
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	UCPD_CFGR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WUPEN	FORCECLK	RXFILT2N3	RXFILTDIS
	Reset value																													0	0	0	0
0x008	Reserved																																
0x00C	UCPD_CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CC2TCDIS	CC1TCDIS	Res.	RDCH	FRSTX	Res.	Res.	Res.	Res.	Res.	CCENABLE[1:0]		ANAMODE	ANASUBMODE[1:0]		PHYCCSEL	PHYRXEN	RXMODE	TXHRST	TXSEND	TXMODE[1:0]	
	Reset value											0	0		0	0						0	0	0	0	0	0	0	0	0	0	0	0
0x010	UCPD_IMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TYPECEVT2IE	TYPECEVT1IE	Res.	RXMSGENDIE	RXOVRIE	RXHRSTDETIE	RXORDDETIE	RXNEIE	Res.	TXUNDIE	HRSTSENTIE	HRSTDISCIE	TXMSGABTIE	TXMSGSENTIE	TXMSGDISCIE	TXISIE
	Reset value																	0	0		0	0	0	0	0		0	0	0	0	0	0	0
0x014	UCPD_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TYPEC_VSTATE_CC2[1:0]		TYPEC_VSTATE_CC1[1:0]		TYPECEVT2	TYPECEVT1	RXERR	RXMSGEND	RXOVR	RXHRSTDET	RXORDDET	RXNE	Res.	TXUND	HRSTSENT	HRSTDISC	TXMSGABT	TXMSGSENT	TXMSGDISC	TXIS
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x018	UCPD_ICR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TYPECEVT2CF	TYPECEVT1CF	Res.	RXMSGENDCF	RXOVRCF	RXHRSTDETCF	RXORDDETCF	Res.	Res.	TXUNDCF	HRSTSENTCF	HRSTDISCCF	TXMSGABTCF	TXMSGSENTCF	TXMSGDISCCF	Res.
	Reset value																	0	0		0	0	0	0			0	0	0	0	0	0	
0x01C	UCPD_TX_ORDSETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXORDSET[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 761. UCPD register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x020	UCPD_TX_PAYSZR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXPAYSZ[9:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x024	UCPD_TXDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXDATA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x028	UCPD_RX_ORDSETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXSOPKINVALID[2:0]			RXSOP3OF4	RXORDSET[2:0]		
	Reset value																										0	0	0	0	0	0	0
0x02C	UCPD_RX_PAYSZR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXPAYSZ[9:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x030	UCPD_RXDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXDATA[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x034	UCPD_RX_ORDEXTR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXSOPX1[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038	UCPD_RX_ORDEXTR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXSOPX2[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x03C - 0x3FF	Reserved																																

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

76 Ethernet (ETH): gigabit media access control (GMAC) with DMA controller

76.1 Ethernet introduction

Portions Copyright (c) Synopsys, Inc. All rights reserved. Used with permission.

The Ethernet peripheral enables to transmit and receive data over Ethernet in compliance with the IEEE 802.3-2015 standard.

The peripheral is configurable to meet the needs of a large variety of consumer and industrial applications, including AV nodes and TSN (time sensitive networking) nodes.

76.2 Ethernet main features

The Ethernet peripheral embeds a dedicated DMA for direct memory interface, a media access controller (MAC) and a PHY interface block supporting several formats.

76.2.1 Standard compliance

The Ethernet peripheral is compliant with the following standards:

- IEEE 802.3-2015 for Ethernet MAC and media independent interface (MII)
- IEEE 1588-2008 for precision networked clock synchronization (PTP)
- IEEE 802.1AS-2011 and 802.1-Qav-2009 for Audio Video (AV) traffic
- IEEE 802.3az-2010 for Energy Efficient Ethernet (EEE)
- IEEE 802.1Qbv-2015, 802.1Qbu-2016, and 802.1AS-Rev D5.0 for Time-Sensitive Networking (TSN) traffic
- AMBA 2.0 for AHB slave port
- AMBA4 for AXI master port
- RGMII specification version 2.6 from HP/Marvell
- RMII specification version 1.2 from RMII consortium

Caution: The gigabit media independent interface (GMII) is only available internally to supply the RGMII adapter. No GMII signals are available off-chip.

76.2.2 MAC features

MAC Tx and Rx common features

- Separate transmission, reception, and control interfaces to the application
- 10, 100, and 1000 Mbit/s data transfer rates with the following PHY interfaces:
  - IEEE 802.3-compliant MII interface to communicate with an external Fast Ethernet PHY
  - RGMII interface to communicate with an external gigabit PHY
  - RMII interface to communicate with an external Fast Ethernet PHY

- Half-duplex operation:
  - CSMA/CD protocol support
  - Flow control using back pressure (based on implementation-specific white papers and UNH Ethernet Clause 4 MAC Test Suite - Annex D)
  - Packet bursting and packet extension in 1000 Mbps Half-duplex operation
- Standard IEEE 802.3az-2010 for Energy Efficient Ethernet in MII and reduced gigabit media independent interface (RGMII) PHYs
- 64-bit data transfer interface on the application side
- Full-duplex flow control operations (IEEE 802.3x Pause packets and Priority flow control)
- Network statistics with RMON or MIB counters (partial support of RFC2819/RFC2665)
- Ethernet packet timestamping as described in IEEE 1588-2002 and IEEE 1588-2008 (64-bit timestamps given in the Tx or Rx status of PTP packet). Both one-step and two-step timestamping are supported in Tx direction.
- Flexibility to control up to two pulse-per-second (PPS) output signals (eth_ptp_pps_outx and ETH_PPS_OUT)
- MDIO (Clause 22 and Clause 45) master interface for PHY device configuration and management

MAC Tx features

- Preamble and start-of-frame data (SFD) insertion
- Separate 32-bit status for each packet transmitted from the application
- Automatic CRC and pad generation controllable on a per-frame basis
- Programmable packet length to support Standard or Jumbo Ethernet packets of up to 16 Kbytes
- Programmable Inter Packet Gap (40–96 bit times in steps of 8)
- IEEE 802.3x Flow Control automatic transmission of zero-quanta Pause packet when flow control input transitions from assertion to de-assertion (in Full-duplex mode)
- Source address field insertion or replacement, and VLAN insertion, replacement, and deletion in transmitted packets with per-packet or static-global control
- Insertion, replacement, or deletion of up to two VLAN tags
- Option to transmit packets with reduced preamble size in Full-duplex mode
- Insert, replace, or delete queue/channel-based VLAN tags
- Frame preemption for MAC Tx

MAC Rx features

- Automatic Pad and CRC stripping options
- Option to disable automatic CRC checking
- Preamble and SFD deletion
- Separate 112-bit or 128-bit status
- Programmable watchdog timeout limit
- Flexible address filtering modes:
  - Four 48-bit perfect (DA) address filters with masks for each byte
  - Four 48-bit SA address comparison check with masks for each byte
  - 64 bit Hash filter for multicast and unicast (DA) addresses

- Option to pass all multicast addressed packets
- Promiscuous mode to pass all packets without any filtering for network monitoring
- Pass all incoming packets (as per filter) with a status report
- Additional packet filtering:
  - VLAN tag-based: Perfect match and Hash-based filtering based either on the outer or inner VLAN tag
  - Layer 3 and Layer 4-based: TCP or UDP over IPv4 or IPv6
  - Extended VLAN tag based filtering with four filter selection
- IEEE 802.1Q VLAN tag detection and option to delete the VLAN tags in received packets
- Detection of remote wake-up packets and AMD magic packets
- Optional forwarding of received Pause packets to the application (in Full-duplex mode)
- Layer 3/Layer 4 checksum offload for received packets
- Stripping of up to two VLAN tags and providing the tags in the status
- Frame preemption for MAC Rx

76.2.3 Transaction layer (MTL) features

MTL Tx and Rx common features

- 64-bit transaction layer block (bridges the application and the MAC)
- Optimization for packet-oriented transfers with packets delimiters
- Programmable burst length, up to half the size of the MTL Rx queue or Tx queue size, to support burst data transfer in the EQOS-MTL configuration
- Programmable threshold capability for each queue (default of 64 bytes)

MTL Tx features

- 4096-byte transmit FIFO with programmable threshold capability
- Two Tx queues on the transmit path with a common memory for all Tx queues
- Store-and-forward mechanism or threshold mode (cut-through) for transmission to the MAC
- Programmable queue size in configurations with multiple queues. Each queue size can be programmed in terms of 256 bytes
- Automatic retransmission of collision packets in Half-duplex mode
- Discard packets on late collision, excessive collisions, excessive deferral, and under-run conditions with appropriate status
- Module to calculate and insert IPv4 header checksum and TCP, UDP, or ICMP checksum on frames transmitted in Store-and-forward mode
- Statistics by generating pulses for packets dropped (because of underflow) in the Tx FIFO
- Optional statistics related to bandwidth consumption for each queue of up to 16 blocks over a 125 µs period
- Packet-level control for
  - VLAN tag insertion or replacement
  - Ethernet source address insertion

  - Layer 3/Layer 4 checksum insertion control
  - One-step timestamp
  - Timestamp control
  - CRC and pad control
- Following scheduling algorithms in configurations with multiple queues
  - Weighted round robin (WRR)
  - Strict priority (SP)
  - Credit-based shaper (CBS)
  - Enhancement to scheduled traffic (EST)
  - Time-based scheduling (TBS)

MTL Rx features

- 4096-byte receive FIFO with configurable threshold
- Two Rx queues on the receive path with a common memory for all Rx queues
- Programmable Rx queue threshold (default fixed at 64 bytes) in threshold (or cut-through) mode
- Option to filter all error packets on reception and not forward them to the application in the store-and-forward mode
- Option to forward the undersized good packets
- Statistics by generating pulses for packets dropped (because of overflow) in the Rx FIFO
- Automatic generation of Pause packet control or backpressure signal to the MAC based on the Rx Queue fill level
- Arbitration among queues when multiple queues are present. The following arbitration schemes are supported:
  - Weighted round robin (WRR)
  - Weighted Strict priority (WSP)
  - Strict priority (SP)

76.2.4 DMA block features

The DMA block exchanges data between the peripheral and the system memory. DMA transfers are driven by software descriptors structure. The application can use a set of registers (see *Section 76.11.2: Ethernet DMA registers*) to control the DMA operations. The DMA block supports the following features:

- 64-bit data transfers
- Separate DMA channel in the transmit path for each queue
- Separate DMA channel in the receive path for each queue
- Optimization for packet-oriented DMA transfers with packet delimiters
- Byte-aligned addressing for data buffer support
- Dual-buffer (ring) descriptor support
- Descriptor architecture allowing large blocks of data transfer with minimum CPU intervention (each descriptor can transfer up to 32 Kbytes of data)
- Comprehensive status reporting normal operation and transfer errors

- Individual programmable burst length for Tx DMA and Rx DMA engines for optimal host bus utilization
- Programmable interrupt options for different operational conditions
- Per-packet transmit or receive complete interrupt control
- Round-robin or fixed-priority arbitration between the receive and transmit engines
- Fixed-priority, weighted-strict-priority or weighted-round-robin arbitration for data read requests from multiple transmit DMA engines
- Start and Stop modes
- Separate ports for host control access and host data interface (AXI)
- Tx DMA channels with TCP segmentation offload (TSO) feature enabled
- Routing of received packets to the DMA channels based on the DA or VLAN priority in multichannel DMA configurations
- Time-sensitive conditional packet transmission by comparing the slot time or IEEE 1588 time information provided in the descriptor (useful for AV applications)
- Programmable control for transmit descriptor posted writes to improve the throughput

76.2.5 Bus interface features

AXI master interface

The AXI master interface features are the following:

- Interfaces with the application through AXI4 compatible interface
- 32-bit address width
- 64-bit data
- OKAY, SLVERR, and DECERR responses
- Software-selected type of AXI burst (fixed and variable length burst) in AXI master interface; extended length fixed bursts of 32, 64, 128, and 256
- Handshaking on AXI read and write data channels

AHB slave interface

The AHB slave interface supports the following features:

- Interfaces with the application through AHB
- AHB slave interface (32-bit) for CSR access
- All AHB burst types

The AHB slave interface does not generate the following responses:

- Split
- Retry

76.2.6 Audio and video features

The Ethernet peripheral can be used in Audio Video (AV) mode. The supported features are compliant with the industry standards for AV traffic:

- Separate channels or queues for AV data transfer in 100 Mbps and 1000 Mbps modes
- Multiple Rx queues on the receive paths for AV traffic, and multiple Tx queues on the Transmit path for AV traffic
- IEEE 802.1-Qav specified credit-based shaper (CBS) algorithm for Transmit channels
- Single Tx FIFO and Rx FIFO (MTL) for all selected queues
- Programmable Slot Interval with range from 1 µs to 4096 µs and granularity of 1 µs.

76.2.7 Time sensitive networking features

The Ethernet peripheral supports the following time sensitive networking (TSN) features:

- IEEE 802.1Qbv-2015, Enhancements to Scheduling Traffic
- IEEE802.1Qbu/802.3br, Frame preemption and Interspersing Express Traffic

76.2.8 Generic queuing features

The Ethernet peripheral supports the following features for generic queuing:

- Programmable control for routing receive packets with Multicast/Broadcast destination address to a programmable receive queue
- Support routing of untagged receive packets to a programmable receive queue
- Programmable control for routing VLAN tagged and untagged IEEE 1588 PTP over Ethernet receive packets to a same or separate programmable receive queue
- Programmable control for routing Unicast/Multicast receive packets that fail the destination address filter to a separate programmable receive queues

76.3 Ethernet pins and internal signals

Table 762 lists the Ethernet inputs and output signals connected to package pins or balls. Active pins depend on the PHY type selected (MII, RMII or RGMII) and on the device configuration.

Table 763 shows the internal Ethernet signals.

Table 762. Ethernet peripheral pins

Port name	Digital port type	Description
ETH_COL	Input	Collision detection signal, MII only.
ETH_CRS	Input	Carrier sense signal, MII only
ETH_REF_CLK	Input	RMII reference clock
ETH_RX_CLK	Input	MII/RGMII timing reference for Rx data transfers
ETH_RXD[3:0]	Input	Receive data. 4 pins for MII and RGMII, 2 for RMII.
ETH_RX_DV	Input	Receive data valid

Table 762. Ethernet peripheral pins (continued)

Port name	Digital port type	Description
ETH_CRS_DV	Input	RMII: Carrier Sense (CRS) and RX_Data Valid (RX_DV) multiplexed on alternate clock cycles. In 10 Mbit/s mode, it alternates every 10 clock cycles.
ETH_RX_CTL	Input	After demultiplexing, this pin provides ETH_RX_DV on the rising edge of ETH_RX_CLK, and ETH_RX_ER (logical XOR between ETH_RX_CTL values on rising edge and falling edge of ETH_RX_CLK).
ETH_RX_ER	Input	Receive error
ETH_TX_CLK	Input	MII timing reference for Tx data transfers
ETH_GTX_CLK	Output	RGMII timing reference for Tx data transfers
ETH_TXD[3:0]	Output	Transmit data. 4 pins for MII and RGMII, 2 for RMII.
ETH_TX_EN	Output	Transmit data enable
ETH_TX_CTL	Output	Multiplexing of ETH_TX_EN on the rising edge of ETH_TX_CLK, and a logical XOR between ETH_TX_EN and ETH_TX_ER values on falling edge of ETH_TX_CLK
ETH_TX_ER	Output	Transmit error
ETH_MDC	Output	Management data clock
ETH_MDIO	Input/output	Management data
ETH_PHY_INTN	Input	PHY interrupt
ETH_PPS_OUT	Output	PTP pulse-per-second output 0
ETH_PTP_AUX_TS(1)	Input	Trigger input for the auxiliary snapshots of the PTP system time

1. This pin is not available on all devices.

Table 763. Ethernet internal input/output signals

Signal name	Signal type	Description
eth_hclk	Digital input	AHB clock
eth_aclk	Digital input	AXI clock
eth_sbd_intr_it	Digital output	Main Ethernet interrupt
lpi_intr_o	Digital output	Sideband signal generated when the transmitter or receiver enters or exits the LPI state.
pmt_intr_o	Digital output	Sideband signal generated when a valid remote wake-up packet is received
eth_tx_clk	Digital input	Tx kernel clock
eth_rx_clk	Digital input	Rx kernel clock
eth_rmii_ref_clk	Digital input	RMII reference kernel clock
eth_ptp_pps_out1	Digital output	PTP pulse-per-second signal
mac_speed_o[1:0]	Digital output	MAC speed information used by the RCC

Table 763. Ethernet internal input/output signals (continued)

Signal name	Signal type	Description
clk_ptp_ref_i	Digital input	PTP reference clock input
eth_ptp_trig[3:0]	Digital input	Trigger input for auxiliary snapshots of the PTP system time

76.4 Ethernet architecture

The Ethernet peripheral is composed of 4 main functional modules:

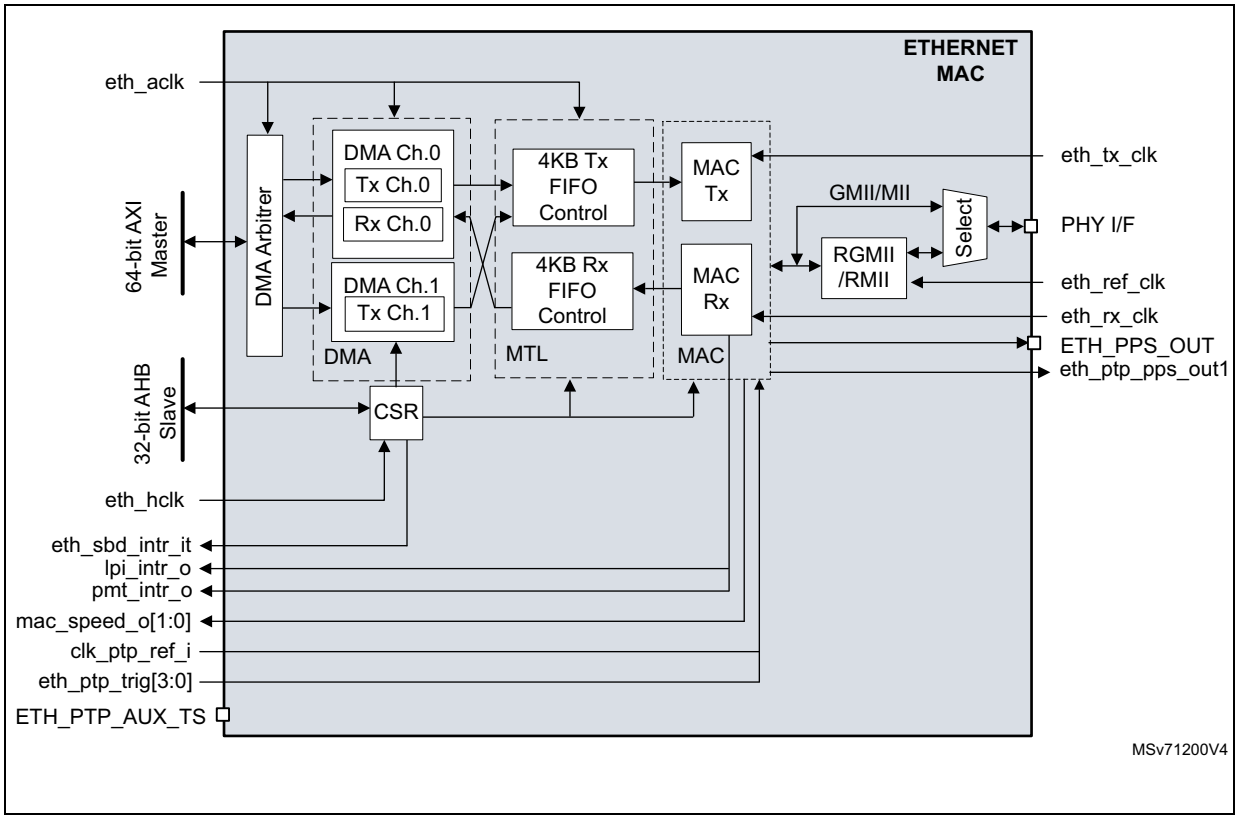
- **The control and status register module (CSR)** that controls the registers access through AHB 32-bit slave interface
- The direct memory access interface (DMA)

  This is the logical DMA module with two physical channel for reception and two for transmission. It controls the data transfers between MAC and system memory through the AMBA AXI4 64-bit master interface.
- **The media access control module (MAC)** in charge of implementing the Ethernet protocol
- **The MAC transaction layer (MTL)** in charge of controlling the data flow between application and MAC.

A protocol adaption module is added to support the RMII or the RGMII PHY Media Independent Interfaces.

Caution: The gigabit media independent interface (GMII) is only available internally to supply the RGMII adapter. No GMII signals are available off-chip.

Figure 1019. Ethernet high-level block diagram



1. For a definition of the internal signals, refer to *Table 763*.
2. Refer to RCC chapter “Clock distribution for Ethernet” for a detailed description of the Ethernet clock architecture.
3. The ETH_PTP_AUX_TS pin is not available on all devices.

76.4.1 DMA controller

The DMA has independent transmit (Tx) and receive (Rx) engines. The Tx engine transfers data from the system memory to the MAC Transaction Layer (MTL), whereas the Rx engine transfers data from the device port (PHY) to the system memory.

The controller uses descriptors to efficiently move data from source to destination with minimal application CPU intervention. The DMA is designed for packet-oriented data transfers such as packets in Ethernet. The controller can be programmed to interrupt the application CPU for situations such as Packet transmit and receive Transfer completion, and other normal or error conditions.

DMA data structures

The DMA and the application communicate through the following two data structures:

- Control and Status registers (CSR)
- Descriptor lists and data buffers

The DMA transfers the data packets received by the MAC to the Rx buffer in system memory and Tx data packets from the Tx buffer in the system memory. The descriptors that reside in the system memory contain the pointers to these buffers.

The base address of each list is written to the respective Tx and Rx registers: *Channel x Tx descriptor list address register (ETH_DMACxTXDLAR)*) and *Channel x Rx descriptor list address register (ETH_DMACxRXDLAR)*.

The descriptor list is forward linked and the next descriptor is always considered at a fixed offset to the current one. The number of descriptors in the list is programmed in the respective Tx/Rx, *Channel x Tx descriptor ring length register (ETH_DMACxTXRLR)* and *Channel x Rx descriptor ring length register (ETH_DMACxRXRLR)*.

Once the DMA processes the last descriptor in the list, it automatically jumps back to the descriptor in the List address register to create a descriptor ring. The descriptor lists reside in the physical memory address space of the application. Each descriptor can point to a maximum of two buffers. This enables two buffers to be used and physically addressed, rather than contiguous buffers in memory.

A data buffer resides in the application physical memory space and consists of an entire packet or part of a packet, but cannot exceed a single packet. Buffers contain only data. The buffer status is saved in the descriptor. Data chaining refers to packets that span multiple data buffers. However, a single descriptor cannot span multiple packets. The DMA skips to the data buffer of next packet when EOP is detected.

Descriptors are specified in *Section 76.10: Descriptors*.

DMA transmission in default mode

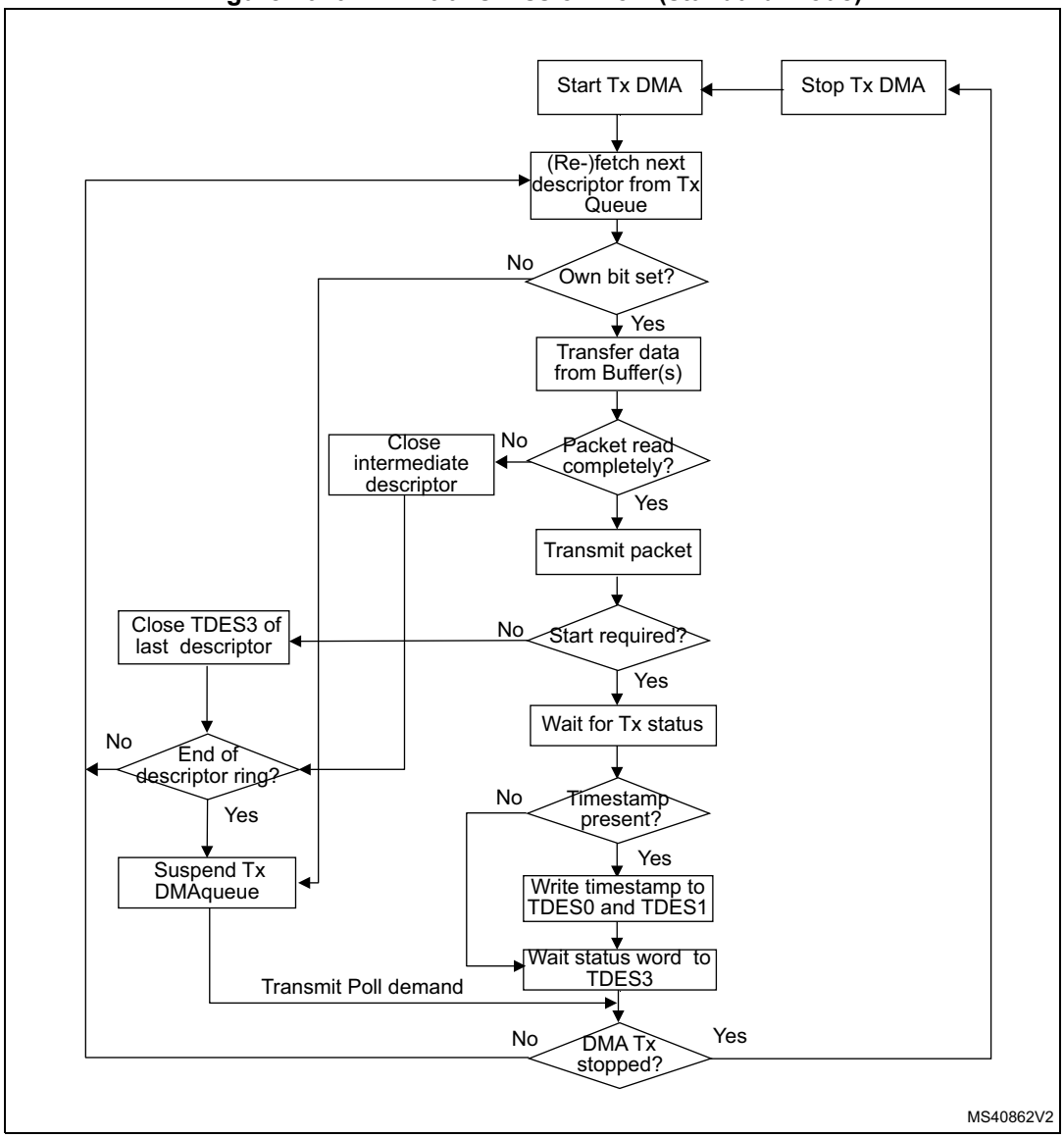
The Tx DMA engine in default mode proceeds as follows:

1. The application sets up the Transmit descriptor (TDES0–TDES3) and sets the Own bit (TDES0[31]) after setting up the corresponding data buffer(s) with Ethernet Packet data.
2. The application shifts the descriptor tail pointer offset value of the Transmit channel.
3. While in the Run state, the DMA runs an arbitration cycle to select the next Tx DMA channel from which the packets requiring transmission should be processed.
4. The DMA fetches the descriptor from the application memory.
5. If the DMA detects one of the following conditions, the transmission from that channel is suspended, bit 2 and 16 of the corresponding DMA channel Status register are set, and the Tx engine proceeds to step 11:
   - The descriptor is flagged as owned by the application (TDES3 [31] = 0).
   - The descriptor tail pointer is equal to the current descriptor pointer in Ring Descriptor list mode.
   - An error condition occurs.
6. If the acquired descriptor is flagged as owned by the DMA (TDES3[31] = 1), the DMA decodes the Transmit Data Buffer address from the acquired descriptor.
7. The DMA fetches the Transmit data from the system memory and transfers the data to the MTL for transmission.
8. If an Ethernet packet is stored over data buffers in multiple descriptors, the DMA closes the intermediate descriptor and fetches the next descriptor. Steps 3 through 7 are repeated until the end-of-Ethernet-packet data is transferred to the MTL.
9. When packet transmission is complete, if IEEE 1588 timestamp feature was enabled for the packet (as indicated in the Tx status), the timestamp value obtained from MTL is written to the Tx descriptor (TDES0 and TDES1) that contains the EOP buffer. The status information is written to this Tx descriptor (TDES3). The application now owns

this descriptor because the Own bit is cleared during this step. If the timestamp feature is disabled for this packet, the DMA does not alter TDES0 and TDES1 contents.

10. Bit 0 of *Channel x status register (ETH_DMACxSR)*) is set after completing transmission of a packet that has Interrupt on Completion (TDES2[31]) set in its Last Descriptor. The DMA engine returns to step 3.
11. In the Suspend state, the DMA tries to acquire the descriptor again (and thereby return to step 3). A poll demand command is triggered by writing any value to the *Channel x Tx descriptor tail pointer register (ETH_DMACxTXDTPR)* when it receives a Transmit Poll demand and the underflow interrupt status bit is cleared. If the application stopped the DMA by clearing Bit 0 of Transmit control register of corresponding DMA channel, the DMA enters the Stop state.

Figure 1020. DMA transmission flow (standard mode)



DMA transmission in OSP (operate on second packet) mode

In Run state, if bit 4 is set in the *Channel x transmit control register (ETH_DMACxTXCR)*), the Transmit process can simultaneously acquire two packets without closing the Status descriptor of the first packet. While the Transmit process completes the first packet transfer, it immediately polls the Transmit descriptor list for the second packet. If the second packet is valid, the Transmit process transfers this packet before writing the status information of the first packet.

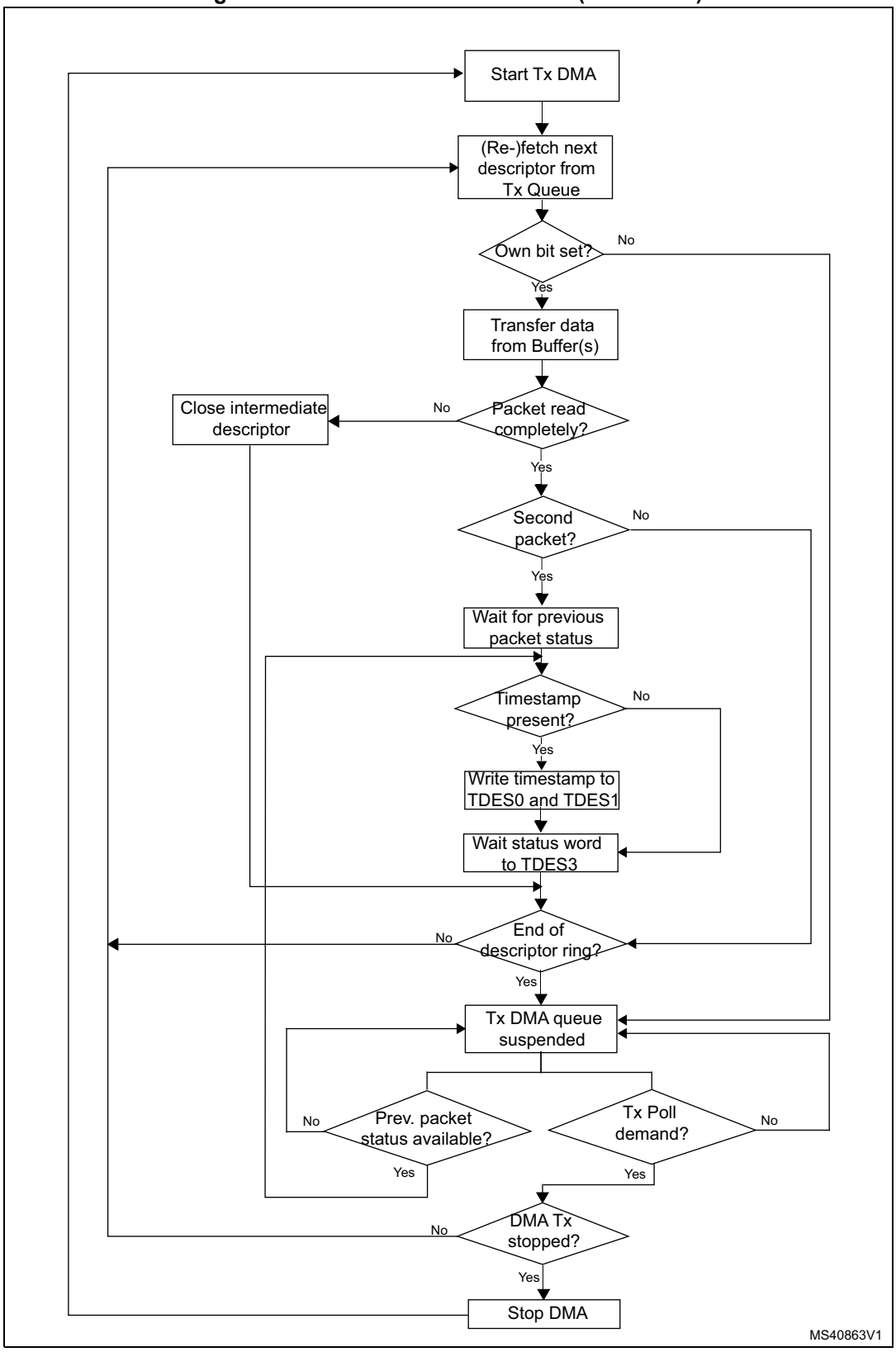
In OSP mode, DMA transmission in the Run state operates as described in the following sequence:

1. The DMA executes steps 1 to 7 of the DMA transmission sequence in default mode (see *Section : DMA transmission in default mode*).
2. The DMA fetches the next descriptor without closing previous packet last descriptor.
3. If the DMA owns the acquired descriptor, the DMA decodes the transmit buffer address in this descriptor. If the DMA does not own the descriptor, the DMA goes into Suspend mode and jumps to step 7.
4. The DMA fetches the Transmit packet from the system memory and transfers the packet to the MTL until the EOP data is transferred, closing the intermediate descriptors if this packet is split across multiple descriptors.
5. The DMA waits for the packet transmission status and timestamp of previous packet. When the status is available, the DMA writes the timestamp to TDES0 and TDES1 if such timestamp was captured (as indicated by a status bit). The DMA writes the status, with a cleared Own bit, to the corresponding TDES3, thus closing the descriptor. If Timestamp feature is not enabled for the previous packet, the DMA does not alter the contents of TDES2 and TDES3.
6. The Transmit interrupt is set (if enabled). The DMA fetches the next descriptor and proceeds to step 3 (when Status is normal). If the previous transmission status shows an underflow error, the DMA goes into Suspend mode (step 7).
7. In Suspend mode, if a pending status and timestamp are received from the MTL, the DMA performs the following operations:
   a) The DMA writes the timestamp (if enabled for the current packet) to TDES2 and TDES3.
   a) The DMA writes the status to the corresponding TDES3.
   a) The DMA sets the relevant interrupts and returns to Suspend mode.

   If no status is pending and the application stopped the DMA by clearing bit 0 of Transmit Control Register of corresponding DMA channel, the DMA enters the Stop state.
8. The DMA can exit Suspend mode and enter the Run state (it goes either to step 1 or to step 2 depending on pending status) only after receiving a Transmit Poll demand in the transmit descriptor tail pointer register of corresponding channel.

A description of the basic DMA transmission flow in OSP mode in given in *Figure 1022: Receive DMA flow*.

Figure 1021. DMA transmission flow (OSP mode)



DMA reception

In the receive path, the DMA reads a packet from the MTL receive queue and writes it to the packet data buffers of the corresponding DMA channel.

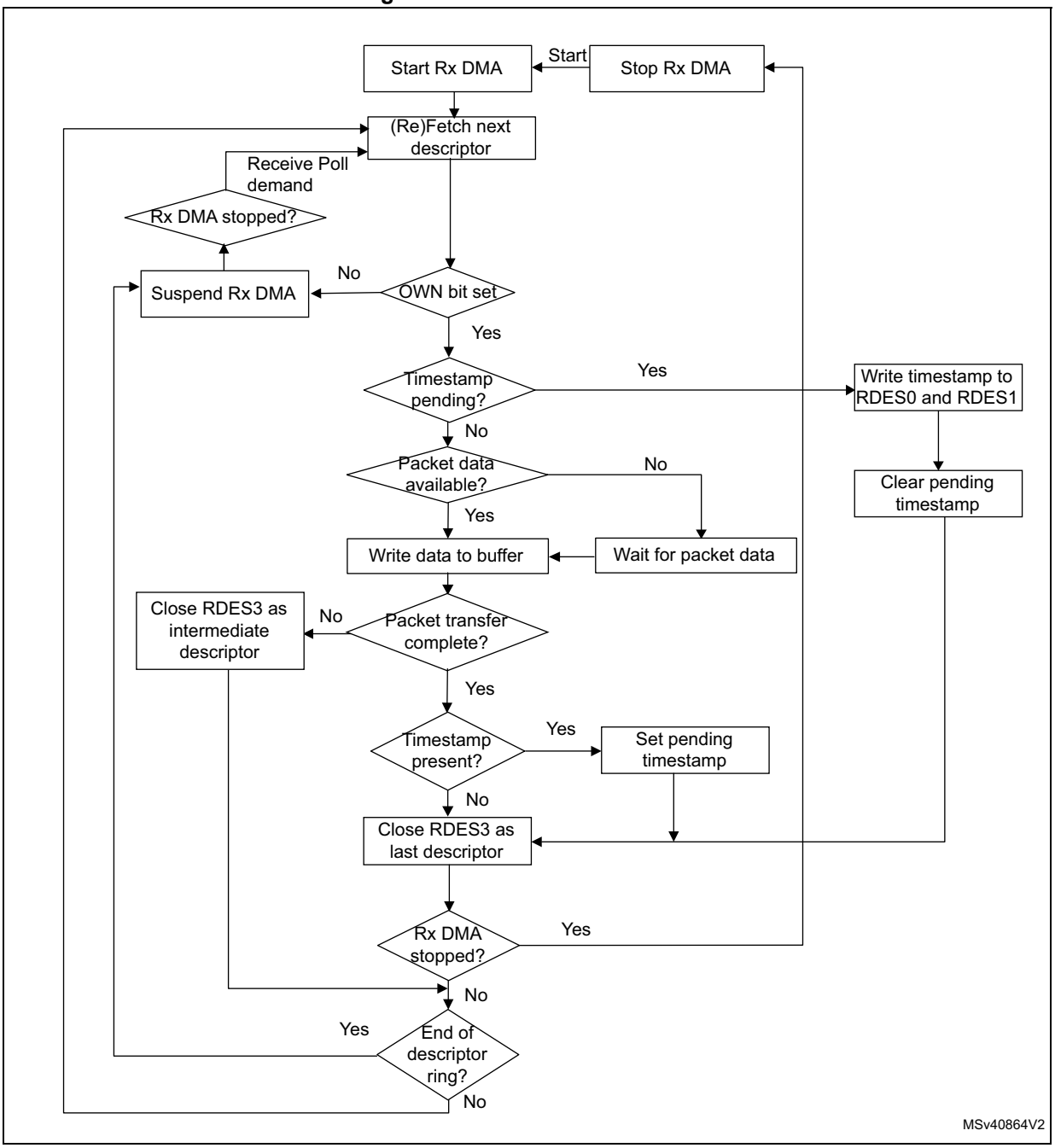
The DMA Rx descriptor ring structure is described in *Section 76.10: Descriptors*.

The reception sequence for Rx DMA engine is as follows (see also *Figure 1022: Receive DMA flow*):

1. The application sets up the Rx descriptors (RDES0-RDES3) and the Own bit (RDES3[31]). The application should set the correct value in the receive descriptor tail pointer register of corresponding DMA channel to indicate the location of the last valid descriptor for the DMA. If the tail pointer points to descriptor N, the last valid descriptor for the DMA is descriptor N - 1.
2. When bit 0 of *Channel x receive control register (ETH_DMACxRXCR)*) is set, the DMA enters the Run state. The DMA looks for free descriptors based on the Rx Current Descriptor and Descriptor tail pointer register values. The descriptors referenced between the current descriptor and the tail pointer registers are available for the DMA. If there are no free descriptors, the DMA channel enters the Suspend state and goes to step 11.
3. The DMA fetches the next available descriptor in the ring and decodes the receive data buffer address from acquired descriptors.
4. If IEEE 1588 timestamping is enabled and the timestamp is available for the previous packet, the DMA writes the timestamp (if available) to the RDES0 and RDES1 of current descriptor and sets the CTXT field (RDES3[30]).
5. The DMA processes the incoming packets and stores them in the data buffers of acquired descriptor.
6. If the current packet transfer is not complete, the DMA closes the current descriptor as intermediate and goes to step 10.
7. The DMA retrieves the status of the receive frame from the MTL and writes the status word to current descriptor with the Own bit cleared and the Last descriptor bit set.
8. The DMA writes the Frame Length to RDES3 and the VLAN tag to RDES0. The DMA also writes the MAC control frame opcode, OAM control frame code, and extended status information (if available) to RDES1 of the last descriptor.
9. The DMA stores the timestamp (if available). The DMA writes the context descriptor after the last descriptor for the current packet (in the next available descriptor).
10. If more descriptors are available in the Rx DMA descriptor ring, go to step 3, otherwise go to the Suspend state (step 11).
11. The receive DMA exits the Suspend state when a receive Poll demand is given, and the application updates the channel receive descriptor tail pointer register to indicate the location of the last valid descriptor for DMA. Then, the engine proceeds to step 2 and fetches again the next descriptor.

Note: Refer to Section : Descriptor tail pointer handling examples for updating the correct value in receive descriptor tail pointer register.

Figure 1022. Receive DMA flow



Priority scheme for Tx DMA and Rx DMA

The DMA arbiter performs the arbitration between the Tx and Rx paths of DMA channel 0 to access descriptors and data buffers.

If the Tx DMA and Rx DMA of a given channel are enabled, the DMA which gets the bus when the channel gets control of the bus must be specified. The priority between the corresponding Tx DMA and Rx DMA can be configured through the TXPR field of the *DMA mode register (ETH_DMAMR)*.

76.4.2 MTL

The MAC Transaction Layer (MTL) provides the FIFO memory interface to buffer and regulate the packets between the application system memory and the MAC. It also enables the data to be transferred between the application clock and MAC clock domains. The MTL layer features two 64-bit wide data paths: the Transmit path and the receive path.

- Transmit path

  The application or internal DMA pushes the Ethernet packets read from the application or system memory into the corresponding queue in Tx FIFO. The packet is then popped out and transferred to the MAC when the queue threshold is reached (threshold mode) or complete packet is in the queue (store-and-forward mode). When EOP is transferred, the status of the transmission is taken from the MAC and transferred back to the application or internal DMA. The Tx queue size is 4096 bytes.

- Receive path

  The MTL Rx module receives the packets from the MAC and pushes them into the Rx queue. The status (fill level) of the queue is indicated to the application or to DMA when it crosses the configured receive threshold (RTC bits[1:0] defined in the operating mode register of the corresponding queue), or when the complete packet was received. The MTL also indicates the queue fill level so that the DMA can initiate preconfigured burst transfers towards the master interface. The Rx queue size is 4096 bytes.

76.4.3 MAC

The MAC is responsible of the Ethernet protocol processing.In Transmission mode, it receives data from MTL before transferring it to the PHY interface. In Reception mode, the MAC receives data from the PHY interface before transferring them to the Rx FIFO of the MTL module.

This section briefly describes transmission and reception sequences.

MAC transmission

The transmission sequence is as follows:

1. Transmission is initiated when the MTL application pushes in data with the SOP (Start of packet) signal asserted.
2. When the SOP signal is detected, the MAC accepts the data and begins the transmission to the GMII or MII.
3. When the EOP (End of packet) is transferred to the MAC, the MAC does one of the following:
   - The MAC completes the normal transmission and provides the transmission status to the MTL.
   - If a normal collision (in Half-duplex mode) occurs during transmission, the MAC provides the Transmit status to the MTL, with the Retry bit set. The MAC provides the Retry request till one of the following is true:

     the packet was successfully transmitted;

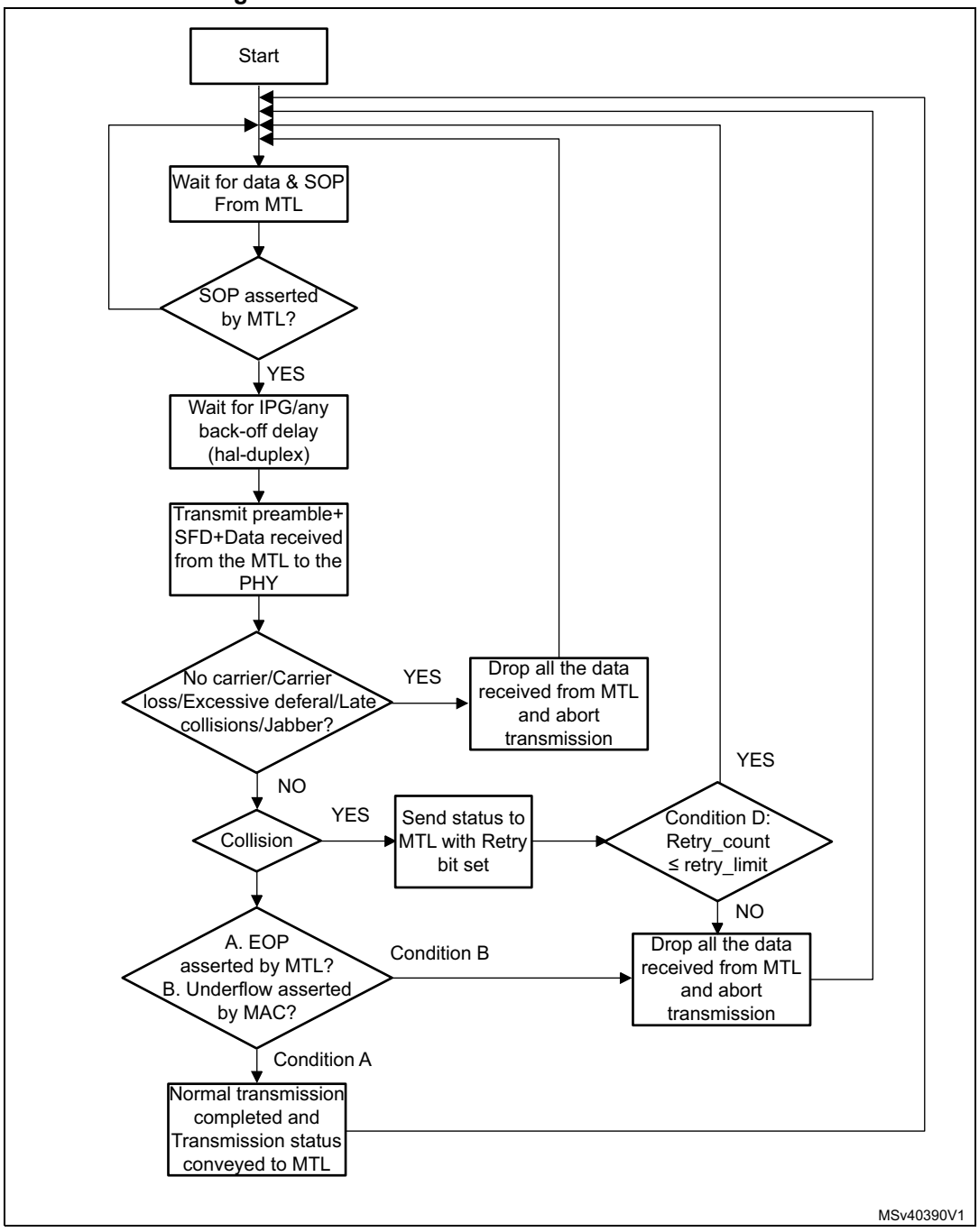
     the maximum number of Retry requests expires. In this case, the MAC aborts the packet transmission with Excessive Collision Transmit status. The MAC accepts and drops all further data until the next SOP is received. The MTL block should retransmit the same packet from SOP when a Retry request (in the Status) is observed from the MAC.

– If any one of the following event happens, the MAC aborts the packet transmission:
  no carrier (Half-duplex mode)
  loss of carrier (Half-duplex mode)
  excessive deferral (Half-duplex mode)
  late collisions (Half-duplex mode)
  jabber
  the MAC accepts and drops all further data until the next SOP is received.

4. The MAC issues an underflow status if the MTL is not able to provide the data continuously during the transmission. The MAC accepts and drops all further data until the next SOP is received.
5. During the normal transfer of a packet from MTL, if the MAC receives a SOP without getting an EOP for the previous packet, it ignores the SOP and considers the new packet as continuation of the previous packet.

Figure 1023: Overview of MAC transmission flow illustrates the MAC transmission process flow.

Figure 1023. Overview of MAC transmission flow



MAC reception

A receive operation is initiated when the MAC detects an SFD on GMII. The MAC strips the preamble and SFD before proceeding to process the packet. The header fields are checked for filtering and the FCS field used to verify the CRC for the packet. The received packet is stored in a shallow buffer until the address filtering is performed. The packet is dropped in the MAC if it fails the address filter.

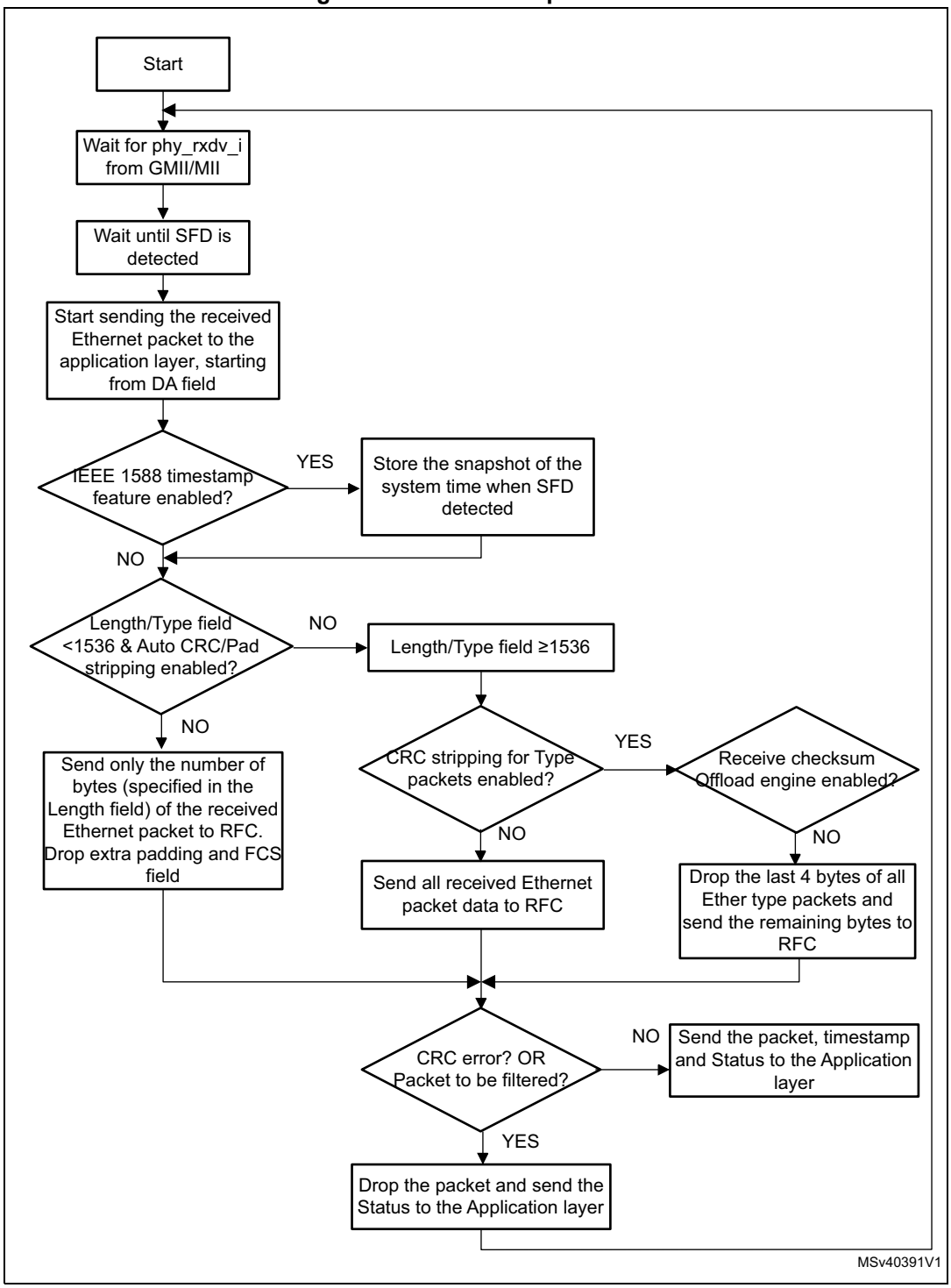
The reception sequence is as follows:

1. When the receive data valid signal (RxDV) of GMII or MII becomes active, the receive State Machine (RSM) starts looking for the SFD field (0x5D byte).

   The state machine drops received packets until it detects SFD.
2. When SFD is detected, the state machine starts sending the data of Ethernet packet to the RPC module, beginning with the first byte following the SFD (destination address).
3. If IEEE 1588 timestamp feature is enabled, the MAC takes a snapshot of the system time at which SFD of any packet is detected on GMII or MII. If this packet is not dropped during MAC filtering, the timestamp is passed to the application.The receive state machine decodes the Length/Type field of the Ethernet packet being received.

   If the Length/Type field is less than 1,536 and if the MAC is programmed for the Auto CRC/Pad Stripping (bit 20 of the *Operating mode configuration register (ETH_MACCR)*), the state machine sends the packet data up to the count specified in the Length/Type field and starts dropping bytes (including the FCS field). The state machine decodes the Length/Type field and checks for the Length interpretation.
4. If the Length/Type field is greater than or equal to 1,536, the RPE module sends all received Ethernet packet data to the RFC module if you have not enabled the CRC stripping for Type packet in Bit 21 of the *Operating mode configuration register (ETH_MACCR)*. However, if the CRC stripping has been enabled for Type packets and not enabled the receive checksum offload engine, the MAC strips and drops the last 4 bytes of all packets of ether type before forwarding the packets to the application.
5. By default, the MAC is programmed for watchdog timer to be enabled, that is, packets above 2,048 (10,240 if Jumbo Packet is enabled) bytes (DA + SA + LT + DATA + PAD + FCS) are cut off at the RPE module. In addition, you can use a programmable watchdog timer (bit 16 of *Watchdog timeout register (ETH_MACWTR)*) to override the fixed timeout of 2,048 or 10,240 bytes. You can disable the watchdog timer by programming bit 19 of *Operating mode configuration register (ETH_MACCR)*. However, even if the watchdog timer is disabled, a packet greater than 32 Kbytes is cut off and a watchdog timeout status is given.

Figure 1024. MAC reception flow



76.4.4 Multiple channels and queues support

This chapter describes how the Ethernet peripheral core supports multiple queues and channels.

Two queues on Rx side and two queues on Tx side are supported. Two DMA channels are implemented on Tx side, one channel per queue. Two DMA channels are implemented on Rx side, one channel per queue.

The multiple queues enable to support the Audio video bridging (AVB) protocol.

Multiple queues and channels support in the Transmit path

The number of Tx DMA channels is equal to the number of Tx queues, and is direct one-to-one mapping. This enables the interleaving of packet/data transfer of multiple Tx DMA on the host/system bus, and each Tx queue is reserved for the corresponding Tx DMA. Therefore, the integrity of packet transfer by each Tx DMA is maintained in the Tx queue and enables efficient utilization of the host bus for data transfers among multiple Tx DMA without depending on the packet length, Transmit buffer availability and so on.

If two Tx DMAs are allowed to transfer data into the same Tx queue, when a Tx DMA starts a packet transfer, the other Tx DMA cannot transfer data to the same Tx queue unless the previous packet transfer is complete. This means that the second DMA remains idle until the first packet transferred is complete, which is effectively a one-to-one mapping of the Tx DMA to the Tx queue.

The Ethernet peripheral supports multiple Tx channels. The priority scheme between both channels is fixed priority. In fixed priority scheme, the channel with the highest priority always wins the arbitration when it requests the bus, as shown in *Table 764*.

Table 764. Fixed priority scheme for DMA channels

Priority level	Channel
0	Channel 0
1	Channel 1

The fixed priority scheme does not necessarily cause the “starving” of lower priority channels for the AXI bus because of the capabilities of the AXI protocol and characteristics of the TxDMA, described as follows:

- Read command requests from a TxDMA for a burst of data transfers are driven on a separate AXI channel, and their execution takes one clock cycle. The actual data transfer starts on a separate AXI channel only after the request is accepted. It takes several clock cycles to complete with relatively large read access latencies.
- The AXI master is free to issue subsequent requests even before the previous data transfers are complete.
- A TxDMA requests another data transfer only after the data transfer corresponding to its previous request is complete and transferred to the MTL Tx queue and space is available in Tx queue to accept the next request.
- The Ethernet peripheral can be programmed to control the maximum number of outstanding requests, requested burst sizes by DMA, and the burst sizes of each data transfer on the AXI bus.

Considering the previous points, the Ethernet peripheral can be programmed so that the requests of all TxDMAs are placed and executed on the AXI bus before the data transfer of the first TxDMA is complete and it raises the next request to the arbiter. For example, in a 64-bit AXI data bus, proceed as follows:

1. TxDMA PBL = 64 (512 bytes).
2. Number of outstanding AXI read requests = 16.
3. Maximum burst length on AXI (BLEN) = 32 (256 bytes).
4. Number of active TxDMAs = 8.

Assuming that all TxDMAs place a request simultaneously and get serviced by the fixed priority algorithm, all the requests are placed on the AXI bus in a window of 32 clocks (two clocks per AXI request, two AXI burst requests per TxDMA request). The data transfer of the first request takes 32 clock cycles + the read-access latencies (assuming that there are tens of cycles). So, the TxDMA arbiter is already IDLE and has completed servicing all the TxDMA requests before this data transfer completes.

Effectively, all the TxDMA channels get equal access to AXI bus and the corresponding Tx queues get filled up at the same rate. The data gets read out from the Tx queues as per the traffic class scheduler in the MTL. So at a steady state, the bandwidth for each channel is controlled by the priorities and settings of the traffic scheduler in the MTL.

Programming guidelines for multiple queues and channels in the Transmit path

See *Section 76.9.8: Programming guidelines for multichannel multiqueue operation on page 4128*.

Multiple queues and channels support in the receive path

The Ethernet controller supports independent selection of the number of Rx DMA channels and the number of Rx queues in the MTL Rx buffer. Therefore, the packets from any Rx queue can be serviced by any of the Rx DMA. As all the packets of Rx queues are stored in a shared RxFIFO buffer in MTL, the Rx scheduler selects the Rx queue whose packet data are to be transferred to the destination Rx DMA, which then forwards the packet data to the host memory. The sequence of events in time is as follows:

1. Each Rx DMA indicates whether it is ready to transfer data, that is, when the Rx DMA has fetched the descriptor and has a receiver buffer ready to accept the packet data. Each Rx DMA also indicates the number of words that it can transfer, based on the space available in the Rx buffer and the value programmed in the RXPBL[5:0] bitfield of its *Channel x receive control register (ETH_DMACxRXCR)*.
2. The MTL Rx scheduler matches the number of bytes/packets available for transfer at the top of all Rx queues with the corresponding Rx DMA ready status and the number of bytes requested by it.
3. If there are multiple successful matches, the scheduler selects the Rx queue to be serviced, based on the arbitration schemes (Strict priority or Weighted Strict priority) decided by the RAA field of the *Operating mode register (ETH_MTLOMR)*. In the Weighted Strict priority scheme, the weight of each Rx queue is set in the RXQ_WEGT[2:0] bitfield of the corresponding *Rx queue x control register (ETH_MTLRXQxCR)*.
4. The MTL Rx controller fetches the data of the selected Rx queue from the shared RxFIFO memory and starts the transfer of the requested PBL number of bytes to the

DMA block or until the end of packet and the corresponding Rx status is transferred (whichever is first).

5. After completing the current request, the scheduler repeats the process by starting a new arbitration cycle. However, it selects the same Rx queue for servicing, if the end of packet is not transferred and the RXQ_FRM_ARBIT field of the *Rx queue x control register (ETH_MTLRXQxCR)* is 1, for the Rx queue.

Rx queue to DMA mapping

The packets in the MTL Rx queues can be routed to any of the multiple DMA channels, by programming the Rx queue and the *Rx queue and DMA channel mapping register (ETH_MTLRXQDMAMR)*.

Two types of Rx queue-to-DMA mapping are possible through programming: static mapping and dynamic (per packet) mapping.

- Static mapping

  In Static mapping mode, all the packets of an Rx queue are connected to a specific DMA channel. For example, all the packets from Rx Queue 0 can be routed to a DMA channel by programming Q0MDMACH and Q0DDMACH of the *Rx queue and DMA channel mapping register (ETH_MTLRXQDMAMR)*.

  Similarly, packets from other Rx queues can be routed to any DMA channel by programming the register fields corresponding to each queue.

- Dynamic (per packet) mapping

  In Dynamic (per packet) mapping mode, the destination DMA channel is decided by the MAC core receiver for each packet.

  In this mode, the destination DMA channel of a packet being read from a Rx queue is not constant but decided independently for each packet. For example, by setting the Q1DDMACH bit of the *Rx queue and DMA channel mapping register (ETH_MTLRXQDMAMR)*, the static mapping is disabled for Rx Queue 1 and the value in Q1MDMACH is ignored. The destination DMA channel is decided by the MAC receiver for each packet, depending on the following, in decreasing order of priority:

  – L3-L4 filter based DMA selection

    When the Layer 3 and Layer 4 packet filtering is enabled, the TCP/UDP and IP header fields of the received packet are matched against the corresponding values programmed and enabled for comparison in the MAC L3 and L4 address control register. If the match is successful, the DMA channel number programmed in the DMCHNx bits of the *L3 and L4 control 0 register (ETH_MACL3L4C0R)* and *L3 and L4 control 1 register (ETH_MACL3L4C1R)* is selected as the destination DMA channel number provided DMCHEN bit of the same register is set.

    If none of the L3-L4 registers give a comparison match, the Ethernet peripheral proceeds to the next step.

  – Extended VLAN based DMA selection

    Extended routing is applicable only if the VLAN filter has passed. Routing is only based on perfect filter result. Each perfect filter has a DMA channel enable and a DMA channel number field which can be programmed. Routing is applicable for a filter only if the DMA channel enable bit is set.

    The frame is routed to the smallest matching filter DMA channel provided it is enabled. If that filter DMA channel number is not enabled, the frame gets routed to Channel 0. For example, if a frame VLAN tag matches filter 1, then the MAC checks if filter 1 DMA channel number is enabled through programming. If yes, the

frame gets routed to the programmed value; otherwise it gets routed to DMA. When the inverse filter is enabled, the frame is routed to the least mismatched filter DMA channel number provided it is enabled. If the DMA channel enable bit is not set, then the frame is routed based on DA-based addressing or to Channel 0.

If Hash filter is also enabled, it is used to determine the filter result only. Routing still depends on the enabled perfect filters. If none of the perfect filters are enabled or if all of them are bypassed, the VLAN filter based routing is not done. The frame is routed through DA-based addressing or to Channel 0. If all the perfect filters give a fail result and the Hash filter has passed, VLAN filter result is a pass. However, routing is based on DA-based addressing or to Channel 0. The behavior is similar when the inverse filtering is enabled.

– Ethernet DA-based DMA selection

The DA address of the received packet is compared against the programmed DA values in MAC address registers. If the address matches any of the programmed values, the corresponding DCS field (when enabled) determines the destination DMA channel number.

If none of the previous operations is able to make a successful match/decision, then the packet is routed to DMA Channel 0 by default.

Programming guidelines for multiple queues and channels in the receive path

See *Section 76.9.8: Programming guidelines for multichannel multiqueue operation on page 4128*.

Selection of the tag priorities assigned to Rx queues

The VLAN tag priorities can be assigned to Rx queues by programming the PSRQ field in the corresponding *Rx queue control 2 register (ETH_MACRXQC2R)*. The bit corresponding to the VLAN tag priority can be set in the PSRQ field to assign the corresponding priority to the Rx queue. As an example, to assign VLAN tag priority 3 to Rx queue 0, set bit3 of PSRQ field in the *Rx queue control 2 register (ETH_MACRXQC2R)*. The VLAN tag priority assigned to a given Rx queue must be unique. This means that multiple Rx queues cannot be assigned the same VLAN tag priority. However, multiple VLAN tag priorities can be assigned to the same Rx queue.

The settings in the PSRQ field is used for VLAN tagged Rx packet routing to Rx queues as well as for PFC based Tx flow control. The received VLAN tagged Rx packet is routed to the Rx queue that matches the VLAN tag priority. In PFC-based Tx flow control, the PSRQ field corresponding to a particular Rx queue is used to enable VLAN tag priorities in the PFC packet that is transmitted when the corresponding Rx queue threshold levels are reached.

Distribution of Rx packets from MAC to Rx queues

The Ethernet peripheral provides multiple Rx queues to support classification and distribution of ingress packets. The MAC receiver pushes the ingress packets to the Rx queues based on packet type (such as tagged, untagged, multicast, PTP, AVB), tag priorities, MAC packet filter results, preemptable or express packet, and so on. The distribution is controlled by the settings of:

- *Rx Queue control register (ETH_MACRXQCR)*
- *Rx queue control 0 register (ETH_MACRXQC0R)*
- *Rx queue control 1 register (ETH_MACRXQC1R)*
- *Rx queue control 2 register (ETH_MACRXQC2R)*
- *Packet filtering control register (ETH_MACPFR)*

Additionally, the Ethernet peripheral supports distribution of ingress packets based on user-defined Type field of the ingress packets when TBRQE bit of the *Rx queue control 1 register (ETH_MACRXQC1R)* is enabled. Up to eight types for matching can be programmed in the *MAC type-based Rx queue mapping register (ETH_MAC_TMRQR)* which is accessible through *MAC indirect access control register (ETH_MAC_IACR)*. The ETH_MAC_TMRQR register contains

- TYP[15:0] field: 16-bit type
- TMRQ[2:0] bitfield: Rx queue number to which ingress packet is pushed when its Type field matches TYP.
- PFEX field: this bit indicates whether the TYP is applicable for express packets or preemptable packets.

Based on the packet filter settings, the MAC receiver decides either to drop or forward the ingress packet (for more details, see *Section 76.5.4: Packet filtering*). However, the results of Layer 3 and Layer 4 filtering, mentioned in *Section : Layer 3 and Layer 4 filtering*, are not considered for the distribution of ingress packets into the Rx queues. Only the results of DA/SA/VLAN filters mentioned in *Section : MAC source or destination address filtering* and *Section : VLAN filtering* are applicable.

The filtering of the broadcast packets is exclusively controlled by the DBF bit of the *Packet filtering control register (ETH_MACPFR)*. The filtering of multicast packets is controlled by the PM bit of the *Packet filtering control register (ETH_MACPFR)*, in addition to the DA/SA match.

By default, packets that fail any of the DA/SA filters, are dropped by the MAC receiver. However, packets that fail the DA/SA/VLAN filter can still be forwarded to the application when the RA field of the *Packet filtering control register (ETH_MACPFR)* is 1. Even when RA field is 0, VLAN filter fail packets can be forwarded if the VTFE field of the *Packet filtering control register (ETH_MACPFR)* is 0.

The ingress packets go through the following operations in decreasing order of priority.

1. RA = 0: ingress packets are dropped if any of the following conditions is true.
   – DA/SA filter fail
   – VLAN filter fail and VTFE field of the *Packet filtering control register (ETH_MACPFR)* is 1
2. RA = 1: ingress packets that fail the DA/SA filter are also forwarded to the Rx queue as mentioned in *Table 765*.

Table 765. Routing DA/SA failed packets to Rx queue

Packet type	TSN type	Destination Rx queue determined by
Broadcast/Multicast	Express	if MFFQE = 1, MFFQ; else if VFFQE = 1, VFFQ; (only for tagged packets) else RxQ0
Unicast	Express	if UFFQE = 1, UFFQ; else if VFFQE = 1, VFFQ; (only for tagged packets) else RxQ0
All	Preemptable	FPRQ field

3. RA = 1 and VTFE = 0: ingress tagged packets that pass the DA/SA filter but fail the VLAN filter are forwarded to the Rx queue as mentioned in *Table 766*.

Table 766. Routing DA/SA pass but VLAN filter fail packets to Rx queue

TSN packet type	Destination Rx queue determined by
Express	if VFFQE = 1, VFFQ; else RxQ0
Preemptable	RQ field

4. Ingress packets that pass the DA/SA/VLAN filters are forwarded to the Rx queue.
   – *Table 767* shows the routing priority in the decreasing order when OMCBCQ field of the *Rx queue control 0 register (ETH_MACRXQC0R)* is 0.

Table 767. Priority routing of filter pass packets when OMCBCQ = 0

Packet type	TSN type	Tagged	Destination Rx queue determined by
Broadcast/Multicast	Express	Yes	if MCBCQEN = 1, MCBCQ; else if TBRQE = 1, TBRQ; else PSRQ
		No	if MCBCQEN = 1, MCBCQ; else if TBRQE = 1, TBRQ; else UPQ
Unicast AVTP control	Express	No	AVCPQ
		Yes	if TACPQE = 1, AVCPQ; else PSRQ
Unicast PTP over Ethernet	Express	No	PTPQ
		Yes	PTPQ or PSRQ based on the value of TPQC.
Remaining Unicast	Express	No	If TBRQE = 1, TBRQ; else UPQ
		Yes	If TBRQE = 1, TBRQ; else PSRQ
All Packets	Preemptable	No	If TBRQE = 1, TBRQ; else FPRQ
		Yes	If TBRQE = 1, TBRQ; else PSRQ

   – *Table 768* shows the routing priority in the decreasing order when OMCBCQ field of the *Rx queue control 1 register (ETH_MACRXQC1R)* is 1.

Table 768. Priority routing of filter pass packets when OMCBCQ = 1

Packet type	TSN type	Tagged	Destination Rx queue determined by
All AVTP Control	Express	No	AVCPQ
		Yes	if TACPQE = 1, AVCPQ; else PSRQ
All PTP over Ethernet	Express	No	PTPQ
		Yes	PTPQ or PSRQ based on TPQC
Remaining Broadcast/Multicast	Express	Yes	if MCBCQEN = 1, MCBCQ; else if TBRQE = 1, TBRQ; else PSRQ
		No	if MCBCQEN = 1, MCBCQ; else if TBRQE = 1, TBRQ; else UPQ

Table 768. Priority routing of filter pass packets when OMCBCQ = 1 (continued)

Packet type	TSN type	Tagged	Destination Rx queue determined by
Remaining Unicast	Express	No	if TBRQE = 1, TBRQ; else UPQ
		Yes	if TBRQE = 1, TBRQ; else PSRQ
All Packets	Preemptable	No	if TBRQE = 1, TBRQ; else FPRQ
		Yes	if TBRQE = 1, TBRQ; else PSRQ

Note: The fields mentioned in Table 765 through Table 768 belong to Rx queue control 0 register (ETH_MACRXQC0R), Rx queue control 1 register (ETH_MACRXQC1R) and Rx queue control 2 register (ETH_MACRXQC2R).

Note: If the RxQ pointed by the various fields in Table 765 through Table 768 is not enabled in the Rx queue control 0 register (ETH_MACRXQC0R) as per the packet type, the packet is written to

- *RxQ0 for Express packets*
- *FPRQ field for Preemptable packets*

If RxQ0/FPRQ is not enabled, the packet is dropped.

Note: If the AV8021ASMEN field of the Timestamp control register (ETH_MACTSCR) is 1, MAC receiver checks only for untagged PTP packet. Packets tagged with PTP payload are considered as Remaining (Unicast/Broadcast/Multicast) tagged packets in Table 765 through Table 768.

The priority field of the outer VLAN tag is considered for routing.

AV feature

The Ethernet peripheral supports the AV data transfer in 100-Mbps and 1000-Mbps modes. The AV feature enables transmission of time-sensitive traffic over bridged local area networks (LANs). The following standards define various aspects of the AV feature implementation:

- IEEE 802.1Qav-2009: enables the bridges to provide time-sensitive and loss-sensitive real-time audio video data transmission (AV traffic). It specifies the priority regeneration and controlled bandwidth queue draining algorithms that are used in bridges and AV traffic sources.
- IEEE 802.1Qat-2009: enables the network resources to be reserved for specific traffic streams traversing a bridged local area network.
- IEEE 802.1AS-2011: specifies the protocol and procedures used to ensure that the synchronization requirements are met for time-sensitive applications such as audio and video across bridged and virtual-bridged LANs consisting of LAN media where the transmission delays are fixed and symmetrical. For example, IEEE 802.3 Full-duplex links include the maintenance of synchronized time during normal operation followed by addition, removal, or failure of network components and network reconfiguration.

While the AVB standards were originally designed for audio and video over standard Ethernet, extending these to control streams would additionally benefit the industrial and automotive sectors. Time sensitive networking (TSN) is a set of standards developed by the Time Sensitive Networking Task Group with the intent to extend the future AVB enhancements to other sectors that could benefit.

Bridges are increasingly used to interconnect devices that support scheduled applications (for example, industrial automation, process control and vehicle control). The IEEE 802.1Qbv-2015 (Enhancements to Scheduling Traffic) provides performance assurances of latency and delivery variation to enable these applications in an engineered LAN while maintaining the existing guarantees for the credit-based shaper and best-effort traffic.

The IEEE 802.3br (Interspersing Traffic) and IEEE 802.1Qbu (frame preemption) enables the suspension of a large preemptable packet being transmitted by the MAC layer to enable one or more express packets to be transmitted before the transmission of the preemptable packet is resumed. This provides the capability to schedule express traffic packets with minimal delays/latencies or at predictable times with efficient utilization of the line bandwidth.

As per the IEEE specification for 802.1Qbv, 802.1Qbu, and 802.3br, the TSN features are supported only in the following modes:

- Full duplex
- 100 Mbps or higher link speed
- Pause or no flow control
- When Qbv schedule is enabled for time-sensitive traffic, Pause or PFC is likely to interfere with the Qbv schedules. So, it is recommended not to enable PFC/PAUSE when Qbv scheduler is enabled.
- Multiple Tx queues (for Qbv) enabled
- One or more Express queues and one or more preemption queues on both transmit and receive (for Qbu/802.3br) enabled.

Transmit path functions

When the AV feature is selected, the Transmit paths of Tx queues are enabled by default.

The Transmit path of queue 0 supports the strict priority algorithm, and it is used for best-effort traffic. For a queue, the strict priority algorithm determines that a packet is available for transmission if the queue contains one or more packets. When the threshold mode for MTL Tx FIFO is enabled, the strict priority algorithm determines that a packet is available for transmission if the queue contains a partial packet of size equal to the programmed threshold limit.

The Transmit path of the AVB Tx queue supports traffic management by using the credit-based shaper algorithm. For a queue, the credit-based shaper algorithm determines that a queue is available for transmission if the following conditions are true:

- The queue contains one or more packets.
- The credit for the queue is positive as per the algorithm.

When the credit-based shaper algorithm is disabled for a queue, the channel uses the default strict priority algorithm.

Each Transmit DMA has a separate descriptor chain for fetching the transmit data. The Transmit channel that gets the access to the system bus depends on the DMA arbiter.

The Transmit path has a shared FIFO (MTL layer) for each queue. The data fetched by the DMA is put in the respective part of the FIFO. The MTL Tx Queue Scheduler controls which part of the FIFO data is transmitted by the MAC. If the credit-based shaper algorithm is enabled for queue 1, the queue is selected if the packet is available in the channel and has a positive or zero credit.

If the credit-based shaper algorithm is disabled for a given Tx queue, the packet to be transmitted is selected based on the fixed priority scheme as described in *Table 769*.

Table 769. Weight for DMA channels

TCW field	Transmit channel weight
000	1
001	2

Receive path functions

When the AV feature is enabled, the receive path of Queue 0 is enabled by default. The whole traffic is received on this channel. The receive paths of additional queues can be enabled. By enabling the receive paths of multiple channels, the received data can be demultiplexed to send the packets into separate receive channels.

To differentiate between the AV and non-AV traffic, the MAC provides a status that indicates if the received packet is an AV packet as well as the corresponding VLAN Priority tag value. This status is updated in the Extended Status field of the receive descriptor as explained in *Section 76.10.4: Receive descriptor*. All received packets with an EtherType field of 0x22F0 are detected as AV packets. The AV packets can be of the following two types:

- AV data packets

  AV data packets are always tagged. Tagged AV data packets are received following the programmed priority value. To specify the channel to which an AV packet with a given priority must be sent, program bits[15:8] in the transmit flow control register of the corresponding queue.

- AV control packets

  The AV control packets can be either tagged or untagged. By default, untagged AV control packets are received on queue 0. To receive these packets on queue 1, program bits[2:0] of the *Rx queue control 1 register (ETH_MACRXQC1R)*. Similarly to the AV data packets, tagged AV control packets are received following the programmed priority value.

In addition to the AV packets, you can receive the untagged PTP packets on any queue. By default, the PTP packets (tagged or untagged) are received on queue 0. To receive these packets on any other queue, program bits[6:4] of the *Rx queue control 1 register (ETH_MACRXQC1R)*.

Credit-based shaper algorithm

The Queue Scheduler uses the credit-based shaper algorithm to arbitrate the AV traffic in all queues and the legacy Ethernet traffic in queue 0. Queue 1 can be programmed to use the credit-based shaper algorithm.

The following sections provide information on the credit-based shaper algorithm implementation:

- Credit value
- idleSlopeCredit and sendSlopeCredit values
- Bandwidth status

Credit value

The credit value is accumulated every transmit clock cycle, that is every 40 ns in 100-Mbps mode and every 8 ns in 1000-Mbps mode. The credit to be added or subtracted per cycle can be fractional depending on the required idleSlope and sendSlope values, as described in *Table 770*.

Table 770. Example of credit value per transmit cycle

Mode	portTransmitRate, idleSlope, sendSlope	Credit value
100 Mbps	– portTransmitRate = 100 Mbps – idleSlope = 70 Mbps (assuming 70% bandwidth reserved for a higher priority traffic class) – sendSlope = 30 Mbps	– credit = 2.8 bits, accumulated per cycle (40 ns) for the higher priority traffic class when best-effort packet is being transmitted. – credit = 1.2 bits, drained per cycle (40 ns) when higher priority traffic class packet is being transmitted.

The DMA stores the queue traffic in the respective part of the Tx FIFO depending on the slot number in the transmit descriptor (if enabled) or the bandwidth availability on the AMBA application bus.

The credit for a queue builds up only when the packet is available, but it cannot be transmitted because the MAC is sending a packet from another queue. The Ethernet peripheral supports another mode in which the credit can build up in advance for a queue in which no packet is available in respective part of the FIFO. This enables sending a burst of high priority traffic in a queue as soon as the data is available. This mode can be enabled through the CC bit of the CBS control register (*Tx queue 1 ETS control register (ETH_MTLTXQ1ECR)*.

- When reset, the accumulated credit parameter in the credit-based shaper algorithm is set to zero if there is positive credit, and there is no packet to transmit in a queue. The credit does not accumulate when there is no packet waiting in a queue and other queues are transmitting.
- When set, the accumulated credit parameter in the credit-based shaper algorithm is not reset to zero if there is positive credit and no packet to transmit in a queue. The credit accumulates even when there is no packet waiting in a queue and other queues are transmitting.

idleSlopeCredit and sendSlopeCredit values

The software must program the idleSlopeCredit and sendSlopeCredit values. The programmed values should be the credit accumulated or drained per clock cycle scaled by 1024 (such as 2.8 x 1024 = 2867 and 1.2 x 1024 = 1229). In addition, the software must program the hiCredit and loCredit values, scaled by 1024, to adjust for scaling of the idleSlopeCredit and sendSlopeCredit values. This means that if computed hiCredit and loCredit values are 12,000 bits and 3,036 bits respectively, the values to be programmed in the hiCredit and loCredit registers of the corresponding channel are 12000 x 1024 bits and two's complement of 3036 x 1024, respectively.

Bandwidth status

The hardware maintains the status of the actual bandwidth consumed by each higher priority queue in the CBS status registers. This enables the software to estimate the average bandwidth consumed by numerically higher traffic classes as compared to the reserved bandwidth.

The CBS status register (*Tx queue x ETS status register (ETH_MTLTXQxESR)*) gives the average number of bits transmitted during the previous programmed slot interval (1, 2, 4, 8, or 16 slots of 125 µs) in a queue. The status register is updated even if the credit-based shaper algorithm is not enabled for a queue. The number of slots over which the average bits transmitted per slot are computed is programmed in bits[6:4] of the CBS control register of the respective queue. For example, if you have programmed two slots, the average bits are computed over slot numbers 0–1, 2–3, 4–5, and so on.

The value programmed in the idleSlopeCredit register of a queue is proportional to the bandwidth reserved for the queue. The software can allocate any bandwidth that is not used by the higher priority queue to the reserved bandwidth of the lower priority queue.

A lower priority queue, which is using the credit-based shaper algorithm, cannot use the unused reserved bandwidth of any higher priority queue that is using the credit-based shaper algorithm. However, a lower priority queue, which is using the strict-priority algorithm, can use the unused reserved bandwidth of any higher priority queue that uses the credit-based shaper algorithm. For example, queue 1 and queue 2 use the credit-based shaper algorithm (with reserved bandwidth of 50% and 25%, respectively) and queue 0 uses the strict-priority algorithm. If queue 1 uses only 40% of reserved bandwidth, the remaining 10% is used by queue 0. Queue 2 cannot exceed the reserved bandwidth of 25%.

Slot number function

When the AV feature is enabled, the slot number function can be used to schedule the data fetching from the system memory by the DMA. This feature is useful when the source AV data needs to be transmitted at specific intervals.

The slot number at which the DMA should fetch the data from system memory can be programmed in the transmit descriptor Word 3 (TDES3). This 4-bit field enables the application to schedule data fetching at up to 16 slots with a programmable slot interval that ranges from 1 µs to 4096 µs and a granularity of 1 µs. This field is applicable only to the AV channels.

When the DMA fetches a Tx descriptor, it compares the slot number of the Tx descriptor with the internally generated reference slot interval. The slot interval is a counter that is updated based on a programmable slot interval (with range from 1 µs to 4096 µs) of the IEEE 1588 system time. In addition, the slot interval counter is initialized to zero when the seconds field of the system time is incremented, that is, when the subsecond counter rolls over. The DMA fetches the data only if it matches the current slot or the next slot. The DMA remains in the descriptor fetch state until a match is found.

To enable the DMA to fetch data only if it matches the current slot or the next two slots, program bit 1 of the *Channel x slot function control status register (ETH_DMACxSFCSR)* of the corresponding DMA channel.

If the slot number in the descriptor is less than the reference slot number, the DMA considers it as a future slot.

The slot number check can be enabled by setting bit 0 in the *Channel x slot function control status register (ETH_DMACxSFCSR)* of corresponding DMA channel. If this check is disabled, the packets are fetched immediately after the descriptor is read. In addition, bits [19:16] indicate the value of the reference slot number in DMA.

Programming guidelines for AV feature

See *Section : Initializing the DMA on page 4137*.

See *Section : Enabling slot number checking on page 4138*.

See *Section : Enabling average bits per slot reporting on page 4138*.

Queue modes

A Tx queue can be enabled for generic or AV traffic. When multiple Tx queues are enabled for generic traffic, queuing is based on WRR or WSP algorithms.

When a Tx queue is enabled for AVB traffic, queuing is based on CBS or SP algorithms.

A Rx queue can be enabled for generic or AV traffic. A particular queue enabled for generic or AV based routing is determined by the RXQ0EN and RXQ1EN fields in the *Rx queue control 0 register (ETH_MACRXQC0R)*.

Below an explanation on how Rx queues are enabled based on the selected features:

- Multiple Rx queues

  When multiple Rx queues are selected, all queues are enabled for generic queuing based on the VLAN tag priority. The VLAN tag priority should match the PSRQ[7:0] bitfield of the *Rx queue control 2 register (ETH_MACRXQC2R)*. By default untagged packets are routed to receive queue specified in UPQ field of *Rx queue control 1 register (ETH_MACRXQC1R)*. Queue 0 is the default value of UPQ[2:0] bitfield. The default value can be overridden with any other value, for the UPQ[2:0] bitfield. The Rx packets can also be routed to a particular DMA channel based on the DCS field of the perfectly-matched MAC Address register (*MAC address x high register (ETH_MACAxHR)*.

- Multiple Rx queues with AV feature

  When AV is enabled for all selected Rx queues, the queuing is done based on the VLAN tag priority. The VLAN tag priority should match the PSRQ[7:0] bitfield of the *Rx queue control 2 register (ETH_MACRXQC2R)*. The Rx packets can also be routed to a particular DMA channel based on the DCS field of perfectly-matched MAC Address register (*MAC address x high register (ETH_MACAxHR)*). The AV control packets (tagged or untagged) are routed based on the AVCPQ[2:0] bitfield of the *Rx queue control 1 register (ETH_MACRXQC1R)*. The PTP over Ethernet packets are routed based on the PTPQ[2:0] bitfield of the *Rx queue control 1 register (ETH_MACRXQC1R)*.

76.5 Ethernet functional description: MAC

76.5.1 Double VLAN processing

The Ethernet peripheral supports the double VLAN (Virtual LAN) tagging feature in which the MAC can process up to two VLAN tags (inner and outer).

The MAC supports the following:

- Insertion, replacement, or deletion of up to two VLAN tags in the Transmit path
- Packet filtering and stripping based on any one of the two VLAN Tags in the receive path. Stripping and providing up to two VLAN Tags in the receive path as a part of the receive status

Transmit path

Table 771: Double VLAN processing features in Tx path describes the features supported by the MAC on the Transmit side.

Table 771. Double VLAN processing features in Tx path

Feature	Description
Support for C-VLAN and S-VLAN tag types	The inner or outer VLAN tag can be of C-VLAN and S-VLAN type. The VLAN type is specified through the CSVL bit of *VLAN inclusion register (ETH_MACVIR)* and *Inner VLAN inclusion register (ETH_MACIVIR)*, respectively. The Ethernet peripheral supports processing of any sequence of outer and inner VLAN tags. However, it does not support the C-VLAN S-VLAN sequence. The MAC does not check whether the packet provided by the application has a valid sequence of the VLAN Tag types or the insertion or replacement operation results in invalid sequence of VLAN Tag type. Therefore, the application must provide correct sequence of VLAN Tag types and program the MAC in such a way that it results in correct sequence of VLAN Tag types in the transmitted packet. The application must ensure the following: – The inner tag should not be S-VLAN when outer C-VLAN Tag insertion is enabled. – The outer tag should not be C-VLAN when inner S-VLAN Tag insertion is enabled. – The inner tag should not be S-VLAN when outer tag should be replaced with C-VLAN. – The outer tag should not be C-VLAN when inner tag should be replaced with S-VLAN.
VLAN tag deletion	VLAN tag deletion can be enabled for outer or inner tag through VLC field in the *VLAN inclusion register (ETH_MACVIR)* or *Inner VLAN inclusion register (ETH_MACIVIR)*, respectively. When VLAN deletion is enabled, the MAC deletes the tag present at the corresponding position. When a packet has only one tag, it is considered as the outer tag. If inner tag deletion is enabled and the packet has only one tag, the MAC does not delete the tag.
VLAN tag insertion or replacement	VLAN tag insertion or replacement can be enabled for outer or inner tag through VLC field in the *VLAN inclusion register (ETH_MACVIR)* or *Inner VLAN inclusion register (ETH_MACIVIR)*, respectively. When VLAN tag insertion or replacement is enabled, the VLTI bit in the previous register is used to determine whether the VLAN tag should be taken from the register or the control word.

Receive path

Table 772: Double VLAN processing in Rx path describes the features supported by the MAC on the receive side and the corresponding bits in the *VLAN tag Control register (ETH_MACVTCR)*.

Table 772. Double VLAN processing in Rx path

Feature	Description
Outer or inner VLAN tag-based filtering	The MAC can filter packets based on the outer or inner VLAN tag through the ERIVLT bit.
C-VLAN or S-VLAN tag-based filtering	The MAC can filter packets based on the C-VLAN or S-VLAN type based on the ERSVLM bit.
Outer and Inner VLAN tag stripping	The MAC can strip the outer and inner VLAN Tags from received frame based on the EVLS and EIVLS bits.
16-bit outer and inner VLAN tag and type in Rx status	The MAC can provide the 16-bit outer and inner VLAN Tag and Type in the Rx status based on the EVLRXS and EIVLRXS bits, respectively.
Disabling or skipping checking of outer VLAN tag type	The MAC can disable or skip checking of outer VLAN Tag type to match C-VLAN or S-VLAN based on the DOVLTC bit.

76.5.2 Source address and VLAN insertion, replacement, or deletion

Source address insertion or replacement

The software can use the SA (source address) insertion or replacement feature to instruct the MAC to do the following for Tx packets:

- Insert the content of the MAC Address registers in the SA field
- Replace the content of the SA field with the content of the MAC Address registers

When SA insertion is enabled, the application must ensure that the packets sent to the MAC do not have the SA field. The MAC does not check whether the SA field is present in the Transmit packet and it inserts the content of MAC Address Registers in the SA field. Similarly, when SA replacement is enabled, the application must ensure that the SA field is present in the packets sent to the MAC. The MAC replaces the six bytes following the Destination Address field in the Transmit packet with the content of the MAC Address Registers.

SA insertion or replacement feature can be enabled for all Transmit packets or selective packets:

- Enabling SA insertion or replacement for all packets

  To enable this feature for all packets, program the SARC field of the *Operating mode configuration register (ETH_MACCR)*.
- Enabling SA insertion or replacement for selective packets

  To enable this feature for selective packets, use the following program the SA Insertion Control field (bits[25:23] of transmit descriptor Word 3/TDES3, refer to *Section 76.10.3: Transmit descriptor*) in the first Transmit descriptor of the packet. When Bit 25 of TDES3 is set, the SA Insertion Control field indicates insertion or replacement by MAC

Address1 registers. When bit 25 of TDES3 is reset, it indicates insertion or replacement by MAC Address 0 registers.

If MAC Address1 registers are not enabled, the MAC Address0 registers are used for insertion or replacement whatever of the value of the most-significant bit of the SA Insertion Control field.

VLAN insertion, replacement, or deletion

The software can use the VLAN insertion, replacement, or deletion feature to instruct the MAC to do the following for Tx packets:

- Delete the VLAN Type and VLAN Tag fields
- Insert or replace the VLAN Type and VLAN Tag fields

  Insertion or replacement is performed based on the setting of VLTI bit in the *VLAN inclusion register (ETH_MACVIR)* as described in *Table 773: VLAN insertion or replacement based on VLTI bit*.

Table 773. VLAN insertion or replacement based on VLTI bit

Condition	Description
VLTI bit is set	The MAC inserts or replaces the following: VLAN Type field (C-VLAN or S-VLAN as indicated by the CSVL bit of *VLAN inclusion register (ETH_MACVIR)*) VLAN Tag field with VT field of Transmit context descriptor of the packet
VLTI bit is reset	The MAC inserts or replaces the following: VLAN Type field (C-VLAN or S-VLAN as indicated by the CSVL bit of *VLAN inclusion register (ETH_MACVIR)*) VLAN Tag field with the VLT field of *VLAN inclusion register (ETH_MACVIR)*

When VLAN replacement or deletion is enabled, the MAC checks if the VLAN Type field (0x8100 or 0x88A8) is present after the DA and SA fields in the Transmit packet. The replace or delete operation does not occur if the VLAN Type field is not detected in two bytes following the DA and SA fields. However, when VLAN insertion is enabled, the MAC does not check the presence of VLAN Type field in the Transmit packet and just inserts the VLAN Type and VLAN Tag fields.

You can enable the VLAN insertion, replacement, or deletion feature for all Tx packets or selective packets:

- To enable this feature for all packets, program the VLC and VLP fields of *VLAN inclusion register (ETH_MACVIR)*.
- To enable this feature for selective packets, program the VTIR field of TDES2 Normal Descriptor (see *Table 823: TDES2 normal descriptor (read format)*).

  In addition, the VLP (VLAN Priority control) bit must be reset in *VLAN inclusion register (ETH_MACVIR)* (for outer VLAN) and *Inner VLAN inclusion register (ETH_MACIVIR)* (in inner VLAN) for the MAC to take the control inputs from the host, depending on the configuration.

76.5.3 Queue/channel-based VLAN tag insertion on Tx

The peripheral supports channel/queue based VLAN tag insertion on all transmitted packets.

Accessing queue/channel specific VLAN tag registers

Queue/channel specific VLAN tag registers are accessed using indirect addressing through the VLAN inclusion register (*VLAN inclusion register (ETH_MACVIR)*). VLAN type and tag value can be independently programmed for each queue/channel.

Enabling queue/channel based VLAN tag insertion on Tx

To enable this feature, set the CBTI bitfield in the VLAN inclusion register (*VLAN inclusion register (ETH_MACVIR)*). The VLAN tag is then inserted on every packet that is transmitted from a queue/channel, with the programmed tag value taken from queue/channel specific VLAN tag register

Programming guidelines for channel based VLAN tag insertion

See *Section 76.9.19: Programming sequence for queue/channel- based VLAN inclusion register on page 4145*.

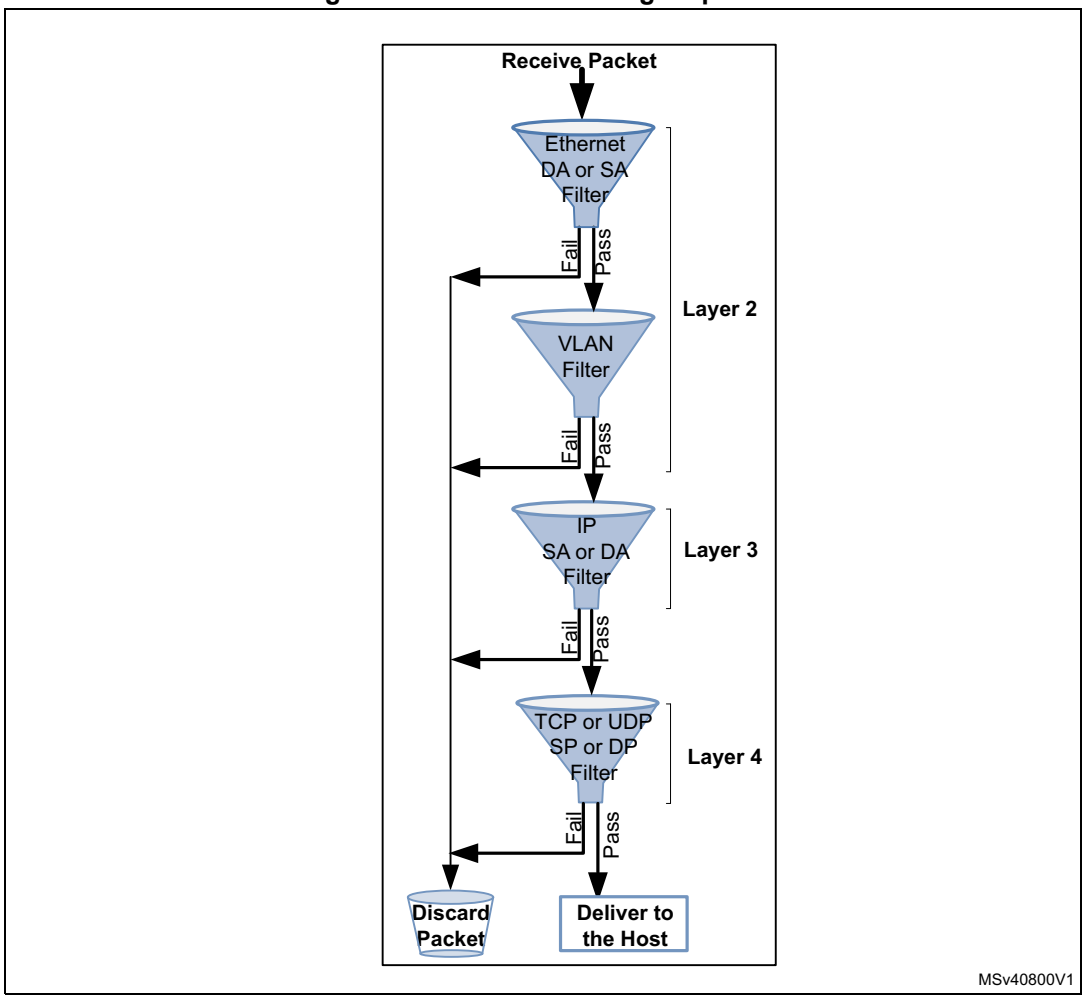
76.5.4 Packet filtering

The MAC supports the following types of filtering for Rx packets:

- *MAC source or destination address filtering*: the Address Filtering Module (AFM) checks the source address and destination address fields of each incoming packet.
- *VLAN filtering*: the MAC supports the VLAN tag-based and VLAN Hash filtering.
- *Layer 3 and Layer 4 filtering*: Layer 3 filtering refers to IP source address and destination address filtering. Layer 4 filtering refers to source port and destination port filtering.

The three filter types can be cascaded. *Figure 1025* shows the filtering sequence for Rx packets.

Figure 1025. Packet filtering sequence



The sequence shown in *Figure 1025* is valid when all the filters (L2, VLAN, L3, L4) are active. If any of the Layer filters are not enabled, that filter is bypassed and the subsequent filter is applied. A packet that fails any of the filters is discarded. However, the discarded packet can be forwarded to the host based on the register control.

For example, when RA bit of *Packet filtering control register (ETH_MACPFR)* is set to 1, all the discarded packets are forwarded to the host but with their packet status indicating the specific filter failure. If RA bit is cleared to 0, VTFE and IPFE bits of *Packet filtering control register (ETH_MACPFR)* control if the packets that fail the VLAN filter and Layer 3-4 filter should be discarded or forwarded to the host.

MAC source or destination address filtering

The MAC address filtering module checks the source address (SA) and destination address (DA) fields of each incoming packet.

Unicast destination address filtering

The MAC supports 4 MAC addresses for unicast perfect filtering. If perfect filtering is selected (HUC bit of *Packet filtering control register (ETH_MACPFR)* is reset), the MAC compares all 48 bits of received unicast address with the programmed MAC address for any match. The default MacAddr0 is always enabled.

The MacAddr1 to MacAddr3 addresses are selected with an individual enable bit. You can mask each byte during comparison with corresponding received DA byte by setting the corresponding Mask Byte Control bit in *MAC address x high register (ETH_MACAxHR)*. This enables group address filtering for the DA.

In Hash filtering mode (when HUC bit is set), the MAC performs imperfect filtering for unicast addresses using a 64-bit Hash table. For Hash filtering, the MAC uses the upper 6 bits CRC of the received destination address to index the content of the Hash table. A value of 00000 selects bit 0 of selected register, and a value of 11111 selects bit 63 of Hash Table register. If the corresponding bit (indicated by the 6-bit CRC) is set to 1, the unicast packet is considered to have passed the Hash filter; otherwise, the packet is considered to have failed the Hash filter.

Multicast destination address filtering

To program the MAC to pass all multicast packets, set the PM bit in *Packet filtering control register (ETH_MACPFR)*. If the PM bit is reset, the MAC performs the filtering for multicast addresses based on the HMC bit of the *Packet filtering control register (ETH_MACPFR)*.

In Perfect filtering mode, the multicast address is compared with the programmed MAC destination address registers. Group address filtering is also supported.

In Hash filtering mode, the MAC performs imperfect filtering using a 64-bit Hash table. The MAC uses the upper 6-bits CRC of received multicast address to index the content of the Hash table. A value of 000000 selects bit 0 of selected register and a value of 111111 selects bit 63 of the Hash Table register. If the corresponding bit is set to 1, the multicast packet is considered to have passed the Hash filter. Otherwise, the packet is considered to have failed the Hash filter.

Hash or Perfect address filtering

To configure the DA filter to pass a packet when its DA matches either the Hash filter or the Perfect filter, set the HPF bit and the corresponding HUC or HMC bits in *Packet filtering control register (ETH_MACPFR)*. This is applicable to both unicast and multicast packets. If the HPF bit is reset, only one of the filters (Hash or Perfect) is applied to receive packet.

Broadcast address filtering

The MAC does not filter any broadcast packets by default. To program the MAC to reject all broadcast packets, set the DBF bit in *Packet filtering control register (ETH_MACPFR)*.

Unicast source address filtering

The MAC can perform perfect filtering based on the source address field of received packets. By default, the MAC compares the SA field with the values programmed in the SA registers. You can configure the MAC Address registers to use SA instead of DA for comparison by setting bit 30 of *MAC address x high register (ETH_MACAxHR)*.

The MAC also supports group filtering with SA. You can filter a group of addresses by masking one or more bytes of the address. The MAC drops the packets that fail the SA filter if the SAF bit is set in *Packet filtering control register (ETH_MACPFR)*. Otherwise, the result of the SA filter is given as a status bit in the receive Status word (see *Table 775*). When the SAF bit is set, the SA filter and DA filter result is ANDed to decide whether the packet needs to be forwarded. This means that the packet is dropped if either filter fails. The packet is forwarded to the application only if the packet passes both filters in-order.

Inverse filtering

For DA and SA filtering, you can invert the filter-match result at the final output by setting the DAIF and SAIF bits of *Packet filtering control register (ETH_MACPFR)*. The DAIF bit is applicable for both Unicast and Multicast DA packets. The result of the unicast or multicast destination address filter is inverted in this mode. Similarly, when the SAIF bit is set, the result of unicast SA filter is reversed.

Table 774 and *Table 775* summarize the DA and SA filtering based on the type of packets received.

Note: When the RA bit of Packet filtering control register (ETH_MACPFR) is set, all packets are forwarded to the system along with the correct result of the address filtering in the Rx status.

Table 774. Destination address filtering

Packet type	PR	HPF	HUC	DAIF	HMC	PM	DBF	DA filter operation
Broadcast	1	X	X	X	X	X	X	Pass
	0	X	X	X	X	X	0	Pass
	0	X	X	X	X	X	1	Fail
Unicast	1	X	X	X	X	X	X	Pass all packets
	0	X	0	0	X	X	X	Pass on Perfect/Group filter match
	0	X	0	1	X	X	X	Fail on Perfect/Group filter match
	0	0	1	0	X	X	X	Pass on Hash filter match
	0	0	1	1	X	X	X	Fail on Hash filter match
	0	1	1	0	X	X	X	Pass on Hash or Perfect/Group filter match
	0	1	1	1	X	X	X	Fail on Hash or Perfect/Group filter match
Multicast	1	X	X	X	X	X	X	Pass all packets
	X	X	X	X	X	1	X	Pass all packets
	0	X	X	0	0	0	X	Pass on Perfect/Group filter match and drop Pause packets if PCF = 0x
	0	0	X	0	1	0	X	Pass on Hash filter match and drop Pause packets if PCF = 0x
	0	1	X	0	1	0	X	Pass on Hash or Perfect/Group filter match and drop Pause packets if PCF = 0x
Multicast	0	X	X	1	0	0	X	Fail on Perfect/Group filter match and drop Pause packets if PCF = 0x
	0	0	X	1	1	0	X	Fail on Hash filter match and drop Pause packets if PCF = 0x
	0	1	X	1	1	0	X	Fail on Hash or Perfect/Group filter match and drop Pause packets if PCF = 0x

Table 775. Source address filtering

Packet type	PR	SAIF	SAF	SA filter operation
Unicast	1	X	X	Pass all packets.
	0	0	0	Pass status on Perfect or Group filter match but do not drop packets that fail
	0	1	0	Fail status on Perfect or Group filter match but do not drop packet
	0	0	1	Pass on Perfect or Group filter match and drop packets that fail
	0	1	1	Fail on Perfect or Group filter match and drop packets that fail

VLAN filtering

The MAC supports Perfect and Hash VLAN filtering. Refer to *Section 76.9.17: Programming guidelines to perform VLAN filtering on the receiver* for detailed programming steps.

VLAN tag Perfect filtering

In VLAN tag Perfect filtering, the MAC compares the VLAN tag of received packet and provides the VLAN packet status to the application. Based on the programmed mode, the MAC compares the lower 12 bits or all 16 bits of received VLAN tag to determine the perfect match.

If VLAN tag Perfect filtering is enabled, the MAC forwards the VLAN-tagged packets along with VLAN tag match status and drops the VLAN packets that do not match. You can also enable the inverse matching for VLAN packets by setting the VTIM bit of *VLAN tag Control register (ETH_MACVTCR)*. In addition, you can enable matching of S-VLAN tagged packets along with the default C-VLAN tagged packets by setting the ESVL bit of *VLAN tag Control register (ETH_MACVTCR)*. The VLAN packet status bit (bit 10 of RDES0) indicates the VLAN tag match status for the matched packets.

Note: The source or destination address (if enabled) has precedence over the VLAN tag filters. This means that a packet that fails the source or destination address filter is dropped irrespective of the VLAN tag filter results.

VLAN tag Hash filtering

The 16-bit VLAN Hash Table is used for group address filtering based on the VLAN tag. The VLAN tag Hash filtering feature can be enabled using the VTHM (VLAN tag Hash Table match enable) bit of the *VLAN tag Control register (ETH_MACVTCR)*. If the VTHM bit is set, the most significant four bits of CRC-32 of VLAN tag are used to index the content of the VLAN Hash Table register. A value of 1 in the VLAN Hash Table register, corresponding to the index, indicates that the VLAN tag of the packet matched and the packet should be forwarded. A value of 0 indicates that VLAN-tagged packet should be dropped.

Note: The 16 or 12 bits of VLAN Tag are considered for CRC-32 computation based on ETV bit in ETH_MACVTR register.

When ETV bit is reset, most significant four bits of CRC-32 of VLAN Tag are inverted and used to index the content of VLAN Hash table register (ETH_MACVHTR).

When ETV bit is set, most significant four bits of CRC-32 of VLAN Tag are directly used to index the content of VLAN tag Control register (ETH_MACVTCR).

The MAC also supports the inverse matching for VLAN packets. In the inverse matching mode, when the VLAN tag of a packet matches the Perfect or Hash filter, the packet should be dropped. If the VLAN perfect and VLAN Hash match are enabled, a packet is considered as matched if either the VLAN Hash or the VLAN perfect filter matches. When inverse

match is set, a packet is forwarded only when both perfect and Hash filters indicate mismatch.

Table 776 shows the different possibilities for VLAN matching and the final VLAN match status. When the RA bit of *Packet filtering control register (ETH_MACPFR)* is set, all packets are received and the VLAN match status is indicated in the VF bit of *RDES2 normal descriptor (write-back format)*. When the RA bit is not set and the VTFE bit is set in *Packet filtering control register (ETH_MACPFR)*, the packet is dropped if the final VLAN match status is Fail. In *Table 776*, value *X* means that this column can have any value.

When VLAN VID is programmed to 0 in the VL field of *VLAN tag Control register (ETH_MACVTCR)*, all VLAN-tagged packets are considered as perfect matched but the status of the VLAN Hash match depends on the VTHM and VTIM bits in *VLAN tag Control register (ETH_MACVTCR)*.

Table 776. VLAN match status(1)

VID	VLAN perfect filter match result	VTHM bit	VLAN hash filter match result	VTIM bit	Final VLAN match status
VID = 0	Pass	0	X	X	Pass
	Pass	1	X	0	Pass
	Pass	1	Fail	1	Pass
	Pass	1	Pass	1	Fail
VID!= 0	Pass	X	X	0	Pass
	Fail	0	X	0	Fail
	Fail	1	Fail	0	Fail
	Fail	1	Pass	0	Pass
	Fail	0	X	1	Pass
	Pass	X	X	1	Fail
	Fail	1	Pass	1	Fail
	Fail	1	Fail	1	Pass

1. In this table, ‘X’ represents any value.

Extended receive VLAN filtering and routing

The MAC receiver can classify the received packets based on VLAN tag, and steer them to a specific Rx DMA channel. The MAC compares the VLAN tag of the received frame with all the enabled and relevant filters, and provides a filtering result. If any of the perfect filters give a pass result and if the respective filter DMA channel number is enabled, the frame is routed to that DMA channel.

In addition to filtering, routing can also be done. For more details about routing, see Extended VLAN-based DMA selection in *Section 76.4.4: Multiple channels and queues support*.

Comparison modes

For each VLAN tag filter, the application has the following comparison options:

- It can program the MAC to compare an outer VLAN tag or an inner VLAN tag with the programmed VID.
- It can choose if 12 or 16 bits of the VID field need to be compared.
- Type check can be disabled or enabled for each filter. If enabled, the application can choose if the VID comparison is for SVLAN or CVLAN type frames only.

  For example, if a filter is enabled for 16-bit comparison, SVLAN type, and Outer VLAN tag, any single or double VLAN-tagged frames with Outer SVLAN tags are compared with this filter, and a pass or fail result is obtained.

Note: The inner VLAN tag comparison is applicable only if Double VLAN tag processing is enabled in VLAN tag Control register (ETH_MACVTCR).

Filtering

When the Extended RX VLAN filtering and routing feature is enabled, the application can enable both perfect and Hash filtering. The overall VLAN filter result is based on the perfect filter result and the Hash filter result (if enabled). The filter result is passed to the application as part of the status bits.

Perfect filtering is done based on the MAC VLAN tag filters defined in *VLAN tag data register (ETH_MACVTDR)*. For each VLAN tag filter, the MAC compares the relevant VLAN tag ID and provides a result. If any one of the VLAN tag filters gives a match, the frame is considered to have passed the VLAN tag filters. If the frame mismatches all the filters, the frame is considered to have failed the VLAN filter. This behavior is applicable only when the inverse filtering is not enabled in the *VLAN tag Control register (ETH_MACVTCR)*.

If inverse filtering is enabled and the frame has mismatched all the relevant filters, then it is considered to have passed the VLAN filter. If the frame matches any one of the relevant filters, then it is considered as a fail. If none of the enabled filters can perform a comparison or if none of the filters are enabled, then the frame is bypassed to the application.

The overall filter result and the values programmed in the VTFE and RA bitfields of the *Packet filtering control register (ETH_MACPFR)* determine if the frame must be dropped or forwarded to the application:

- When RA = 1 or VTFE = 0

  The MAC forwards the frame, irrespective of the filter result.
- When RA = 0 and VTFE = 1

  The MAC forwards the frame only if the VLAN tag filter status is pass. After forwarding a frame to the application, the relevant filter result is indicated through the status bits.

Programming guidelines for Extended VLAN filtering and routing on reception

See *Section 76.9.18: Programming guidelines for extended VLAN filtering and routing on reception on page 4144*.

VLAN filter status

The Extended receive VLAN filtering and routing feature provides two status bits to indicate the comparison result of the VLAN tags.

By default, the MAC indicates the VLAN filter status through one bit in the status (VF) in RDES2. When the Extended receive VLAN filtering and routing is enabled, two status bits

are used to indicate the comparison result of VLAN tags. The Outer VLAN tag filter pass and Inner VLAN tag filter pass bits are defined in the following positions in various configurations. The status indicated through these bits is highly dependent on how the fields are programmed:

- In RDES2:
  - Bit 15: Outer VLAN tag filter status
  - Bit 14: Inner VLAN tag filter status
- Outer VLAN tag filter status (OTS)
  - In perfect filtering, without inverse filtering enabled, if this bit is set, it indicates that the frame Outer VLAN tag has matched one of the VLAN tag filters.
  - If this bit is reset, it indicates that the frame Outer VLAN tag has either failed the relevant Outer VLAN tag filters or bypassed them.
  - If none of the filters are enabled for Outer VLAN tag comparison, then this bit is reset.
  - If inverse filtering is enabled and this bit is set, then the frame VLAN tag has passed all the relevant VLAN tag filters. If it is reset, then it has failed at least one of the filters or bypassed all the filters programmed for Outer VLAN tag comparison.
  - This bit is valid for both Single and Double VLAN-tagged frames.
- Inner VLAN Tag Filter Status (ITS)
  - In perfect filtering, without inverse filtering enabled, if this bit is set, it indicates that the frame Inner VLAN tag has matched one of the VLAN tag filters.
  - If this bit is reset, it indicates that the frame Inner VLAN tag has either failed the relevant Inner VLAN tag filters or bypassed them.
  - If none of the filters are enabled for Inner VLAN tag comparison, then this bit is reset.
  - If inverse filtering is enabled and this bit is set, then the frame VLAN tag has passed all the relevant VLAN tag filters. If it is reset, then it has failed at least one of the filters or bypassed all the filters programmed for Inner VLAN tag comparison.
  - This bit is valid for only Double VLAN-tagged frames, when Double VLAN processing is enabled.

The application must monitor the status bits and the programming to determine if the frame has passed or failed the VLAN filter.

Table 777 and *Table 778* show the possible filter combinations and the corresponding filter results. These tables explain the scenarios when Double VLAN processing and Hash VLAN filter are enabled.

Table 777 shows the possible values of OTS and ITS status bits when at least one perfect filter is enabled.

Legend for *Table 777*:

- VTIM - VLAN Tag Inverse Match Enable: bit 17 of *VLAN tag Control register (ETH_MACVTCR)*.
- HFO - Hash Filter enabled for Outer VLAN Tag Comparison: bit 26 and bit 27 of *VLAN tag Control register (ETH_MACVTCR)*.
- HFI - Hash Filter enabled for Inner VLAN Tag Comparison): bit 26 and bit 27 of *VLAN tag Control register (ETH_MACVTCR)*.
- PFO - Perfect filter comparison enabled for Outer VLAN Tag: any of the MAC VLAN tag filters in *VLAN tag data register (ETH_MACVTDR)* is enabled (bit 16 is set) and programmed for Outer VLAN tag comparison (bit 20 is set to 0).
- PFI - Perfect Filter comparison enabled for Inner VLAN Tag: any of the MAC VLAN tag filters in *VLAN tag data register (ETH_MACVTDR)* is enabled (bit 16 is set) and programmed for Inner VLAN tag comparison (bit 20 is set to 1).
- OTS - Outer VLAN tag filter status
- ITS - Inner VLAN tag filter status

Table 777. OTS and ITS bit values with at least one perfect filter enabled

VTIM	HFO	HFI	PFO	PFI	OTS	ITS
0	0	0	0	1	0	1/0
0	0	0	1	0	1/0	0
0	0	0	1	1	1/0	1/0
0	1	0	1	1	1/0	1/0
0	1	0	1	0	1/0	0
0	1	0	0	1	1/0	1/0
0	0	1	1	1	1/0	1/0
0	0	1	1	0	1/0	1/0
0	0	1	0	1	0	1/0
1	0	0	0	1	0	1/0
1	0	0	1	0	1/0	0
1	0	0	1	1	1/0	1/0
1	1	0	1	1	1/0	1/0
1	1	0	1	0	1/0	0
1	1	0	0	1	1/0	1/0
1	0	1	1	1	1/0	1/0
1	0	1	1	0	1/0	1/0
1	0	1	0	1	0	1/0

Table 778 shows the possible values of OTS and ITS status bits when none of the perfect filters are enabled and only the VLAN Hash filter is enabled.

Table 778. OTS and ITS bit values with only VLAN Hash filter enabled

VTIM	HFO	HFI	OTS	ITS
0	0	0	0	0
0	1	0	1/0	0
0	0	1	1/0	1/0
1	0	0	1/0	0
1	1	0	1/0	0
1	0	1	1/0	1/0

- When no perfect filters are enabled, any VLAN packet is considered to have bypassed the perfect filter.
- When VLAN Hash filter is enabled for one of the tags, the respective status bit depends on the result of the filter. Status bits are set to 0 when VLAN Hash filter is not enabled.
- The value 1/0 for the ITS/OTS field indicates that the final result depends on the result of the enabled relevant filter.

Example 1

The second row of *Table 778* indicates that at least one perfect filter is enabled for Outer VLAN tag comparison and none of the filters are enabled for Inner VLAN tag comparison. Inverse VLAN filtering is not enabled. The bit OTS is given as 1/0. If the received frame passes at the least one of the enabled Outer VLAN tag filters, then the bit is set to 1. If the frame does not pass any of the enabled Outer VLAN tag filters, then the bit is set to 0.

Example 2

The last row of *Table 778* indicates that inverse filtering is enabled, Hash filter and at least one perfect filter is enabled for Inner VLAN tag comparison. If the received frame Inner VLAN tag mismatches with both the Hash filter and all the enabled perfect filters, then the frame has the ITS bit set to 1. Otherwise, the bit is set to 0. OTS bit is set to 0 as comparison is not performed.

VLAN stripping

Each of the VLAN tags has individual control over stripping. The programming options of always strip, never strip, strip on pass and strip on fail are available. Inner or Outer VLAN tag stripping is based on the pass or fail results of the individual tag. If a tag is bypassed by all the relevant filters, stripping is not applicable for the tag.

- If strip-on-pass is enabled for the outer VLAN tag, stripping is done only if the Outer VLAN tag has passed the relevant filters. The Outer VLAN tag filter result bit is set.
- If strip-on-fail is enabled for the outer VLAN tag, stripping is done only if the Outer VLAN tag has failed the relevant filters. The Outer VLAN Tag Filter Result Bit is reset.
- If the Outer VLAN tag of the received frame is bypassed by the entire filter (no comparison has been made), the tag is not stripped, though the status bit is still 0.
- As multiple filters are enabled, it is possible that the received VLAN frame matches more than one filters. The VLAN tag value is not always deterministic from the filter status bits.
- If the application strips the VLAN tag based on the filter result, it might lose the VID. So, if stripping is enabled for any of the tags, the tag can be put in the status. The

application must enable the respective "VLAN Tag in Status" bit[24] or bit[31] in the *VLAN tag Control register (ETH_MACVTCR)*.

VLAN filter fail packet queue

When VLAN filtering is enabled, the VLAN filter fail packets can be routed to a programmable queue (VFFQ) when RA = 1 or VTFE = 0 and the enable bit (VFFQE) for the queue is set:

- RA and VTFE bits are implemented in *Packet filtering control register (ETH_MACPFR)*.
- VFFQ and VFFQE fields are implemented in *Rx Queue control register (ETH_MACRXQCR)*.

The packets that pass the VLAN filtering are routed based on the VLAN tag priority field. The VLAN tag priorities can be assigned to Rx queues by programming the PSRQ bitfield in *Rx queue control 2 register (ETH_MACRXQC2R)*. The packets that fail the VLAN filter are discarded if RA = 0 or VTFE = 1. However when RA = 1 or VTFE = 0, the VLAN filter fail packets are still forwarded to the application. In such a scenario, when the VLAN Filter Fail Queue Enable (VFFQE) bit is set, the VLAN filter fail packets are forwarded to the Rx queue number programmed in the VFFQ bit. If VFFQE = 0, the Rx queue number is determined by the VLAN priority mapping as per the PSRQ bitfield.

Table 779 shows the Rx queue routing table for Unicast tagged packets, with DA/SA filter enabled.

Table 779. Rx Queue routing table for Unicast-tagged packets(1)

RA	VFTE	SA/DA filter result	VLAN filter result	VFFQE	Queue routing
X	X	PASS	PASS	X	PSRQ
0	0	PASS	FAIL	0	PSRQ
0	0	PASS	FAIL	1	VFFQ
0	X	FAIL	X	X	DROPPED
0	1	PASS	FAIL	X	DROPPED
1	X	FAIL	X	0	UFFQ(2)/PSRQ
1	X	FAIL	X	1	UFFQ(2)/VFFQ
1	X	PASS	FAIL	0	PSRQ
1	X	PASS	FAIL	1	VFFQ

1. X = Don't care condition.
2. When UFFQE is enabled, else PSRQ.

Layer 3 and Layer 4 filtering

The MAC supports Layer 3 and Layer 4 based packet filtering. The Layer 3 filtering refers to the IP Source or Destination Address filtering in the IPv4 or IPv6 packets whereas Layer 4 filtering refers to the Source or Destination Port number filtering in TCP or UDP.

The Layer 3 and Layer 4 packet filtering feature automatically enables the IPC Full Checksum Offload Engine on the receive side. For Layer 3 or Layer 4 filtering operation, you must set the IPC bit of the *Operating mode configuration register (ETH_MACCR)* to enable the Rx Checksum Offload engine.

When Layer 3 and Layer 4 filtering is enabled, the packets are filtered in the following way:

- Matched packets

  The MAC forwards the packets that match all enabled fields to the application along with the status. The MAC gives the matched field status only if the IPC bit of *Operating mode configuration register (ETH_MACCR)* is set and one of the following conditions is true:

  – All enabled Layer 3 and Layer 4 fields match.

  – At least one of the enabled field matches and other fields are bypassed or disabled

    When multiple Layer 3 and Layer 4 filters are enabled, any filter match is considered as a match. If more than one filter matches, the MAC provides the status of the lowest filter where Filter 0 is the lowest filter and Filter 3 is the highest filter. For example, if Filter 0 and Filter 1 match, the MAC gives the status corresponding to filter 0.

Note: The source or destination address and VLAN tag filters (if enabled) have precedence over Layer 3 and Layer 4 filter. This means that a packet which fails the source or destination address or VLAN tag filter is dropped irrespective of the Layer 3 and Layer 4 filter results.

- Unmatched packets

  The MAC drops the packets that do not match any of the enabled fields. You can use the inverse match feature to block or drop a packet with specific TCP or UDP over IP fields and forward all other packets.The aborted or partial packets are dropped in the MTL Rx FIFO. If the Rx FIFO operates in the Threshold (cut-through) mode and the threshold is programmed to a small value. Such packet transfer to the application starts before the failed Layer 3 and Layer 4 filter results are available, the application may receive a partial packet with appropriate abort status.

- Non-TCP or UDP IP Packets

  By default, all non-TCP or UDP IP packets are bypassed from the Layer 3 and Layer 4 filters. You can optionally program the MAC to drop all non-TCP or UDP over IP packets.

Layer 3 filtering

The MAC supports perfect matching or inverse matching for IP Source Address and Destination Address. In addition, you can match the complete IP address or mask the lower bits matching, that is, compare all bits of the address except the specified lower mask bits.

For IPv6 packets filtering, you can enable the last four data registers of a register set to contain the 128-bit IP Source Address or IP Destination Address. The IP Source Address or Destination Address should be programmed in the order defined in the IPv6 specification, that is, the first byte of the IP Source Address or Destination Address in the received packet is in the higher byte of the register and the subsequent registers follow the same order.

For IPv4 packet filtering, you can enable the second and third data registers of a register set to contain the 32-bit IP Source Address and IP Destination Address. The remaining two data registers are reserved. The IP Source Address or Destination Address should be programmed in the order defined in the IPv4 specification, that is, the first byte of IP Source Address and Destination Address in the received packet in the higher byte of the respective register.

Layer 4 filtering

The MAC supports perfect matching or inverse matching for TCP or UDP Source and Destination Port numbers. However, you can program only one type (TCP or UDP) at a time. The first data register contains the 16-bit Source and Destination Port numbers of TCP or UDP, that is, the lower 16 bits for Source Port number and higher 16 bits for Destination Port number.

The TCP or UDP Source and Destination Port numbers should be programmed in the order defined in the TCP or UDP specification, that is, the first byte of TCP or UDP Source and Destination Port number in the received packet is in the higher byte of the register.

Layer 3 and Layer 4 filters register set

The MAC implements two sets of registers for Layer 3 and Layer 4 based packet filtering. In a register set, there is a control register, such as *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, to control the packet filtering. In addition, there are five address registers to program the Layer 3 and Layer 4 fields to be matched, such as:

- *Layer4 address filter 0 register (ETH_MACL4A0R)*
- *Layer3 address 0 filter 0 register (ETH_MACL3A00R)*
- *Layer3 address 1 filter 0 register (ETH_MACL3A10R)*
- *Layer3 address 2 filter 0 register (ETH_MACL3A20R)*
- *Layer3 address 3 filter 0 register (ETH_MACL3A30R)*

The second, and independent set of registers are: *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, *Layer 4 address filter 1 register (ETH_MACL4A1R)*, *Layer3 address 0 filter 1 Register (ETH_MACL3A01R)*, *Layer3 address 1 filter 1 register (ETH_MACL3A11R)*, *Layer3 address 2 filter 1 Register (ETH_MACL3A21R)* and *Layer3 address 3 filter 1 register (ETH_MACL3A31R)*.

76.5.5 IEEE 1588 timestamp support

The IEEE 1588 standard defines a precision time protocol (PTP) which allows precise synchronization of clocks in measurement and control systems implemented with technologies such as network communication, local computing, and distributed objects. The PTP applies to systems communicating by local area networks supporting multicast messaging, including (but not limited to) Ethernet. This protocol enables heterogeneous systems that include clocks of varying inherent precision, resolution, and stability to synchronize. The protocol supports system-wide synchronization accuracy in the submicrosecond range with minimal network and local clock computing resources.

This chapter contains the following sections:

- *IEEE 1588 timestamp support*
- *IEEE 1588 system time source*
- *IEEE 1588 auxiliary snapshots*
- *Flexible pulse-per-second output*

- *PTP timestamp offload function*
- *One-step timestamp*”

IEEE 1588 timestamp support

The Ethernet peripheral supports the IEEE 1588-2002 (version 1) and IEEE 1588-2008 (version 2). The IEEE 1588-2002 supports PTP transported over UDP/IP. The IEEE 1588 2008 supports PTP transported over Ethernet. The peripheral provides programmable support for both standards. It supports the following features:

- Support of both timestamp formats
- Optional snapshot of all packets or only PTP type packets
- Optional snapshot of only event messages
- Optional snapshot based on the clock type: ordinary, boundary, end-to-end transparent, and peer-to-peer transparent
- Optional selection of the node to act as master or slave for ordinary and boundary clock
- Identification of the PTP message type, version, and PTP payload in packets sent directly over Ethernet and sends the status
- Optional measurement subsecond time in digital or binary format

Clock types

The MAC supports the following clock types defined in the IEEE 1588-2008 specifications:

- Ordinary clock

  The ordinary clock of a domain supports a single copy of the protocol. It has a single PTP state and a single physical port. In typical industrial automation applications, an ordinary clock is associated with an application device such as a sensor or an actuator. In telecom applications, the ordinary clock can be associated with a timing demarcation device.

  The ordinary clock can be a grandmaster or a slave clock. It supports the following features:

  – Transmission and reception of PTP messages. The timestamp snapshot can be controlled as described in *Timestamp control register (ETH_MACTSCR)*.

  – Maintenance of the data sets such as timestamp values.

  The table below shows the messages for which you can take the timestamp snapshot on the receive side for master and slave nodes.

Table 780. Ordinary clock: PTP messages for snapshot

Master	Slave
Delay_Req	SYNC

  For an ordinary clock, you can take the snapshot of either of the following PTP message types: version 1 or version 2. You cannot take the snapshots for both PTP message types. You can take the snapshot by setting the TSVER2ENA bit and selecting the snapshot mode in *Timestamp control register (ETH_MACTSCR)*.

- Boundary clock

The boundary clock typically has several physical ports which communicate with the network. The messages related to synchronization, master-slave hierarchy, and signaling end in the protocol engine of the boundary clock. Such messages are not forwarded. The PTP message type status given by the MAC helps to identify the type of message and take appropriate action.

The boundary clock is similar to the ordinary clock except for the following features:

– The clock data sets are common to all ports of the boundary clock.

– The local clock is common to all ports of the boundary clock.

- End-to-end transparent clock

  The end-to-end transparent clock supports the end-to-end delay measurement mechanism between the slave clocks and the master clock. The end-to-end transparent clock forwards all messages like normal bridge, router, or repeater. The residence time of a PTP packet is the time taken by the PTP packet from the ingress port to the egress port.

  The residence time of a SYNC packet inside the end-to-end transparent clock is updated in the correction field of the associated Follow_Up PTP packet before it is transmitted. Similarly, the residence time of a Delay_Req packet, inside the end-to-end transparent clock, is updated in the correction field of the associated Delay_Resp PTP packet before it is transmitted. Therefore, the snapshot needs to be taken at both ingress and egress ports only for the messages mentioned in *Table 781*. You can take the snapshot by setting the SNAPTYPSEL bits to 10 in the *Timestamp control register (ETH_MACTSCR)*.

Table 781. End-to-end transparent clock: PTP messages for snapshot

PTP messages
SYNC
Delay_Req

- Peer-to-peer transparent clock

  In the peer-to-peer transparent clock, the computation of the link delay is based on an exchange of Pdelay_Req, Pdelay_Resp, and Pdelay_Resp_Follow_Up messages with the link peer.

  The peer-to-peer transparent clock differs from the end-to-end transparent clock in the way it corrects and handles the PTP timing messages. In all other aspects, it is identical to the end-to-end transparent clock.

  The residence time of the Pdelay_Req and the associated Pdelay_Resp packets is added and inserted into the correction field of the associated Pdely_Resp_Followup packet. Therefore, support for taking snapshot for the event messages related to Pdelay is added as shown in *Table 789*.

Table 782. Peer-to-peer transparent clock: PTP messages for snapshot

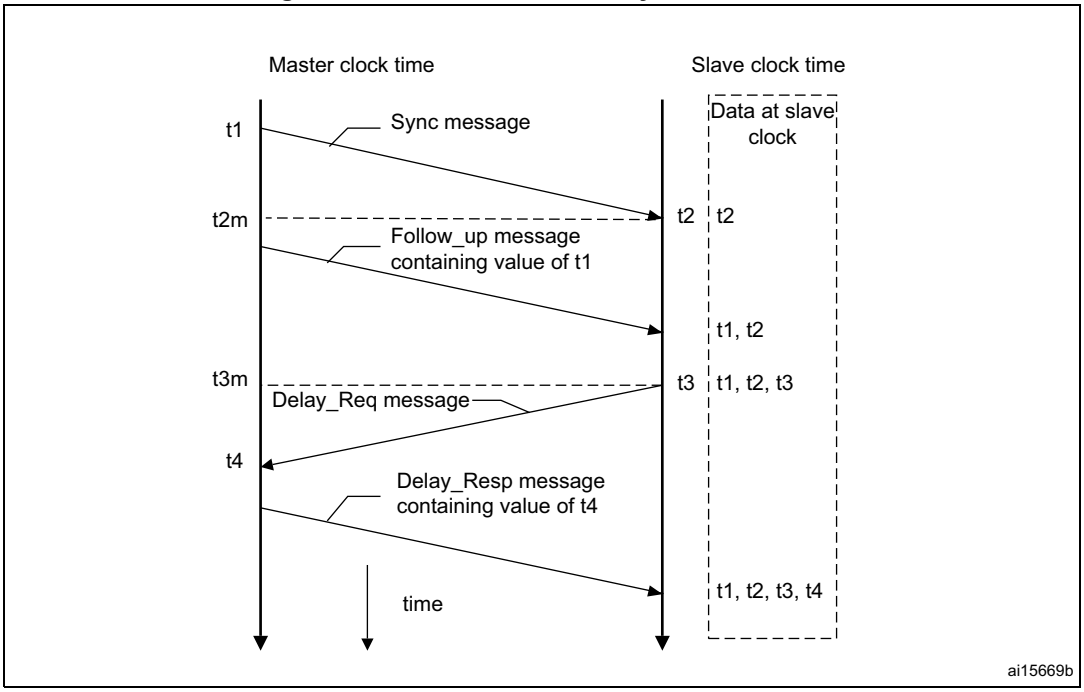
PTP messages
SYNC
Pdelay_Req
Pdelay_Resp

You can take the snapshot by setting the SNAPTYPESEL bit to 11 in *Timestamp control register (ETH_MACTSCR)*.

Delay request-response mechanism

The system or network is classified into the master and slave nodes for distributing the timing and clock information. *Figure 1026* shows the process that PTP uses for synchronizing a slave node with a master node by exchanging PTP messages.

Figure 1026. Networked time synchronization



As shown in *Figure 1026*, the PTP uses the following process:

1. The master broadcasts the PTP Sync messages to all its nodes. The Sync message contains the reference time information of the master. This message leaves the system of the master at t_1. This time must be captured for Ethernet ports at GMII or MII.
2. The slave receives the SYNC message and also captures the exact time, t_2, using its timing reference.
3. The master sends a Follow_up message to the slave, which contains t_1 information for later use.
4. The slave sends a Delay_Req message to the master and notes the exact time, t_3, at which this packet leaves the GMII or MII interface.

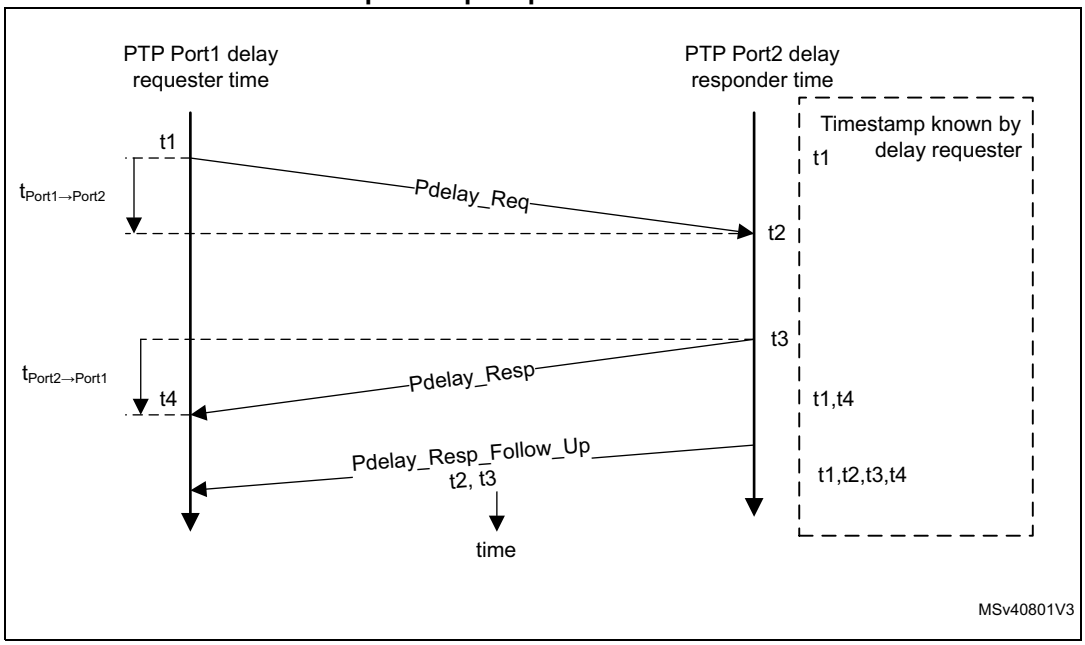
5. The master receives the message, capturing the exact time t_4, at which the message enters its system.
6. The master sends the t_4 information to the slave in the Delay_Resp message.
7. The slave uses the four values of t_1, t_2, t_3, and t_4 to synchronize its local timing reference with the timing reference of the master.

Most of the PTP implementation is done in the software above the UDP layer. However, the hardware support is required to capture the exact time when specific PTP packets enter or leave the Ethernet port at the GMII or MII interface. This timing information must be captured and returned to the software for proper implementation of PTP with high accuracy.

Peer-to-peer PTP transparent clock (P2P TC) message support

The IEEE 1588-2008 standard supports peer-to-peer PTP (Pdelay) message in addition to the Sync, Delay Request, Follow_up, and Delay Response messages. *Figure 1027* shows the method to calculate the propagation delay in clocks supporting peer-to-peer path correction.

Figure 1027. Propagation delay calculation in clocks supporting peer-to-peer path correction



As shown in *Figure 1027*, the propagation delay is calculated as follows:

1. Port 1 issues a Pdelay_Req message and generates a timestamp (t_1) for the Pdelay_Req message.
2. Port 2 receives the Pdelay_Req message and generates a timestamp (t_2) for this message.

3. Port 2 returns a Pdelay_Resp message and generates a timestamp (t_3) for this message.

   To minimize errors caused by frequency offset between the two ports, Port 2 returns the Pdelay_Resp message as quickly as possible after the receipt of the Pdelay_Req message. Port 2 returns any one of the following:

   – Difference between the timestamps t_2 and t_3 in the Pdelay_Resp message
   – Difference between the timestamps t_2 and t_3 in the Pdelay_Resp_Follow_Up message
   – Timestamps t_2 and t_3 in the Pdelay_Resp and Pdelay_Resp_Follow_Up messages, respectively

4. Port 1 generates a timestamp (t_4) on receiving the Pdelay_Resp message.
5. Port 1 uses all four timestamps to compute the mean link delay.

Timestamp correction

According to the IEEE 1588 specifications, a timestamp must be captured when the message timestamp point (leading edge of the first bit of the octet immediately following the Start Frame Delimiter octet) crosses the boundary between the node and the network.

As the MAC takes the timestamp at an internal point far from the actual boundary of the node and network, this captured timestamp is corrected/updated for the ingress/egress path latency (including the delay in the PHY layers). Further correction is done for the inaccuracies/errors introduced due to the clock (GMII Tx, Rx clock) being different at the capture point as compared to the PTP clock (clk_ptp_ref_i) that is used to generate the time. The resultant CDC (clock domain crossing) circuits add an error the depends on the clock period of the GMII and PTP clocks.

Ingress correction

In the receive side, the timestamp captured at the internal snapshot point is delayed (later in time) as compared to the time at which that packet SFD bit is received at the port boundary. Therefore, the captured timestamp must be reduced by the ingress latency and the errors in CDC sampling. This correction value must be determined/calculated by the software and written into the *Timestamp ingress correction nanosecond register (ETH_MACTSICNR)*.

The correction value consists of the following three components:

1. External latency in the PHY layer between boundary point and the input of the core

   If the PHY is compliant with the IEEE 802.3 Clause 45 MMD registers, it has registers indicating the maximum and minimum ingress latency. The software can read these registers and determine the average ingress latency in the PHY. Alternatively (if the PHY does not support these registers), the ingress latency must be determined from the PHY datasheet or timing characteristics.

2. Internal latency from the input of the core to the internal capture point

   The internal ingress latency can be read from the *Timestamp ingress latency register (ETH_MACTSILR)*. This is a read-only register that gives the latency in scaledNanoseconds format as defined in IEEE 1588 Clause 5.3.2. The latency differs based on the active PHY interface (such as MII or RMII) and the operating speed. Therefore, the software must read this register after any speed change in the MAC, to determine the current internal latency.

3. CDC synchronization

   The CDC synchronization error is almost equal to twice the clock-period of the PTP clock (clk_ptp_ref_i).

The values determined from these three components should be added by the software and must be written into the TSIC field of the *Timestamp ingress correction nanosecond register (ETH_MACTSICNR)*.

Note: The value written to the register must be negative (two's complement), because it has to be subtracted from the captured timestamp. The MAC receiver adds the value in this register to the captured timestamp and then gives the resultant value as the timestamp of the received packet.

When TSCTRLSSR bit in *Timestamp control register (ETH_MACTSCR)* is set, the nanoseconds field of the captured timestamp is in decimal format with a granularity of 1 ns. So bit 31 of TSIC must be set to 1 (for negative value) and bits 30 to 0 must be programmed with "10^9 – total ingress_correction_value[nanosecond part]" represented in binary. For example, if the required correction value is –5 ns, then the value is 0xBB9A C9FB.

When TSCTRLSSR bit in *Timestamp control register (ETH_MACTSCR)* is reset, the nanoseconds field of the captured timestamp is in binary format with a granularity of ~0.466 ns. Therefore, bits[30:0] must be written with "2^{31} – total ingress_correction_value" represented in binary with bit[31] = 1.

Egress correction

In the Transmit side, the timestamp captured at the internal snapshot point is earlier (advanced in time) as compared to the time at which that packet SFD bit is output at the port boundary. Therefore, the captured timestamp must be compensated by the egress latency and the errors in CDC sampling. This correction value must be determined/calculated by the software and written into the *Timestamp egress correction nanosecond register (ETH_MACTSECNR)*.

The correction value consists of the following three components:

1. External latency in the PHY layer between the output of the core and the boundary of the port and the network

   If the PHY is compliant with the IEEE 802.3 Clause 45 MMD registers, it has registers indicating the maximum and minimum egress latency. The software can read these registers and determine the average egress latency in the PHY. Alternatively (if the PHY does not support these registers), the egress latency must be determined from the PHY datasheet or timing characteristics.

2. Internal latency from the internal capture point and the output of the core

   This internal egress latency can be read from the *Timestamp egress latency register (ETH_MACTSELR)*. This is a read-only register that gives the latency in scaledNanoseconds format as defined in IEEE 1588 Clause 5.3.2. The latency differs based on the active PHY interface and the operating speed. Therefore, the software must read this register after any speed change in the MAC, to determine the current internal latency.

3. CDC synchronization error

   The timestamp correction because of synchronization is compensated by adding

   $$\text{EGRESS_SYNC_CORR} = (1 \times \text{PTP_CLK_PER} + 4 \times \text{TX_CLK_PER})$$

Table 783 lists the egress and ingress latency values for various PHY interfaces:

Table 783. Egress and ingress latency for PHY interfaces

PHY interface		Egress latency	Ingress latency
RGMII	1 Gbps	12	12
RGMII	100 Mbps	40	40
RGMII	10 Mbps	400	400
RMII	100 Mbit/s	40	120
RMII	10 Mbit/s	400	800

Frequency range of reference timing clock

The timestamp information are transferred across asynchronous clock domains, from the MAC clock domain to the application clock domain. Therefore, a minimum delay is required between two consecutive timestamp captures. This delay is four clock cycles of GMII or MII and three clock cycles of PTP clocks. If the delay between two timestamp captures is less than this delay, the MAC does not take a timestamp snapshot for the second packet.

The PTP clock frequency limitations are the following:

- Maximum PTP clock frequency

  The maximum PTP clock frequency is limited by the maximum resolution of the reference time (10 ns at 100 MHz). In addition, the resolution or granularity of the reference time source determines the accuracy of the synchronization. Therefore, a higher PTP clock frequency gives better system performance.

- Minimum PTP clock frequency

  The minimum PTP clock frequency depends on the time required between two consecutive SFD bytes and the time taken for synchronizing the time with the GMII or MII clock domain. This relationship is given by the following equation:

  3 × PTP clock period + 4 × GMII/MII clock period ≤ Minimum gap between two SFDs

  The GMII or MII clock frequency is fixed by IEEE specifications. Therefore, the minimum PTP clock frequency required for proper operation depends on the operating mode and operating speed of the MAC as shown in *Table 784*.

Table 784. Minimum PTP clock frequency example

Mode	Minimum gap between two SFDs	Minimum PTP frequency with internal timestamp
10 Mbit/s full-duplex	168 MII clocks (128 clocks for a 64-byte packet + 24 clocks of min IFG + 16 clocks of preamble)	5 MHz
10 Mbit/s half-duplex	48 MII clocks (8 clocks for a JAM pattern sent just after SFD because of collision + 24 IFG + 16 preamble)	5 MHz
100 Mbit/s full duplex	168 MII clocks (128 clocks for a 64-byte packet + 24 clocks of min IFG + 16 clocks of preamble)	5 MHz

Table 784. Minimum PTP clock frequency example (continued)

Mode	Minimum gap between two SFDs	Minimum PTP frequency with internal timestamp
100 Mbit/s half-duplex	48 MII clocks (8 clocks for a JAM pattern sent just after SFD because of collision + 24 IFG + 16 preamble)	5 MHz
1000 Mbps Full-duplex	84 GMII clocks (64 clocks for a 64-byte packet + 12 clocks of min IFG + 8 clocks of preamble)	5 MHz
1000 Mbps Half-duplex	24 GMII clocks (4 clocks for a JAM pattern sent just after SFD because of collision + 12 IFG + 8 preamble)	18.75 MHz

PTP processing and control

Table 785 shows the common message header for the PTP messages. This format is taken from the IEEE 1588-2008 specifications.

Table 785. Message format defined in IEEE 1588-2008

Bits								Octet	Offset
7	6	5	4	3	2	1	0		
transportSpecific				messageType				1	0
Reserved				versionPTP				1	1
messageLength								2	2
domainNumber								1	4
Reserved								1	5
flagField								2	6
correctionField								8	8
Reserved								4	16
sourcePortIdentity								10	20
sequenceId								2	30
controlField(1)								1	32
logMessageInterval								1	33

1. The controlField is used in version 1. In version 2, the messageType field is used for detecting different message types.

Some fields of the Ethernet payload can be used to detect the PTP packet type and control the snapshot to be taken. These fields are different for the following PTP packets:

- PTP packets over IPv4
- PTP frames over IPv6
- PTP packets over Ethernet

PTP packets over IPv4

Table 786 provides information about the fields that are matched to control the snapshot for the PTP packets sent over UDP over IPv4 for IEEE 1588 version 1 and 2. The octet positions for the tagged packets are offset by 4. This is based on the IEEE 1588-2008, Annex D and the message format defined in *Table 785*.

Table 786. Message format defined in IEEE 1588-2008

Matched field	Octet position	Matched value	Description
MAC packet type	12, 13	0x0800	IPv4 datagram
IP version and header length	14	0x45	IP version is IPv4
Layer 4 protocol	23	0x11	UDP
IP multicast address (IEEE 1588 version 1)	30, 31, 32, 33	0xE0, 0x00, 0x01, 0x81 (or 0x82 or 0x83 or 0x84)	Multicast IPv4 addresses allowed: 224.0.1.129 224.0.1.130 224.0.1.131 224.0.1.132
IP multicast address (IEEE 1588 version 2)	30, 31, 32, 33	0xE0, 0x00, 0x01, 0x81 (Hex) 0xE0, 0x00, 0x00, 0x6B (Hex)	PTP Primary multicast address: 224.0.1.129 PTP Pdelay multicast address: 224.0.0.107
UDP destination port	36, 37	0x013F, 0x0140	0x013F: PTP event messages(1) 0x0140: PTP general messages
PTP control field (IEEE 1588 version 1)	74	0x00, 0x01, 0x02, 0x03, 0x04	0x00: SYNC 0x01: Delay_Req 0x02: Follow_Up 0x03: Delay_Resp 0x04: Management
PTP message type field (IEEE 1588 version 2)	42 (nibble)	0x0, 0x1, 0x2, 0x3, 0x8, 0x9, 0xB, 0xC, 0xD	0x0: SYNC 0x1: Delay_Req 0x2: Pdelay_Req 0x3: Pdelay_Resp 0x8: Follow_Up 0x9: Delay_Resp 0xA: Pdelay_Resp_Follow_Up 0xB: Announce 0xC: Signaling 0xD: Management
PTP version	43 (nibble)	0x1 or 0x2	0x1: Supports PTP version 1 0x2: Supports PTP version 2

1. PTP event messages are SYNC, Delay_Req (IEEE 1588 version 1 and 2) or Pdelay_Req, Pdelay_Resp (IEEE 1588 version 2 only)

PTP frames over IPv6

Table 787 provides information about the fields that are matched to control the snapshots for the PTP packets sent over UDP over IPv6 for IEEE 1588 version 1 and 2. The octet positions for the tagged packets are offset by 4. This is based on the IEEE 1588-2008, Annex D and the message format defined in *Table 785*.

Table 787. IPv6-UDP PTP packet fields required for control and status

Matched field	Octet position	Matched value	Description
MAC packet type	12, 13	0x86DD	IP datagram
IP version	14 (bits [7:4])	0x6	IP version is IPv6
Layer 4 protocol	20(1)	0x11	UDP
PTP multicast address	38 – 53	FF0x:0:0:0:0:0:0:181 (Hex) FF02:0:0:0:0:0:0:6B (Hex)	PTP primary multicast address: FF0x:0:0:0:0:0:0:0:181 (Hex) PTP Pdelay multicast address: FF02:0:0:0:0:0:0:0:6B (Hex)
UDP destination port	56, 57a	0x013F, 0x140	0x013F: PTP event message 0x0140: PTP general messages
PTP control field (IEEE 1588 version 1)	94a	0x00, 0x01, 0x02, 0x03, or 0x04	0x00: SYNC 0x01: Delay_Req 0x02: Follow_Up 0x03: Delay_Resp 0x04: Management (version1)
PTP message type field (IEEE 1588 version 2)	62a (nibble)	0x0, 0x1, 0x2, 0x3, 0x8, 0x9, 0xB, 0xC, or 0xD	0x0: SYNC 0x1: Delay_Req 0x2: Pdelay_Req 0x3: Pdelay_Resp 0x8: Follow_Up 0x9: Delay_Resp 0xA: Pdelay_Resp_Follow_Up 0xB: Announce 0xC: Signaling 0xD: Management
PTP Version	63 (nibble)	0x1 or 0x2	0x1: Supports PTP version 1 0x2: Supports PTP version 2

1. The Extension header is not defined for PTP packets.

PTP packets over Ethernet

Table 788 provides information about the fields that are matched to control the snapshots for the PTP packets sent over Ethernet for IEEE 1588 version 1 and 2. The octet positions for the tagged packets are offset by 4. This is based on the IEEE 1588-2008, Annex D and the message format.

Table 788. Ethernet PTP packet fields required for control and status

Matched field	Octet position	Matched value	Description
MAC destination multicast address(1)	0–5	01-1B-19-00-00-00 01-80-C2-00-00-0E	All PTP messages can use any of the following multicast addresses(2): 01-1B-19-00-00-00 01-80-C2-00-00-0E(3)
MAC packet type	12, 13	0x88F7	PTP Ethernet packet
PTP control field (IEEE 1588 version 1)	46	0x00, 0x01, 0x02, 0x03, or 0x04	0x00: SYNC 0x01: Delay_Req 0x02: Follow_Up 0x03: Delay_Resp 0x04: Management
PTP message type field (IEEE 1588 version 2)	14 (nibble)	0x0, 0x1, 0x2, 0x3, 0x8, 0x9, 0xB, 0xC, or 0xD	0x0: SYNC 0x1: Delay_Req 0x2: Pdelay_Req 0x3: Pdelay_Resp 0x8: Follow_Up 0x9: Delay_Resp 0xA: Pdelay_Resp_Follow_Up 0xB: Announce 0xC: Signaling 0xD: Management
PTP version	15 (nibble)	0x1 or 0x2	0x1: Supports PTP version 1 0x2: Supports PTP version 2

1. The unicast address match of destination addresses (DA), programmed in MAC address 0 to 31, is used if the TSENMACADDR bit of *Timestamp control register (ETH_MACTSCR)* is set.
2. IEEE 1588-2008, Annex F
3. The MAC does not consider the PTP version 1 messages with Peer delay multicast address (01-80-C2-00-00-0E) as valid PTP messages.

Transmit path functions

The MAC captures a timestamp when the start packet delimiter (SFD) of a packet is sent on the GMII or MII interface. The packets, for which a timestamp has to be captured, can be controlled on per-packet basis. Each Transmit packet can be marked to indicate whether a timestamp should be captured for it.

The MAC does not process the transmitted packets to identify the PTP packets. The packets for which a timestamp has to be captured must be specified. The packets can be defined by using the control bits in the transmit descriptor (see *Section 76.10.3: Transmit descriptor*). The MAC returns the timestamp to the software inside the corresponding

Transmit descriptor, thus automatically connecting the timestamp to the specific PTP packet.

The 64-bit timestamp information is written to the TDES0 and TDES1 fields. The TDES0 field holds the 32 least significant bits of the timestamp.

Receive path functions

The MAC can be programmed to capture the timestamp of all packets received on the GMII or MII interface or to process packets to identify the valid PTP messages. The snapshot of the time to be sent to the application can be controlled by using the following options of the *Timestamp control register (ETH_MACTSCR)*:

- Enable snapshot for all packets
- Enable snapshot for IEEE 1588 version 1 or version 2 timestamp
- Enable snapshot for PTP packets transmitted directly over Ethernet or UDP-IP-Ethernet
- Enable timestamp snapshot for the received packet for IPv4 or IPv6
- Enable timestamp snapshot only for EVENT messages (SYNC, DELAY_REQ, PDELAY_REQ, or PDELAY_RESP)
- Enable the node to be a master or slave and select the snapshot type

  This feature controls the type of messages for which snapshots are taken.

Note: The peripheral also supports the PTP messages over VLAN packets.

Table 789 indicates the PTP messages for which a snapshot is taken depending on the SNAPTYPSEL field in *Timestamp control register (ETH_MACTSCR)*.

Table 789. Timestamp snapshot dependency on ETH_MACTSCR bits

SNAPTYPSEL	TSMSTRENA	TSEVNTENA	PTP messages
00	X	0	SYNC, Follow_Up, Delay_Req, Delay_Resp
00	0	1	SYNC
00	1	1	Delay_Req
01	X	0	SYNC, Follow_Up, Delay_Req, Delay_Resp, Pdelay_Req, Pdelay_Resp, Pdelay_Resp_Follow_Up
01	0	1	SYNC, Pdelay_Req, Pdelay_Resp
01	1	1	Delay_Req, Pdelay_Req, Pdelay_Resp
10	X	X	SYNC, Delay_Req
11	X	X	Pdelay_Req, Pdelay_Resp

The DMA returns the timestamp to the software application inside the corresponding receive descriptor. The extended status, containing the timestamp message status and the IPC status, is written in the RDES1 normal descriptor and the snapshot of the timestamp is written in RDES0 and RDES1 fields of the context descriptor. The RDES0 field holds the 32 least significant bits of the timestamp.

Programming guidelines for IEEE 1588 timestamping (system time correction)

See *Section : System time correction* in *Section 76.9.10: Programming guidelines for IEEE 1588 timestamping on page 4132*.

IEEE 1588 system time source

To get a snapshot of the time, the MAC requires a reference time in 64-bit format as defined in the IEEE 1588-2002 (80-bit format as defined in the IEEE 1588-2008).

Description of IEEE 1588 system time source

The peripheral uses the reference clock input and uses it to internally generate the Reference time (also called the system time) and capture timestamps.

The timestamp has the following fields:

- UInteger48 seconds field

  The seconds field is the integer portion of the timestamp in units of seconds. It is 48-bit wide. For example, 2.000000001 seconds are represented as seconds Field = 0x0000 0000 0002.
- UInteger32 nanosecondsField

  The nanoseconds field is the fractional portion of the timestamp in units of nanoseconds. For example, 2.000000001 seconds are represented as nanoSeconds = 0x0000 0001.

  The nanoseconds field supports the following two modes:
  - **Digital rollover mode**: In this mode, the maximum value in the nanoseconds field is 0x3B9A C9FF, that is, (10e9-1) nanoseconds.
  - **Binary rollover mode**: In this mode, the nanoseconds field rolls over and increments the seconds field after value 0x7FFF FFFF. Accuracy is ~0.466 ns per bit.

    These modes can be set through TSCTRLSSR bit in *Timestamp control register (ETH_MACTSCR)*.

System time register module

The 64-bit PTP time is updated using the PTP input reference clock, clk_ptp_ref_i. This PTP time is used as a source to take snapshots (timestamps) of the Ethernet frames being transmitted or received at the GMII.

The system time counter can be initialized or corrected using either the coarse or the fine correction method.

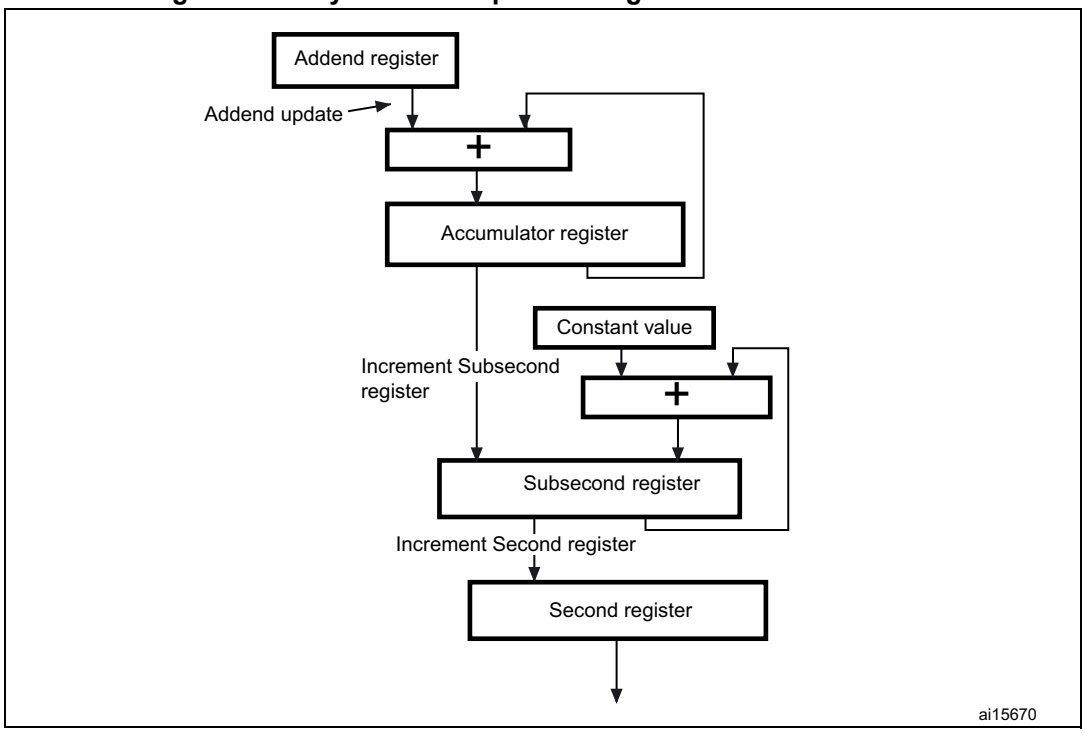
In the coarse correction method, the initial value or the offset value is written to the timestamp update register. For initialization, the system time counter is programmed with the value in the timestamp update registers, whereas for system time correction the offset value (timestamp update register) is added to or subtracted from the system time.

In the fine correction method, the slave clock (reference clock) frequency drift with respect to the master clock (as defined in IEEE 1588-2002 specifications) is corrected over a period of time, unlike in the coarse correction method where it is corrected in a single clock cycle. The longer correction time helps maintain linear time and does not introduce drastic changes (or a large jitter) in the reference time between PTP Sync message intervals. In this method, an accumulator sums up the contents of the Addend register as shown in *Figure 1028*. The arithmetic carry that the accumulator generates is used as a pulse to

increment the system time counter. The accumulator and the addend are 32-bit registers. The accumulator acts as a high-precision frequency multiplier or divider.

This system time update algorithm is shown in *Figure 1028*.

Figure 1028. System time update using fine correction method



The system time update logic requires a 50 MHz clock frequency to achieve 20 ns accuracy. The frequency division is the ratio of the reference clock frequency to the required clock frequency. For example, if the reference clock (clk_ptp_ref_i) is 66 MHz, this ratio is calculated as 66 MHz/50 MHz = 1.32. Therefore, the default addend value to be set in the register is 2^{32} / 1.32, 0xC1F07C1F.

If the reference clock drifts lower, for example, to 65 MHz, the ratio is 65 / 50, that is 1.3 and the value to set in the addend register is 2^{32} / 1.30, or 0xC4EC 4EC4.

If the clock drifts higher, for example to 67 MHz, the addend register must be set to 0xBF0B 7672. When there is not clock drift, the default addend value of 0xC1F0 7C1F (2^{32} / 1.32) must be programmed.

In *Figure 1028*, the constant value used to accumulate the subsecond register is decimal 43, which achieves a system time accuracy of 20 ns (in other words, it is incremented in 20 ns steps).

The software must calculate the drift in frequency based on the SYNC messages and accordingly update the Addend register.

Initially, the slave clock is set with FreqCompensationValue0 in the Addend register. This value is as follows:

$$\text{FreqCompensationValue}_0 = 2^{32} / \text{FreqDivisionRatio}$$

If MasterToSlaveDelay is initially assumed to be the same for consecutive Sync messages, the algorithm given in this section must be applied. After a few Sync cycles, frequency lock occurs. The slave clock can then determine a precise MasterToSlaveDelay value and re-synchronize with the master using the new value.

The algorithm is as follows:

1. At time $MasterSyncTime_n$ the master sends the slave clock a SYNC message. The slave receives this message when its local clock is $SlaveClockTime_n$ and computes $MasterClockTime_n$ as follows:

$$MasterClockTime_n = MasterSyncTime_n + MasterToSlaveDelay_n$$

2. The master clock counts for current Sync cycle, $MasterClockCount_n$ is

$$MasterClockCount_n = MasterClockTime_n - MasterClockTime_{n-1}$$

(assuming that MasterToSlaveDelay is the same for Sync cycles n and n – 1)

3. The slave clock count for current Sync cycle, $SlaveClockCount_n$ is

$$SlaveClockCount_n = SlaveClockTime_n - SlaveClockTime_{n-1}$$

4. The difference between master and slave clock counts for current Sync cycle, $ClockDiffCount_n$ is

$$ClockDiffCount_n = MasterClockTime_n - SlaveClockTime_n$$

5. The frequency-scaling factor for slave clock, $FreqScaleFactor_n$ is

$$FreqScaleFactor_n = (MasterClockCount_n + ClockDiffCount_n) / SlaveClockCount_n$$

6. The frequency compensation value for Addend register, $FreqCompensationValue_n$ is

$$FreqCompensationValue_n = FreqScaleFactor_n \times FreqCompensationValue_{n-1}$$

In theory, this algorithm achieves the lock in one Sync cycle. However, it may take several cycles, because of changing network propagation delays and operating conditions. This algorithm is self-correcting. If the slave clock is initially set to an incorrect value by the master, the algorithm corrects it at the cost of additional Sync cycles.

Refer to *Section 76.9.10: Programming guidelines for IEEE 1588 timestamping* for detailed programming steps.

IEEE 1588 auxiliary snapshots

The auxiliary snapshot feature enables to store a snapshot of the system time based on an external event. The event is considered to be the rising edge of the eth_ptp_trgx (where x = 1 to 4) sideband signal.

Up to four auxiliary snapshot inputs can be configured and up to four snapshots can be stored. A FIFO is accessible through registers: *Auxiliary timestamp seconds register (ETH_MACATSSR)* and *Auxiliary timestamp nanoseconds register (ETH_MACATSNR)*.

The snapshots taken for any input are stored in a common FIFO; only 64 bits are kept. The application can read the *Timestamp status register (ETH_MACTSSR)* to know the timestamp of which input is available for reading at the top of this FIFO.

When a snapshot is stored, the MAC indicates this to the application with an interrupt. The value of the snapshot is read through a FIFO register access. If the FIFO becomes full and an external trigger to take the snapshot is asserted, a snapshot trigger-missed status (ATSSTM) is set in the *Timestamp status register (ETH_MACTSSR)*. This indicates that the latest auxiliary snapshot of the timestamp is not stored in the FIFO. The latest snapshot is not written to the FIFO when it is full.

When an application reads the 64-bit timestamp from the FIFO, the space becomes available to store the next snapshot. You can clear a FIFO by setting the ATSFC bit in *Auxiliary control register (ETH_MACACR)*. When multiple snapshots are present in the FIFO, the count is indicated in bits[27:25] of *Timestamp status register (ETH_MACTSSR)*.

Flexible pulse-per-second output

The MAC supports either a fixed pulse-per-second output mode (also called fixed mode) or a flexible pulse-per-second output mode for the ETH_PPS_OUT and eth_ptp_pps_out1 outputs:

- Fixed pulse-per-second output

  In this mode, only the frequency of the PPS output can be changed by setting the PPSCTRL0 field in the *PPS control register (ETH_MACPPSCR)*.
- Flexible pulse-per-second output

  In this mode, the software has the flexibility to program the start or stop time, width, and interval of the pulse generated on the eth_ptp_pps_out output:

  The start and stop times are programmed through *PPS x target time seconds register (ETH_MACPPSTTSxR)* and *PPS x target time nanoseconds register (ETH_MACPPSTTNxR)*.

  The PPS width and interval are programmed in terms of granularity of system time (number of the units of subsecond increment value) through *PPS x width register (ETH_MACPPSWxR)* and *PPS x interval register (ETH_MACPPSIxR)*, respectively.

Note: By default, the peripheral is in Fixed mode and indicates one second interval. When Fixed mode is selected by clearing PPSEN0 to 0 in the PPS control register (ETH_MACPPSCR):

- *the output on all PPS outputs is controlled by the value programmed in the PPSCTRL_PPSCMD field. Independent control of individual PPS output is not supported in Fixed mode.*
- *PPS x target time seconds register (ETH_MACPPSTTSxR)* and *PPS x target time nanoseconds register (ETH_MACPPSTTNxR) are used only for generating target time reached interrupt; they are not used for PPS output generation.*
- *TRGTMODSEL0/1/2/3 must be programmed to 0.*
- *the frequency of the PPS output can be changed by setting the PPSCTRL0 field in the PPS control register (ETH_MACPPSCR).*

Description of flexible pulse-per-second (PPS) output

The peripheral supports the following features with the flexible PPS outputs:

- Control of up to two independent PPS outputs, available externally (ETH_PPS_OUT) and internally (eth_ptp_pps_out1)
- Programming the start or stop time in terms of system time.
- Programming the start point of the single pulse and start and stop points of the pulse train in terms of 64-bit system time. The Target Time registers are used to program the start and stop time.
- Programming the stop time in advance, that is, the stop time can be programmed before the actual start time has elapsed.
- Programming the width between the rising edge and corresponding falling edge of PPS signal output in terms of number of units of subsecond increment value programmed in the *Subsecond increment register (ETH_MACSSIR)*. The pulse width can be programmed from 1 to 232–1 units of subsecond increment value.
- Programming the interval, between the rising edges of PPS signal, in terms of number of units of subsecond increment value. You can program the interval between pulses from 1 to 232–1 units of subsecond increment value.
- Option to cancel the programmed PPS start or stop request.
- Error if the start or stop time being programmed has already elapsed.

Note: The PTP reference clock mentioned in the following sections is the clock at which the system time is updated. When the TSCFUPDT bit of Timestamp control register (ETH_MACTSCR) is set to 0, this clock is similar to the clk_ptp_ref_i clock. In Fine correction mode, this is the clock tick at which the system time is updated (using Subsecond increment register (ETH_MACSSIR) (as shown in Figure 1028).

Refer to *Section 76.9.14: Programming guidelines for flexible pulse-per-second (PPS) output* for further details on how configuring flexible pulse output.

PPS start and stop times

The initial start time can be programmed in the Target Time registers.

Each flexible PPS output is associated to independent Target Time registers, *PPS x target time seconds register (ETH_MACPPSTTSxR)* and *PPS x target time nanoseconds register (ETH_MACPPSTTNxR)*.

If required, the start or stop time can be programmed again. However, this can be done only after the earlier programmed value is synchronized with the PTP clock domain. Bit 31 of *PPS x target time nanoseconds register (ETH_MACPPSTTNxR)* indicates that the synchronization is complete. This enables to program the start or stop time in advance even before the earlier stop or start time has elapsed.

To ensure proper PPS signal output, it is recommended to program advanced system time for the start or stop time. If the application programs a start or stop time that has already elapsed, the MAC sets an error status bit indicating the programming error. If enabled, the MAC also sets the Target Time Reached interrupt event. The application can cancel the start or stop request only if the corresponding start or stop time has not elapsed. If the time has elapsed, the cancel command has no effect.

PPS width and interval

The PPS width and interval are programmed in terms of granularity of system time, that is, number of the units of subsecond increment value. For example, to obtain a PPS pulse width of 40 ns and an interval of 100 ns with a PTP reference clock of 50 MHz, program the width and interval to values 2 and 5, respectively. Smaller granularity can be achieved by using a faster PTP reference clock.

Before giving the command to trigger a pulse or pulse train on the PPS output, program or update the interval and width of the PPS signal output.

Each flexible PPS output is associated to independent interval and width registers, *PPS x interval register (ETH_MACPPSIxR)* and *PPS x width register (ETH_MACPPSWxR)*.

PTP timestamp offload function

This feature enables the automatic generation of specific PTP packets to be performed, when the MAC operates as a specific node in the PTP network.

These packets can be generated periodically or triggered by the host software. In other modes, this feature can parse the incoming PTP packets on the receiver, and automatically generate and respond to the required PTP packets. It helps to offload certain PTP node functions with better accuracy and lower response latency.

The PTP offload feature is selected through *PTP offload control register (ETH_MACPOCR)*. 80-bit PTP node identity is configured through the following three registers: *PTP source port identity 0 register (ETH_MACSPI0R)*, *PTP Source port identity 1 register (ETH_MACSPI1R)* and *PTP Source port identity 2 register (ETH_MACSPI2R)*.

Description of PTP offload function

Depending on the programmed mode, the MAC generates PTP Ethernet messages periodically or from the application, or based on reception of a particular PTP message. *Table 790* indicates the PTP message generation criteria.

Table 790. PTP message generation criteria

Programming			Mode	Criteria for generation of PTP messages	PTP message type generated
SNAPTYPSEL	TSMSTRENA	TSEVNTENA			
00	0	1	Ordinary or Boundary Slave	SYNC message reception	Delay_Req
00	1	1	Ordinary or Boundary Master	Periodic or on trigger from application	SYNC
				Delay_Req message reception	Delay_Resp

Table 790. PTP message generation criteria (continued)

Programming			Mode	Criteria for generation of PTP messages	PTP message type generated
SNAPTYPSEL	TSMSTRENA	TSEVNTENA			
01	0	1	Transparent Slave	Periodic or on trigger from application	Pdelay_Req
				Pdelay_Req message reception	Pdelay_Resp
				SYNC message reception	Delay_Req
01	1	1	Transparent Master	Periodic or on trigger from application	Pdelay_Req
				Pdelay_Req message reception	Pdelay_Resp
				Periodic or on trigger from application	SYNC
				Delay_Req message reception	Delay_Resp
11	X	X	Peer-to-Peer Transparent	Periodic or on trigger from application	Pdelay_Req
				Pdelay_Req message reception	Pdelay_Resp
All other programming combinations are invalid for the PTP offload feature.					

Note: Clocks supporting peer delay mechanism must not generate delay request/delay response messages, according to IEEE 1588-2008 specifications. However, the peripheral controller supports this for flexibility, with a programmable control bit (DRRDIS) in the PTP offload control register (ETH_MACPOCR).

The DRRDIS bit can be used to control the response generation for delay request/delay response message. For example, in transparent slave mode, delay request is generated in response to received SYNC *only when the bit is reset.*

When the MAC is set as an Ordinary or Boundary Slave clock in the PTP network, it can respond to the reception of SYNC messages with an automatic generation and transmission of the corresponding Delay_Req message. Similarly, various other modes of operation are explained in *Table 790*.

The MAC supports the multicast communication model for the generation of SYNC and Pdelay_Req PTP messages. For instance, the Destination Address field of the generated PTP over Ethernet packet is the defined special multicast addresses (0x011B 1900 0000 for

all except peer delay mechanism messages and 0x0180 C200 000E for peer delay mechanism messages).

When the MAC responds to received SYNC, Delay_Req and Pdelay_Req PTP messages with special multicast destination address, it also uses the corresponding special multicast address in the DA field of the automatically generated Delay_Req, Delay_Resp, and Pdelay_Resp PTP messages, respectively.

When the MAC responds to received SYNC, Delay_Req and Pdelay_Req PTP messages with unicast destination address, it takes the SA field of the received packets and makes them as the DA field of the automatically generated Delay_Req, Delay_Resp, and Pdelay_Resp PTP messages, respectively.

At the same time, all the received PTP messages are forwarded to the application along with Rx status, indicating whether the response was generated by the MAC, if it satisfies the packet filtering logic of the MAC receiver

When the MAC automatically generates a PdelayReq or responds with a Delay_Req, the egress timestamp of these two PTP messages are provided in the Tx TS status (Tx Timestamp Status register and interrupt generated).

In addition to messageType and versionPTP fields match for basic PTP over Ethernet message detection, the following additional fields are matched to qualify the received PTP message type:

1. The domainNumber field is checked for a match against the value programmed in the CSR.
2. The twoStepFlag in flagField field is checked for one-step indication (0b0).
3. The transportSpecific field is checked for Default PTP over Ethernet (0b0000) or 802.1AS mode (0b1111) when enabled.

PTP packet generation

This section explains the format and content of the automatically generated PTP packets by the MAC when this mode is enabled. It provides the template of the common PTP message header, as well as the detailed description of the fields of the specific PTP packets generated.

Table 791. Common PTP message header fields

Bits								Octets	Offset
7	6	5	4	3	2	1	0		
transportSpecific				messageType				1	0
Reserved				versionPTP				1	1
messageLength								2	2
domainNumber								1	4
Reserved								1	5
flagField								2	6
correctionField								8	8
Reserved								4	16
sourcePortIdentity								10	20

Table 791. Common PTP message header fields (continued)

Bits								Octets	Offset
7	6	5	4	3	2	1	0		
sequenceId								2	30
sequenceId								2	30
controlField								1	32
logMessageInterval								1	33

PTP message header fields

- **messageType**

  The following encoded values are used for PTP message types:
  - SYNC: 0000
  - Delay_Req: 0001
  - Pdelay_Req: 0010
  - Pdelay_Resp: 0011
  - Delay_Resp: 1001
- **transportSpecific**

  The following transport protocol encoding is used:
  - Default PTP over Ethernet: 0000
  - 802.1AS mode: 0001
- **versionPTP**

  It is always set to 2 because PTP version 2 is supported.
- **domainNumber**

  This field contains the value from the *PTP offload control register (ETH_MACPOCR)*.
- **flagField**

  The following values are used:
  - alternateMasterFlag (Octet 0 bit 0): 0 for SYNC and Delay_Resp
  - twoStepFlag (Octet 0 bit 1): 0 for SYNC and Pdelay_Resp
  - unicastFlag (Octet 0 bit 2): 0 for Multicast Address, 1 for Unicast Address
- **correctionField**

  For more information, see *Table 792*.
- **sourcePortIdentity**

  This field takes the value programmed in the *PTP source port identity 0 register (ETH_MACSPI0R)*, *PTP Source port identity 1 register (ETH_MACSPI1R)* and *PTP Source port identity 2 register (ETH_MACSPI2R)*.
- **sequenceId**

  Pdelay_Resp and Delay_Resp use the same sequenceId field from received Pdelay_Req and Delay_Req PTP messages. For SYNC/Delay_Req, Pdelay_Req, a separate sequenceId counter is maintained. These sequenceId counters get incremented by 1 every time the corresponding message is generated and transmitted.
- **controlField**

The following encoded values are used for controlField:

- SYNC: 0000 0000
- Delay_Req: 0000 0001
- Pdelay_Req: 0000 0010
- Pdelay_Resp: 0000 0101
- Delay_Resp: 0000 0011

- **logMessageInterval**
  - SYNC:

    This field contains logSyncInterval from the corresponding MAC_Log_Message_Interval register.
  - Delay_Resp:

    This field contains the sum of DRSYNCR and logSyncInterval value taken from the *Log message interval register (ETH_MACLMIR)* for a multicast PTP message and 0111 1111 for unicast PTP message.
  - Delay_Req, Pdelay_Req and Pdelay_Resp: 0111 1111

    where logSyncInterval = log2 (Mean Value of Interval in seconds)

The MAC supports values of –15 to 15 for logSyncInterval fields, which translates to a range from 32.768 micro second (2–15) to 215 second. For a given value of log sync interval (N), the time interval between two SYNC packets is given by the following:

- $2^{(30+N)}$ ns, when N is negative (–1 to –15)
- 2^{N} seconds, when N is positive (0 to 15)

For example:

- When logSyncInterval is programmed to 1, the interval is 2^{1}; therefore, the SYNC message is sent once every 2 seconds.
- When logSyncInterval is programmed to -1, the interval is 2^{-1} = 0.536 seconds; therefore, the SYNC message is sent once every 536 milliseconds. The value is 0.536 seconds, because 2^{-30} = 1 ns.
- When logSyncInterval is programmed to –-5, the interval is 2^{-5} = 33.55 ms; therefore, the SYNC message is sent once every 33.55 ms.

Note: The MAC uses the PTP system time to generate the intervals for periodic packet transmission. For negative values of log message interval programmed, the generated period may deviate from the value given by the equation $2^{(30+N)}$, because of the non-binary nature of the nanoseconds field of the system time.

PTP message-specific fields

The message-specific fields are the following:

- **messageLength**

  There is no suffix supported, so this field contains the length of the PTP message that includes 34-byte PTP common header and the body specific to the message type.

  For SYNC and Delay_Req packets, this field contains 44, whereas for Delay_Resp, Pdelay_Req and Pdelay_Resp, it contains 54.
- **originTimestamp**

  This field is the captured egress timestamp for SYNC, Delay_Req, and Pdelay_Req PTP messages.
- **receiveTimestamp**

  For Delay_Resp PTP message, this is the ingress timestamp of the corresponding received Delay_Req PTP message.
- **requestingPortIdentity**

  For Delay_Resp and Pdelay_Resp PTP messages, this is the sourcePortIdentity field taken from the corresponding received Delay_Req and Pdelay_Req PTP messages.
- **requestReceiptTimestamp**

  For the Pdelay_Resp PTP message, this field is set to 0.

One-step timestamp

The MAC supports the one-step timestamp feature that enables to identify the offset in the packet and inserts the timestamp received from the application at that offset.

MAC Transmit PTP mode for one-step timestamp

Depending upon the type of message and its mode, the MAC updates the following fields of Transmit PTP packets:

- correctionField in the PTP header of messages
- originTimestamp in SYNC, Delay_Req, and Pdelay_Req messages

Table 792 shows how the PTP mode is selected based on the settings of SNAPTYPSEL, TSMSTRENA, and TSEVNTENA bits of the *Timestamp control register (ETH_MACTSCR)* and the fields that are updated for the incoming PTP packets based on the message type in that mode, during the one-step timestamping operation.

Table 792. MAC Transmit PTP mode and one-step timestamping operation

Programming			Mode	Per packet control(1)			Messages processed on transmission
SNAPTYPSEL	TSMSTRENA	TSEVNTENA		TTSE (2)	OSTC (3)	TTS(4)	
X	X	X	N/A	1	X	X	Timestamp is captured and returned to application
X	X	X	N/A	X	0	X	OST operation is not performed (PTP packet is not modified)

Table 792. MAC Transmit PTP mode and one-step timestamping operation (continued)

Programming			Mode	Per packet control[1]			Messages processed on transmission
SNAPTYPSEL	TSMSTRENA	TSEVNTENA		TTSE[2]	OSTC[3]	TTS[4]	
00	X	0	End-to-end transparent	0	1	Ingress TS	Sync (correction field for residence time and ingress asymmetric correction)
							Delay_Req (correction field for residence time and egress asymmetric correction)
00	0	1	Ordinary or Boundary Slave	1	1	X	Delay_Req (originTimestamp field) Delay_Req (correction field for egress asymmetric correction)
00	1	1	Ordinary or Boundary Master	0	1	X	Sync (originTimestamp field) Sync (correction field for subnanosecond correction)
01	X	0	End-to-end Transparent with support for peer delay mechanism	0	1	Ingress TS	Sync (correction field for residence time and ingress asymmetric correction)
						Ingress TS	Pdelay_Req (correction field for residence time and egress asymmetric correction)
						Ingress TS	Pdelay_Resp (correction field for residence time and ingress asymmetric correction)
01	0	1	Ordinary or Boundary Slave with support for peer delay mechanism or Peer-to-peer transparent	0	1	Ingress TS	Sync (correction field for residence time and ingress asymmetric correction) (applicable only for Peer to Peer transparent clock operation)
				1	1	X	Delay_Req (originTimestamp field) Delay_Req (correction field for egress asymmetric correction)
				1	1	X	Pdelay_Req (originTimestamp field) Pdelay_Req (correction field for egress asymmetric correction)
				0	1	Ingress TS for Pdelay_Req	Pdelay_Resp (correction field for turnaround time and ingress asymmetric correction)

Table 792. MAC Transmit PTP mode and one-step timestamping operation (continued)

Programming			Mode	Per packet control[1]			Messages processed on transmission
SNAPTYPSEL	TSMSTRENA	TSEVNTENA		TTSE[2]	OSTC[3]	TTS[4]	
01	1	1	Ordinary or Boundary Master with support for peer delay mechanism	0	1	X	Sync (originTimestamp field) Sync (correction field for subnanosecond correction)
				1	1	X	Pdelay_Req (originTimestamp field) Pdelay_Req (correction field for egress asymmetric correction)
				0	1	Ingress TS for Pdelay_Req	Pdelay_Resp (correction field for turnaround time and ingress asymmetric correction)
10	X	X	End-to-end transparent	0	1	Ingress TS	Sync (correction field for residence time and ingress asymmetric correction)
						Ingress TS	Delay_Req (correction field for residence time and egress asymmetric correction)
11	X	X	Peer-to-peer transparent	0	1	Ingress TS	Sync (correction field for residence time and ingress asymmetric correction)
				1	1	X	Pdelay_Req (originTimestamp field) Pdelay_Req (correction field for egress asymmetric correction)
				0	1	Ingress TS for Pdelay_Req	Pdelay_Resp (correction field for turnaround time and ingress asymmetric correction)

1. The per-packet control values provided here are the recommended settings used by devices in typical PTP operation and for the programmed mode.
2. TTSE represents TTSE bit of TDES2 transmit normal descriptor. The TTSE function is independent from the OST function and the programmed operation mode for OST. The MAC captures and returns the timestamp when the TTSE bit is set.
3. OSTC represents OSTC bit of TDES3 transmit context descriptor
4. TTS represents the timestamp value provided in the TTSH, TTSL fields of TDES0 and TDES1 transmit normal descriptor (write-back format).

Note: Residence time/ turnaround time is calculated as the difference between the captured timestamp (egress timestamp) and the ingress timestamp. Clocks supporting peer delay mechanism do not use delay request or response, but it is included in OST for flexibility.

Enabling one-step timestamp

The one-step timestamp feature can be enabled for a given packet by setting bit 20 (OSTC) in TDES3 context descriptor. To update the correction field in certain PTP packets, the ingress timestamp must be given in the TSSL and TSSH fields.

The one-step timestamp feature is supported only for the PTP over Ethernet packets. It is not supported for PTP over IPv4/IPv6 packets.

76.5.6 Time sensitive networking (TSN)

This section describes how the Ethernet peripheral core supports time sensitive network (TSN) features.

TSN is a set of standards, developed by the IEEE 802.1 work group, which define the mechanisms for the time-sensitive transmission of data over Ethernet.

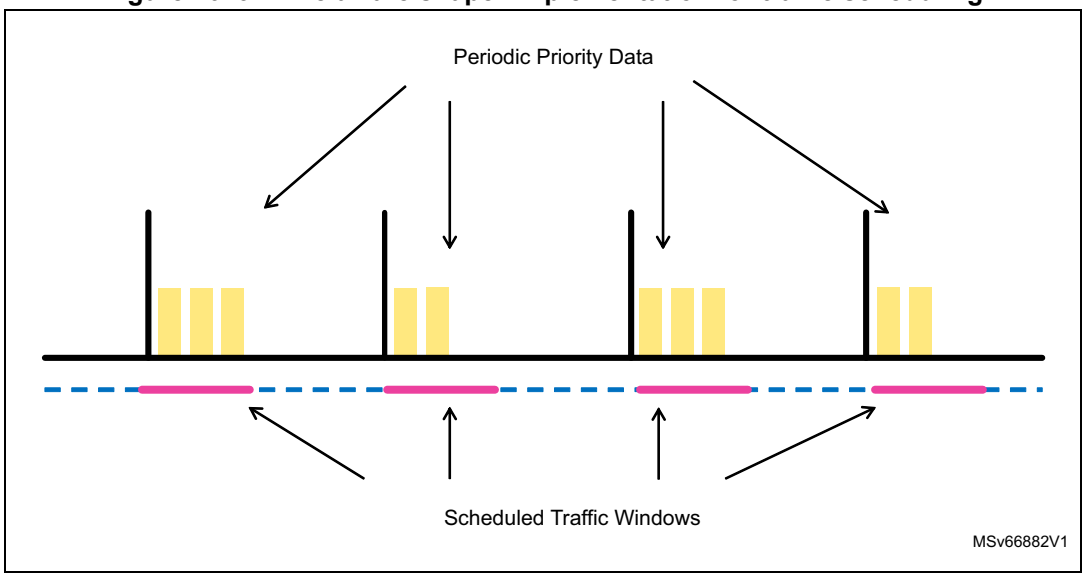
The present section describes:

- *Enhancements for scheduled traffic (EST)*
- *Frame preemption (FPE)*
- *Time-based scheduling (TBS)*

Enhancements for scheduled traffic (EST)

The IEEE 802.1Qbv-2015 defines the schedule for each queue on every egress port that makes the peripheral aware of the traffic arrival schedule. This information can be used to block lower priority traffic from being transmitted in this time window/slot. This ensures that scheduled traffic is forwarded from sender to receiver through all the network nodes with a deterministic delay.

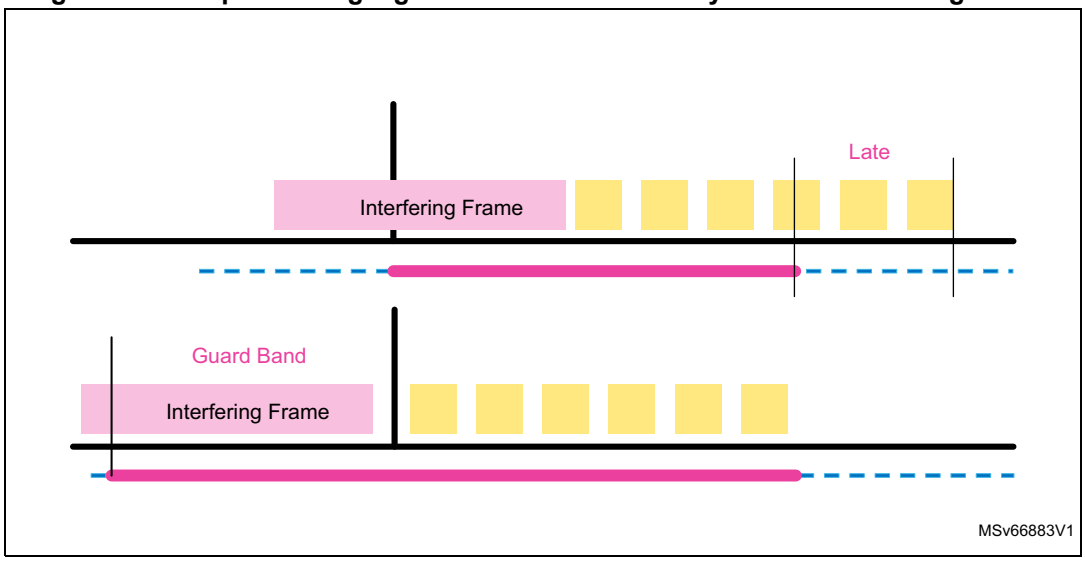
Figure 1029. Time aware shaper implementation for traffic scheduling



One of the important requirements to achieve a low latency is to make sure there are no interfering frames during the scheduled windows that are reserved to high priority traffic. The use of scheduled traffic imposes limitations while starting a transmission.

As shown in *Figure 1030*, if an interfering frame transmission begins just before the start of a reserved time period, the transmission can be extended into the reserved window and potentially interfere with higher priority traffic. Therefore, a guard band whose size is equal to the largest possible interfering frame is required before the window starts.

Figure 1030. Implementing a guard band to avoid delays due to interfering frames



A larger guard band equates to a less efficient use of network bandwidth. However, with the implementation of IEEE802.1Qbu (frame preemption), this issue is addressed. Frame preemption breaks the interfering frame into smaller fragments. Therefore, the guard band needs to be only as large as the largest possible interfering fragment instead of the largest possible interfering frame.

During the guard band, only the frames for which the entire frame can be complete before the next gate close event are permitted. This ensures that the high priority traffic can always start at the beginning of the window reserved for it.

The transmit scheduling embeds the following feature to be able to support EST:

- Gate control list (GCL)
- Enforced gate controls in the scheduler
- Gate open duty cycle taken into account in the computation of the idleSlope (CBS)

Gate control list implementation (GCL)

Figure 1031 shows the block diagram extracted from the IEEE 802.1Qbv specifications that illustrates how the gate control list governs the gate close (C) and open (O) events based on the schedule provided for each event.

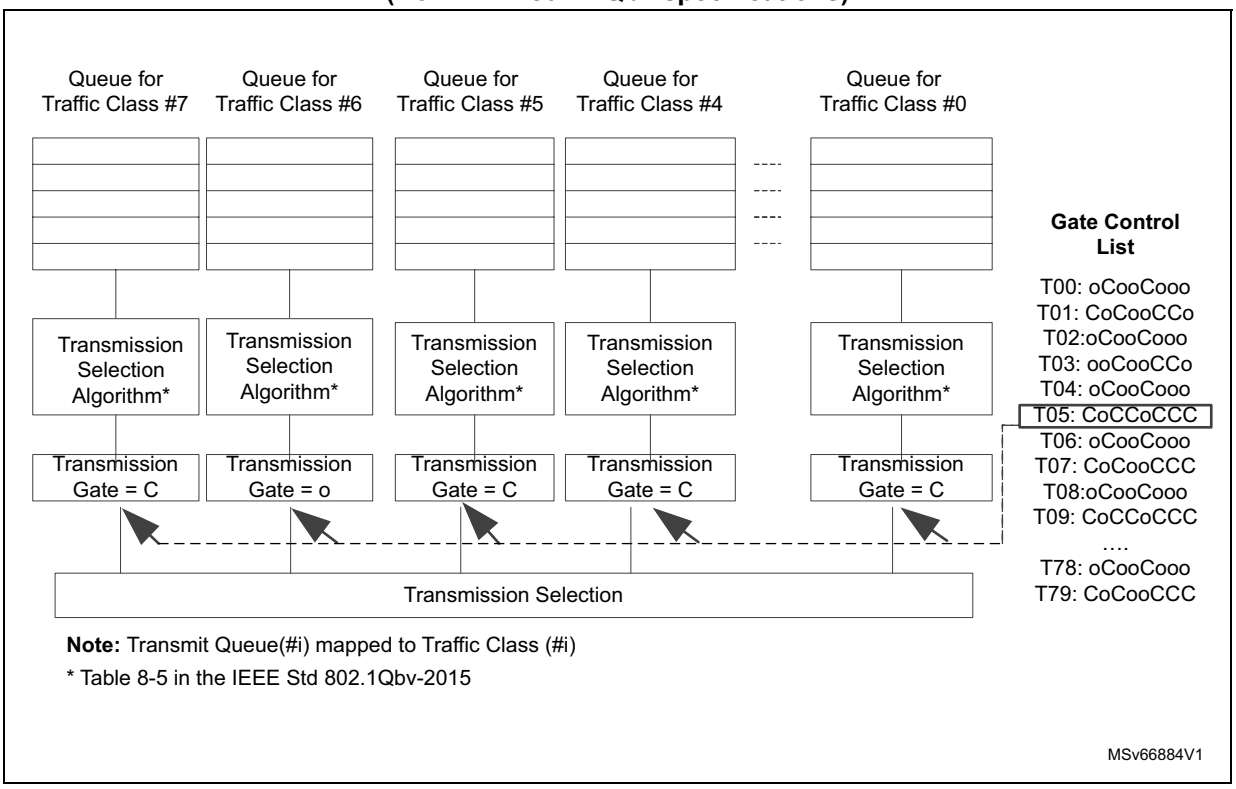
The GCL consists of two parts:

- Time interval

  It defines the time (expressed in ns) during which the gate controls are valid and should be applied before reading the next gate controls from the list. The maximum time interval is 16 ms. The left shift of the time interval up to 7 bits is supported to be able to apply a multiplication factor ranging from 1 to 128 ns (in steps of 2^n). The maximum value (post shifting) of this field must be 999,999,999 ns.
- Gate control

  Defines the Open (O) represented by logic 1 or Close (C) represented by logic 0 state for the gate of each traffic class (TC).

Figure 1031. GCL governing gate close and open events block diagram (from IEEE 802.1 Qbv specifications)



The implementation of GCL consists of the following two gate-control lists:

- Hardware owned list (HWOL), which can be accessed only by hardware
- Software owned list (SWOL), which can be accessed only by software

The access to these lists is mutually exclusive. The ownership of the list is set by hardware in the SWOL field of *EST Status register (ETH_MTLESTSR)*. *Table 793* shows how the GCL is implemented.

Table 793. External memory used for holding the two gate-control lists

Gate control (up to 8 bits)	Time interval in ns	Gate-control list
OOCCCCCC	T0	HWOL or SWOL
OOOOCCCC	T1	
CCCCOOCC	T2	
CCCCCCOO	T3	
I	I	
I	I	
OCOCOCOO		
OOOOCCCC	Last T	

Table 793. External memory used for holding the two gate-control lists (continued)

Gate control (up to 8 bits)	Time interval in ns	Gate-control list
OOOOCCCC	T0	SWOL or HWOL
OOOOCCCC	T1	
OOCCCCCC	T2	
OOOOCCCC	T3	
I	I	
I	I	
OCOCCCCC		
OOCCCCCC	Last T	

Four registers are related to the gate control lists (one for each GCL). These registers are implemented through Indirect addressing using *EST Gate Control List register (ETH_MTLESTGCLCR)* and *EST Gate Control List Data register (ETH_MTLESTGCLDR)*.

- **64-bit base time register (BTR)**

  The base time register is a 64-bit register that holds the start time to execute the GCL. The format of the BTR is the same as the PTP format (upper 32-bit holds time in seconds and lower 32-bit hold time in ns).

  Once the execution of a given list starts, the peripheral can update the BTR value to indicate the next list execution start time.

- **40-bit cycle time (CTR)**

  The cycle time register is a 40-bit register that holds the time at which the execution of the GCL should be repeated. The cycle time register consists of an 8-bit value in seconds and a 32-bit value in ns (similar to the PTP time format with truncated second register).

  For a given GCL, the start time is given by the below formula:

  $$\text{Start time} = \text{Base time} + N \times \text{Cycle time}$$

  where N is an integer value representing the iteration number starting from 0 for the first iteration.

  If the GCL execution takes longer than the cycle time, then the list is truncated at the cycle time and the subsequent loop starts at the cycle time.

- **32-bit time extension (TER)**

  The time extension register (TER) is a 32-bit register that holds the amount of time (in ns) of which current GCL can be extended before switching to the new GCL. This is useful to avoid small fragments of the current list before switching to a new list. The time extension value is coded on 31 bits and stored in a 32-bit register, with the most significant bit forced to 0.

- **7-bit [gate control] list length (LLR)** (where n depends on the configured GCL depth)

  The list length register (LLR) is a 7-bit register that holds the integer value of the GCL length, which corresponds to the number of valid rows in each GCL. The processing of the GCL stops after the number of rows read is equal to the LLR value.

Transmission gating

A bridge or an end station can be used to allow transmission from each TC that is yet to be scheduled relative to a known timescale. To achieve this, a transmission gate is associated to each TC. The state of the transmission gate determines if the queued frames can be selected for transmission.

The state of a TC transmission gate can be in one of the following:

- Open

  Queued frames are selected for transmission according to the definition of the transmission selection algorithm associated to the TC.
- Closed

  Queued frames are not selected for transmission.

A frame on a traffic class queue is not available for transmission if the transmission gate is in closed state or if there is insufficient time available for transmitting the whole frame before the next gate-close event associated to this queue.

The peripheral has visibility on the current schedule of gate controls and the immediate next schedule of the gate controls. As a result, the maximum gate open period does not exceed the sum of the two time intervals since a frame is selected for transmission only if the gate is currently open and the duration of gate open interval is greater than the time taken to transfer the entire frame.

The implementation must know the frame size before the transmission, so that transmission overruns can be avoided and only the frames that can complete are scheduled. This can be achieved by:

- programming the OVHD field ([5:0] bits) of *EST Extended Control register (ETH_MTLESTECR)* with fixed overhead of the IPG or EIPG, preamble, and scheduler delay
- automatically predicting the variable overhead due to the offloads enabled for a packet in the MAC transmitter.

Consider the following while programming the EST related fields to ensure correct packet scheduling:

- Speed: 1 Gbps/100 Mbps
- PTOV (in ns): 9 PTP clock cycles
- CTOV (in ns): 3 PTP + 3 Transmit clock cycles
- The slot interval per packet (in ns) is given by the following formula:

$$\text{Slot interval per packet} = (\text{Packet size} + \text{Overhead} + \text{IPG or EIPG} + \text{preamble}) \times \text{time to transmit 1 byte}$$

The overhead and scheduler delays in bytes are equal to 17 Tx + (10 + X) application clock cycles, where X = 2 if VLAN tag insertion is enabled, X = 0 otherwise.

The peripheral adequately compensates for the data transfer delays from the buffer to the line by offsetting the current time with all the relevant delays (provided by the CTOV field of *EST Control register (ETH_MTLESTCR)*). This ensures that the provided schedule is always accurately implemented at the line.

Note: An overhead is added to the packet size to take into account the scheduler delay, IPG or EIPG, and preamble overheads on the line. The gate open window should therefore be able to accommodate at least 128 + overhead bytes to transmit a 128-byte frame.

In 100-Mbps mode of operation, the rounding down error is about 0.4 %. For example, the gate open duration should be at least 1035 bytes for transmitting 1000-byte frames at a speed of 100 Mbps (31 bytes for scheduler delay and 4 bytes for rounding down error).

The minimum time interval for GCL entry is the following:

$$\text{Min time interval for GCL entry} = \text{Time to transmit fragment size} + \text{Scheduler delay} + \text{IPG/EIPG} + \text{Preamble overheads}$$

Idle slope computation updates

When EST is enabled, the credit is accumulated only when the gate is open. The effective data rate of the idleSlope must therefore be increased to reflect the duty cycle for the transmission gate associated to the queue.

The idle slope (idleSlope) is computed based on the gate open time (GateOpenTime) and operating cycle time (OperCycleTime) values. Program the idleSlope registers (one per CBS enabled TC) based on the following equation:

$$\text{idleSlope} = (\text{operIdleSlope(N)} \times \text{OperCycleTime} / \text{GateOpenTime})$$

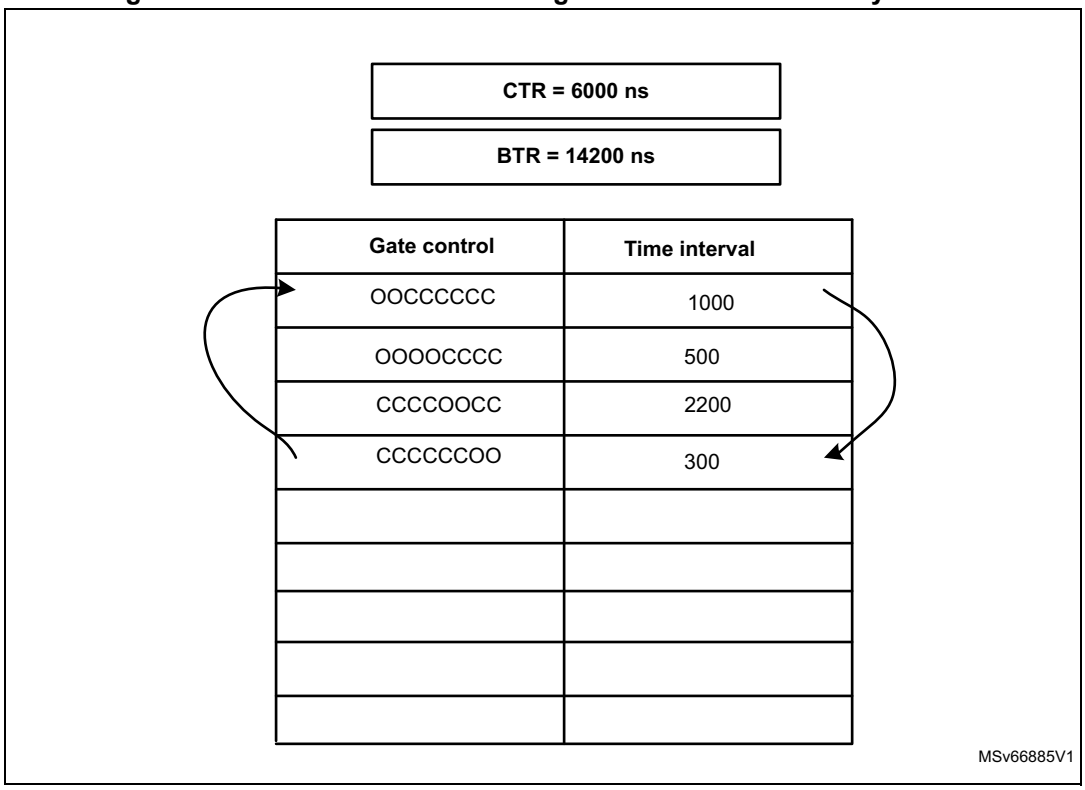
Operational details of GCL

Set the switch to software list through the SSWL bitfield in *EST Control register (ETH_MTLESTCR)*, so that the hardware can access the programmed gate control list. The first set of gate controls is applied when the current time is equal to the value in the base time register (BTR) and is held until the programmed time interval value.

To avoid transmission overruns, an additional gate control event is always read ahead from the list. This enables the GCL to determine the next gate close events (if any) for the TCs that are open.

The scheduling is based only on the gate open state and time interval of the current and subsequent schedule. An internal accumulator is used to add the time intervals when gate controls are applied. The sum of BTR + accumulator holds the time at which the next set of gate controls are planned to be applied.

Figure 1032. GCL and associated registers - BaseTime and CycleTime



The GCL is read progressively from the first row adhering to the schedule. The read operations continue until the list length (defined in the LLR register) is reached. The execution of the list restarts at BTR + CTR time. At this point the BTR is incremented by the value in CTR to mark the beginning of a new cycle. In the absence of gate controls, all the gates are deemed to be in open state during the execution of the list.

In cases where the list execution time is greater than the CycleTime, the list is truncated and the next iteration starts when the current time equals BTR + CTR.

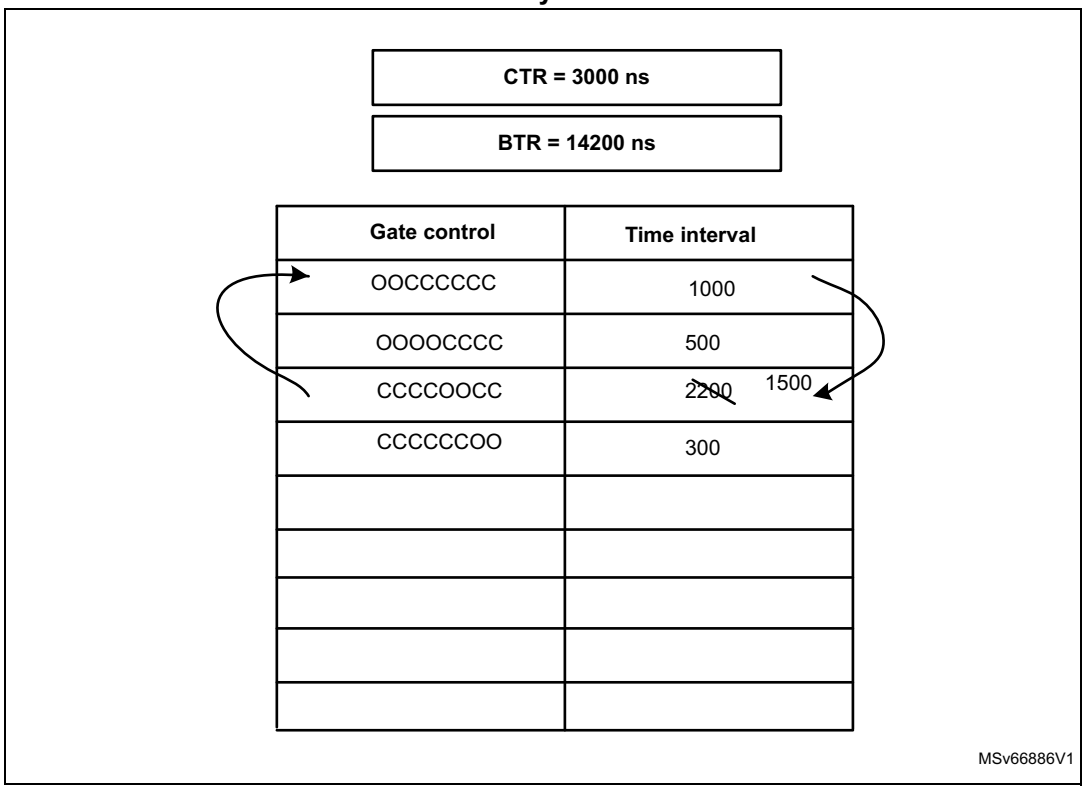
Table 794. GCL and associated registers - BTR and CTR

Current time	Gate control applied	Accumulator value	BTR (with updates)
14200	OOCCCCCC	1000	14200
15200	OOOOCCCC	1500	14200
15700	CCCCOOCC	3700	14200
17900	CCCCCCOO	4000	14200
18200	OOOOOOOO	0	20200
20200	OOCCCCCC	1000	20200
21200	OOOOCCCC	1500	20200

In the example given in *Table 794*, the execution starts at 14200 and the first set of gate controls, OOCCCCCC is immediately applied. Since the time interval is 1000 ns, the next set of gate controls is applied at 14200 (BTR) + 1000 (accumulator) = 15200 ns as shown in *Table 794*. Since there are no gate controls available after the execution of the last gate

control and before the next iteration of the loop, the gates are deemed to be in open state during that time period as shown at time 18200 in *Table 794*.

Figure 1033. GCL and associated registers - BaseTime and CycleTime list execution time > CycleTime



Since the list execution takes longer than the allocated CycleTime, the list is truncated and the list is started from the BTR+CTR as shown in *Table 795*.

Table 795. GCL and associated registers - BTR and CTR, execution time > CycleTime

Current time	Gate control applied	Accumulator value	BTR (with updates)
14200	OOCCCCCC	1000	14200
15200	OOOOCCCC	1500	14200
15700	CCCCOOCC	3700	14200
17200	OOCCCCCC	1000	17200
18200	OOOOCCCC	1500	17200
18700	CCCCOOCC	3700	17200

While applying the third set of gate controls, BTR + CycleTime (17200 ns) is lower than BTR + accumulator (17900), so the list is truncated and execution switches to a new iteration at 17200.

Installing a new GCL

When a new software-programmed GCL is available and must be executed at the new BTR value, the switch to the new GCL can be done in one of the following ways:

- New base time aligned with current schedule
- New base time unaligned with current schedule

New base time aligned with current schedule

If the cycle time value for the new gating cycle is the same as the cycle time of the current gating cycle, and if the BTR chosen for the new gating cycle (new BTR) is an integer multiple of the current cycle time (+ current BTR), then the new gating cycle exactly lines up with the old gating cycle, that is, the cycle start times for the new gating cycle are the same as they would have been for the old configuration. This can be considered as the ideal case and enables the new gating cycle to be installed and executed with no timing issues. The peripheral completes the execution of an iteration of the current list and switches to the new list at the beginning of the BaseTime listed in the new list.

If (New BaseTime ≥ Current time) ConfigChangeTime = New BaseTime

Else If (New BaseTime < Current Time)

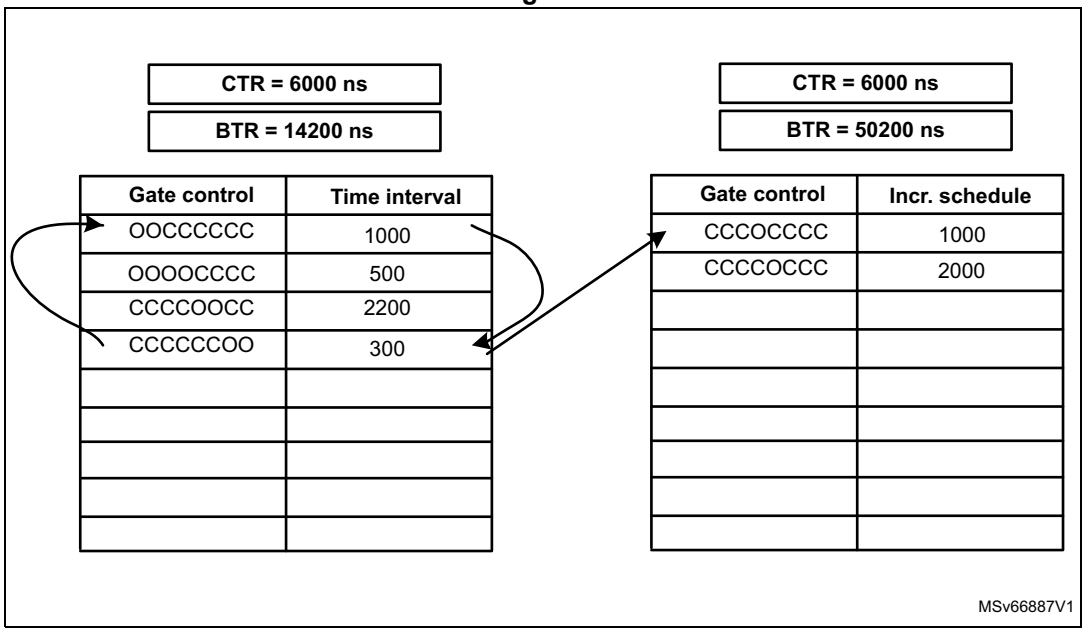
1. Set the BTRError (BTRE bit in ETHMTLESTR)
2. ConfigChangeTime = New BaseTime + N × New CycleTime)

where N is the smallest integer for which the relation ConfigChangeTime ≥ CurrentTime and (N ≤ 8) is TRUE.

When N is higher than 8, the hardware cannot auto-recover and the loop count value in BTR error loop count (BTRL bitfield in ETH_MTLESTSR) is set to 0b1111 1111 requiring the software to reprogram the new base time.

Figure 1034 illustrates the installation of the new GCL along with the time lines.

Figure 1034. Switching to a new configuration that is aligned with the existing configuration



In the example given in *Table 796*, after the sixth iteration of the first GCL, the BTR values of the old and new GCL are equal. At that point the new GCL is processed as a natural extension to the existing GCL.

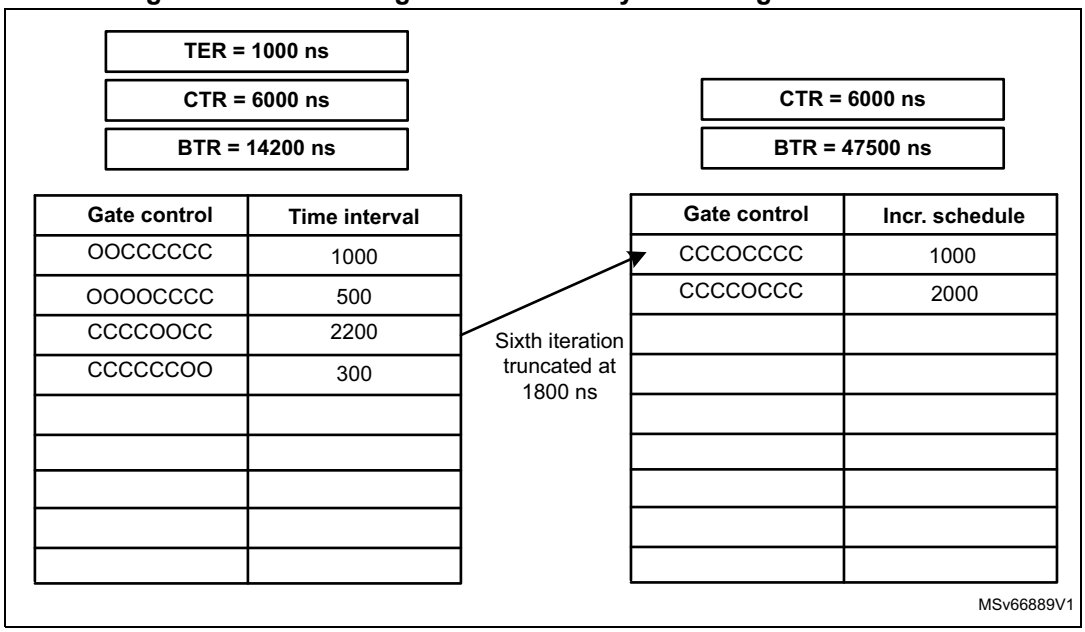
Table 796. GCL and associated registers - BTR and CTR

Current time	Gate control applied	Accumulator value	BTR (with updates)
44200	OOCCCCCC	1000	44200
45200	OOOOCCCC	1500	44200
45700	CCCCOOCC	3700	44200
47900	CCCCCCOO	4000	44200
48200	OOOOOOOO	0	50200
50200	CCCOCCCC	1000	50200
51200	CCCCOCCC	3000	50200
53200	OOOOOOOO	0	56200

New base time unaligned with current schedule

If the new CycleTime differs from the current CycleTime or the new BaseTime is in the future and is not an integer multiple of current CycleTime, then the old and new cycles do not line up. When a new BaseTime is reached (that is when the new configuration is installed and starts to execute), the last old cycle is normally truncated to start the first new cycle. This can be undesirable if it results in a very short last old cycle. It would be better to simply extend the penultimate old cycle by that small amount, rather than starting a very short cycle. The cycle time extension register (TER, related to the current configuration list) enables this extension of the last old cycle to be done in a defined way. If the last complete old cycle ends normally in less than the current cycle time extension (TER in ns) before the new BaseTime, then the last complete cycle before the new BaseTime is reached is extended so that it ends at the new BaseTime.

Figure 1035. Switching to the new list by truncating the current list



At the end of the fifth iteration, current time + cycle time extension (TER) < New BTR, so the sixth iteration of the current configuration has started. During the sixth iteration of the current list, when the new BTR value is smaller than the next schedule in the current list, it switches to the new list.

Table 797. Extending to new list by truncating the current list

Current time	Gate control applied	Accumulator value	BTR (with updates)
44200	OOCCCCCC	1000	44200
45200	OOOOCCCC	1500	44200
45700	CCCCOOCC	3700	44200
47500	CCCOCCCC	4000	44200
48500	CCCCOCCC	0	50200
50500	OOOOOOOO	1000	50200

Figure 1036 shows an example where the current configuration list is extended instead of starting a new iteration since the extension time of 800 ns is less than the allowed cycle extension time (TER) of 1000 ns.

Figure 1036. Switching to new list by extending the current list

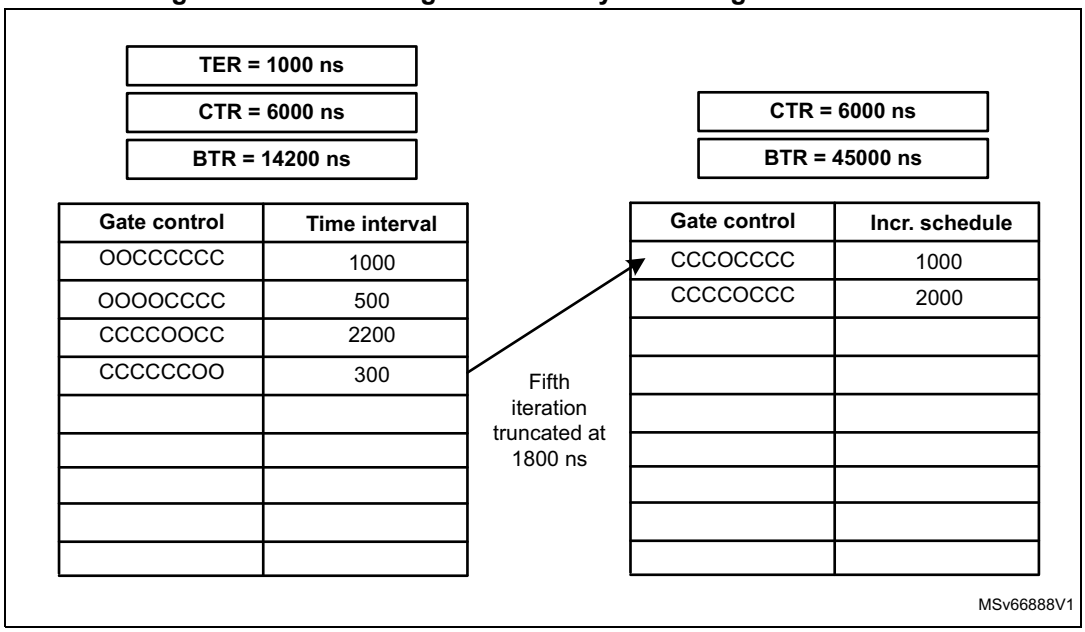


Table 798. Switching to new list by extending the current list

Current time	Gate control applied	Accumulator value	BTR (with updates)
38200	OOCCCCCC	1000	38200
39200	OOOOCCCC	1500	38200
39700	CCCCOOCC	3700	38200
41900	CCCCCCOO	4000	38200
42200	OOOOOOOO	0	44200
45000	CCCOCCCC	1000	45000
46000	CCCCOCCC	3000	45000
48000	OOOOOOOO	0	51000

Impact of transmit scheduling algorithms on EST

When EST is used in isolation, the Gate Control List manages the final open/close state of the queues along with the checks carried out by the transmission selection algorithm in MTL. Since the gate controls operate on a predefined repetitive schedule, it is recommended to use strict priority or credit based shaper (CBS) scheduling algorithms.

Other algorithms such as the weighted round robin (WRR), deficit weighted round robin (DWRR) and weighted fair queuing (WFQ) implement the masking of the queues based on the current winning queue. The algorithm is applied only to the group of queues that open simultaneously. To ensure the queues whose gates are open get priority, these algorithms are modified to treat gate open queues and gate closed queues as separate groups, giving priority to gate open queues.

For example, consider four queues (Q3, Q2, Q1, Q0) with respective weights 4:3:2:1. Q3 and Q2 are in open state in one slot, while Q1 and Q0 are in open state in another slot. In this case, the scheduler works as follows:

1. In the first slot, Q3 and Q2 are serviced in the ratio of 4:3 for the duration during which the slot is open.
2. In the second slot, Q1 and Q0 are serviced in the ratio of 2:1.
3. Fresh arbitration is started every time a slot is opened.

In other words, the traffic does not get distributed in the intended ratio of 4:3:2:1; but as two groups with different ratios, and only for the duration of the slot when the gates are open continuously.

EST programming guidelines

For details about programming guidelines for EST, see *Section 76.9.20: Programming guidelines for EST on page 4146*.

Frame preemption (FPE)

Frame preemption overview

One of the important requirements of IEEE 802.1Qbv-2015 standard is to ensure there are no interfering frames during the scheduled windows that are reserved for high priority traffic. To achieve this goal, a guard band, whose size is equal to the largest possible interfering frame, is required before the window starts. However a larger guard band equates to a less efficient use of network bandwidth.

Frame preemption compliant with IEEE802.1Qbu enables reducing the guard band by breaking the interfering frame into smaller fragments. Therefore, the guard band needs to be only as large as the largest possible interfering fragment instead of the largest possible interfering frame.

Note: Internal loopback is not supported for preempted frames

Description of the frame preemption

Preemption enables one or more higher priority (express) frames to interrupt the transmission of a lower priority (preemptable) frame. The preemptable frame transmission is resumed and completed after the express frame transmission is complete. The following two abstractions of the MAC are used to support frame preemption:

- A preemptable MAC (pMAC), which carries the preemptable traffic
- An express MAC (eMAC), which carries the express traffic

Only the part of the MAC that holds the state during preemption is replicated and represented as pMAC and eMAC. There are two ways for the MAC to put on hold the transmission of the preemptable traffic in the presence of express traffic:

- The MTL scheduler interrupts the preemptable traffic that is currently being transmitted.

  When the preemption capability is enabled, the MAC interrupts the transmission and reception of preemptable frames. A preempted fragment can be followed by zero or more express frames, before the continuation fragments. The preemptable frame can be fragmented any number of times. However, the minimum final and non-final fragment length criterion must be met.

  Interleaving of more than one preemptable packet is not permitted. This implies that if a preemptable packet is fragmented by an express packet, another preemptable packet

cannot be transferred until all the remaining fragments of the first preempted packet are transferred.

- The MTL scheduler prevents starting the transmission of preemptable traffic

  When the preemption capability is disabled, the pMAC entity is disabled and only express traffic is transmitted or received.

Note: Queue0 can be an express queue when FPE is enabled and EST is disabled.

Enabling frame preemption

Enable the frame preemption feature by setting EFPE field of *FPE control and status register (ETH_MACFPECSR)*.

Modifying GCL to support FPE

In the EST-only configuration, the GCL entry has 24 bits of time interval and 8 high order bits representing the gate open/close state requirements as shown in *Table 799*.

Table 799. Gate control list when FPE is disabled

Gate control	Time interval (ns)
OOCCCCCC	T0
OOOOCCCC	T1
CCCCOOCC	T2
CCCCCCOO	T3
I	I
I	I
OCOCOCOO	T126
OOOOCCCC	T127

EST only supports SetGate operations, which implies that the gates are set either to open or close at a given time interval.

However, when both FPE and EST are enabled, the GCL also supports Set-and-Hold-MAC and Set-and-release-MAC operations, in addition to the SetGate operation.

To support hold and release operations, the format of the GCL is slightly changed while configuring and enabling FPE. The first Queue (Q0) is always programmed to carry preemption traffic and therefore it is always open. The bit corresponding to Q0 is renamed as Release/Hold MAC control.

The TX queues whose packets are preemptable are identified by setting the PEC field of *FPE control and status register (ETH_MACFPECSR)*. The GCL bit of the corresponding queue is ignored and considered as always open. As a result, even if the software writes a 0 (C), it is ignored for such queues.

Table 800. Gate control list when FPE is enabled

Gate control (up to 7 bits)	Release/Hold indication	Time interval (ns): 16, 20 or 24 bits
CCCOOOO	0	T0
CCCCOOO	0	T1

Table 800. Gate control list when FPE is enabled (continued)

Gate control (up to 7 bits)	Release/Hold indication	Time interval (ns): 16, 20 or 24 bits
OCCCOOO	1	T2
COCCOOO	1	T3
CCOCOOO	1	T4
CCCOOOO	0	T5
CCCCOOO	0	T6

When the Release/Hold bit transitions from a 0 to 1, a Set-and-Hold-MAC operation is performed. This marks the cease of the preemptable traffic. This is achieved by sending a Hold request to MTL *ha* ns in advance (where *ha* is the time interval mentioned in the Hold advance (HADV) field of *FPE Frame Preemption Advance register (ETH_MTLFPEAR)*.When the Release/Hold bit transitions from a '1' to '0', a Set-and-Release-MAC operation is performed. This marks the resuming of the preemptable traffic. This is achieved by sending a Release request to MTL *ra* ns in advance (where *ra* is the time interval mentioned in the Release Advance (RADV) field of *FPE Frame Preemption Advance register (ETH_MTLFPEAR)*).The preemptable traffic is blocked for the time duration the Release/Hold bit is set to a '1' in the gate control list.

Note: HADV is the advance time to initiate the hold state. Ideally HADV should be programmed to a value of minimum fragment size, which should be smaller than the time interval (minimum fragment size + IPG + Preamble overheads in ns).

Impact of preemption on CBS algorithm

When the CBS algorithm is used, the definition of *Transmit* is as follows:

- TRUE for the duration of frame transmission from the queue;
- FALSE when frame transmission from the queue is complete.

When the CBS algorithm is used along with frame preemption, the value of *Transmit* is TRUE only during the frame transmission by the MAC. If the frame transmission is delayed or interrupted (for instance, a preemptable frame transmission is interrupted to allow the transmission of an express frame from a different queue, or the start of an express frame is delayed because a preemptable frame is being transmitted), the value of *Transmit* is FALSE until the transmission of the frame begins or is resumed.

In addition, the value of *Transmit* is FALSE during the transmission of any overhead resulting from a frame preemption. For example, additional frame overhead (mCRC, Fragment Count) that is added to the preemptable frame.

At any given time, if there are no frames in the queue, the value of *Transmit* is FALSE, and the credit is positive, the credit value is set to zero if there is no preemptable frame from the queue for which transmission is in progress but has been interrupted.

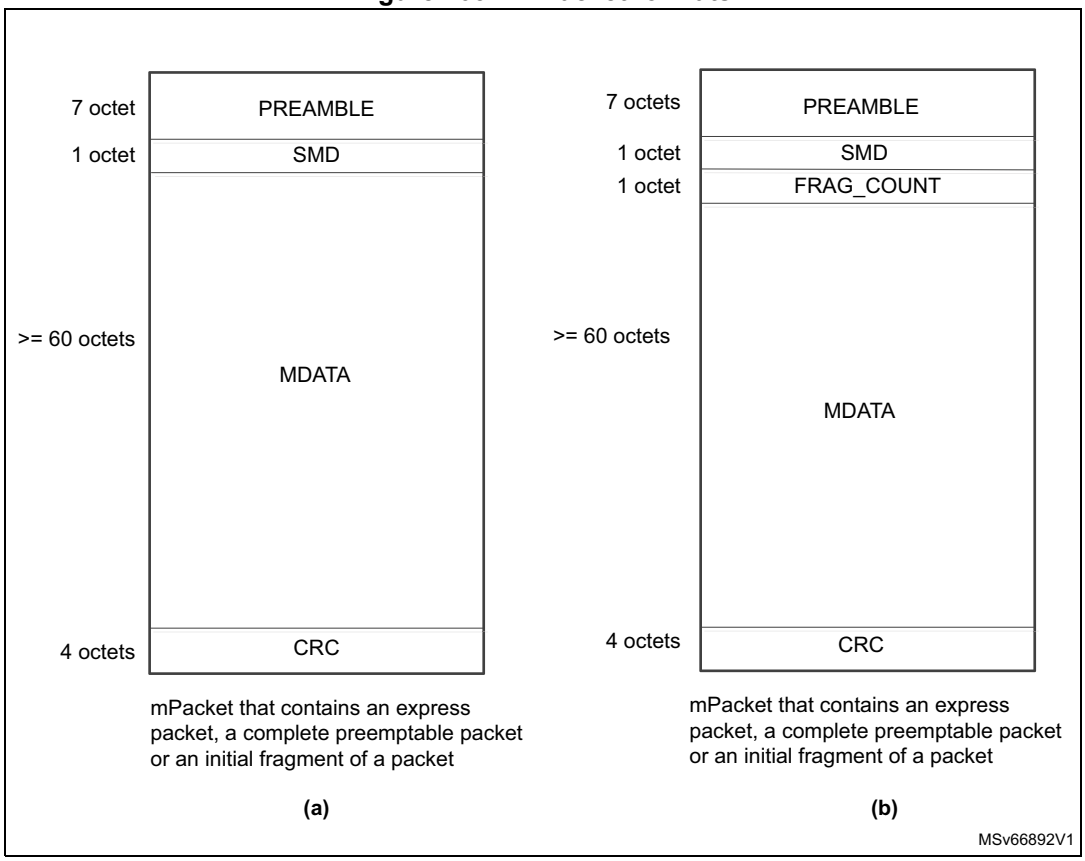
mPacket Format

When the preemption capability is enabled, MAC sends mPackets to the PHY. An mPacket can be:

- An express packet
- A preemptable packet
- An initial fragment of a preemptable packet
- A continuation fragment of a preemptable packet

Figure 1037 shows the format of the mPacket. It contains an express packet, a complete preemptable packet or the initial fragment of a preemptable packet. *Figure 1037* (b) shows the format of an mPacket containing a continuation fragment of a packet.

Figure 1037. mPacket formats



Preamble

The preamble in the mPacket shown in *Figure 1037* (a) contains seven octets. The preamble in the mPacket shown in *Figure 1037* (b) contains six octets. Each octet contains the value 0x55, transmitted from left to right, 10101010.

Start mPacket Delimiter (SMD)

The value of the SMD indicates whether the mPacket contains an express packet, the start of a preemptable packet (initial fragment or complete packet), or any continuation fragments of a preemptable packet. *Table 801* shows the valid SMD values.

Table 801. Valid SMD Values of mPacket

mPacket type	Notation	Frame count	Value
verify packet	SMD-V	-	0x07
Respond packet	SMD-R	-	0x19
Express packet	SMD-E	-	0xD5
Preemptable packet start	SMD-S0	0	0xE6
	SMD-S1	1	0x4C
	SMD-S2	2	0x7F
	SMD-S3	3	0xB3
Continuation fragment	SMD-C0	0	0x61
	SMD-C1	1	0x52
	SMD-C2	2	0x9E
	SMD-C3	3	0x2A

frag_count

A frag_count is a modulo-4 counter that increments for each continuation fragment of the preemptable packet. The frag_count protects against mPacket reassembly errors by enabling detection of the loss of up to three packet fragments.

The frag_count field is present only in mPackets with SMD-C notation (continuation fragment). The frag_count is zero in the first continuation fragment of each preemptable packet.

Table 802. Valid frag_count values

frag_count	Value
0	0xE6
1	0x4C
2	0x7F
3	0xB3

mData

The contents of the packet from the MAC, starting with the first byte after the SFD to the last byte before the FCS, are sent in the mData fields of one or more mPackets for that frame. The minimum size of the mData field is 60 bytes.

CRC

The CRC field contains a cyclic redundancy check (CRC) and provides an indication on the final mPacket of a frame. In the final mPacket of a frame, the CRC field contains the last four octets of the MAC frame (FCS field).

For other mPackets, the CRC field contains an mCRC value. The mCRC is calculated on the octets of the packet from the first octet of the frame (octet following the SFD of

preemption frames) to the last octet of the packet transmitted in that mPacket, by performing an XOR of the calculated 32-bit CRC value of the fragment with 0x0000FFFF.

Summary of Packet Formats

- Express packet
  Seven bytes of preamble, SMD-E, data and CRC
- Complete Preemptable packet
  Seven bytes of preamble, current preemptable packet SMD, data, CRC
- Initial fragment (non-final) of preemptable packet
  Seven bytes of preamble, current preemptable packet SMD, data, mCRC
- Continuation fragments (non-final) of preemptable packet
  Six bytes of preamble, current preemptable continuation fragment SMD, current preemptable continuation fragment FC, data, mCRC
- Final fragment of preemptable packet
  Six bytes of preamble, current preemptable continuation fragment SMD, current preemptable continuation fragment FC, data, CRC

Note: When frame preemption is enabled, the MAC receiver communicates the exceptions related to received preempted fragments (such as incorrect fragment sequencing or missed fragment) to the MTL layer by reporting it as CRC error. The software should not set the DCRCC (Disable CRC Checking for Received Packets) bit of Extended operating mode configuration register (ETH_MACECR) to 1, otherwise the MTL layer might forward the unintended preempted fragments to the application.

Table 803. Current and Previous SMD Values

Previous preemptable packet SMD	Current preemptable packet SMD
SMD-S0	SMD-S1
SMD-S1	SMD-S2
SMD-S2	SMD-S3
SMD-S3	SMD-S0

Table 804. Current and previous SMD values

Previous preemptable fragment SMD	Previous preemptable fragment FC	Current preemptable continuation fragment SMD	Current preemptable continuation fragment FC
SMD-S0	NA	SMD-C0	FC0
SMD-S1	NA	SMD-C1	FC0
SMD-S2	NA	SMD-C2	FC0
SMD-S3	NA	SMD-C3	FC0
SMD-C0	FC0	SMD-C0	FC1
SMD-C0	FC1	SMD-C0	FC2
SMD-C0	FC2	SMD-C0	FC3
SMD-C0	FC3	SMD-C0	FC0

Table 804. Current and previous SMD values (continued)

Previous preemptable fragment SMD	Previous preemptable fragment FC	Current preemptable continuation fragment SMD	Current preemptable continuation fragment FC
SMD-C1	FC0	SMD-C1	FC1
SMD-C1	FC1	SMD-C1	FC2
SMD-C1	FC2	SMD-C1	FC3
SMD-C1	FC3	SMD-C1	FC0
SMD-C2	FC0	SMD-C2	FC1
SMD-C2	FC1	SMD-C2	FC2
SMD-C2	FC2	SMD-C2	FC3
SMD-C2	FC3	SMD-C2	FC0
SMD-C3	FC0	SMD-C3	FC1
SMD-C3	FC1	SMD-C3	FC2
SMD-C3	FC2	SMD-C3	FC3
SMD-C3	FC3	SMD-C3	FC0

Transmit preemption

MTL Tx preemption

To support preemption, the MAC should have more than one TX queue with at least one queue designated as express queue, plus one queue designated as preemption queue. When FPE is enabled (EFPE set to 1 in *FPE control and status register (ETH_MACFPECSR)*), the MTL preempts a preemptable frame. When a Hold request is asserted (EST configured and enabled), express frames are available for transmission, that is frames are present in the MTL FIFO and qualified for arbitration, after ensuring that the minimum mPacket mData field size is met. Therefore, preemption occurs only if at least 60 bytes of the preemptable frame have been transmitted and at least 64 bytes (including the frame CRC) remain to be transmitted.

The earliest starting position of preemption is controlled by the AFSZ field of *FPE control and status register (ETH_MACFPECSR)*. Preemption does not occur until at least 64 × (1 + AFSZ) - 4 bytes of the preemptable frame have been sent.

When preemption occurs, all the preemptable queues are blocked and only the express queues are allowed to arbitrate (if more than one express queue has traffic) and transmit.

The continuation fragment of the preempted frame is the first frame to be transmitted after a Release request is asserted (EST configured and enabled) and all the express traffic transmission completes.

Note: All the PTP packets should be transmitted as express packets.

MTL should wait for the previous fragment status to be received before resuming the continuation fragments of a preempted frame.

MAC Tx preemption

The MAC supports the preemption by implementing the functionality required to generate the mPackets as described in *Section : mPacket Format*.

When the preemption is enabled, the MAC replaces the SFD of a preemption packet with an SMD-S value. A 2-bit rolling frame count is encoded in the SMD-S value.

The SMD-E value is the same as the SFD value, so the SFD of an express packet does not need to be replaced.

Receive preemption

MAC receive preemption

When FPE is enabled, the MAC receiver passes the incoming packets and differentiates between express packets and preemptable packets. An SMD containing an SMD-E indicates express packet while and SMD containing an SMD-S indicates the first mPacket of a preemptable packet.

If an mPacket containing an SMD-S is received when the reception of the previous preempted packet has not be completed, the MAC sets a CRC error status for the previously received partial packet.

When an SMD-S is detected, the MAC records the frame count indicated by the SMD, and begins sending data on the MRI interface.

The MAC checks the last four bytes of the mPacket. If they do not match CRC, it indicates the end of the packet with or without a CRC error as per the CRC check result. If the last four octets of the mPacket match, it indicates that the packet was preempted.

An SMD containing an SMD-C indicates an mPacket that continues the data for a preempted packet. Upon receiving an SMD value of SMD-C, the MAC checks the following:

- A preempted packet is in progress.
- The frame count indicated by the SMD matches the frame count of the packet in progress.
- The frag_count value indicates the next fragment count.

If any of these checks fails, the mPacket is discarded and the MAC sets a CRC error status for the partially received packet.

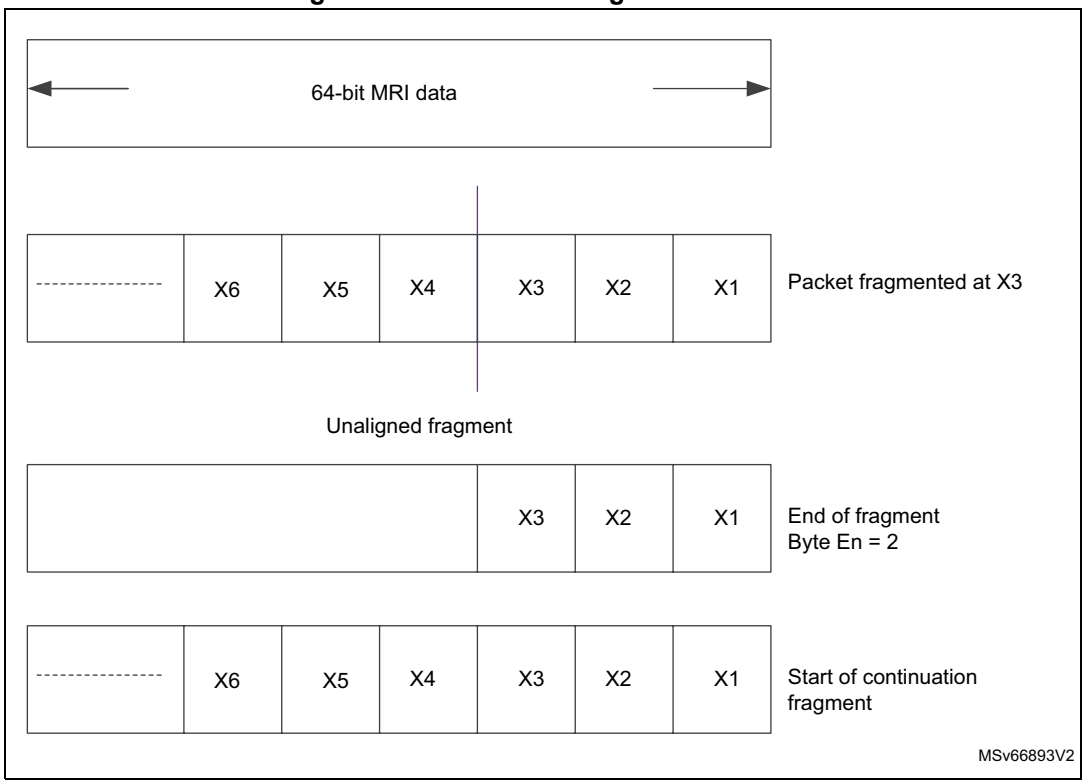
If all the checks pass, the next fragment count is incremented modulo 4.

When a packet is preempted, the MAC saves the state of the partially received packet (such as filter check status, timestamp, length fields) and can process any received express packets before the continuation fragment is received.

Data alignment

When a received frame cannot be fragmented on any byte boundary, the MAC retains the unaligned bytes of data in the previous fragment and resends them with the next fragment as shown in *Figure 1038*.

Figure 1038. MAC data alignment feature



The MTL can choose to ignore the partial data (based on byte enable value) that comes with end of fragment and use instead the data-width-aligned data received in the first beat of the next continuation fragment.

MTL receive preemption

The MTL Rx must have at least two Rx queues to support FPE function since the preemptable packets and the express packets must be routed to separate Rx queues. The destination Rx queue of a received packet is controlled by ETH_MACRXQCiR registers. Program these registers so that preemptable traffic and express traffic are not routed to the same Rx queue. As the queue mapping for tagged packets is based on VLAN user-priority field, this implies that priority of preemptable and express packets are mutually exclusive. In other words, packets of a certain priority (traffic class) are either express or preemptable but cannot be both.

For the preemptable frames, in addition to the PSRQ (Priority Selected in Rx queue) based queue routing, a programmable frame preemption residue queue (FPRQ) is supported to route all the other received preemption packets (untagged, SA/ DA or VLAN filter fails forwarded due to RA being set or VTFE being reset).

Table 805. Preemption control values on MRI interface for various frame types

Preemption packet type	Queue routing
AV tagged packets passing the SA/DA and VLAN filters	PSRQ(1)
Generic tagged packets passing the SA/DA and VLAN filters	PSRQ(2)
Tagged packets failing the SA/DA and VLAN filters without clearing RA to 0 or setting VTFE to 1	Dropped
All other packets OR when RA is set to 1	FPRQ

1. Only if the queue is enabled for AV feature.
2. Only if the queue is enabled for Generic feature.

The express frames use Queue 0 as a default queue. As a result, Queue 0 must not be used for receiving preemption frames in case of filter failures.

When a fragment is received, it is pushed into its destination Rx queue. When this packet is preempted and is followed by an express packet, it is pushed/stored in a different Rx queue. The MTL saves the context of the preempted frame and processes one or more express frames. When the continuation fragment arrives, it restores the saved context and pushes the remainder of the fragmented frame into its Rx queue.

MTL receive arbitration

On the DMA interface, data is fetched from the MTL Rx FIFO based on the arbitration selected in *Operating mode register (ETH_MTLOMR)*. Frame based arbitration can be used only when all the MTL preemption queues operate in Store and Forward mode. Otherwise, there is a loss of bandwidth on the ARI interface since all the fragments of a preemptable packet are not available. Therefore, express packets received between the fragments are blocked until all the fragments are received and transferred; this defeats the purpose of express traffic.

When operating either in Threshold (cut through) or Store and Forward operating mode, PBL-based arbitration is recommended over frame-based arbitration. If PBL-based arbitration is used, the watermark check is always performed as well as the arbitration/transfer of data in terms of chunks of PBL size of data, which is almost similar to the concepts of fragments. Therefore the express queue packets is also blocked for less time since the DMA interface transfers the data without loss of efficiency.

Received preemption frames support in offload engines

When FPE is enabled, all the packets processed by the offload engines (ARP, PTO) and PMT, PTP, PAUSE, and PFC packets must be received as express/normal packets. These offload engine logic and functions do not recognize preemption frames.

Verify and respond mPackets

When FPE function is present, the MAC can receive and detect the Verify and Respond mPackets, even when FPE is not enabled by software. When the MAC detects valid verify/respond mPackets, it notifies the software by setting the RVER and RRSP fields of *FPE control and status register (ETH_MACFPECSR)*, respectively. Optionally an interrupt can be generated. As such packets have empty (all-zero) data payload, they are dropped inside the MAC and not forwarded to the MRI.

The software can set the SVER and SRSP fields of *FPE control and status register (ETH_MACFPECSR)* to request the MAC to transmit Verify and Respond mPackets respectively. Upon successful transmission of these frames, the MAC clears the SVER/SRSP bits and sets the TVER and TRSP bitfields in *FPE control and status register (ETH_MACFPECSR)*. Optionally an interrupt can be generated when these events occur.

Note: 1. For the preemption frames received with CRC error indication, ignore the following status fields: Packet length, Dribble error indication, Runt frame indication.

2. When FPE is configured, the forward undersized good packets (FUP) bit of ETH_MTLOMR register should not be set.

3. When FPE is configured, do not disable CRC checking on receive. This is required because, FPE error fragments are identified internally using CRC checks. For more details, refer to the DCRCC bitfield in Extended operating mode configuration register (ETH_MACECR).

4. Express and preemption traffic must be routed to different DMAs.

Frame preemption and MMC counter and interrupt registers

The following MMC counters and associated Interrupt registers are present in the MAC.

Table 806. MMC Counters and Associated Interrupt Registers

Frame assembly error counter	Description	Associated MMC counter
Frame assembly error counter	32-bit counter providing the number of MAC frames with reassembly errors on the receiver due to a mismatch in the fragment count value.	*MMC Rx packet assembly error register (ETH_RX_PACKET_ASM_ERR)*
Frame SMD error counter	32-bit counter providing the number of received MAC frames rejected because they arrived with an SMD-C when there was no preceding preempted frame	*MMC Rx packet SMD error register (ETH_RX_PACKET_SMD_ERR)*
Frame assembly OK counter	32-bit counter providing the number of MAC frames that were successfully reassembled	*MMC Rx packet assembly OK register (ETH_RX_PACKET_ASM_OKR)*
MAC Rx fragment counter	32-bit counter providing the number of additional mPackets received due to preemption	*MMC Rx FPE fragments counter register (ETH_RX_FPE_FRAG_CR)*
MAC Tx fragment counter	32-bit counter providing the number of additional mPackets transmitted due to preemption	*MMC FPE Tx fragment counter register (ETH_MMC_FPE_TX_FCR)*
Hold request counter	32-bit counter containing the number of hold requests sent to the MAC	*MMC Tx hold request counter register (ETH_MMC_TX_HRCR)*

The following additional registers are associated to MMC interrupts for the MMC error counters:

- *MMC FPE Tx interrupt status register (ETH_MMC_FPE_TX_ISR)*
- *MMC FPE Tx interrupt mask register (ETH_MMC_FPE_TX_IMR)*
- *MMC FPE Rx interrupt status register (ETH_MMC_FPE_RX_ISR)*
- *MMC FPE Rx interrupt mask register (ETH_MMC_FPE_RX_IMR)*

Time-based scheduling (TBS)

The time-based scheduling feature is suitable for traffic whose periodicity and rate are predictable. To improve the quality-of-service of such traffic:

- The transmit DMA fetches the packet from the host memory for transmission at designated time. This helps the software to set up the Transmit descriptors in advance, even before the packet is ready and available. It reduces the overhead on the software and avoids constant monitoring of the time and preparing descriptors just in time when the packet is targeted to be transmitted.
- The MAC transmits the packets only at the designated/predetermined time even if the packets are fetched in advance. This helps maintaining a constant transmission rate that can be consumed by the receiver, therefore avoiding congestion and excessive buffering in the network.

Description of time-based scheduling

The time-based scheduling feature supports fetching and launching of an Ethernet packet at (or after) a predetermined time. The time-based scheduling is supported only in the following modes and configurations:

- Full duplex mode
- Link speed of 100 Mbps or higher

Note: Do not enable time-based scheduling for channels on which the TSO feature is enabled.

The ESTM mode bit setting impacts the time-based scheduling feature only when the DUT processes the GCL list.

The slot number is meaningful only after the GCL is active. So, start the GCL list before enabling the time-based scheduling (TBS) and enhancements to scheduled traffic (EST) features. If the GCL is not started, all gates are deemed open. In addition, if the packets do not make progress, the Tx queue creates back pressure.

Definitions

The following are the definitions specific to the time-based scheduling feature:

- Launch time: time beyond which the MTL can schedule the packet for transmission.
- GCL slot number (GSN)
- Launch expiry time: time beyond which the packet is dropped by the MTL.
- Fetch time: time beyond which the Tx DMA can schedule a packet fetch from the host memory.

Launch time

The launch time is specified in the enhanced normal descriptor in the DMA configuration. For more details, refer to *Section 76.10.5: Enhanced descriptor for time-based scheduling*.

When the launch time is specified, it is valid only for the specific frame that is scheduled for transmission. The launch time is reset after the transmission of one frame.

The following are the two launch time formats:

- **Normal/absolute format**

  In this format, the launch time is an absolute time value at which the packet is launched for transmission. The launch time is interpreted in the normal/absolute format if the ESTM bit of the *TBS control register (ETH_MTLTBSCR)* is cleared.

The launch time is a 32-bit value where the most-significant eight bits represent the time in seconds and the remaining 24 bits the time in 256 ns. The launch time is compared with the IEEE 1588-based system/PTP time (bits[39:8]) and rolls over after 256 seconds.

The maximum value of the lower 32 bits of system time is 999,999,999 in decimal (0x3B9A C9FF), and it wraps to 0 when reaching this value (representing a full second). Therefore, the maximum value of the lower 24 bits of the launch time (after multiplying by 256) must be 0x3B9AC9.

As the maximum value of the launch time is 256 seconds:

– The launch time is greater than the current system time when its value is between System Time[39:8] and System Time[39:8] + 128 s.

– The launch time is less than the current system time when its value is between System Time[39:8] + 128 s and System Time[39:8] + 256 s, because this is a modulo 256 computation.

- **EST/offset format**

  In this format, the launch time is an offset value relative to the time indicated by the base time register (BTR) of the GCL list provided in the GCL slot number (GSN). The value in the BTR is always updated to the start time of the current loop of the GCL.

  For each packet, the GSN and the launch time values are specified in the Enhanced Transmit descriptor in the DMA configurations, or as control word in the MTL configurations.

  The launch time offset is a 32-bit value where the upper eight bits represent the time in seconds and the remaining 24 bits the time in 256 ns that is added to the BTR value corresponding to the GCL slot number. The value of the launch time offset must be smaller than the value of the cycle time (it is specified in the CTR register that is implemented when EST is enabled).

  If the CTR is greater than or equal to 1 s, the maximum value of the lower 24 bits of launch time offset should be 999 999 999 in decimal (0x3B9A C9FF).

  In this format, the launch time is a 64-bit value, which is interpreted as follows:

$$\text{Launch time} = \text{Launch GSN BTR}[63:0] + (\text{Launch Time Offset}[31:0] \ll 8)$$

  It is compared with the System Time[63:0].

  GSN BTR is the base time value at which the launch GSN loop started execution.

GCL slot number (GSN)

The GCL slot number (GSN) is a modulo 16 count of the GCL loop count. The count is incremented for every new GCL loop. The installation of a new GCL list does not impact the count. The current GCL slot number can be obtained by reading the CGSN field of the *EST Status register (ETH_MTLESTSR)*.

The maximum value of GSN is 15. Therefore GSN values between the current GSN (CGSN field of *EST Status register (ETH_MTLESTSR)*) and CGSN + 8 represent the current or future slots; all the other GSN values are interpreted as elapsed slots or past slots. The GSN value must consequently be between CGSN and CGSN + 8, for a correct interpretation of time.

Launch expiry time

- **Normal/absolute mode**

  In Normal mode, when the LEOV (launch expiry offset valid bit of *TBS control register (ETH_MTLTBSCR)*) is set, the Launch Expiry Offset (LEOS field of *TBS control register (ETH_MTLTBSCR)*) determines the maximum amount of time during which a frame is eligible for launch, starting from the time at which the frame becomes eligible for launch.

  The Launch Expiry Offset is a 24-bit value defined in 256 ns units, with a maximum value of 999,999,999 ns (0x3B9A C9FF).

  $$\text{Launch expiry time} = (\text{Launch Time}[39:8] + \text{LEOS}[32:8]) \times 256\ \text{ns}$$

  The packet with a specific launch time is considered eligible for transmission when the launch time is less than the system time and (if LEOV is set) the system time is less than the launch expiry time.

  When the system time is greater than the launch expiry time, the frame is categorized as expired and is dropped from the MTL FIFO.

Note: For correct interpretation and meaningful operation, the fetch, launch, and launch expiry times should never be set to a value higher than the current system time + 128 s, since such a value is interpreted as time that has already elapsed.

In Full duplex mode, the frames dropped from the MTL FIFO have the Error Summary (bit 15) and Excessive Deferral (bit 3) of the Tx status set.

- **EST/offset mode**

  In EST mode of operation, when the LEOV field of the *TBS control register (ETH_MTLTBSCR)* is set, the Launch Expiry GSN Offset (LEGOS field of the *TBS control register (ETH_MTLTBSCR)*) and Launch Expiry Offset (LEOS field of *TBS control register (ETH_MTLTBSCR)*) determine the maximum amount of time during which a frame remains eligible for launch, starting from the time at which the frame becomes eligible for launch.

  $$\text{Launch Expiry Offset} = \text{LEGOS} / \text{LEOS}$$

  LEGOS holds the GSN offset (multiples of CTR time) and LEOS holds the maximum value of CTR (subCTR values) in ns.

  The Launch Expiry Offset is a 24-bit value defined in units of 256 ns, with a maximum value which is the smaller of 999,999,999 ns and CTR - 1 ns.

  The Launch Expiry GSN is computed as follows:

  $$\text{Launch Expiry GSN} = \text{Launch GSN} + \text{LEGOS} + \text{CCMA}$$

  Where:

  CCMA is the CTR carry due to modulo addition. This value is 1 if ((Launch Time Offset + LEOS) << 8) is equal to or greater than CTR.

  – When CCMA = 1: Launch Expiry Offset = (Launch Time Offset + LEOS) - CTR.
  – When CCMA = 0, Launch Expiry Offset = (Launch Time Offset + LEOS).

The launch expiry time is computed as follows:

$$\text{Launch expiry time} = \text{Launch Expiry GSN BTR}[63:0] + \text{Launch Expiry Time Offset}$$

When LEOV is not set, the launch expiry time is not checked.

When LEOV is set:

- If the system time is greater than the launch expiry time, the frame is dropped from the MTL FIFO. The frame is considered as expired.
- If the launch time is smaller than the system time and the launch expiry time is greater than the system time, the frame is considered eligible for launch.

Note: The maximum value of LEGOS is 7. This implies that when LEOV is set, the frame has a maximum life time of less than 8 GCL loop iterations after it becomes eligible for launch.

The slot number of the first GCL list executed each time after EST is enabled, is zero.

Fetch time

The fetch time can also be indicated by the software for each packet. This is done by setting the FTOV field of DMA TBS Control Register x (ETH_DMATBSCTRL0R).

If the FTOV filed is not set, the Fetch Time Offset is not valid and the DMA fetches packets without any time constraints.

The fetch time accounts for all possible delays in the DMA fetch operation and ensures that the frame is present in the MTL FIFO before the launch time.

- **Normal/absolute mode**

  In Normal mode, the fetch time is derived/calculated by reducing the time specified in the Fetch Time Offset (FTOS) field of DMA TBS Control Register x (ETH_DMATBSCTRL0R) from the given launch time.

  In Normal mode of operation, the fetch launch time is computed as follows:

  $$\text{Fetch time}[39:8] = \text{Launch Time}[39:8] - \text{FTOS}[31:8]$$

  The fetch time is a 32-bit value. It is compared against System Time[39:8] to determine the eligibility for fetching the frame:

  – The fetch time is defined as greater than the system time if the fetch time is in the range of System Time[39:8] and System Time[39:8] + 128 s. The frame is considered as not-eligible for fetch.

  – The fetch time is defined as smaller than the system time if the fetch time is in the range of System Time[39:8] + 128 s and System Time[39:8] + 256 s. The frame is considered as eligible for fetch.

  This is a modulo 256 computation.

- **EST/offset mode**

  In EST/offset mode, the Fetch GSN Offset (FGOS field of DMA TBS Control Register x (ETH_DMATBSCTRL0R) provides the slot number offset to be deducted from the launch GSN. In this case, the FTOS value should be the smaller of 999,999,999 ns and CTR – 1 ns.

  If Launch Time Offset ≥ FTOS:

  – Fetch Time Offset = ((Launch Time Offset – FTOS) × 256 ns)

  – CBFS (CTR Borrow for Fetch Subtraction) = 0

  If Launch Time Offset < FTOS:

  – Fetch Time Offset = CTR + ((Launch Time Offset – FTOS) × 256 ns)

  – CBFS (CTR Borrow for Fetch Subtraction) = 1

  The Fetch GSN is computed as follows:

  – Fetch GSN = Launch GSN – FGOS – CBFS

  – Fetch Time = Fetch GSN BTR[63:0] + Fetch Time Offset

Note: The frame is marked eligible for DMA fetch when the fetch time is smaller than the system time.

The maximum value of FGOS is 7. This implies that when FTOV is set, the frame can be fetched at a maximum of 8 GCL loop iterations before it becomes eligible for Launch.

- **DMA operation sequence**

  The following is the sequence of operation when FTOV is set:

  a) Fetch the first enhanced normal descriptor (FD is set).

  b) If LTV is set and fetch enabled, compute the fetch time based on the launch time. Wait for the system time to be greater than the fetch time

  c) Read the frame (data) from the host memory and transfer it to the MTL FIFO.

  d) Close the normal descriptor.

  e) Fetch the next normal descriptor (if the previous descriptor was not the last).

  f) Repeat steps **d** to **f** until the last descriptor of the frame (LD is set).

  After the last descriptor of a frame, the next enhanced normal descriptor should be programmed with a new launch time and with LTV bit set. Otherwise, the subsequent frames are processed without any time restrictions.

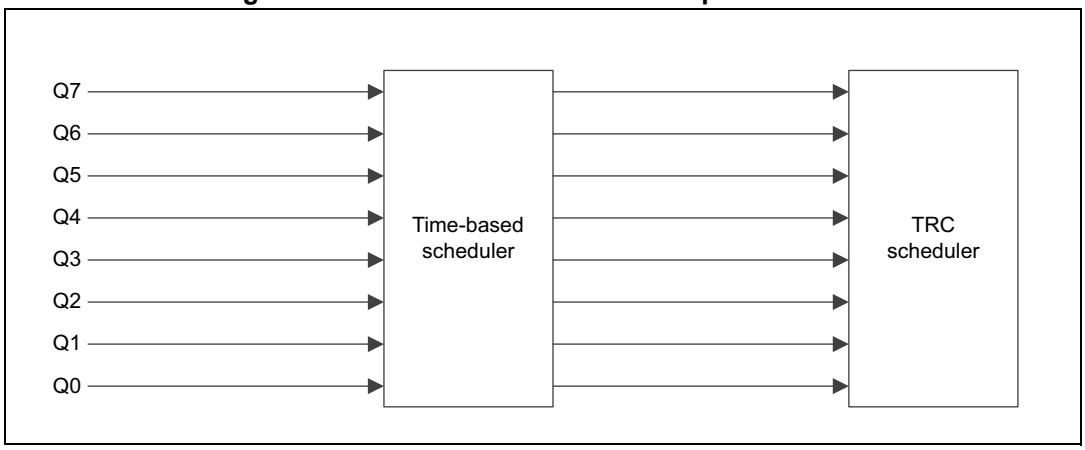
- **Impact on the transmission selection algorithm**

  The time-based scheduler does not directly influence the transmission selection algorithm but has an indirect influence as it determines if a queue is eligible to participate in the scheduling.

  If a frame of a specific queue has a valid launch time, the time-based scheduler ensures that the frame participates in transmit scheduling (TRC scheduler) only after the launch time has elapsed.

  The gating done by the time-based scheduler is only for the scheduling (picking frame for transmission). It might not alter any other characteristics and behavior of the frame. For example, the frame waiting for the launch time to elapse continues to gain credits in CBS, and the CBS credits are not lost as the frame is marked as available but not ready. Strict priority and CBS are recommended transmit scheduling algorithms for the TBS implementation because scheduling for TBS enabled queues is more predictable.

Figure 1039. Time-based scheduler implementation



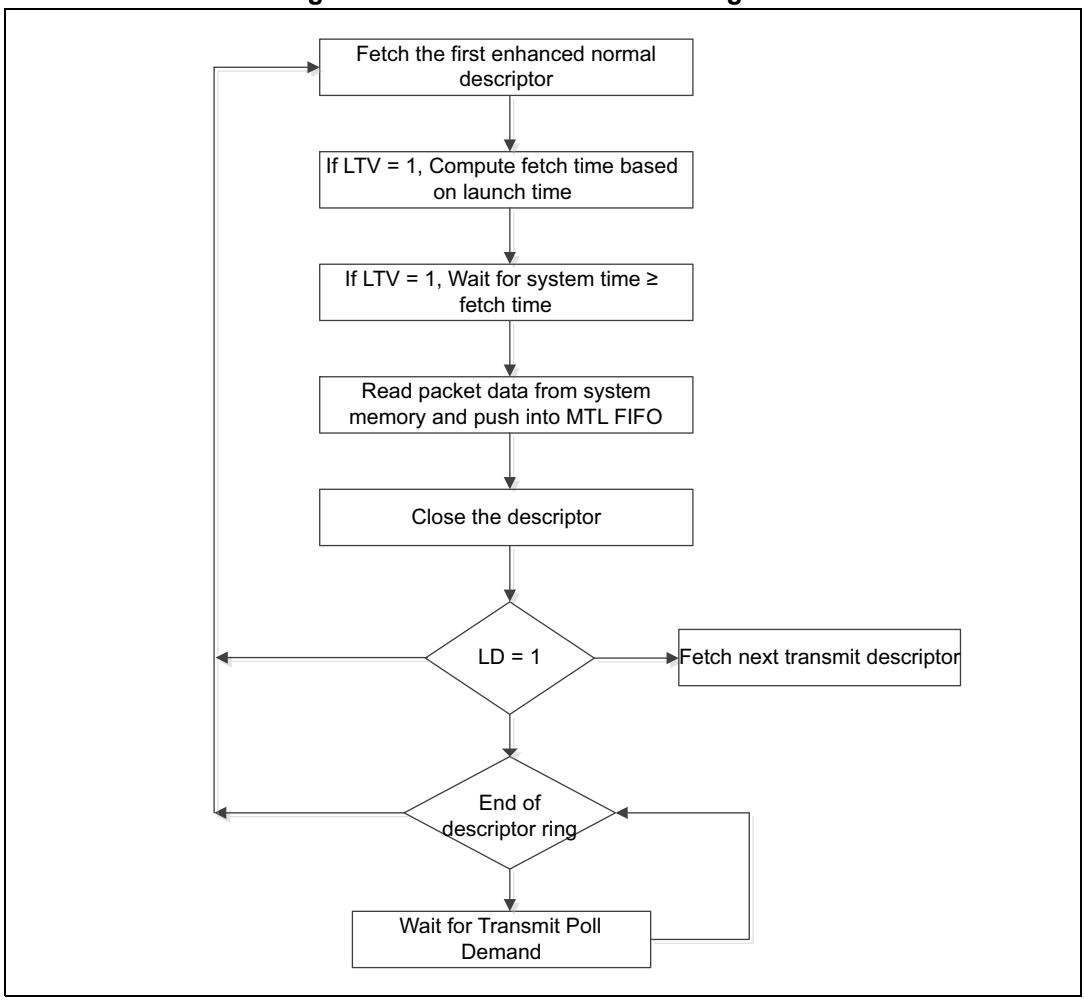
Programming guidelines for time-based scheduling

For details about programming guidelines, see *Section 76.9.21: Programming the launch time in time-based scheduling*.

Time-based scheduling flow

Figure 1040 shows the time-based scheduling flow.

Figure 1040. Time-based scheduling flow



76.5.7 Checksum offload engine

Communication protocols such as TCP and UDP implement checksum fields, which help determine the integrity of data transmitted over a network. The most widespread use of Ethernet is to encapsulate TCP and UDP over IP datagrams. The MAC has a Checksum Offload Engine (COE) to support checksum calculation and insertion in the Transmit path, as well as error detection in the receive path.

Transmit checksum offload engine

In the transmit path, the MAC calculates the checksum and inserts it in the Tx packet. This feature helps reducing the load on the software and can improve the overall system throughput.

The COE module supports two types of checksum calculation and insertion. The checksum engine can be controlled for each packet by setting the CIC bits (TDES3 bits[17:16]).

Note: The checksum for TCP, UDP, or ICMP is calculated over a complete packet, and then inserted into its corresponding header field. Because of this requirement, the Tx FIFO

automatically operates in the Store-and-forward mode even if the MAC is configured in Threshold (cut-through) mode.

Make sure that the Tx FIFO is deep enough to store a complete packet before that packet is transferred to the MAC transmitter, the reason being that when space is not available to accept the programmed burst length of data, then the MTL Tx FIFO starts reading to avoid deadlock. In such a case, the COE fails as the start of the packet header is read out before the payload checksum can be calculated and inserted. Therefore, the checksum insertion must be enabled only in the packets that are less than the number of bytes, given by the following equation:

$$\text{Packet size} < \text{TxQSize} - (\text{PBL} + 5) \times 9$$

where

TxQSize corresponds to the TQS bitfield of *Tx queue x operating mode register (ETH_MTLTXQxOMR)*

PBL corresponds to the TxPBL bitfield of *Channel x transmit control register (ETH_DMACxTXCR)*

Refer to IETF specifications RFC 791, RFC 793, RFC 768, RFC 792, RFC 2460, and RFC 4443 for IPv4, TCP, UDP, ICMP, IPv6, and ICMPv6 packet header specifications, respectively.

IP header checksum engine

In IPv4 datagrams, the integrity of the header fields is indicated by the 16-bit Header Checksum field (the eleventh and twelfth bytes of the IPv4 datagram). The COE detects an IPv4 datagram when the Type field of Ethernet packet has the value 0x0800 and the Version field of IP datagram has the value 0x4. The checksum field of the input packet is ignored during calculation and replaced with the calculated value.

Note: IPv6 headers do not have a checksum field. Therefore, the COE does not modify the IPv6 header fields.

The result of this IP header checksum calculation is indicated by the IP Header Error status it in the Transmit status (bit 0 in *Table 828: TDES3 normal descriptor (write-back format)*).

This status bit is set whenever the values of the Ethernet Type field and the Version field of IP header are not consistent, or when the Ethernet packet does not have enough data, as indicated by the IP header Length field. In other words, this bit is set when an IP header error is asserted under the following circumstances:

- For IPv4 datagrams:
  - The received Ethernet type is 0x0800, but the Version field of IP header is not equal to 0x4.
  - The IPv4 Header Length field indicates a value less than 0x5 (20 bytes).
  - The total packet length is less than the value given in the IPv4 Header Length field.
- For IPv6 datagrams:
  - The Ethernet type is 0x86DD but the IP header Version field is not equal to 0x6.
  - The packet ends before the IPv6 header (40 bytes) or extension header (as given in the corresponding Header Length field in an extension header) is completely received.

TCP/UDP/ICMP checksum engine

The TCP/UDP/ICMP Checksum Engine processes the IPv4 or IPv6 header (including extension headers) and determines whether the encapsulated payload is TCP, UDP, or ICMP. The checksum is calculated for the TCP, UDP, or ICMP payload and inserted into its corresponding field in the header. The Tx COE can work in the following two modes:

- The TCP, UDP, or ICMPv6 pseudo-header is not included in the checksum calculation and is assumed to be present in the Checksum field of the input packet. This engine includes the Checksum field in the checksum calculation, and then replaces the Checksum field with the final calculated checksum.
- The engine ignores the Checksum field, includes the TCP, UDP, or ICMPv6 pseudo-header data into the checksum calculation, and overwrites the checksum field with the final calculated value.

Note: For ICMP-over-IPv4 packets, the Checksum field in the ICMP packet must always be 0x0000 in both modes, because pseudo-headers are not defined for such packets. If it does not equal 0x0000, an incorrect checksum may be inserted into the packet.

The result of this operation is indicated by the Payload Checksum Error status bit in the Transmit Status vector (bit 12 in *Table 828: TDES3 normal descriptor (write-back format)*). This engine sets the Payload Checksum Error status bit when it detects that the packet has been forwarded to the MAC transmitter engine in the store-and-forward mode without the end of packet (EOP) being written to the FIFO, or when the packet ends before the number of bytes indicated by the Payload Length field in the IP Header is received. When the packet is longer than the indicated payload length, the COE ignores them as stuff bytes, and no error is reported. When this engine detects the first type of error, it does not modify the TCP, UDP, or ICMP header. For the second error type, it still inserts the calculated checksum into the corresponding header field.

Table 807 describes the functions supported by Transmit Checksum Offload engine based on the packet type. When the MAC does not insert the checksum, it is indicated as “No” in the table.

Note: Do not enable checksum insertion for IPv4 or IPv6 packets that are greater than the frame size constraint specified in Section : Transmit checksum offload engine because it might result in incorrect checksum insertion or unexpected behavior.

Table 807. Transmit checksum offload engine functions for different packet types

Packet type	Hardware IP header checksum insertion	Hardware TCP/UDP checksum insertion
Non-IPv4 or IPv6 packet	No	No
IPv4 packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad support for 2K packets is enabled in the MAC) but less than or equal to the frame size constraint specified in *Section : Transmit checksum offload engine*.	Yes	Yes
IPv6 packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad support for 2K packets is enabled in MAC) but less than or equal to the frame size constraint specified in *Section : Transmit checksum offload engine*.	Not applicable	Yes

Table 807. Transmit checksum offload engine functions for different packet types (continued)

Packet type	Hardware IP header checksum insertion	Hardware TCP/UDP checksum insertion
IPv4 with TCP, UDP, or ICMP	Yes	Yes
IPv4 packet has IP options (IP header is longer than 20 bytes)	Yes	Yes
Packet is an IPv4 fragment	Yes	No
IPv6 packet with the following next header fields in main or extension headers: – Hop-by-hop options (in IPv6 main header) – Hop-by-hop options (in IPv6 extension header) – Destinations options – Routing (with segment left 0) – Routing (with segment left > 0) – TCP, UDP, or ICMP – Authentication – Any other next header field in main or extension headers	 – Not applicable – Not applicable – Not applicable – Not applicable – Not applicable – Not applicable – Not applicable – Not applicable	 – Yes – No – Yes – No – No – Yes – No – No
IPv4 packet has TCP header with Options fields	Yes	Yes
IPv4 Tunnels: – IPv4 packet in an IPv4 tunnel – IPv6 packet in an IPv4 tunnel	 – Yes (IPv4 tunnel header) – Yes (IPv4 tunnel header)	 – No – No
IPv6 Tunnels: – IPv4 packet in an IPv6 tunnel – IPv6 packet in an IPv6 tunnel	 – Not applicable – Not applicable	 – No – No
IPv4 packet has 802.3ac tag (with C-VLAN tag or S-VLAN Tag when enabled).	Yes	Yes
IPv6 packet has 802.3ac tag (with C-VLAN tag or S-VLAN Tag when enabled).	Not applicable	Yes
IPv4 frames with security features (such as encapsulated security payload)	Yes	No
IPv6 frames with security features (such as encapsulated security payload)	Not applicable	No

Receive checksum offload engine

The receive checksum offload engine is used to detect errors in IP packets by calculating the header checksum and further matching it with the received header checksum. This engine also identifies a TCP, UDP, or ICMP payload in received IP packets and calculates the checksum of such payloads appropriately.

The receive checksum offload engine (Rx COE) can be enabled by setting the IPC bit of *Operating mode configuration register (ETH_MACCR)*. When this bit is set, both IPv4 and IPv6 packet in the received Ethernet packets are detected and processed for data integrity. The MAC receiver identifies IPv4 or IPv6 packets by checking for value 0x0800 or 0x86DD, respectively, in the Type field of the received Ethernet packet. This identification is applicable to single VLAN-tagged packets. It is also applicable to double VLAN-tagged packets when the EDVLP bit of the *VLAN tag Control register (ETH_MACVTCR)* is set.

The Rx COE calculates the IPv4 header checksums and checks that they match the received IPv4 header checksums. The result of this operation (pass or fail) is given to the RFC module for insertion into the receive status word. The IP Header Error bit is set for any mismatch between the indicated payload type (Ethernet Type field) and the IP header version, or when the received packet does not have enough bytes, as indicated by the Length field of the IPv4 header (or when fewer than 20 bytes are available in an IPv4 or IPv6 header).

Packets with TCP/IP errors (header or payload) are dropped in MTL when DIS_TCP_EF bit of the *Rx queue x operating mode register (ETH_MTLRXQxOMR)* is reset and FEP bit is set.

This engine also identifies a TCP, UDP, or ICMP payload in the received IP datagrams (IPv4 or IPv6) and calculates the checksum of such payloads properly, as defined in the TCP, UDP, or ICMP specifications. This engine includes the TCP, UDP, or ICMPv6 pseudo-header bytes for checksum calculation and checks whether the received checksum field matches the calculated value. The result of this operation is given as a Payload Checksum Error bit in the receive status word. This status bit is also set if the length of the TCP, UDP, or ICMP payload does not match the expected payload length given in the IP header.

Table 808: Receive checksum offload engine functions for different packet types describes the functions supported by the Rx COE based on the packet type. When the payload of an IP packet is not processed (indicated as "No" in the table), the information (whether the checksum engine is bypassed or not) is given in the receive status.

Note: The MAC does not append any payload checksum bytes to the received Ethernet packets.

Table 808. Receive checksum offload engine functions for different packet types

Packet type	Hardware IP header checksum checking	Hardware TCP/UDP/ICMP checksum checking
Non-IPv4 or IPv6	No	No
IPv4 packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad support for 2K packets is enabled in the MAC)	Yes	Yes
IPv6 packet is greater than 1,522 bytes (2,000 bytes when IEEE 802.3ad Support for 2K packets is enabled in the MAC)	Not applicable	Yes
IPv4 with TCP, UDP, or ICMP	Yes	Yes
IPv4 header's protocol field contains a protocol other than TCP, UDP, or ICMP	Yes	No
IPv4 packet has IP options (IP header is longer than 20 bytes)	Yes	Yes
Packet is an IPv4 fragment	Yes	No
IPv6 packet with the following next header fields in main or extension headers: – Hop-by-hop options (in IPv6 main header) – Hop-by-hop options (in IPv6 extension header) – Destinations options – Routing (with segment left 0) – Routing (with segment left > 0) – TCP, UDP, or ICMP – Any other next header field in main or extension headers	 – Not applicable – Not applicable – Not applicable – Not applicable – Not applicable – Not applicable – Not applicable	 – Yes – No – Yes – Yes – No – Yes – No
IPv4 packet has TCP header with Options fields	Yes	Yes
IPv4 Tunnels: – IPv4 packet in an IPv4 tunnel – IPv6 packet in an IPv4 tunnel	 – Yes (IPv4 tunnel header) – Yes (IPv4 tunnel header)	 – No – No
IPv6 Tunnels: – IPv4 packet in an IPv6 tunnel – IPv6 packet in an IPv6 tunnel	 – Not applicable – Not applicable	 – No – No
IPv4 packet has 802.3ac tag (with C-VLAN Tag or S-VLAN Tag when enabled).	Yes	Yes
IPv6 packet has 802.3ac tag (with C-VLAN Tag or S-VLAN Tag when enabled).	Not applicable	Yes
IPv4 frames with security features (such as encapsulated security payload)	Yes	No
IPv6 frames with security features (such as encapsulated security payload)	Not applicable	No

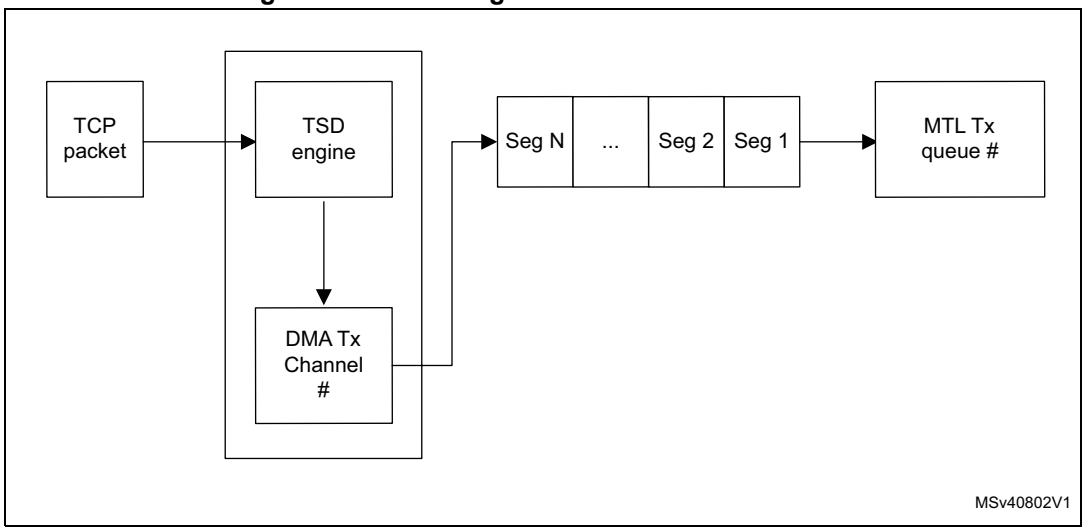
76.5.8 TCP segmentation offload

The MAC supports the TCP segmentation offload (TSO) feature in which the DMA splits a large TCP packet into multiple small packets and passes these packets to the MTL as shown in *Figure 1041*.

This feature is enabled by programming the TSE bit of corresponding ETH_DMACxCR register (see *Channel x transmit control register (ETH_DMACxTXCR)*). It is only supported when the MAC operates in Full-duplex mode.

For detailed programming steps, refer to *Section 76.9.16: Programming guidelines for TSO*.

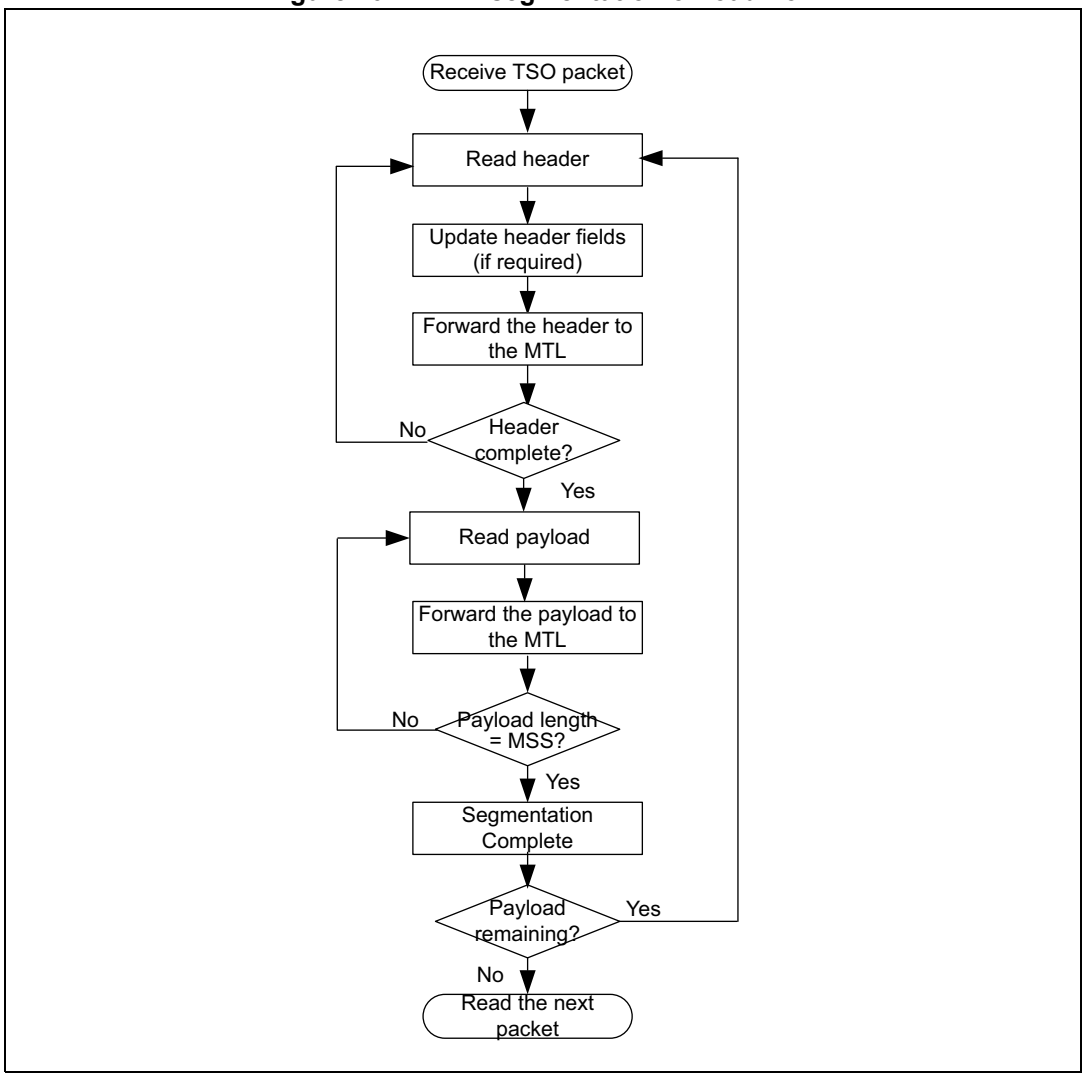
Figure 1041. TCP segmentation offload overview



DMA operation with TSO feature

Figure 1042 shows the TSO flow.

Figure 1042. TCP segmentation offload flow



For the TSO feature, the Tx DMA operation is as follows:

1. The application sets up the Transmit descriptor (TDES0-TDES3) and sets the Own bit (TDES3[31]) after setting up the corresponding data buffer(s) with Ethernet packet data.
2. The application increases the offset value of the descriptor tail pointer of the DMA Tx channel.
3. While in the Run state, the DMA fetches the next available descriptor and performs one of the following actions:
   - If the descriptor is a context descriptor and the context is not between the first and last descriptors of a packet, the DMA stores the context values.
   - If the descriptor is a context descriptor and the context is between the first and last descriptors of a packet, the DMA closes the context descriptor indicating a Context Descriptor Error (TDES3[23]) and fetches the next descriptor.

– If the descriptor is a normal descriptor, the DMA checks the TSE bit. If the TSE bit is not set, the DMA continues with the default mode of operation or OSF operation (if enabled).

4. The DMA calculates the number of segments from the TCP payload length (TDES3[17:0]) and the MSS value.
5. The DMA goes through channel arbitration to fetch the data buffers. The DMA fetches the header and payload separately.
6. For the first segment, the DMA fetches the header from the system memory and stores it in the TSO memory (if present and when the length of header is not greater than the TSO memory size). If the current segmented packet is not the first segment, it fetches again the header buffer in system memory, as done for the first segment. In such cases, the DMA does not close the first descriptor containing the header buffer until the header for last segment is fetched.
7. The required fields in the header bytes are modified/updated as per the segmentation requirements and written into the corresponding MTL Tx queue.
8. The DMA then takes the payload buffer pointer, fetches the MSS number of payload bytes from the system memory, and directly pushes it into the MTL Tx queue. If the buffer(s) in the descriptor do(es) not have enough data for the MSS payload (except for the last segment), the DMA closes this descriptor.
9. The DMA jumps to Step 3 and repeats the process until the last segment is written into the Tx queue.
10. The DMA closes the last descriptor and the first descriptor (containing the header buffer when it is not stored in TSO memory), and then moves on to the next packet transfer.

The DMA repeats all these steps if more descriptors are available. When no more descriptor are available, the DMA enters the suspend state.

Note: The TSO engine determines whether to perform TSO or USO operation based on the THL field (TCP Header Length) in TDES3 of first Normal Tx descriptor for the packet. The value of 2 indicates USO and any value greater than or equal to 5 indicates TSO.

TCP/IP header fields

While segmenting a TCP packet, the DMA automatically updates the TCP/IP header fields. *Table 809* describes how the TCP and IP headers are updated.

Table 809. TSO: TCP and IP header fields

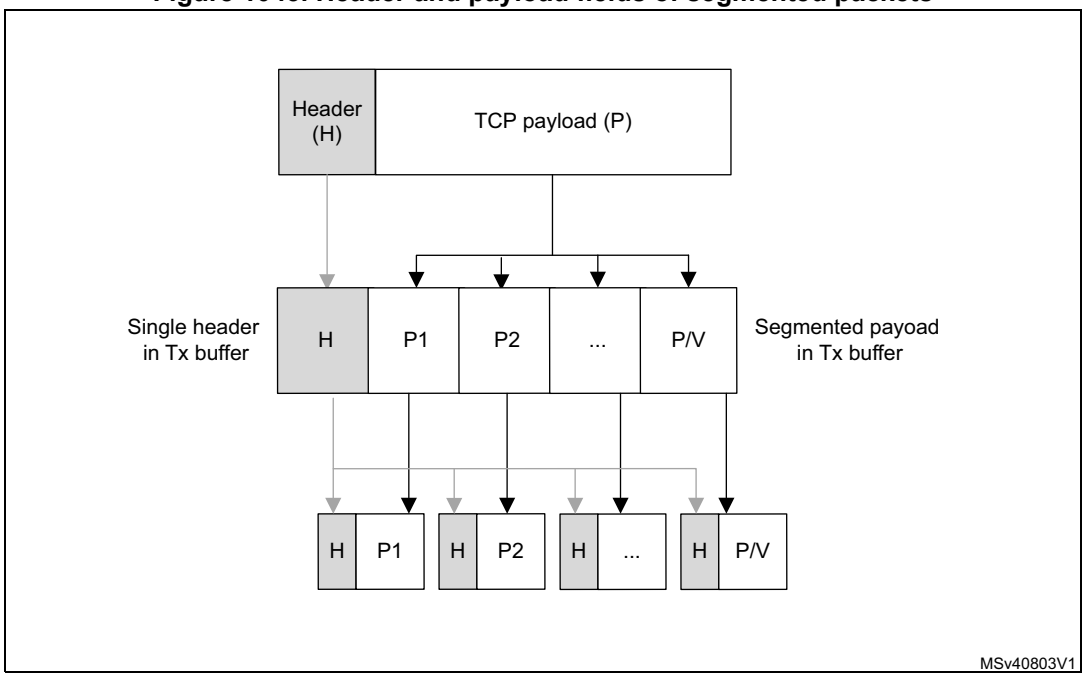
Packet sequence	TCP header	IP header
First packet	1. The sequence number is not updated. The value provided in the header is used. 2. If set, the FIN and PSH flags are cleared. 3. The TCP checksum is calculated again.	IPv4 Header – Total Length = MSS + TCP Header Length + IP Header Length – Identification field is not modified. It is sent as per the header provided by the software. – IPv4 Header Checksum is recalculated. IPv6 Header – Payload Length = MSS + TCP Header Length + IP Extension Header Length
Subsequent packets	1. The sequence number is updated. The MSS value is added to the sequence number value of previous segment. 2. If set, the FIN and PSH flags are cleared. 3. The TCP checksum is calculated again.	IPv4 Header – Total Length = MSS + TCP Header Length + IP Header Length – Identification field = Previous Identification Field + 1 – IPv4 Header Checksum is recalculated IPv6 Header – Payload Length = MSS + TCP Header Length + IP Extension Header Length
Last packet	1. The sequence number is updated. The MSS value is added to the sequence number value of previous segment. 2. If FIN and PSH flags were set in original header, these flags are set. 3. The TCP checksum is calculated again.	IPv4 Header – Total Length = Remaining Payload + TCP Header Length + IP Header Length – Identification Field = Previous Identification Field + 1 – IPv4 header Checksum is recalculated IPv6 Header – Payload Length = Remaining Payload Length + TCP Header Length + IP Extension Header Length

Header and payload fields of segmented packets

After segmentation, the split packets use the same header as the parent TCP packet for header fields other than the ones described in *Table 809: TSO: TCP and IP header fields*. *Figure 1043: Header and payload fields of segmented packets* shows how same header is used for the header fields of segmented packets.

The application must create the header in Buffer 1 of the first descriptor of the packet to be segmented and provide the header length in TDES2 of the first descriptor (FD = 1). When the FD bit is set, the DMA reads the header from the header buffer to which the TDES0 is pointing. Buffer 2 of the first descriptor can be used for payload and TDES0 and TDES1 of subsequent descriptors. For subsequent descriptors (FD = 0), the address to which the TDES0 and TDES1 are pointing is treated as payload buffer address of the same packet.

Figure 1043. Header and payload fields of segmented packets



Context descriptor sequence

The context descriptor can provide the maximum segment size (MSS) value for segmentation. The application must provide the context descriptor before the normal descriptor to be used for the corresponding TCP packet. If the application needs to provide a new MSS, it must create the context descriptor in the descriptor list before the first normal descriptor of the packet to be segmented with the new MSS value. The MSS value in the context descriptor is valid only if the TCMSSV bit of TDES3 in context descriptor is set and the OSTC bit is reset (refer to *Section 76.10.3: Transmit descriptor*).

When the application provides a context descriptor with a valid MSS value, the DMA internally stores the MSS value and uses this value for all subsequent packets for which the TSO is enabled through the TSE bit of TDES3 normal descriptor.

If the application places a context descriptor in the middle of a packet (between the first and last descriptors of a packet), the DMA does the following:

1. The DMA ignores the context and closes the descriptor.
2. The DMA indicates the error in descriptor status.
3. The DMA generates an interrupt if the CDEE bit is set in the Interrupt enable register corresponding to a DMA channel (see *Channel x interrupt enable register (ETH_DMACxIER)*).

The application can read the interrupt status through CDE bit of Status register corresponding to a DMA channel (see *Channel x status register (ETH_DMACxSR)*).

Building the Descriptor and the packet for the TSO feature

To enable segmentation for a packet, the application must set the TSE bit of TDES3 of first normal descriptor (see *Section 76.10.3: Transmit descriptor*). If the TSE bit is set in TDES3 for a non TCP/IP packet, the DMA behavior is unpredictable.

The application must program the length of the TCP packet payload in TDES3[17:0] and the TCP header in TDES3[22:19]. The maximum length of TCP packet payload that can be segmented is 256 Kbytes.

Note: The TDES3[22:19] bitfield is used as slot number in case of AV configurations. If both TSO and AV are enabled for the same DMA Tx channel (TSE bit is set in Channel x transmit control register (ETH_DMACxTXCR), then do not enable slot checking for that DMA Tx channel (do not set ESC bit in Channel x slot function control status register (ETH_DMACxSFCSR).

The header of the packet including Ethernet header, L3 header and L4 header should be provided in Buffer1 of the first normal descriptor of the TSO packet. Only buffer 1 of the first normal descriptor of a packet enabled for TSO is taken as the buffer containing the header.

The TCP payload can begin from buffer 2 of the first normal descriptor and continue to buffer1 and buffer 2 of second normal descriptor and subsequent descriptors.

The TCP payload may span across multiple buffers and multiple descriptors. The size of buffers containing the TCP payload should add up to be equal to the TCP payload length provided in TDES3[17:0] of the first normal descriptor.

The MAC always calculates and appends CRC and inserts Padding (if required) for all packets segmented by the DMA. If the TSE bit of TDES3 is enabled, the CRC PAD Control (CPC) field of TDES3 is reserved. To determine the size of a TCP packet after segmentation, the DMA uses the Maximum Segment Size (MSS) provided by the application through context descriptor. The DMA segments only those packets which have payload size greater than MSS. The application must provide the MSS by either programming the MSS value in ETH_DMACxCR (see *Channel x control register (ETH_DMACxCR)*) or by providing a context descriptor. The DMA uses the last programmed value of MSS or the last MSS value provided through context (whichever is provided later).

The header length plus the MSS size (which is equal to the size of each TCP segment) should not exceed 16383 bytes otherwise the MAC transmitter truncates the packet after 16383 bytes causing a CRC error.

The header length plus MSS size plus programmed PBL value in ETH_DMACxTXCR register (see *Channel x transmit control register (ETH_DMACxTXCR)*) should be lesser than the Tx queue size programmed in TQS field of ETH_MTLTXQxOMR register (see *Tx queue x*

operating mode register (ETH_MTLTXQxOMR)). A MSS plus header equal to half the programmed Tx queue size is recommended.

The DMA also supports segmentation of VLAN-tagged TCP/IP frames. If the TCP packet has a VLAN tag, then the same tag is used for all the segments irrespective of the VLAN tag type provided (C-VLAN or S-VLAN). The VLAN tag insert/replace control bits are used for all segments.

If the Double VLAN feature is selected, then the DMA passes both tags for all segments irrespective of the VLAN tag types provided (C-VLAN or S-VLAN). The VLAN tag Insert/Replace control bits for both tags is applicable to all segments. If the Double VLAN feature is not selected, then the application must not set the TSE bit in TDES3 for a TCP/IP packet with two tags. The DMA behavior in this scenario is unpredictable.

If the TSE bit is set in TDES3 for the packet and TCP header length provided is less than 5 (meaning this is an invalid TCP header because it is less than 20 bytes), the DMA does not perform segmentation, instead it transmits the entire packet as a single packet. In this scenario, the CRC pad control bits are forced by DMA to 00 (MAC does CRC and padding) and checksum insertion control bits are forced to 11 (hardware does the checksum calculation for both header and payload).

76.5.9 IPv4 ARP offload

The MAC supports the Address Recognition Protocol (ARP) Offload for IPv4 packets. This feature allows to process the IPv4 ARP request packet in the receive path and to generate the corresponding ARP response packet in the transmit path.

The MAC generates the ARP reply packets for appropriate ARP request packets**.** ARP packets for IPv4 are L2 layer packets with Length/Type of 0x0806.

The ARP offloading sequence is as follows:

1. The MAC receiver gets an ARP request if the request Target Protocol Address matches the IPv4 address programmed in the MAC L3 register.
2. The MAC generates an ARP reply packet.
3. The MAC copies the Sender Hardware Address field in the ARP request to the following fields:
   – DA field of the Ethernet packet header
   – Target Hardware Address field of the ARP reply packet
4. The MAC copies the Sender Protocol Address field in the ARP request to the Target Protocol Address field in the ARP reply packet.
5. The MAC places its MAC address in the following fields:
   – SA field of the Ethernet packet header
   – Sender Hardware Address field of the ARP reply packet
6. The MAC copies the Target Protocol Address field in the ARP request to the Sender Protocol Address field in the ARP reply packet.
7. The MAC sets the opcode field in ARP reply packet to 2 indicating ARP reply.
8. The MAC recalculates the CRC and performs padding for the generated ARP reply packet.
9. The MAC transmitter sends the ARP reply

The MAC processes only one ARP request at a time. It does not store the fields of multiple ARP requests. If the MAC receives an ARP request when it is already processing an earlier

ARP request, the MAC does not generate the ARP reply for new ARP request. The MAC forwards the new ARP request packet to the application with ARP Reply Not Generated status bit set (bit 34). However, in power-down mode, if the MAC receives an ARP request when it is already processing an earlier ARP request, the MAC drops the new ARP request. If the Disable CRC check (DCRCC) bit of the *Extended operating mode configuration register (ETH_MACECR)* is set, then the MAC does not check for valid CRC of a ARP request packet. It can generate an ARP response packet if the other conditions are valid. The ARP request packet must always have a valid CRC.

Note: When the received ARP request is less than the 64-byte packet length, the MAC does not send an ARP response. It is treated as a normal packet and forwarded to the application based on the MAC filter settings.

76.5.10 Loopback

The MAC supports loopback of transmitted packets to its receiver.

Guidelines for using loopback mode

Below some guidelines for using the loopback mode:

- Enable loopback only with the Full-duplex mode. In Half-duplex mode, the carrier sense signal or collision signal inputs get sampled which may result into issues such as packet dropping.
- If the loopback mode is enabled without connecting a PHY chip, externally generate the Tx and Rx clocks and provide these clocks to the MAC.
- Do not loop back big packets since they may get corrupted in the loopback FIFO.

The transmit and receive clocks can have an asynchronous timing relationship. Therefore, an asynchronous FIFO is used to make the loopback path of the transmitted data to the receive path. The FIFO is free-running to write on the write clock (eth_tx_clk) and read on every read clock (eth_rx_clk). At the start of each packet read from the FIFO, the write and read pointers are reinitialized to have an offset of two or four (in 10/100 Mbit/s mode). This avoids overflow or underflow during a packet transfer, and ensures that the overflow or underflow occurs only during the IPG period between the packets. The FIFO depth of five or nine is sufficient to prevent data corruption for packet sizes up to 9,022 bytes with a difference of 200 ppm between (G)MII transmit and receive clock frequencies.

Therefore, bigger packets should not be looped back because they may get corrupted in this loopback FIFO.

At the end of every received packet, the receive Protocol Engine module generates received packet status and sends it to the receive Packet Controller module. The control, missed packet, and filter fail status are added to the receive status in the receive Packet Controller module. The MAC does not process ARP or PMT packets that are looped back.

Enabling loopback mode

To enable this feature, program the LM bit of the *Operating mode configuration register (ETH_MACCR)*. Loopback can be enabled for all PHY interfaces. The data is always looped back through internal asynchronous FIFO on to the internal receive MII or GMII interface, irrespective of which PHY interface is selected.

The loopback data is also passed through the corresponding transmit PHY interface block, onto the Ethernet line.

Note: During loopback, the data/packet is reflected on the gmii_txd signal.

Preemption is not supported in loopback mode.

76.5.11 Flow control

The transmit flow control involves transmitting Pause packets in Full-duplex mode and back-pressure in Half-duplex mode to control the flow of packets from the remote end. This section describes the flow control for transmit and receive paths.

Flow control in Full-duplex mode

In Full-duplex mode, the MAC uses IEEE 802.3x Pause packets for flow control. *Table 810* describes the fields of a Pause packet.

Table 810. Pause packet fields

Field	Description
DA	Contains the special multicast address
SA	Contains the MAC address 0
Type	Contains 8808
MAC Control opcode	Contains 0001 for IEEE 802.3x Pause Control packets
PT	Contains Pause time specified in the PT field of the *Tx Queue 0 flow control register (ETH_MACQ0TXFCR)*

When the FCB bit is set, the MAC generates and transmits a single Pause packet. If the FCB bit is set again after the Pause packet transmission is complete, the MAC sends another Pause packet irrespective of whether the pause time is complete or not. To extend the pause or terminate the pause prior to the time specified in the previously-transmitted Pause packet, the application should program the Pause Time register with appropriate value and then again set the FCB bit.

Flow control in Half-duplex mode

In Half-duplex mode, the MAC uses the deferral mechanism for the flow control (backpressure). When the application requests to stop receiving packets, the MAC sends a JAM pattern of 32 bytes when it senses a packet reception, provided the transmit flow control is enabled. This results in a collision and the remote station backs off. If the application requests a packet to be transmitted, it is scheduled and transmitted even when the backpressure is activated. If the backpressure is kept activated for a long time (and more than 16 consecutive collision events occur), the remote stations abort the transmission because of excessive collisions.

Table 811 describes the flow control in the Tx path for Queue 0 based on the setting of the following bits:

- EHFC bit of *Rx queue x operating mode register (ETH_MTLRXQxOMR)*
- TFE bit of *Tx Queue 0 flow control register (ETH_MACQ0TXFCR)*
- DM bit of *Operating mode configuration register (ETH_MACCR)*

Flow control is similar for all queues.

Table 811. Tx MAC flow control

EHFC	TFE	DM	Description
X	0	X	The MAC transmitter does not perform the flow control or backpressure operation.
0	1	0	The MAC transmitter performs backpressure when Bit 0 of *Tx Queue 0 flow control register (ETH_MACQ0TXFCR)* is set.
1	1	0	The MAC transmitter performs backpressure when Bit 0 of *Tx Queue 0 flow control register (ETH_MACQ0TXFCR)* is set. In addition, the MAC Tx performs backpressure when Rx queue level crosses the threshold set by Bits[10:8] of *Rx queue x operating mode register (ETH_MTLRXQxOMR)*.
0	1	1	The MAC transmitter sends the Pause packet when Bit 0 of *Tx Queue 0 flow control register (ETH_MACQ0TXFCR)* is set.
1	1	1	The MAC transmitter sends the Pause packet when Bit 0 of *Tx Queue 0 flow control register (ETH_MACQ0TXFCR)* is set. In addition, the MAC Tx sends a Pause packet when Rx queue level crosses the threshold set by Bits[10:8] of *Rx queue x operating mode register (ETH_MTLRXQxOMR)*.

Transmit flow control

The transmit flow control is enabled when TFE bit is set in *Tx Queue 0 flow control register (ETH_MACQ0TXFCR)*.

Flow control trigger

The application can request the MAC to send a Pause packet or initiate back-pressure by setting the FCB bit in the corresponding *Tx Queue 0 flow control register (ETH_MACQ0TX-FCR)*.

The flow control is also triggered based on Rx Queue Threshold: the flow control operation of the MAC is enabled when the EHFC bit in the corresponding *Rx queue x operating mode register (ETH_MTLRXQxOMR)* is set. The flow control signal to the MAC is asserted when the fill level of the Rx queue crosses the threshold configured in RFA field of *Rx queue x operating mode register (ETH_MTLRXQxOMR)*. This flow control signal is de-asserted when the fill-level of the queue is lower than the threshold configured in the RFD field.

Note: The RFA sets the threshold at which the flow control is triggered and the MAC schedules to send a PAUSE frame to be sent by the local transmitter.

The remote transmitter can continue to send packets until it receives this PAUSE packet. As a results, the RXQ needs space to take in this data even after the flow control is triggered, in order to avoid overflow and loss of packets.

The maximum space required is calculated theoretically. It can be 2 max-sized packets + a little more assuming that the read from the RxFIFO is not occurring during this whole time.

Theoretically, this makes RFA = 4 or more, which implies that RxQ must have a minimum size of 4 Kbytes.

Practically, the system works with RFA = 2 (full size – 2 Kbytes, with very little probability of overflow).

The RFA bitfield in Rx queue x operating mode register (ETH_MTLRXQxOMR) is used for activating the flow control. These bits control the threshold (fill-level of Rx queue) at which the flow control is activated.

With an RXQ size of 4 Kbytes, RFA must be set to 2 Kbytes to receive the whole packet.

Receive flow control

In the receive path, the flow control is functional only in full-duplex mode. If any Pause packet is received in half-duplex mode, the packet is considered as a normal control packet.

Description of receive flow control

Table 812 describes the flow control in the Rx path based on the setting of the following bits:

- RFE bit of Rx flow control register (ETH_MACRXFCR)
- DM bit of Operating mode configuration register (ETH_MACCR)

Table 812. Rx MAC flow control

RFE	DM	Description
0	x	The MAC receiver does not detect the received Pause packets.
1	0	The MAC receiver does not detect the received Pause packets but recognizes such packets as Control packets.
1	1	The MAC receiver detects or processes the Pause packets and responds to such packets by stopping the MAC transmitter.

The following sequence describes the Rx flow control:

1. The MAC checks the destination address (DA) of the received Pause packet for either of the following:
   - Multicast destination address: the DA matches the unique multicast address specified for the control packet (0x0180 C200 0001).
   - Unicast destination address: the DA matches the content of the MAC Address 0 registers (MAC address 0 high register (ETH_MACA0HR) and MAC address x low register (ETH_MACAxLR)) and the UP bit of Rx flow control register (ETH_MACRXFCR) is set.

     If the UP bit is set, the MAC processes Pause packets with unicast destination address in addition to the unique multicast address.
2. The MAC decodes the following fields of the received packet:
   - Type field: this field is checked for 0x8808.
   - Opcode field: this field is checked for 0x0001 (Pause packet).
   - Pause time: the Pause time (for Pause packet) is captured to determine the time for which the transmitter needs to be blocked.
3. If the byte count of the status indicates 64 bytes and there is no CRC error, the MAC transmitter pauses the transmission of any data packet for the duration of the decoded Pause Time value multiplied by the slot time (64 byte times).

If subsequent Pause packets are received before the earlier Pause Time expires, the MAC updates the Pause Timer with new value.

Enabling receive flow control

Set the RFE bit in the *Rx flow control register (ETH_MACRXFCR)* to enable the Pause flow control.

76.5.12 MAC management counters

The peripheral supports storing the statistics about the received and transmitted packets in registers that are accessible through the application.

The counters in the MAC management counters (MMC) module can be viewed as an extension of the register address space of the CSR module. The MMC module maintains a set of registers for gathering statistics on the received and transmitted packets. The register set includes a control register for controlling the behavior of the registers, two 32-bit registers containing interrupts generated (receive and transmit), and two 32-bit registers containing masks for the Interrupt register (receive and transmit). These registers are accessible from the Application through the AHB slave interface in the same way the CSR registers are accessed. The organization of these registers is shown in *Section 76.11.4: Ethernet MAC and MMC registers*.

The MMC counters are free running. There is no separate enable for the counters to start. A particular MMC counter starts counting when corresponding packet is received or transmitted.

The receive MMC counters are updated for packets that are passed by the Address Filter (AFM) block. The statistics of packets dropped by the AFM module, are not updated unless they are runt packets of less than 6 bytes (DA bytes are not received fully). To get statistics of all packets, set bit 0 in the *Packet filtering control register (ETH_MACPFR)*. The MMC module gathers statistics on encapsulated IPv4, IPv6, TCP, UDP, or ICMP payloads in received Ethernet packets.

In addition to control registers, two sets of registers are implemented:

- Six registers used for collision, error, and good packet counters:
  - *Tx single collision good packets register (ETH_TX_SINGLE_COLLISION_GOOD_PACKETS)*
  - *Tx multiple collision good packets register (ETH_TX_MULTIPLE_COLLISION_GOOD_PACKETS)*
  - *Tx packet count good register (ETH_TX_PACKET_COUNT_GOOD)*
  - *Rx CRC error packets register (ETH_RX_CRC_ERROR_PACKETS)*
  - *Rx alignment error packets register (ETH_RX_ALIGNMENT_ERROR_PACKETS)*
  - *Rx unicast packets good register (ETH_RX_UNICAST_PACKETS_GOOD)*
- Four registers to record LPI mode transition:
  - *Tx LPI microsecond timer register (ETH_TX_LPI_USEC_CNTR)*
  - *Tx LPI transition counter register (ETH_TX_LPI_TRAN_CNTR)*
  - *Rx LPI microsecond counter register (ETH_RX_LPI_USEC_CNTR)*
  - *Rx LPI transition counter register (ETH_RX_LPI_TRAN_CNTR)*

Definitions

The following terminology is used in MMC register descriptions:

- Transmitted packets are considered “good” if transmitted successfully. In other words, a transmitted packet is good if the packets transmission is not aborted because of any of the following errors:
  - Jabber timeout
  - No carrier or loss of carrier
  - Late collision
  - Packet underflow
  - Excessive deferral
  - Excessive collision
- Received packets are considered “good” if none of the following errors exists:
  - CRC error
  - Runt packet (shorter than 64 bytes)
  - Alignment error (in 10/100 Mbit/s only)
  - Length error (non-Type packet only)
  - Out of range (non-Type packet only, longer than 1518 bytes)
  - GMII_RXER input error
- The maximum transmit frame size depends on the frame type, as follows:
  - Untagged frame maxsize = 1,518
  - VLAN frame maxsize = 1,522
  - Jumbo frame maxsize = 9,018
  - JumboVLAN frame maxsize = 9,022
- The maximum receive packet size depends on the packet type and control bits (JE, S2KP, GPSLCE and EDVLP), as shown in the *Table 813*.

Table 813. Size of the maximum receive packet

JE	S2KP	GPSLCE	EDVLP	Untagged frame maximum size in bytes	Single VLAN frame maximum size in bytes	Double VLAN Frame maximum size in bytes
1	X	X	1	9018	9022	9026
0	1	X	X	2000	2000	2000
0	0	1	1	GPSL	GPSL+4	GPSL+8
0	0	0	1	1518	1522	1526
1	X	X	0	9018	9022	9022
0	0	1	0	GPSL	GPSL+4	GPSL+4
0	0	0	0	1518	1522	1522

76.5.13 Interrupts generated by the MAC

Interrupts can be generated from the MAC as a result of various events. These interrupt events are combined with the events in the DMA on the eth_sbd_intr_it signal. The MAC interrupts are of level type, that is, the interrupt remains asserted (high) until it is cleared by the application or software.

The *Interrupt status register (ETH_MACISR)* describes the events that can cause an interrupt from the MAC. The MAC interrupts are enabled by default. Each event can be prevented from asserting the interrupt on the eth_sbd_intr_it signals by setting the corresponding mask bits in the *Interrupt enable register (ETH_MACIER)*.

The interrupt register bits only indicate the block from which the event is reported. You must read the corresponding status registers and other registers to clear the interrupt.

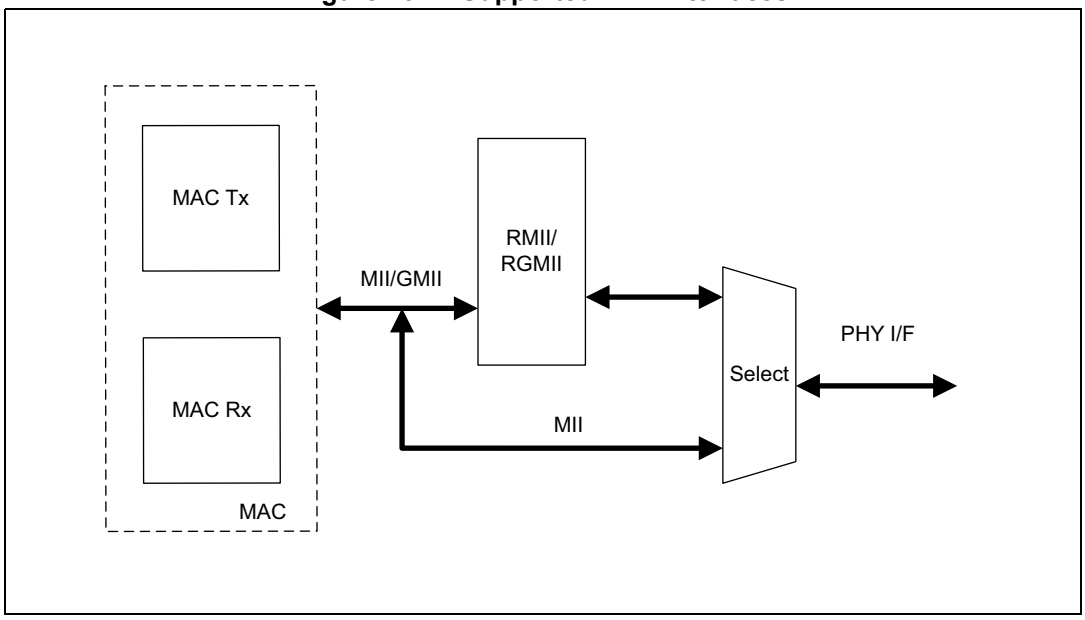
76.5.14 MAC and MMC register descriptions

Refer to *Section 76.11.4: Ethernet MAC and MMC registers*.

76.6 Ethernet functional description: PHY interfaces

The Ethernet peripheral support several PHY interfaces. The root interface is the MII/GMII one. All other interfaces are derived from it as shown in *Figure 1044*.

Figure 1044. Supported PHY interfaces



This section describes the SMA module used for PHY control and different PHY interfaces. It contains the following sections:

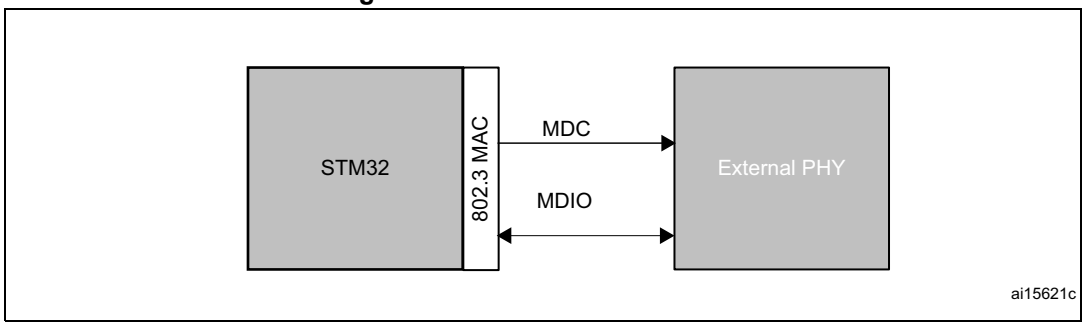
- *Station management agent (SMA)*
- *Media independent interface (MII)*
- *Reduced media independent interface (RMII)*
- *Reduced gigabit media independent interface (RGMII)*

76.6.1 Station management agent (SMA)

The application can access the PHY registers through the station management agent (SMA) module. The SMA includes a two-wire station management interface (MIM).

The SMA module supports accessing up to 32 PHYs. The application can address one of the 32 registers from any 32 PHYs. Only one register in one PHY can be addressed at a time. The application sends the control data to the PHY and receives status information from the PHY through the SMA module, as shown in *Figure 1045*.

Figure 1045. SMA Interface block



SMA functional overview

The MAC initiates the management write or read operation with respect to the MDC clock. The MDC clock is derived from the CSR clock (eth_hclk). The maximum operating frequency of the ETH_MDC pin is 2.5 MHz, as specified in IEEE 802.3 specifications. However, a different divider can be selected if the system supports higher clock frequencies. The division factor depends on the clock range setting through CR[3:0] in the *MDIO address register (ETH_MACMDIOAR)* register. The MDC clock is selected as follows:

Table 814. MCD clock selection

Selection	MDC clock
0000	eth_hclk/42
0001	eth_hclk/62
0010	eth_hclk/16
0011	eth_hclk/26
0100	eth_hclk/102
0101	eth_hclk/124
0110	eth_hclk/204
0111	eth_hclk/324
1000	eth_hclk/4
1001	eth_hclk/6
1010	eth_hclk/8
1011	eth_hclk/10
1100	eth_hclk/12
1101	eth_hclk/14
1110	eth_hclk/16
1111	eth_hclk/18

The data exchange between the MAC and the PHY is performed through a three-state buffer and brought out as ETH_MDIO line connected to the PHY.

Figure 1046 shows the structure of a Clause 45 MDIO packet, while *Table 815* provides a detailed description of the packet fields.

Figure 1046. MDIO packet structure (Clause 45)

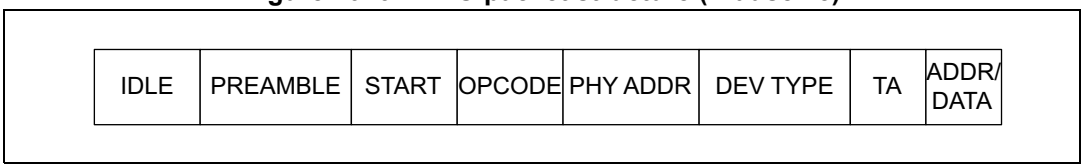


Table 815. MDIO Clause 45 frame structure

Field	Description
IDLE	The ETH_MDIO line is three-state; there is no clock on ETH_MDC.
PREAMBLE	32 continuous bits of value 1
START	Start of packet is 0b00
OPCODE	– 0b00: Address – 0b01: Write – 0b10: Read+ Address – 0b11: Read
PHY ADDR	5-bit address select for one of 32 PHYs
DEV TYPE	5-bit device type
TA	Turnaround – 0bZ0: Read and post-read increment address – 0b10: Write and address MDIO accesses where Z is the tri-state level
DATA	16-bit value: for an address cycle (OPCODE = 0b00), this frame contains the address of the register to be accessed on the next cycle. For the data cycle of a write frame, this field contains the data to be written to the register. For read or post-read increment address frames, this field contains the contents of the register read from the PHY. – In address and data write cycles, the peripheral drives the ETH_MDIO line during the transfer of these 16 bits. – In read and post-read increment address cycles, the PHY drives the ETH_MDIO line during the transfer of these 16 bits.

The frame structure for Clause 22 frames is also supported. The C45E bit in *MDIO address register (ETH_MACMDIOAR)* can be programmed to enable Clause 22 or Clause 45 mode of operation. *Figure 1047* shows the structure of a Clause 22 MDIO packet, while *Table 816* provides a detailed description of the packet fields.

Figure 1047. MDIO packet structure (Clause 22)

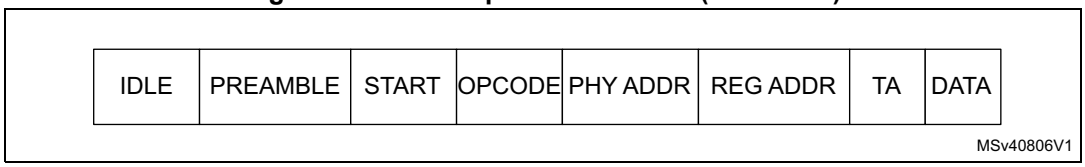


Table 816. MDIO Clause 22 frame structure

Field	Description
IDLE	The ETH_MDIO line is three-state; there is no clock on ETH_MDC.
PREAMBLE	32 continuous bits of value 1
START	Start of packet is 0b01
OPCODE	0b10 for Read and 0b01 for Write
PHY ADDR	5-bit address select for one of 32 PHYs
REG ADDR	Register address in the selected PHY
TA	Turnaround – 0bZ0: Read and post-read increment address – 0b10: Write and address MDIO accesses where Z is the tri-state level
DATA	Any 16-bit value. In a Write operation, the MAC drives ETH_MDIO. In a Read operation, the PHY drives it.

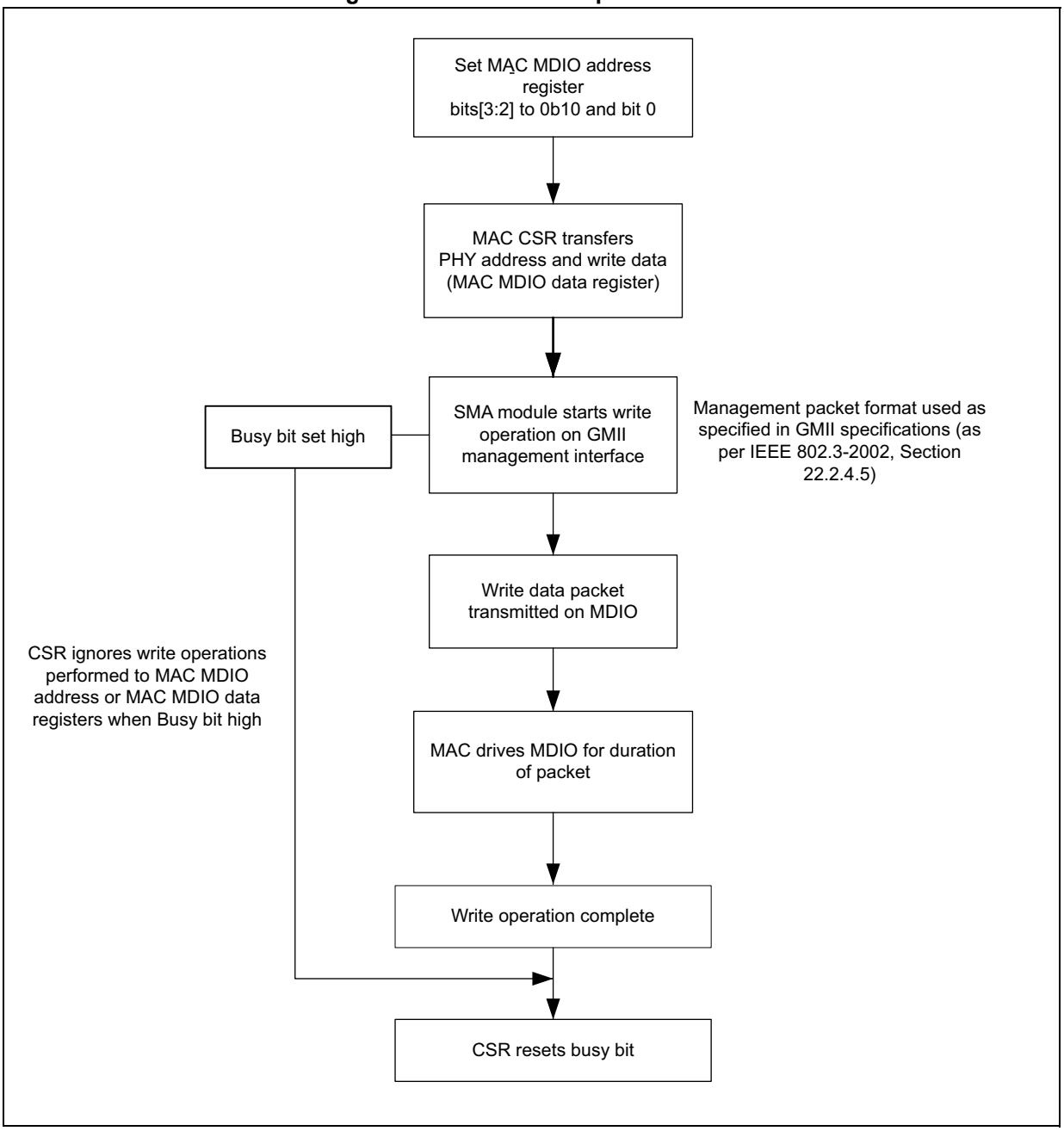
In addition to normal read and write operations, the SMA also supports post-read increment address while operating in Clause 45 mode.

GMII/MII management write operations

After the station management agent receives the PHY address and the write data from the MAC CSR module, the SMA starts a write operation to the PHY registers.

Figure 1048 illustrates the flow for a write operation from the SMA module to the PHY registers.

Figure 1048. SMA write operation flow



When bits[3:2] are set to 01 and bit 0 to 1 in the *MDIO address register (ETH_MACMDIOAR)*, the MAC CSR module transfers the PHY address, the register address in PHY, and the write data (*MDIO data register (ETH_MACMDIODR)*) to the SMA to initiate a Write operation into the PHY registers. At this point, the SMA module starts a Write operation on the GMII management Interface using the management packet format specified in the GMII specifications (as per IEEE 802.3-2002 specifications, *Section 22.2.4.5*).

When the SMA module starts a Write operation, the write data packet is transmitted on the MDIO line. The MAC drives the MDIO line for complete duration of the packet. The Busy bit is set high until the write operation is complete. The CSR ignores the Write operations per-

formed to the *MDIO address register (ETH_MACMDIOAR)* or the *MDIO data register (ETH_MACMDIODR)* during this period (the Busy bit is high). When the Write operation is complete, the SMA module indicates this to the CSR, and the CSR resets the Busy bit. The packet format for the Write operation is as follows:

Figure 1049. Write data packet

IDLE	PREAMBLE	START	OPCODE	PHY ADDR	REG ADDR	TA	DATA	IDLE
Z	1111..11	01	01	AAAAA	RRRRR	10	DDD...DDD	Z

MSv40807V1

GMII/MII management read operation

When bits[3:2] are set to 11 and bit 0 to 1 in the *MDIO address register (ETH_MACMDIOAR)*, the MAC CSR module transfers the PHY address and the register address in PHY to the SMA to initiate a Read operation in the PHY registers. At this point, the SMA module starts a Read operation on the GMII/MII management interface using the management packet format specified in the GMII/MII specifications (as per IEEE 802.3-2002 specifications, *Section 22.2.4.5*).

When the SMA module starts a Read operation on the MDIO, the CSR ignores the Write operations to the *MDIO address register (ETH_MACMDIOAR)* or *MDIO data register (ETH_MACMDIODR)* register during this period (the Busy bit is high) and the transaction is completed without any error on the MCI interface. When the Read operation is complete, the SMA indicates this to the CSR. The CSR resets the Busy bit and updates the *MDIO data register (ETH_MACMDIODR)* with the data read from the PHY. The MAC drives the MDIO line for the complete duration of the frame except during the Data fields when the PHY is driving the MDIO line. For more information about the communication from the application to the PHYs, see the Reconciliation Sublayer and Media Independent Interface Specifications sections of the IEEE 802.3z, 1000BASE Ethernet.

The packet format for the Read operation is as follows:

Figure 1050. Read data packet

IDLE	PREAMBLE	START	OPCODE	PHY ADDR	REG ADDR	TA	DATA	IDLE
Z	1111..11	01	10	AAAAA	RRRRR	Z0	DDD...DDD	Z

MSv40808V1

Preamble suppression

The IEEE standard specifies 32-bit preamble (all-ones) for the MDIO frames. The peripheral provides controls to support preamble suppression. It transmits MDIO frames with only 1 preamble bit. The preamble suppression can be enabled by setting the PSE bit in *MDIO address register (ETH_MACMDIOAR)*.

Trailing clocks and back-to-back transactions

The peripheral drives the ETH_MDC clock for the duration of the MDIO frame. There is no clock driven during the idle period. The trailing clock feature can be used if the PHY needs the ETH_MDC clock to be active for some cycles after the MDIO frame. The NTC[2:0] bitfield in *MDIO address register (ETH_MACMDIOAR)* allows the programming of trailing clocks from 0 to 7.

The peripheral supports back-to-back transactions which allow starting the next MDIO frame even before the trailing clocks are complete for previous MDIO frame. This feature can be enabled by setting BTB bit in *MDIO address register (ETH_MACMDIOAR)* when the trailing clock feature is also enabled. When back-to-back transactions are enabled, the GMII busy bit (GB) is cleared immediately after MDIO frame completion. This allows the software to issue the next command, which is executed by the peripheral while trailing clocks are still on for the previous MDIO frame. When (GB) transactions are not enabled, the GMII busy bit is cleared after the trailing clocks are complete for the MDIO frame.

Interrupt for MDIO transaction completion

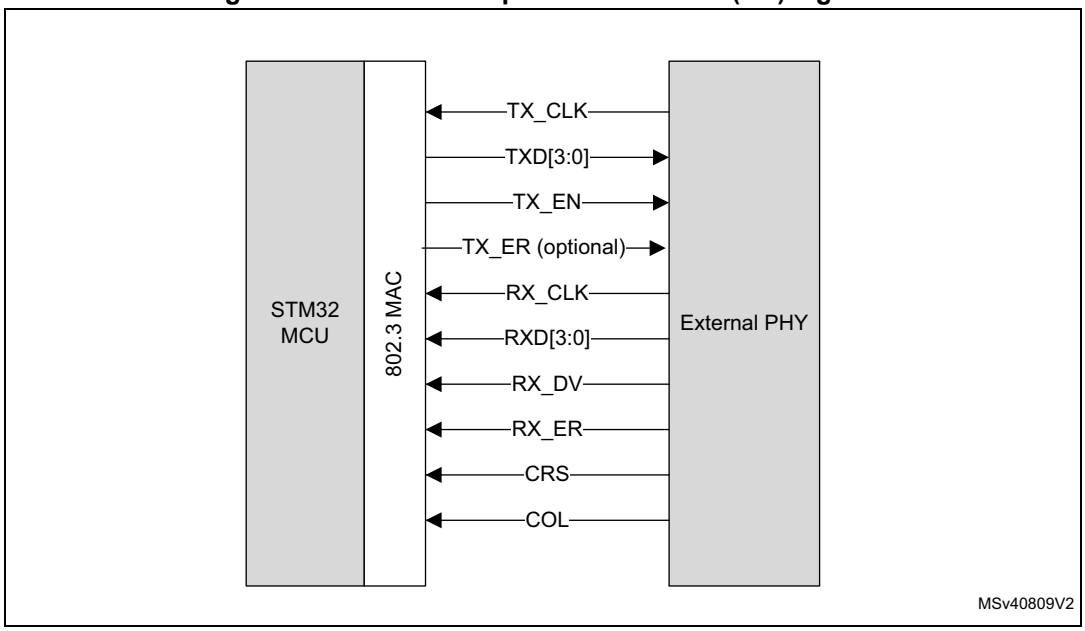
The peripheral can generate an interrupt on completion of MDIO read or write transactions. Therefore, the application need not poll the GMII busy bit of the *MDIO address register (ETH_MACMDIOAR)* to know the completion of MDIO commands.

76.6.2 Media independent interface (MII)

The media-independent interface (MII) defines the interconnection between the MAC sublayer and the PHY for data transfer at 10 Mbit/s and 100 Mbit/s.

MII signals are given in *Figure 1051: Media independent interface (MII) signals*.

Figure 1051. Media independent interface (MII) signals



- TX_CLK: continuous clock that provides the timing reference for Tx data transfers. The nominal frequency is 2.5 MHz at 10 Mbit/s and 25 MHz at 100 Mbit/s.
- TXD[3:0]: transmit data.

  TXD is a bundle of 4 data signals driven synchronously by the MAC sublayer and qualified (valid data) on the assertion of the TX_EN signal. TXD[0] is the least significant bit, TXD[3] is the most significant bit. While TX_EN is deasserted, the transmit data must have no effect upon the PHY.
- TX_EN: transmission enable signal indicating that the MAC is presenting nibbles on the MII for transmission. It must be asserted synchronously (TX_CLK) with the first nibble of the preamble and must remain asserted while all nibbles to be transmitted are presented to the MII.
- TX_ER (optional): required only for Energy Efficient Ethernet (EEE). The transmit error is indicated by inverting the CRC. The remote station can detect the Transmit error through incorrect CRC.
- RX_CLK: continuous clock that provides the timing reference for Rx data transfers. The nominal frequency is 2.5 MHz at 10 Mbit/s, 25 MHz at 100 Mbit/s.
- RXD[3:0]: receive data

  RXD is a bundle of 4 data signals driven synchronously by the PHY and qualified (valid data) on the assertion of the RX_DV signal. RXD[0] is the least significant bit, RXD[3] is the most significant bit. While RX_EN is deasserted and RX_ER is asserted, a specific RXD[3:0] value is used to transfer specific information from the PHY.

- RX_DV: receive data valid

  This signal indicates that the PHY is presenting recovered and decoded nibbles on the MII for reception. It must be asserted synchronously (RX_CLK) with the first recovered nibble of the frame and must remain asserted through the final recovered nibble. It must be deasserted prior to the first clock cycle that follows the final nibble. In order to receive the frame correctly, the RX_DV signal must encompass the frame, starting no later than the SFD field.

- RX_ER: receive error

  This signal must be asserted for one or more clock periods (RX_CLK) to indicate to the MAC sublayer that an error was detected somewhere in the frame. This error condition must be qualified by RX_DV assertion.

- CRS: carrier sense.

  This signal is asserted by the PHY when either the transmit or receive medium is non idle. It is deasserted by the PHY when both transmit and receive media are idle. The PHY must ensure that the CS signal remains asserted throughout the duration of a collision condition. This signal is not required to transition synchronously with respect to the Tx and Rx clocks. In Full-duplex mode the state of this signal is don't care for the MAC sublayer.

- COL: collision detection signal

  This signal must be asserted by the PHY upon detection of a collision on the medium and must remain asserted while the collision condition persists. This signal is not required to transition synchronously with respect to the Tx and Rx clocks. In Full-duplex mode, the state of this signal is don't care for the MAC sublayer.

76.6.3 Reduced media independent interface (RMII)

The reduced media independent interface (RMII) specification reduces the pin count between Ethernet PHYs and STM32 MCU (only in 10/100 mode). According to the IEEE 802.3u, an MII contains 16 pins for data and control. RMII specification reduces the pin count to 7.

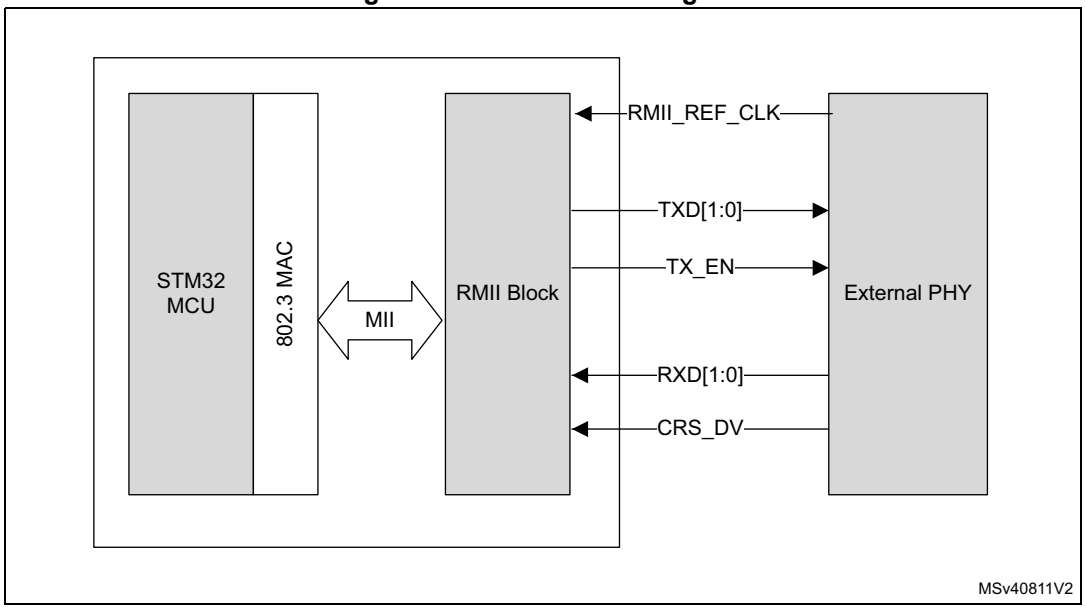
Part of the Ethernet peripheral, the RMII module is instantiated at the MAC output. This helps in translating the MII of the MAC into the RMII. The RMII block has the following characteristics:

- Supports 10 Mbit/s and 100 Mbit/s operating rates. It does not support the 1000 Mbit/s operation.
- Provides independent 2-bits wide transmit and receive paths by sourcing two clock references externally.

RMII block diagram

Figure 1052: RMII block diagram shows the position of the RMII block relative to the MAC and RMII PHY. The RMII block is placed in front of the MAC to translate the MII signals to RMII signals.

Figure 1052. RMII block diagram

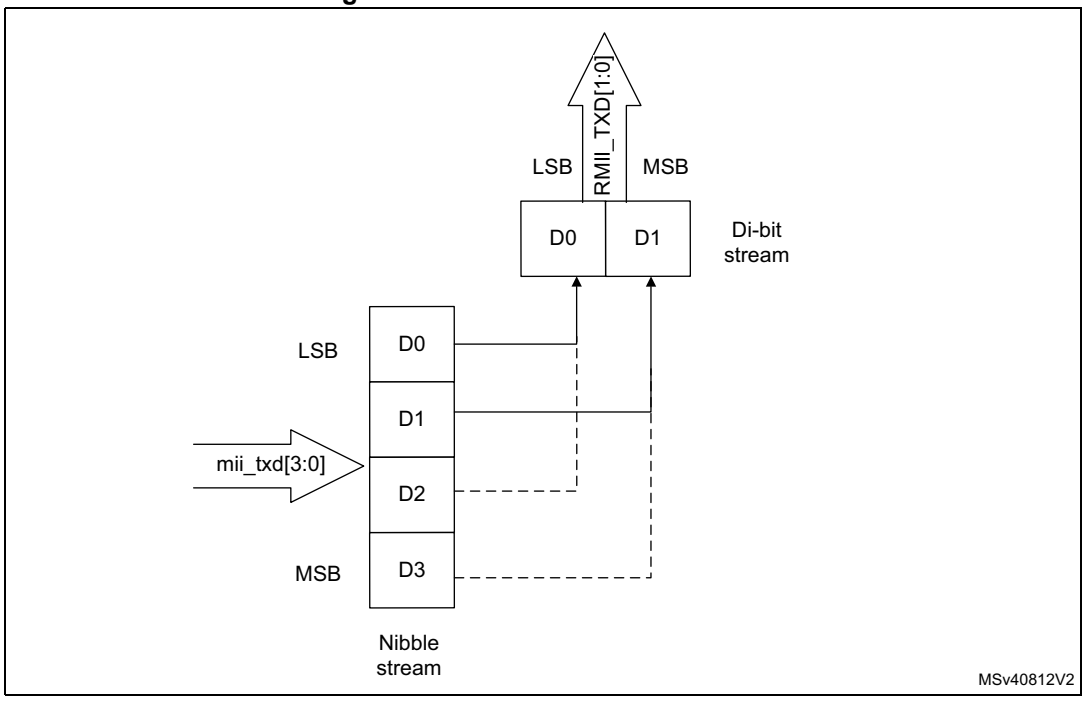


- RMII_REF_CLK: continuous 50 MHz reference clock input
- TXD[1:0]: transmit data
- TX_EN: transmit data enable.
  When high, this bit indicates that valid data are being transmitted on TXD[1:0].
- RXD[1:0]: receive data
- CRS_DV: carrier Sense (CRS) and RX_Data Valid (RX_DV) multiplexed on alternate clock cycles. In 10 Mbit/s mode, it alternates every 10 clock cycles.

Transmit bit order

Each nibble from the MII interface must be transmitted on the RMII interface di-bit at a time with the order of di-bit transmission shown in *Figure 1053: Transmission bit order*. The lower order bits (D1 and D0) are transmitted first followed by higher order bits (D2 and D3).

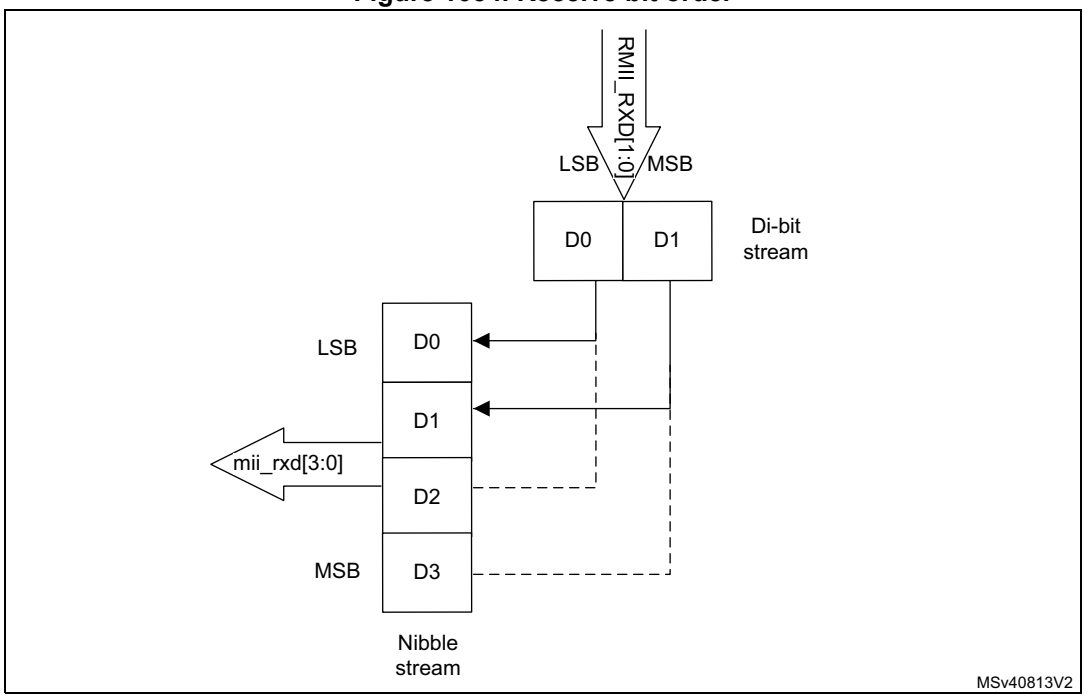
Figure 1053. Transmission bit order



Receive bit order

Each nibble is transmitted to the MII interface from the di-bit received from the RMII interface in the nibble transmission order shown in *Figure 1054: Receive bit order*. The lower order bits (D0 and D1) are received first, followed by the higher order bits (D2 and D3).

Figure 1054. Receive bit order



76.6.4 Reduced gigabit media independent interface (RGMII)

The reduced gigabit media independent interface (RGMII) specification reduces the pin count of the interconnection between the MAC and the PHY for GMII and MII interfaces. To achieve this, the data path and control signals are reduced and multiplexed together with both edges of the transmit and receive clocks. For Gigabit operation, the clocks operate at 125 MHz; for 10/100 operation, the clock rates are 2.5/25 MHz.

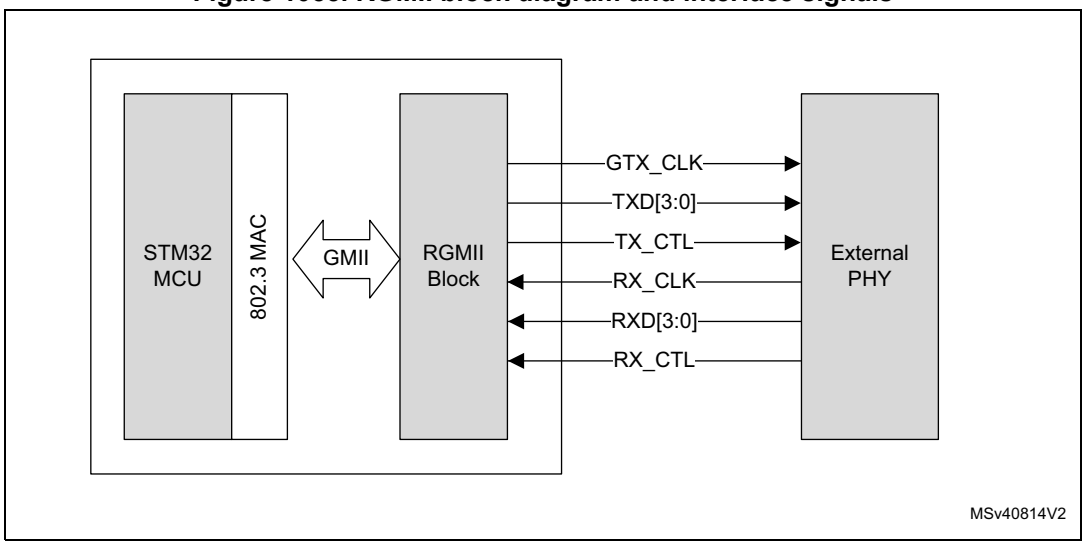
The RGMII module is instantiated between the GMII of the MAC and the PHY to translate the control and data signals between the GMII and RGMII protocols. The RGMII block has the following characteristics:

- Support of 10, 100 and 1000 Mbps operation rates.
- No extra clock required since both edges of the incoming clocks are used.
- Extraction of the in-band (link speed, link status, and duplex mode) status signals from the PHY and provides them to the MAC for link detection.

RGMII block diagram

Figure 1055: RGMII block diagram and interface signals shows the position of the RGMII block relative to the MAC and RGMII PHY. The RGMII block is placed between the GMII and the PHY to translate the GMII signals to RGMII signals.

Figure 1055. RGMII block diagram and interface signals



- **Transmit path**: this block translates all GMII transmit signals to RGMII signals. The block registers all GMII input signals at the rising edge of eth_gmii_gtx_clk and generates all RGMII transmit signals at the rising and falling edges of eth_gmii_gtx_clk.
- **Receive path:** this block translates all RGMII receive signals to GMII receive signals. The block registers all RGMII input signals at the rising and falling edges of eth_rx_clk and generates all GMII receive signals at the rising edge of eth_rx_clk.
- GTX_CLK: RGMII clock for Tx data transfers, 125 MHz, 25 MHz or 2.5 MHz
- TXD[3:0]: transmit data
- TX_CTL: multiplexing of TX_EN on the rising edge of GTX_CLK, and a logical XOR between TX_EN and TX_ER values on falling edge of GTX_CLK
- RX_CLK: RGMII clock for Rx data transfers
- RXD[3:0]: receive data
- RX_CTL: after demultiplexing, this pin provides ETH_RX_DV on the rising edge of ETH_RX_CLK, and ETH_RX_ER (logical XOR between ETH_RX_CTL values on rising edge and falling edge of ETH_RX_CLK).

Signal conversion

The RGMII block performs data conversions in both directions.

Transmit data conversion

As defined in the RGMII specification, multiplexing of data and control in Gigabit mode is accomplished by using both edges of the Transmit clock. The RGMII block accepts the 8-bit GMII transmit data from the MAC GMII (gmii_txd[7:0]) on the rising edge of eth_gmii_gtx_clk. For Gigabit operation, the RGMII block then converts gmii_txd[7:0] to two 4-bit nibbles and transmits the data on the RGMII transmit data port (rgmii_txd_o[3:0]) as follows:

- At the rising edge of eth_gmii_gtx_clk, rgmii_txd_o[3:0] contains gmii_txd[3:0]
- At the falling edge of eth_gmii_gtx_clk, rgmii_txd_o[3:0] contains gmii_txd[7:4]

For 10/100 mode, the Transmit data to the PHY is 4 bits. Therefore, for 10/100 mode, the RGMII block drives the gmii_txd[3:0] data to the PHY on rgmii_txd_o[3:0] at the rising edge of eth_gmii_gtx_clk.

Receive data conversion

In 1000-Mbps mode, the RGMII block registers the 4-bit data on the rgmii_rxd_i[3:0] signal at both edges of the Receive clock (eth_rx_clk) and transfers the resulting 8-bit data to the MAC GMII on the rising edge of eth_rx_clk clock, as follows:

- gmii_rxd[3:0] is the value received on rgmii_rxd_i[3:0] at the rising edge of eth_rx_clk
- gmii_rxd[7:4] is the value received on rgmii_rxd_i[3:0] at the falling edge of eth_rx_clk

In 10/100 mode, the RGMII block registers the 4-bit data on rgmii_rxd_i[3:0] only at the rising edge of eth_rx_clk and transfers the data to the MAC GMII on the rising edge of eth_rx_clk, as follows:

- gmii_rxd[3:0] is the value received on rgmii_rxd_i[3:0] at the rising edge of eth_rx_clk.
- gmii_rxd[7:4] is 0x0.

76.7 Ethernet low-power modes

76.7.1 Low-power management

The Ethernet peripheral supports the following techniques to save power:

- Magic packet
- Remote wake-up

The magic packet and remove wake-up techniques are used to save power in the host system when it is in low-power mode, and has to be woken up only at the reception of specific packets from the Ethernet network. In low-power mode, the clocks on most of the host logic, along with the majority of the peripherals (except the MAC receiver logic), can be disabled. On receiving the specific packets from the network, the MAC generates the trigger to wake up the host system and come back to normal state. Refer to the power control (PWR) section of the reference manual for the list of the system operating modes that support Ethernet low-power modes.

The Energy Efficient Ethernet (EEE) mode is compliant with the IEEE 802.3az-2010 standard. It is primarily targeted to save power in the Ethernet port when there is no traffic on the line. In this mode, the host indicates to the far-end that it does not have any packets to transmit in the near future and puts the transmitter port (MAC controller, PCS and PHY

layers) in low-power mode. Similarly, the receiver port can also be put in low-power mode when the far-end host indicates that it does not have any traffic to transfer. This allows significant saving of power in the Ethernet port (mainly in the PHY) with intermittent and burst traffic profile. The triggering of entry and exit out of the EEE mode is controlled by the MAC and is supported within the peripheral.

Simultaneous operation of the EEE mode along with any or both the other power saving modes is also supported.

Description of magic packet mode

This section describes how to save power through magic packet detection.

Note: The magic packet feature is based on the magic packet technology white paper from Advanced Micro Device (AMD).

The watchdog timeout limit for a magic packet is 2,048 bytes irrespective of the value programmed in WD bit in Operating mode configuration register (ETH_MACCR) and PWE bit in Watchdog timeout register (ETH_MACWTR).

In the magic-packet-based power saving mode, the reception of a valid magic packet by the MAC receiver triggers an exit from low-power mode. The MAC enters power saving mode when PWRDWN bit of *PMT control status register (ETH_MACPCSR)* is programmed to 1. Exit from the magic-packet-based power saving mode is enabled by setting the MGKPKTEN bit of *PMT control status register (ETH_MACPCSR)* to 1.

The magic packet contains a unique pattern at any offset after the Destination address, Source address, and Length/Type fields. In addition to the unique pattern matching, the MAC receiver also checks for the following, to detect the received packet as a valid magic packet:

- The packet must be addressed to it (Destination Address of the received packet should perfect match the *MAC address 0 high register (ETH_MACA0HR)* and *MAC Address 0 low register (ETH_MACA0LR)*) or with multicast/broadcast address.
- The packet must not have any length error, FCS error, dribble bit error, GMII error, and collision.
- The packet must not be runt (length including Ethernet header and FCS is at least 64 bytes).

Magic packet data format

The content of the unique pattern in magic packets is described as follows:

- Six bytes of all-ones (0xFF FF FF FF FF FF) called synchronization stream. There can be more than six bytes of 0xFF, but only the last six are considered.
- The synchronization stream is immediately followed by 16 repetitions of the Destination address field of the packet (*MAC address 0 high register (ETH_MACA0HR)* and the *MAC Address 0 low register (ETH_MACA0LR)*) or multicast/broadcast address.
- No break or interruption between synchronization stream and first repetition of Destination address field or within its 16 repetitions.

If the MAC address of a node is 0x00 11 22 33 44 55, the MAC scans for the following data sequence:

Destination Address Source Address Length/Type…… FF FF FF FF FF FF
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55
00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55 00 11 22 33 44 55
…CRC

Description of remote wake-up packet mode

This section describes the power saving mode based on remote wake-up packet.

Note: The remote wake-up packet feature implementation is based on the Device Class Power Management Reference Specification and various implementation-specific white papers.

The watchdog timeout limit for a magic packet is 2,048 bytes irrespective of the value programmed in WD bit in Operating mode configuration register (ETH_MACCR) and PWE bit in Watchdog timeout register (ETH_MACWTR).

In the remote-wake-up-magic-packet-based power saving mode, the reception of expected remote wake-up packet by the MAC receiver triggers the exit from low-power mode. The MAC enters power saving mode when PWRDWN bit in *PMT control status register (ETH_MACPCSR)* is programmed to 1. Exit from the remote-wake-up-magic-packet-based power saving mode is enabled by programming RWKPKTEN bit of *PMT control status register (ETH_MACPCSR)* to 1.

The MAC implements a filter lookup table (programmed through *Remote wake-up packet filter register (ETH_MACRWKPFR)* in which CRC, offset, and byte mask of the pattern embedded in remote wake-up packet and the filter operation commands are programmed.

The pattern embedded in the remote wake-up packet is located at any offset after the Destination address and Source address fields. In addition to the CRC match for the pattern, the MAC receiver also checks the following, to detect the received packet as a valid remote wake-up packet:

- The packet must be addressed to it (Destination Address of the received packet should perfect match the *MAC address 0 high register (ETH_MACA0HR)* and *MAC Address 0 low register (ETH_MACA0LR)*) or with multicast/broadcast address.
- The packet must not have any length error, FCS error, dribble bit error, GMII error, and collision.
- The packet must not be runt (length including Ethernet header and FCS is at least 64 bytes).

When a valid remote wake-up packet is received, the MAC receiver sets the RWKPRCVD bit in *PMT control status register (ETH_MACPCSR)* and triggers the interrupt on pmt_intr_o output port. The PMTIS bit in *Interrupt status register (ETH_MACISR)* is set when power-gating is not enabled in low-power mode. An interrupt is triggered to the application on the sbd_intr_o output port when interrupt is enabled (PMTIE bit in *Interrupt enable register (ETH_MACIER)* is set) and CSR clock is not gated off in low-power mode.

Remote wake-up packet filters

When the remote-wake-up-based power saving mode is enabled, four remote wake-up filters can be selected. The structure of the remote wake-up filters is shown in *Table 817: Remote wake-up packet filter register*.

Table 817. Remote wake-up packet filter register

ETH_MACRWKPFR value	Field							
0	Filter 0 byte mask							
1	Filter 1 byte mask							
2	Filter 2 byte mask							
3	Filter 3 byte mask							
4	Reserved	Filter 3 command	Reserved	Filter 2 command	Reserved	Filter 1 command	Reserved	Filter 0 command
5	Filter 3 offset		Filter 2 offset		Filter 1 offset		Filter 0 offset	
6	Filter 1 CRC - 16				Filter 0 CRC - 16			
7	Filter 3 CRC - 16				Filter 2 CRC - 16			

The remote wake-up filter fields are described in *Table 818: Description of the remote wake-up filter fields*.

Table 818. Description of the remote wake-up filter fields

Register	Description
Filter *i* Byte mask	The filter *i* byte mask register defines the bytes of the packet that are examined by filter *i* (0, 1, 2 or 3) to determine whether or not a packet is a wake-up packet. – The MSB (31st bit) must be zero. – Bit j[30:0] is the byte mask. – If Bit j (byte number) of the byte mask is set, the CRC block processes the Filter *i* Offset + j of the incoming packet; otherwise Filter *i* Offset + j is ignored.
Filter *i* Command	The 4-bit filter *i* command controls the filter *i* operation. – Bit 3 specifies the address type, defining the destination address type of the pattern. When the bit is set, the pattern applies to only multicast packets; when the bit is reset, the pattern applies only to unicast packet. – Bit 2 (Inverse mode), when set, reverses the logic of the CRC16 Hash function signal, to reject a packet with matching CRC_16 value. Bit 2, along with Bit 1, allows a MAC to reject a subset of remote wake-up packets by creating filter logic such as "Pattern 1 AND NOT Pattern 2". – Bit 1 (And_Previous) implements the Boolean logic(1). When set, the result of the current entry is logically ANDed with the result of the previous filter. This AND logic allows a filter pattern longer than 32 bytes by splitting the mask among two, three, or four filters. This depends on the number of filters that have the And_Previous bit set. – Bit 0 is the enable for filter *i*. If Bit 0 is not set, filter *i* is disabled.

Table 818. Description of the remote wake-up filter fields (continued)

Register	Description
Filter *i* Offset	This filter *i* offset register defines the offset (within the packet) from which the filter *i* examines the packets. – This 8-bit pattern-offset is the offset for the filter *i* first byte to be examined. – The minimum allowed offset is 12, which refers to the 13th byte of the packet. – The offset value 0 refers to the first byte of the packet.
Filter *i* CRC-16	This filter *i* CRC-16 register contains the CRC_16 value calculated from the pattern and also the byte mask programmed to the wake-up filter register block. – The 16-bit CRC calculation uses the following polynomial: $G(x) = x^{16} + x^{15} + x^2 + 1$ Each mask, used in the Hash function calculation, is compared with a 16-bit value associated with that mask. Each filter has the following: – 32-bit Mask: Each bit in this mask corresponds to one byte in the detected packet. If the bit is 1', the corresponding byte is taken into the CRC16 calculation. – 8-bit Offset Pointer: Specifies the byte to start the CRC16 computation. The pointer and the mask are used together to locate the bytes to be used in the CRC16 calculations.

1. The And_Previous bit setting is applicable within a set of four filters.
Setting And_Previous bit of a filter that is not enabled has no effect, that is setting And_Previous bit of lowest number filter in the set of four filters has no effect. For example, setting And_Previous bit of Filter 0 has no effect.
If And_Previous bit is set for a given filter to form an AND chained filter, the AND chain breaks when it finds a disabled filter. For example: If Filter 2 And_Previous bit is set (bit 1 in Filter 2 command is set) but Filter 1 is not enabled (bit 0 in Filter 1 command is reset), then only Filter 2 result is considered. If Filter 2 And_Previous bit is set (bit 1 in Filter 2 command is set), Filter 3 And_Previous bit is set (bit 1 in Filter 3 command is set), but Filter 1 is not enabled (bit 0 in Filter 1 command is reset), then only Filter 2 result ANDed with Filter 3 result is considered. If Filter 2 And_Previous bit is set (bit 1 in Filter 2 command is set), Filter 3 And_Previous bit is set (bit 1 in Filter 3 command is set), but Filter 2 is not enabled (bit 0 in Filter 2 command is reset), then since setting Filter 2 And_Previous bit has no effect, only Filter 1 result ORed with Filter 3 result is considered.
If filters chained by And_Previous bit setting have complementary programming, then a frame may never pass the AND chained filter. For example, if Filter 2 And_Previous bit is set (bit 1 in Filter 2 command is set), Filter 1 Address_Type bit is set (bit 3 in Filter 1 command is set) indicating multicast detection and Filter 2 Address_Type bit is reset (bit 3 in Filter 2 command is reset) indicating unicast detection or vice versa, then a remote wake-up frame does not pass the AND chained filter as a remote wake-up frame cannot be of both unicast and multicast address types.

The remote wake-up filter registers are implemented as eight indirect access registers (wkuppktfilter_reg#i) for four remote wake-up filters, and accessed by the application through *Remote wake-up packet filter register (ETH_MACRWKPFR)*. The entire set of wkuppktfilter_reg registers must be written to program the remote wake-up filters. The wkuppktfilter_reg register is programmed by sequentially writing the eight register values in *Remote wake-up packet filter register (ETH_MACRWKPFR)* for wkuppktfilter_reg0 to wkuppktfilter_reg3, respectively. The wkuppktfilter_reg register is read in a similar way. The MAC updates the wkuppktfilter_reg register current pointer value in RWKPTR field of *PMT control status register (ETH_MACPCSR)*.

Note: If the Remote wake-up packet filter register (ETH_MACRWKPFR) is accessed in byte or half-word mode, the internal counter to access the appropriate wkuppktfilter_reg is incremented when the CPU accesses Lane 3.

When Remote wake-up packet filter register (ETH_MACRWKPFR) is written, the content is transferred from CSR clock domain to PHY receive clock domain after the write operation. There should not be any further write to the Remote wake-up packet filter register (ETH_MACRWKPFR) until the first write is updated in PHY receive clock domain. Otherwise, the second write operation does not get updated to the PHY receive clock domain. Therefore, the delay between two write operations to the Remote wake-up packet filter register (ETH_MACRWKPFR) should be at least 4 cycles of the PHY receive clock.

PMT interrupt

The PMT interrupt signal is asserted when a valid remote wake-up packet is received.

Table 819 lists the remote wake-up scenarios in which PMT interrupt is generated.

Table 819. Remote wake-up packet and PMT interrupt generation(1)

Filter i Command			Frame Type and CRC Status		Interrupt Generation
CAST	INV	EN	Received Frame Cast Type	CRC Status	RWK INTR
0	0	1	Unicast	MATCH	Remote Wake-up packet is detected and PMT interrupt is generated
0	1	1	Unicast	MISMATCH	Remote Wake-up packet is detected and PMT interrupt is generated
1	0	1	Multicast	MATCH	Remote Wake-up packet is detected and PMT interrupt is generated
1	1	1	Multicast	MISMATCH	Remote Wake-up packet is detected and PMT interrupt is generated

1. In all other combinations, the Remote Wake-up packet is not detected and PMT interrupt is not generated.

In addition to sbd_intr_o signal, the pmt_intr_o (synchronous to Rx clock) signal is asserted. The pmt_intr_o signal, synchronous to the Rx clock domain, is provided so that the application clock can be stopped by software when the MAC is in the power-down mode. It is ORed with lpi_intr_o signal (see *Section : LPI interrupt*) and tied to the EXTI peripheral.

As the pmt_intr_o signal is generated in the PHY Rx clock domain, it is not cleared immediately when the *PMT control status register (ETH_MACPCSR)* is read. This is because the resultant clear signal has to cross to the PHY Rx clock domain, and then clear the interrupt source. This delay is at least 4 clock cycles of Rx clock and can be significant when the peripheral is operating in the 10 Mbit/s mode. When the application clears the PWRDWN bit in *Remote wake-up packet filter register (ETH_MACRWKPFR)*, the MAC comes out of the power-down mode, but this event does not generate the PMT interrupt.

Power-down sequence

The software must perform the following tasks to initiate the power-down sequence:

- Disable the Transmit DMA (if applicable) by clearing the ST bit of the *Channel x transmit control register (ETH_DMACxTXCR)*.
- Wait for any previous frame transmissions to complete. You can check this by reading TFCSTS[1:0] and TPESTS bits in *Debug register (ETH_MACDR)* and TXQSTS bit in *Tx queue x debug register (ETH_MTLTXQxDR)* of all MTL Tx Queues.
- Disable the MAC transmitter and MAC receiver by clearing TE and RE bits in *Operating mode configuration register (ETH_MACCR)*.
- Wait till the Receive DMA empties all frames from the Rx FIFO. You can check this by reading PRXQ[13:0] in *Rx queue x debug register (ETH_MTLRXQxDR)* of all Rx Queues. If these bits are zero, it indicates that the Rx FIFO is empty.
- Configure the magic packet (MGKPKTEN) and/or remote wake-up (RWKPKTEN) detection in the *PMT control status register (ETH_MACPCSR)*.
- Set bit 31 (ARPEN) in the *Operating mode configuration register (ETH_MACCR)*.
- Enable the MAC receiver by setting RE bit and then set PWRDWN bit in the *PMT control status register (ETH_MACPCSR)* to initiate the power-down sequence in MAC.

Note: If the feature is enabled and the MAC transmitter is in the LPI mode when it is put into the power-down mode, then the MII interface gets clamped to assert the LPI pattern. If the MAC transmitter is not in the LPI mode when it is put into the power-down mode, the GMII or MII interface gets clamped to all-zero.

Power-up sequence

The MAC wakes up on receiving the magic packet or remote wake-up frame. The power-up sequence is as follows:

- The MAC asserts pmt_intr_o. When only clock-gating is employed in low-power mode, the pmt_intr_o signal can be used to start the clocks that were gated-off after entering low-power mode.
- The software performs the following tasks:
  - De-assert the pmt_intr_o by reading the *PMT control status register (ETH_MACPCSR)*.
  - Perform a write operation (with reset values) to the *PMT control status register (ETH_MACPCSR)* and the *Remote wake-up packet filter register (ETH_MACRWKPFR)* so that the corresponding values in the always-on block gets synchronized. Otherwise, the values of these registers are different.
  - Perform write operations to the *Operating mode configuration register (ETH_MACCR)*, *MAC address 0 high register (ETH_MACA0HR)* and *MAC Address 0 low register (ETH_MACA0LR)* to synchronize the values in the CSR module and the respective bits in the always-on block. Otherwise, the MAC receiver is on even though the Receive Enable bit is set to 0.

After completing these steps, the software must initialize all registers, enable the transmitter, and program the DMA (in DMA configurations) to resume the normal operation.

76.7.2 Energy-efficient Ethernet (EEE)

EEE is an operational mode that enables the IEEE 802.3 Media Access Control (MAC) sublayer along with a family of physical layers to operate in Low-power idle (LPI) mode. The

EEE operational mode supports the IEEE 802.3 MAC operation at 100 Mbit/s and 1000 Mbps. The peripheral supports the IEEE 802.3az-2010 for EEE.

The LPI mode allows saving power by switching off the parts of the communication device functionality when there is no data to be transmitted and received. The systems on both sides of the link can disable some functionalities to save power during the periods of low-link utilization. The MAC controls whether the system should enter or exit the LPI mode and communicates this to the PHY.

The EEE specifies the capabilities negotiation methods that the link partners can use to determine whether EEE is supported, and then select the set of parameters that are common to both devices.

Transmit path functions

The transmit path functions include tasks that the MAC must perform to make the PHY enter the LPI state.

In the transmit path, the software must set the LPIEN bit of the *LPI control and status register (ETH_MACLCSR)* to indicate to the MAC to stop transmission and initiate the LPI protocol. The MAC completes the transmission in progress, generates its transmission status, and starts transmitting the LPI pattern instead of the IDLE pattern if the link status has been up continuously for a period specified in the LPI LS TIMER LST[9:0] bitfield of *LPI timers control register (ETH_MACLTCR)*. The PHY Link Status PLS bit of the *LPI control and status register (ETH_MACLCSR)* indicates the link status of the PHY.

Note: The EEE feature is not supported when the MAC is configured to use the RMII.

According to the standard (IEEE 802.3az-2010), the PHY must not stop the TxCLK clock during the LPI state in the MII (10 or 100) mode. However, the MAC can stop the GTX_CLK clock during the LPI state in the GMII (1000) code.

To make the PHY enter the LPI state, the MAC performs the following tasks:

1. De-asserts TX_EN.
2. Asserts TX_ER.
3. Sets TXD[3:0] to 0x1 (for 100 Mbit/s) or TXD[7:0] to 0x01 (for 1,000 Mbps).
4. Updates the status (TLPIEN bit of *LPI control and status register (ETH_MACLCSR)*) and generates an interrupt.

Note: The MAC maintains the same state of the TX_EN, TX_ER, and TXD signals for the entire duration during which the PHY remains in the LPI state.

To bring the PHY out of the LPI state, that is when the software resets the LPIEN bit, the MAC performs the following tasks:

1. Stops transmitting the LPI pattern and starts transmitting the IDLE pattern.
2. Starts the LPI TW TIMER:

   The MAC cannot start the transmission until the wake-up time specified for the PHY expires. The auto-negotiated wake-up interval is programmed in the TWT field of the *LPI timers control register (ETH_MACLTCR)*.
3. Updates the LPI exit status (TLPIEX bit of the *LPI control and status register (ETH_MACLCSR)*) and generates an interrupt.

Figure 1056 shows the behavior of TX_EN, TX_ER, and TXD[3:0] signals during the LPI mode transitions.

Note: The MAC does not stop the TX_CLK clock. The application can stop this clock (as shown in Figure 1056) if the PHY supports it and when the MAC sets the sbd_tx_clk_gating_ctrl_o signal to 1. The sbd_tx_clk_gating_ctrl_o signal is asserted after nine Tx Clock Cycles, one Pulse Synchronizer delay, and one CSR clock cycle. The assertion of the sbd_tx_clk_gating_ctrl_o signal depends on the LPITCSE bit of the LPI control and status register (ETH_MACLCSR) and can be done automatically as shown on Figure 1057.

If RGMII Interface is selected, the Tx clock is required for transmitting the LPI pattern and so the Tx Clock cannot be gated.

If the MAC is in the Tx LPI mode and the Tx clock is stopped, the application should not write to CSR registers that are synchronized to Tx clock domain.

If the MAC is in the LPI mode and the application issues a soft reset or hard reset, the MAC transmitter comes out of the LPI mode.

Figure 1056. LPI transitions (Transmit, 100 Mbds)

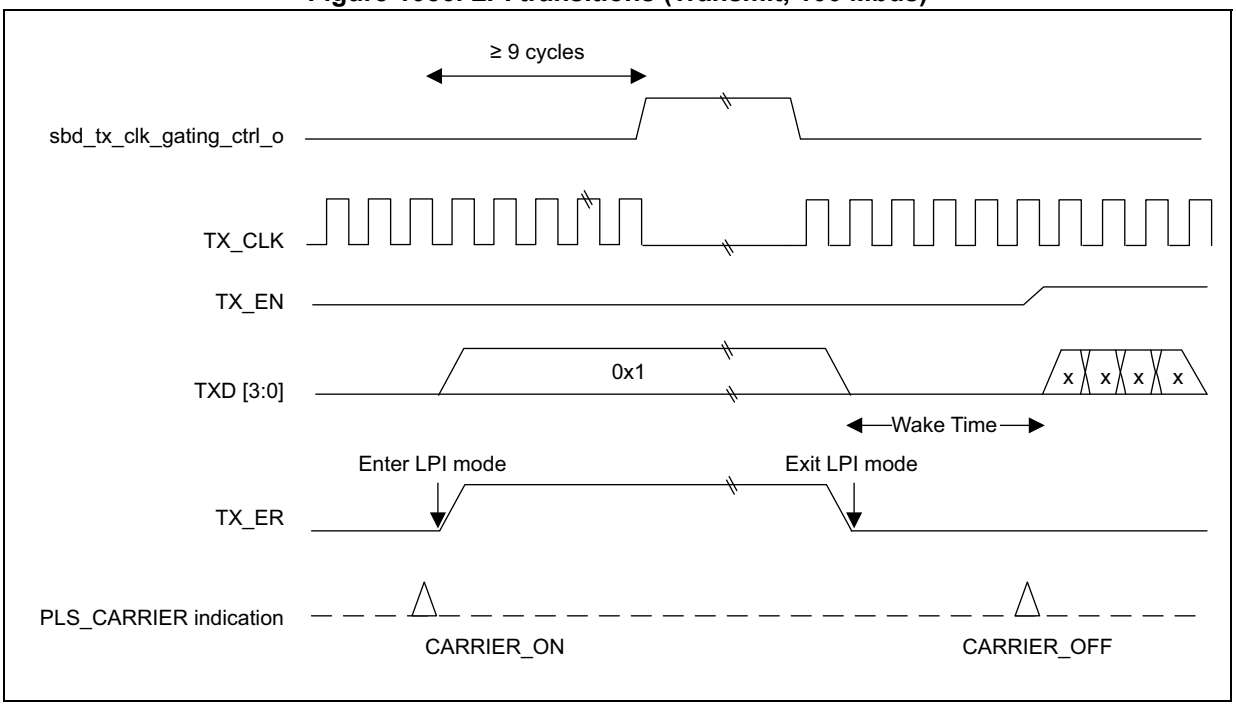
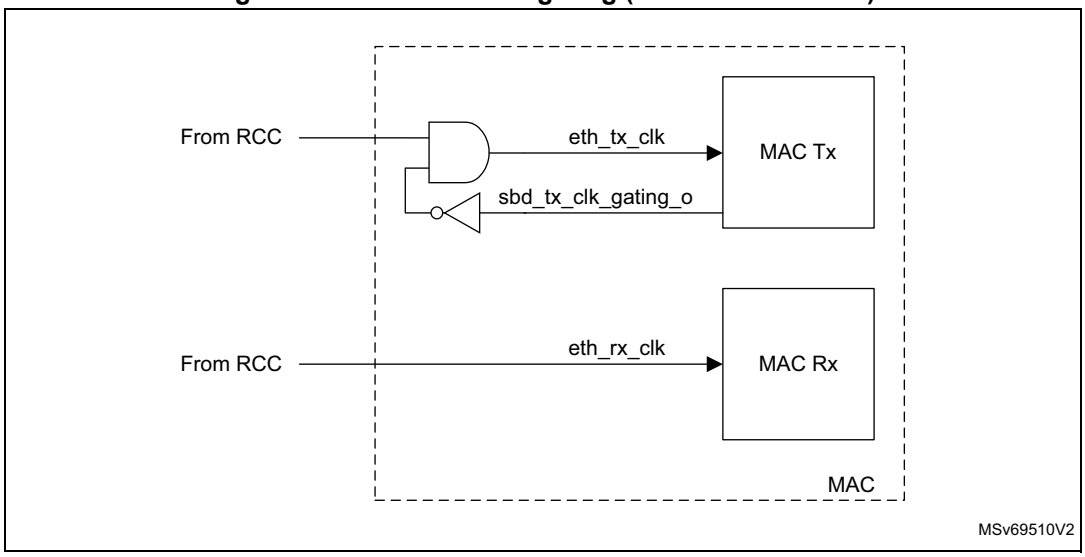


Figure 1057. LPI Tx clock gating (when LPITCSE = 1)



Automated entry/exit from LPI mode in transmit path

The MAC transmitter can be programmed to enter and exit LPI Idle mode automatically based on whether it is Idle for a specific period of time or has a packet to transfer. These modes are enabled and controlled by the *LPI control and status register (ETH_MACLCSR)*.

When LPITXA and LPIEN of *LPI control and status register (ETH_MACLCSR)* are set, the MAC transmitter enters LPI Idle state when the MAC transmit path (including the MTL layers and DMA layers) are idle. The MAC transmitter exits the LPI Idle state and clears the LPITXEN bit as soon as any of functions in the TX path (DMA, MTL or MAC) becomes non-idle due to initiation of a packet transfer.

In addition, when LPITE is also set, the MAC transmitter enters LPI Idle state only if the Transmit path remains in idle state (no activity) for the time period indicated by the value in *LPI entry timer register (ETH_MACLETR)*. In this mode also, the MAC transmitter exits the LPI Idle state as soon as any of the functions becomes non-idle. However, the LPIEN bit is not cleared but remains active so that reentry to LPI Idle state is possible without any software intervention when the MAC becomes idle again.

When both LPITE and LPITXA bits are cleared, the application can directly control the entry and exit of LPI Idle state by programming the LPIEN bit.

Receive path functions

The receive path functions include the tasks that the PHY and MAC must perform when the PHY receives signals from the link partner to exit the LPI state.

In the receive path, when the PHY receives the signals from the link partner to enter into the LPI state, the PHY and MAC perform the following tasks:

1. The PHY asserts RX_ER.
2. The PHY sets RXD[3:0] to 0x1 (for 100 Mbit/s) or RXD[7:0] to 0x01 (for 1,000 Mbps).
3. The PHY de-asserts RX_DV.
4. The MAC updates the RLPIEN bit of the *LPI control and status register (ETH_MACLCSR)* and immediately generates an interrupt.

Note: The PHY maintains the same state of the RX_ER, RXD, and RX_DV signals for the entire duration during which it remains in the LPI state.

If the LPI pattern is detected for a very short duration (that is, less than two cycles of Rx clock), the MAC does not enter the Rx LPI mode.

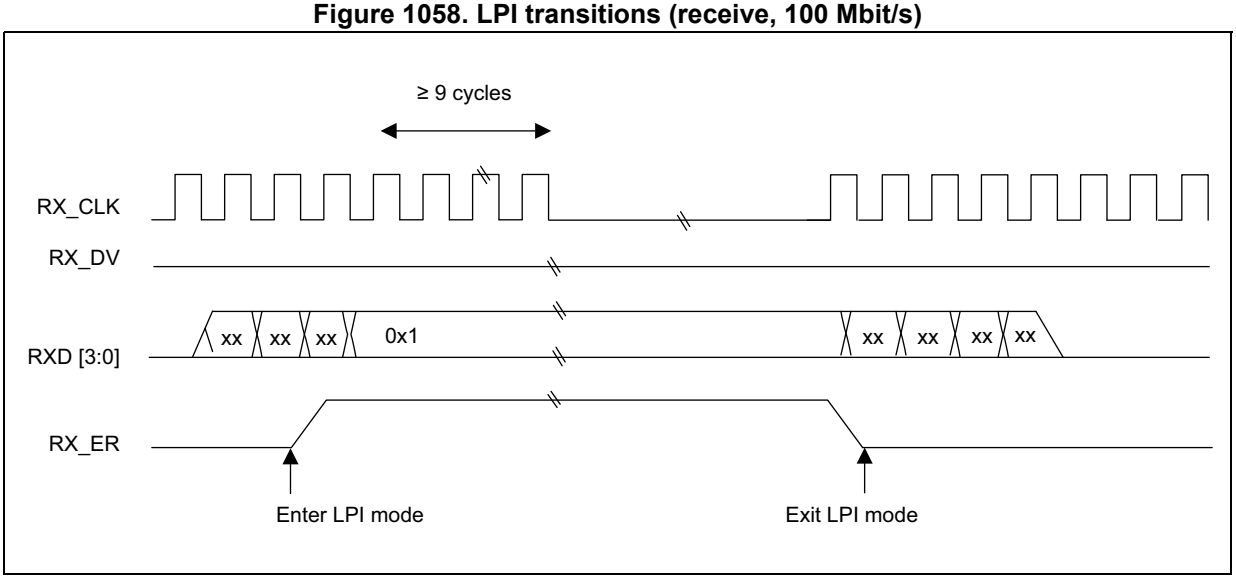
If the duration between the end of the current Rx LPI pattern and the start of the next Rx LPI pattern is very short (that is, less than two Rx clock cycles), then the MAC exits and enters again the Rx LPI mode. The MAC does not trigger the Rx LPI Exit and Entry interrupts.

When the PHY receives signals from the link partner to exit the LPI state, the PHY and MAC perform the following tasks:

1. The PHY de-asserts RX_ER and returns to a normal inter-packet state.
2. The MAC updates the RLPIEX bit of the *LPI control and status register (ETH_MACLCSR)* and generates an interrupt immediately. The sideband signal lpi_intr_o (synchronous to Rx clock) is also asserted.

Figure 1058 shows the behavior of RX_ER, RX_DV, and RXD[3:0] signals during the LPI mode transitions.

Figure 1058. LPI transitions (receive, 100 Mbit/s)



Note: If the RX_CLK_stoppable bit (in the PHY register written through MDIO) is asserted when the PHY is indicating LPI to the MAC, the PHY may halt the RX_CLK at any time more than nine clock cycles after the start of the LPI state as shown in Figure 1058.

If the MAC is in LPI mode and the application issues a soft reset or hard reset, the MAC receiver exits from LPI mode during reset. If the LPI pattern is still received after the reset is de-asserted, the MAC receiver enters again the LPI state.

If the RX clock is stopped in the RX LPI mode, the application should not write to the CSR registers that are being synchronized to the RX clock domain.

When the PHY sends the LPI pattern, if EEE feature is enabled, the MAC automatically enters the LPI state. There is no software control to prevent the MAC from entering the LPI state.

LPI timers

The transmitter maintains the LPI LS TIMER, LPI TW TIMER, and LPI AUTO ENTRY TIMER timers.

The following LPI timers are loaded with the respective values from the *LPI timers control register (ETH_MACLTCR)* and *LPI entry timer register (ETH_MACLETR)*:

- LPI LS TIMER

  The LPI LS TIMER counts, in milliseconds, the time expired since the link status is up. You can enable the monitoring of the LUD bit of the *PHYIF control status register (ETH_MACPHYCSR)* by setting the PLSEN bit of *LPI control and status register (ETH_MACLCSR)*.

  The link status is indicated to the MAC by the LNKSTS bit of the *PHYIF control status register (ETH_MACPHYCSR)* or the value programmed by the software in the PLS bit of the *LPI control and status register (ETH_MACLCSR)*. If the link status is not available in the *PHYIF control status register (ETH_MACPHYCSR)*, the software should get the PHY link status by reading the PHY register and accordingly update the PLS bit.

  This timer is cleared every time the link goes down. It starts to increment when the link is up again and continues to increment until the value of the timer becomes equal to the terminal count. Once the terminal count is reached, the timer remains at the same value as long as the link is up. The terminal count is the value programmed in the LST[9:0] bitfield in the *LPI timers control register (ETH_MACLTCR)*. The GMII interface does not assert the LPI pattern unless the terminal count is reached. This ensures a minimum time for which no LPI pattern is asserted after a link is established with the remote station. This period is defined as 1 second in the IEEE 802.3-az-2010. The LPI LS TIMER is 10-bit wide. The software can program up to 1023 milliseconds.

- LPI TW TIMER

  The LPI TW TIMER counts, in microseconds, the time expired since the de-assertion of LPI. The terminal count should be programmed in Bit[15:0] of *LPI timers control register (ETH_MACLTCR)*. The terminal count of the timer is the value of resolved Transmit TW that is the auto-negotiated time after which the MAC can resume the normal transmit operation. After exiting the LPI mode, the MAC resumes its normal operation after the TW timer reaches the terminal count.

  The MAC supports the LPI TW TIMER in units of microsecond. The LPI TW TIMER is 16-bit wide. Therefore, the software can program up to 65535 micro seconds.

- LPI AUTO ENTRY TIMER

  This timer counts in steps of eight microseconds, the time for which the MAC transmit path has to remain in idle state (no activity), before the MAC transmitter enters the LPI IDLE state and starts transmitting the LPI pattern. This timer is enabled when LPITE bit in *LPI control and status register (ETH_MACLCSR)* is set.

Note: Program the PLS bit of LPI control and status register (ETH_MACLCSR) to 1'b0 before switching between the GMII and MII modes. This resets the internal timers. If the mode is changed after the LPI LS TIMER or LPI TW TIMER starts, the change in the Tx clock frequency can result in incorrect timeout.

LPI interrupt

The MAC generates the LPI interrupt when the Tx or Rx side enters or exits the LPI state. The interrupt sbd_intr_o is asserted when the LPI interrupt status is set. The LPI interrupt can be cleared by reading the *LPI control and status register (ETH_MACLCSR)*.

When the MAC exits the Rx LPI state, then in addition to the sbd_intr_o, the sideband signal lpi_intr_o (synchronous to Rx clock) is asserted. The lpi_intr_o signal can be used to trigger the external clock-gating circuitry to restore the application clock to the MAC. The lpi_intr_o signal, synchronous to the Rx clock domain, is provided so that the application clock can be stopped by software when the MAC is in the LPI state. It is ORed with pmt_intr_o signal (see *Section : PMT interrupt*) and tied to the EXTI peripheral.

The lpi_intr_o signal is generated in the Rx clock domain. It may not be cleared immediately after the *LPI control and status register (ETH_MACLCSR)* is read. This is because the clear signal, generated in CSR clock domain, has to cross the Rx clock domain, and then clear the interrupt source. This delay is at least four clock cycles of Rx clock and can be significant when the peripheral is operating in the 10 Mbit/s mode.

Programming guidelines for Energy Efficient Ethernet

For detailed guidelines on the programming guidelines, see *Section 76.9.13: Programming guidelines for Energy Efficient Ethernet (EEE) on page 4139*.

76.8 Ethernet interrupts

The Ethernet peripheral generates a single interrupt signal (eth_sbd_intr_it). This signal can be raised as a result of various events. These events are captured in status registers and interrupt enables are provided for each source of interrupt such that the interrupt signal is asserted for an event only when the corresponding interrupt enable is set.

The interrupt status and corresponding enable registers are organized in a hierarchical manner so that it is easier for software to traverse and identify the source of interrupt event quickly. When interrupt is asserted, the *Interrupt status register (ETH_DMAISR)* register is first level that indicates the major blocks for the interrupt event source. This register is read-only, and it contains bits corresponding to each DMA channel (TX & RX pair), the MTL, and the MAC. The software application must then read one (or more) of the following registers corresponding to the bits that are set:

- ETH_DMAC0SR: channel 0 status (see *Channel x status register (ETH_DMACxSR)*))
- ETH_DMAC1SR: channel 1 status (see *Channel x status register (ETH_DMACxSR)*)
- ETH_MTLISR: interrupt status (see *Interrupt status register (ETH_MTLISR)*)
- ETH_MACISR: interrupt status (see *Interrupt status register (ETH_MACISR)*)

76.8.1 DMA interrupts

Interrupt registers description

The ETH_DMAC0SR: Channel 0 Status register (see *Channel x status register (ETH_DMACxSR)*) captures all the interrupt events of that TxDMA and RxDMA channel. The ETH_DMAC0IER: Channel 0 Interrupt Enable register (see *Channel x interrupt enable register (ETH_DMACxIER)*) contains the corresponding enable bits for each of the interrupt event.

There are two groups of interrupts in the DMA channel namely Normal and Abnormal interrupts.They are indicated by Bits[15:14] of ETH_DMAC0SR register respectively. The normal group is for events that happen during the normal transfer of packets (TI: transmit interrupt, RI: receive interrupt, TBU: Transmit buffer unavailable) while the abnormal interrupt events are for error events.

Interrupts are not queued. If the same interrupt event occurs again before the driver responds to the previous one, no additional interrupts are generated. An interrupt is generated only once for multiple events. The driver must scan the *Interrupt status register (ETH_DMAISR)* for the cause of the interrupt and clear the source in the respective Status register. The interrupt is cleared only when all the bits of *Interrupt status register (ETH_DMAISR)* are cleared.

Periodic scheduling of transmit and receive interrupt

It is not preferable to generate interrupts for every packet transferred by DMA (RI and TI) for system throughput performance reasons. The Ethernet peripheral gives the flexibility to schedule the interrupt at regular intervals using two methods:

1. Set Interrupt on Completion bit in Transmit descriptor (TDES2[31] in *Table 823: TDES2 normal descriptor (read format)*) once for every “required” number of packets to be transmitted.
2. Similarly, set the IOC (RDES3[30] in*Table 836: RDES3 normal descriptor (read format)*) bit only at some specific intervals of Receive descriptors. This way, whenever

a received packet transfer to system memory is complete and any of the descriptors used for that packet transfer has the IOC bit set, only then the RI event is generated.

In addition to above, an interrupt timer (ETH_DMACxRXIWTR: Channel i Rx Interrupt Watchdog Timer) is given for flexible control and periodic scheduling of Receive interrupt. When this interrupt timer is programmed with a nonzero value, it gets activated as soon as the Rx DMA completes a transfer of a received packet to system memory without asserting the Receive interrupt because the corresponding interrupt of completion IOC bit (RDES3[30] in *Table 836: RDES3 normal descriptor (read format)*) is not set. When this timer runs out as per the programmed value, RI bit is set and the interrupt is asserted if the corresponding RIE is enabled in ETH_DMAC0IER register (see *Channel x interrupt enable register (ETH_DMACxIER)*). The timer is stopped and cleared before it expires, if the RI is set for a packet transfer whose descriptor's IOC was set. The timer is reactivated automatically after the next packet transfer is complete without the RI event being generated.

Per channel transfer complete interrupt

The Transmit Transfer complete interrupt (TI) and Receive Transfer complete interrupt (RI) is reflected in the Channel Status register (*Channel x status register (ETH_DMACxSR)*). The TI bit is set whenever the Tx DMA channel closes the descriptor in which the IOC bit is set (Interrupt On Completion - TDES2[31]). Similarly, the RI bit is set whenever the Rx DMA channel closes the descriptor with the LD bit set and, in any of the descriptors used for transferring that packet, IOC bit is set (Interrupt Enable on completion - RDES3[30])).

The interrupt signal is asserted for the Transfer complete interrupts only when the corresponding interrupts are enabled in the channel interrupt enable register (*Channel x interrupt enable register (ETH_DMACxIER)*).

The behavior of the RI/TI interrupts changes depending on the settings of INTM field (bits[17:16]) in the ETH_DMAMR register (*DMA mode register (ETH_DMAMR)*). *Table 820* explains the behavior of the Transfer Complete interrupt.

Table 820. Transfer complete interrupt behavior

Interrupt mode	Behavior of TI/RI and eth_sbd_intr_it
INTM=0	The TI/RI status signals are set whenever the Transfer complete event is detected. The bits get cleared whenever the software driver writes 1 to these bits. The eth_sbd_intr_it is asserted whenever the corresponding interrupts are also enabled in ETH_DMACxIER register.
INTM=1	The TI/RI is set as explained above. However, the eth_sbd_intr_it is not asserted for any RI/TI events.
INTM=2	The RI/TI status bits are set whenever the Transfer Complete event is detected and gets reset whenever software driver clears those bits by writing 1. However, if another Transfer Complete event is detected before it is cleared (serviced) by the software, these status bits is automatically set again. However, the eth_sbd_intr_it is not generated based on TI/RI.

76.8.2 MTL interrupts

MTL interrupt events are combined with the events in the DMA to generate the interrupt signal.

The register *Interrupt status register (ETH_MTLISR)* report the queue number responsible for the event. ETH_MTLQ0ICSR: Queue 0 Interrupt Control Status or ETH_MTLQ1ICSR: Queue 1 Interrupt Control Status must be read for event description.

The MTL interrupts are enabled by default. Each event can be prevented from asserting the interrupt by setting the corresponding mask bits in the *Interrupt status register (ETH_MTLISR)* register.

MTL interrupt signal is driven by one of these events on Queue 0 and Queue 1:

- Receive Queue Overflow Interrupt
- Transmit Queue Underflow

76.8.3 MAC interrupts

MAC interrupt events are combined with the events in the DMA to generate the interrupt signal.

The MAC interrupts are of level type, that is, the interrupt remains asserted (high) until it is cleared by the application or software.

The *Interrupt status register (ETH_MACISR)* describes the events that can cause an interrupt from the MAC. The MAC interrupts are enabled by default. Each event can be prevented from asserting the interrupt by setting the corresponding mask bits in the *Interrupt status register (ETH_MACISR)*.

The interrupt register bits only indicate the block from which the event is reported. You must read the corresponding status registers and other registers to clear the interrupt.

MAC interrupt signal is driven by one of these events:

- Receive status interrupt
- Transmit status interrupt
- Timestamp interrupt status
- MMC interrupt status
  - MMC receive checksum offload interrupt status
  - MMC transmit interrupt status
  - MMC Receive interrupt status
- LPI interrupt status
- PMT interrupt status
- PHY interrupt
- RGMII interrupt status

Note: Two sidebands signals are generated together with LPI and PMT interrupts: lpi_intr_o and pmt_intr_o. They are used for wake-up event detection at EXTI level.

76.9 Ethernet programming model

This chapter provides the instructions for initializing the DMA or MAC registers in the proper sequence. It contains the following sections:

- DMA initialization (see *Section 76.9.1*)
- MTL initialization (see *Section 76.9.2*)
- MAC initialization (see *Section 76.9.3*)
- Performing normal receive and transmit operation (see *Section 76.9.4*)
- Stopping and starting transmission (see *Section 76.9.5*)
- Programming guidelines for multichannel multiqueue operation (see *Section 76.9.8*)
- Programming guidelines for GMII link state transitions (see *Section 76.9.9*)
- Programming guidelines for IEEE 1588 timestamping (see *Section 76.9.10*)
- Programming guidelines for AV feature (see *Section 76.9.12*)
- Programming guidelines for energy-efficient Ethernet (see *Section 76.9.13*)
- Programming guidelines for flexible pulse-per-second (PPS) output (see *Section 76.9.14*)
- Programming guidelines for IEEE 1588 auxiliary snapshot (see *Section 76.9.15*)
- Programming guidelines for TSO (see *Section 76.9.16*)
- Programming guidelines for VLAN filtering on the receiver (see *Section 76.9.17*)
- Programming guidelines for extended VLAN filtering and routing on reception (see *Section 76.9.18*)
- Programming guidelines for queue/channel-based VLAN inclusion register (see *Section 76.9.19*)
- Programming guidelines for EST (see *Section 76.9.20*)
- Programming the launch time in time-based scheduling (see *Section 76.9.21*)
- Enabling the frame preemption function (see *Section 76.9.22*)

76.9.1 DMA initialization

Complete the following steps to initialize the DMA:

1. Provide a software reset to reset all MAC internal registers and logic (bit 0 of *DMA mode register (ETH_DMAMR)*).
2. Wait for the completion of the reset process (poll bit 0 of the *DMA mode register (ETH_DMAMR)*, which is cleared when the reset operation is completed).
3. Program the following fields to initialize the *System bus mode register (ETH_DMASBMR)*:
   a) AAL, ONEKBBE, RD_OSR_LMT, and WR_OSR_LMT
   b) Fixed burst or undefined burst
   c) Outstanding requests limit value (OSR_LMT).
   d) If fixed length value is enabled, select the maximum burst length possible on the AXI Bus (bits [7:1])
4. Create a transmit and a receive descriptor list. In addition, ensure that the receive descriptors are owned by the DMA (set bit 31 of TDES3/RDES3 descriptor). For more information on descriptors, refer to *Section 76.10: Descriptors*.

Note: Descriptor address from start to end of the ring should not cross the 4GB boundary.

5. Program ETH_DMAC0TXRLR and ETH_DMAC0RXRLR registers (see *Channel x Tx descriptor ring length register (ETH_DMACxTXRLR)* and *Channel x Rx descriptor ring length register (ETH_DMACxRXRLR)*). The programmed ring length must be at least 4.
6. Initialize receive and transmit descriptor list address with the base address of transmit and receive descriptor (*Channel x Tx descriptor list address register (ETH_DMACxTXDLAR)*, *Channel x Rx descriptor list address register (ETH_DMACxRXDLAR)*). In addition, program the transmit and receive tail pointer registers that inform the DMA about the available descriptors (see *Channel x Tx descriptor tail pointer register (ETH_DMACxTXDTPR)* and *Channel x Rx descriptor tail pointer register (ETH_DMACxRXDTPR)*).
7. Program ETH_DMAC0CR, ETH_DMAC0TXCR and ETH_DMAC0RXCR registers (see *Channel x control register (ETH_DMACxCR)*, *Channel x transmit control register (ETH_DMACxTXCR)* and *Channel x receive control register (ETH_DMACxRXCR)*) to configure the parameters such as the maximum burst-length (PBL) initiated by the DMA, descriptor skip lengths, OSP for TxDMA, RBSZ[13:0] for RxDMA, and so on.
8. Enable the interrupts by programming the ETH_DMAC0IER register (see *Channel x interrupt enable register (ETH_DMACxIER)*).
9. Start the Receive and Transmit DMAs by setting SR (bit 0) of *Channel x receive control register (ETH_DMACxRXCR)* and ST (bit 0) of the ETH_DMAC0TXCR (see *Channel x transmit control register (ETH_DMACxTXCR)*).
10. Repeat steps 4 to 9 for all the Tx DMA channels selected in the hardware.

76.9.2 MTL initialization

Complete the following steps to initialize the MTL registers:

1. Program the Tx Scheduling (SCHALG) and Receive Arbitration Algorithm (RAA) fields in *Operating mode register (ETH_MTLOMR)* to initialize the MTL operation when multiple Tx and Rx queues are used.
2. Program the Receive Queue to DMA mapping in *Rx queue and DMA channel mapping register (ETH_MTLRXQDMAMR)*.
3. Program the following fields to initialize the operating mode in *Tx queue x operating mode register (ETH_MTLTXQxOMR)*.
   a) Transmit Store And Forward (TSF) or Transmit Threshold Control (TTC) if the Threshold mode is used.
   b) Transmit Queue Enable (TXQEN) to value 2‘b10 to enable Transmit Queue 0.
   c) Transmit Queue Size (TQS).
4. Program the following fields to initialize the operating mode in the ETH_MTLRXQxOMR register (see *Rx queue x operating mode register (ETH_MTLRXQxOMR)*):
   a) Receive Store and Forward (RSF) or RTC if Threshold mode is used.
   b) Flow Control Activation and De-activation thresholds for MTL Receive FIFO (RFA and RFD).
   c) Error Packet and undersized good Packet forwarding enable (FEP and FUP).
   d) Receive Queue Size (RQS).
5. Repeat the previous two steps for all the MTL Tx and Rx queues selected in the configuration

76.9.3 MAC initialization

The MAC configuration registers establish the operating mode of the MAC. If possible, these registers must be initialized before initializing the DMA. The following MAC Initialization operations can also be performed after DMA initialization. If the MAC initialization is complete before the DMA is configured, enable the MAC receiver (last step in the following sequence) only after the DMA is active. Otherwise, received frames fill the Rx FIFO and overflow.

1. Provide the MAC address registers: *MAC address x low register (ETH_MACAxLR)*, *MAC address 0 high register (ETH_MACA0HR)* and *MAC address x high register (ETH_MACAxHR)*.
2. Program the following fields to set the appropriate filters for the incoming frames in the *Packet filtering control register (ETH_MACPFR)*:
   a) Receive All.
   b) Promiscuous mode.
   c) Hash or Perfect Filter.
   d) Unicast, multicast, broadcast, and control frames filter settings.
3. Program the following fields for proper flow control in the *Tx Queue 0 flow control register (ETH_MACQ0TXFCR)*:
   a) Pause time and other Pause frame control bits.
   b) Transmit Flow control bits.
   c) Flow Control Busy.
4. Program the *Interrupt enable register (ETH_MACIER)* as required, if it is applicable for your configuration.
5. Program the appropriate fields in the *Operating mode configuration register (ETH_MACCR)* register. For example, Inter-packet gap while transmission and jabber disable.
6. Set bit 0 and 1 in *Operating mode configuration register (ETH_MACCR)* register to start the MAC transmitter and receiver.

To support Jumbo Transmit/Receive packets, follow these steps:

- In the *Operating mode configuration register (ETH_MACCR)*
  a) Set JE bit to 1.
  b) Set JD and WD bits to 0 to avoid giant packet error reporting.
  c) Set GPSLCE bit to 1
  d) Set GPSL bitfield of the *Extended operating mode configuration register (ETH_MACECR)* to a value > 9026

To support Transmit/Receive packets, up to 16K, follow these steps:

- In the *Operating mode configuration register (ETH_MACCR)*
  a) Set JD and WD bits to 1 to avoid giant packet error reporting.
  b) Set GPSLCE bit to 1.
  c) Set GPSL bitfield of the *Extended operating mode configuration register (ETH_MACECR)* to 16383.

76.9.4 Performing normal receive and transmit operation

For normal operation, complete the following steps:

1. For normal transmit and receive interrupts, read the interrupt status. Then, poll the descriptor by reading the status of the descriptor owned by the Host (either transmit or receive).
2. Set the descriptors to appropriate values. Make sure that transmit and receive descriptors are owned by the DMA to resume the transmission and reception of data.
3. If the descriptors are not owned by the DMA (or no descriptor is available), the DMA goes into Suspend state. The transmission or reception can be resumed by freeing the descriptors and writing the ETH_DMAC0TXDTPR (see *Channel x Tx descriptor tail pointer register (ETH_DMACxTXDTPR)* and ETH_DMAC0RXDTPR (see *Channel x Rx descriptor tail pointer register (ETH_DMACxRXDTPR)*).
4. In debug mode, the values of the current host transmitter or receiver descriptor address pointer can be read in ETH_DMAC0CATXDR and ETH_DMAC0CARXDR registers (see *Channel x current application transmit descriptor register (ETH_DMACxCATXDR)* and *Channel x current application receive descriptor register (ETH_DMACxCARXDR)*).
5. In debug mode, the values of the current host transmit buffer address pointer and receive buffer address pointer can be read in ETH_DMAC0CATXDR and ETH_DMAC0CARXDR registers (see *Channel x current application transmit descriptor register (ETH_DMACxCATXDR)* and *Channel x current application receive descriptor register (ETH_DMACxCARXDR)*).

76.9.5 Stopping and starting transmission

Complete the following steps to pause the transmission for some time:

1. Disable the Transmit DMA (if applicable) by clearing Bit 0 (ST) of ETH_DMAC0TXCR register (see *Channel x transmit control register (ETH_DMACxTXCR)*).
2. Wait for any previous frame transmissions to complete. You can check this by reading the appropriate bits of *Tx queue x debug register (ETH_MTLTXQxDR)* (TRCSTS[1:0] is not 01 and TXQSTS = 0).
3. Disable the MAC transmitter and MAC receiver by clearing RE and TE bits of the *Operating mode configuration register (ETH_MACCR)* Register.
4. Disable the Receive DMA (if applicable), after making sure that the data in the Rx FIFO is transferred to the system memory (by reading the appropriate bits of *Tx queue x debug register (ETH_MTLTXQxDR)*, PRXQ=0 and RXQSTS[1:0] = 00).
5. Make sure that both Tx queue and Rx queue are empty (TXQSTS is 0 in *Tx queue x debug register (ETH_MTLTXQxDR)* and RXQSTS[1:0] is set to 00.
6. To restart the operation, first start the DMAs, and then enable the MAC transmitter and receiver.

Note: Do not change the configuration (such as duplex mode, speed, port, or loopback) when the MAC is actively transmitting or receiving. These parameters are changed by software only when the MAC transmitter and receiver are not active.

Similarly, do not change the DMA-related configuration when transmit and receive DMA are active.

76.9.6 Programming guidelines for switching to new descriptor list in RxDMA

Switching to a new descriptor list is different in the Rx DMA compared to the Tx DMA. Switching to a new descriptor list is permitted when the RxDMA is in Suspend state, as explained below:

- Generally, RxDMA prepares the descriptors in advance.
- If the RxDMA goes to Suspend state due to descriptors not being available, a major failure occurs (software is not able to free the filled-up descriptors/buffers). If this issue is not rectified immediately, frames are lost because of an RxFIFO overflow. Therefore, the software is allowed to create a new descriptor list and program the RxDMA to start using it immediately, without going into Stop state.

76.9.7 Programming guidelines for switching the AXI clock frequency

To dynamically change the AXI clock frequency (without applying soft reset or hard reset), follow these steps:

1. Disable the Transmit DMA (if applicable) and wait for any previous frame transmissions to complete. When the frame transmissions is complete, the Tx FIFO becomes empty and the Tx DMA enters Stop state. The Tx FIFO status is given in the *Tx queue x debug register (ETH_MTLTXQxDR)* and the status of DMA is given in *Debug status register (ETH_DMADSR)*.
2. Disable the MAC transmitter and the MAC receiver by clearing the appropriate bits in *Operating mode configuration register (ETH_MACCR)*.
3. Disable the Receive DMA (if applicable) after making sure that the data in the Rx FIFO is transferred to the system memory. The Rx FIFO empty status is given in *Rx queue x debug register (ETH_MTLRXQxDR)*.
4. Ensure that the application does not perform any register read or write operation.
5. Change the frequency of the AXI clock.
6. Enable the MAC transmitter or the MAC receiver and the Transmit or Receive DMA. These steps ensure that no valid data is present in the Tx FIFO or Rx FIFO at the time of clock frequency switching and prevent any data corruption.

76.9.8 Programming guidelines for multichannel multiqueue operation

Transmit path

1. Program the Transmit queue size in the TQS field in ETH_MTLTXQ**x**OMR register. The size of the queue is determined by the value programmed in the TQS field.

   During the Transmit operation, the number of channels is equal to the number of the queues. As a result, the channel to queue mapping is fixed.
2. To use a queue, the queue must be enabled by setting the TXQEN bit in the corresponding ETH_MTLTXQ**x**OMR register. The ST bit in *Channel x transmit control register (ETH_DMACxTXCR)* and the corresponding TXQEN bit in ETH_MTLTXQ**x**OMR register must be enabled.
3. Program the scheduling method in SCHALG bit of ETH_MTLOMR register.
4. If CBS algorithm is enabled on AV queue 1, ETH_MTLTXQ1ECR, ETH_MTLTXQ1SSCR**,** ETH_MTLTXQ1HCR and ETH_MTLTXQ1LCR registers also need to be programmed as required.

Receive path

1. Program the Receive queue size in the RQS field in ETH_MTLRXQ**x**OMR register. The size of the queue is determined by the value programmed in the RQS field.
2. Enable the Receive queues 0 to 1 in the RXQ0EN to RXQ1EN fields of ETH_MACRXQ0CR register for AV.

   Enable SR bit of statically or dynamically mapped *Channel x receive control register (ETH_DMACxRXCR)* and corresponding RXQiEN in *Rx queue control 0 register (ETH_MACRXQC0R)*.
3. The MAC routes the Rx packets to the Rx queues depending on following packet types:
   a) AV PTP packets are routed according to the value of PTPQ in ETH_MACRXQC1R register
   b) AV untagged control packets are routed according to the value of AVCPQ in ETH_MACRXQC1R register.
   c) VLAN tag priority field in VLAN tagged packets

      Program PSRQ1 and PSRQ0 of the ETH_MACRXQC2R register to configure the routing of tagged packets based on the USP (user priority) field of the received packets to the Rx queues 0 to 1.
   d) The AV tagged control and data packets are also routed according to PSRQ field of the ETH_MACRXQC2R register.

Note: The priorities set in PSRQ0 and PSRQ1 must be unique.

4. If multiple RX DMA channels are enabled, the following programming should be done for proper arbitration and mapping:
   a) Program the RAA field of *Operating mode register (ETH_MTLOMR)* to select the arbitration algorithm to decide which Rx queue is read out from the RxFIFO memory.
   b) Program the *Rx queue x control register (ETH_MTLRXQxCR)* to decide the weights and the packet arbitration for each Rx queue.
   c) If static mapping is programmed in Rx queue and *Rx queue and DMA channel mapping register (ETH_MTLRXQDMAMR)* (QiDDMACH is reset to 0), bits QiMDMACH need to be programmed to select the channel for which each queue is mapped.
   d) Set RXQiDDMACH bit in Rx queue and *Rx queue and DMA channel mapping register (ETH_MTLRXQDMAMR)* to select dynamic mapping of packets in each Rx queue.
   e) In dynamic channel mapping, the routing of a packet to a specific RxDMA channel is decided by the value of DCS field in the lowest MAC address register.

Programming guidelines for recovering from DMA channel failure

When the DMA channel issues a bus error, follow the steps described below to recover from the failure.

Recovering from the receive DMA channel failure

Follow these steps if a bus error occurred in the Receive DMA channel:

1. Set the RPF bit to 1. This flushes all the packets one after the other. This step is optional. However, setting this bit prevents HOL (head-of-line) blocking in the Rx queues when packets sent to the RXDMA are stopped due to bus error.
2. Reprogram the specific registers of the DMA channel.
3. Start the DMA channel.

Recovering from the transmit DMA channel failure

1. Stop the specific DMA channel, even if it is in active state.
2. Flush the corresponding MTL queue.
3. Reprogram the specific registers of the DMA channel.
4. Start the DMA channel.

Note: Due to the known limitations, reprogramming the DMA channel registers might not always be successful to recover from a bus error. If the peripheral is not fully functional after reprogramming the DMA, as a workaround, issue a soft reset to recover from the bus error.

Programming guidelines for disabling receive queue

Follow these steps to disable specific receive queue:

1. Disable the receiver by setting the RE bit of the *Operating mode configuration register (ETH_MACCR)* to 0.
2. Read the *Debug register (ETH_MACDR)* (ensure that RPESTS and RFCFCSTS fields are 0) and *Rx queue x debug register (ETH_MTLRXQxDR)* of all the Rx queues.
3. When all *Rx queue x debug register (ETH_MTLRXQxDR)* are read as 0, disable the intended Rx queue by programming the corresponding fields in the *Rx queue control 0 register (ETH_MACRXQC0R)*.
4. Enable the receiver by setting the RE bit of *Operating mode configuration register (ETH_MACCR)* to 1.

76.9.9 Programming guidelines for GMII link state transitions

Transmit and receive clocks are running when the link is down

Complete the following steps when the link is down while the transmit and receive clocks are running:

1. Disable the Transmit DMA (if applicable) by clearing bit 0 (ST) of *Channel x control register (ETH_DMACxCR)*.
2. Disable the MAC receiver by clearing RE bit of *Operating mode configuration register (ETH_MACCR)*.
3. Wait for any previous frame transmissions to complete. You can check this by reading the appropriate bits of *Tx queue x debug register (ETH_MTLTXQxDR)* (TRCSTS[1:0] is not 01).

   or

   Flush the Tx FIFO for faster empty operation.
4. Disable the MAC transmitter by clearing TE bit of the *Operating mode configuration register (ETH_MACCR)* Register.
5. Make sure that both Tx and Rx queues are empty (TXQSTS is set to 0 in *Tx queue x debug register (ETH_MTLTXQxDR)* and RXQSTS[1:0] to 00 in *Rx queue x debug register (ETH_MTLRXQxDR)*).
6. After the link is up, read the PHY registers to identify the latest configuration and program the MAC registers accordingly.
7. Restart the operation by starting the Tx DMA. Then enable the MAC transmitter and receiver.

   The Rx DMA does not need to be enabled: since the receiver is disabled, there are no data in the Rx FIFO.

Transmit and receive clocks are stopped when the link is down

Complete the following steps when the link is down and the transmit and receive clocks are stopped:

1. Disable the MAC transmitter and receiver by clearing RE and TE bits in the *Operating mode configuration register (ETH_MACCR)*. This does not take immediate effect as the clocks are absent.
2. Wait till the link is up and the clocks are restored.
3. Wait until the transfer of any partial frame is complete if any was ongoing when the Transmit/Receive clock is stopped. This can be checked by reading the *Debug register (ETH_MACDR)* (all bits should be set to 0). Some old packets may still remain in the TXFIFO as the MAC transmitter is stopped.
4. Read the PHY registers to identify the latest operating mode and program the MAC registers accordingly.
5. Restart the MAC transmitter and receiver by setting RE and TE bits.

76.9.10 Programming guidelines for IEEE 1588 timestamping

Initializing the System time generation

The timestamp feature can be enabled by setting bit 0 of the *Timestamp control register (ETH_MACTSCR)*. However, it is essential that the timestamp counter is initialized after this bit is set. Complete the following steps to perform the peripheral initialization:

1. Mask the Timestamp Trigger interrupt by clearing bit 12 of *Interrupt enable register (ETH_MACIER)*.
2. Set bit 0 of *Timestamp control register (ETH_MACTSCR)* to enable timestamping.
3. Program *Subsecond increment register (ETH_MACSSIR)* based on the PTP clock frequency.
4. If you use the Fine Correction method, program *Timestamp addend register (ETH_MACTSAR)* and set bit 5 of *Timestamp control register (ETH_MACTSCR)*.
5. Poll the *Timestamp control register (ETH_MACTSCR)* until bit 5 is cleared.
6. Program bit 1 of *Timestamp control register (ETH_MACTSCR)* to select the Fine Update method (if required).
7. Program *System time seconds update register (ETH_MACSTSUR)* and *System time nanoseconds update register (ETH_MACSTNUR)* with the appropriate time value.
8. Set bit 2 in *Timestamp control register (ETH_MACTSCR)*.

   The timestamp counter starts as soon as it is initialized with the value written in the timestamp update registers. If one-step timestamping is required:

   a) Enable one-step timestamping by programming bit 27 of the TDES3 Context Descriptor.

   b) Program *Timestamp ingress asymmetric correction register (ETH_MACTSIACR)* to update the correction field in PDelay_Req PTP messages.
9. Enable the MAC receiver and transmitter for proper timestamping.

Note: If timestamp operation is disabled by clearing bit 0 of Timestamp control register (ETH_MACTSCR), repeat all these steps to restart the timestamp operation.

System time correction

To synchronize or update the system time in one shot (coarse correction method), complete the following steps:

1. Set the offset (positive or negative) in the timestamp update registers (*System time seconds update register (ETH_MACSTSUR)* and *System time nanoseconds update register (ETH_MACSTNUR)*).
2. Set bit 3 (TSUPDT) of the *Timestamp control register (ETH_MACTSCR)*.

   The value in the timestamp update registers is added to or subtracted from the system time when the TSUPDT bit is cleared.

To synchronize or update the system time to reduce system-time jitter (fine correction method), complete the following steps:

1. With the help of the algorithm described in *Section : System time register module*, calculate at which rate you intend to increment or decrement the system time.
2. Update the *Timestamp addend register (ETH_MACTSAR)* with the new value and set bit 5 of the *Timestamp control register (ETH_MACTSCR)* Register.
3. Wait for the time during which you want the new value of the Addend register to be active. This can be done by enabling the Timestamp Trigger interrupt after the system time reaches the target value.
4. Program the required target time in *PPS x target time seconds register (ETH_MACPPSTTSxR)* and *PPS x target time nanoseconds register (ETH_MACPPSTTNxR)*.
5. Enable the Timestamp interrupt in bit 12 of *Interrupt enable register (ETH_MACIER)*.
6. Set bit 3 in Register *Timestamp control register (ETH_MACTSCR)*.
7. When this trigger generates an interrupt, read *Interrupt status register (ETH_MACISR)*.
8. Reprogram *Timestamp addend register (ETH_MACTSAR)* with the old value and set bit 5 again.

76.9.11 Programming guidelines for PTP offload feature

Programming guidelines to enable automatic periodic generation of PTP sync messages

Follow these steps to enable automatic periodic generation of PTP sync messages:

1. Program SNAPTYPSEL, TSMSTRENA, and TSEVNTENA fields of *Timestamp control register (ETH_MACTSCR)* to 0, 1, and 1 respectively, to configure the node as Ordinary or Boundary Master (1, 1, and 1 for Transparent Master).
2. Program the PTOEN bit and DN field of *PTP offload control register (ETH_MACPOCR)* to enable PTP Offload feature and domain number to send in egress PTP Sync message.
3. Program the ASYNCEN bit of *PTP offload control register (ETH_MACPOCR)* to enable periodic generation of PTP Sync messages.
4. Program the 80-bit Source Port Identity in *PTP source port identity 0 register (ETH_MACSPI0R)*, *PTP Source port identity 1 register (ETH_MACSPI1R)* and *PTP Source port identity 2 register (ETH_MACSPI2R)* to send in egress PTP Sync message.
5. Program the LSI field of *Log message interval register (ETH_MACLMIR)* to program the periodicity of the PTP Sync messages.

   For example, a value of 1 corresponds to 2^1 which translates to PTP Sync message every 2 seconds, and a value of 0xFF (twos complement of -1) corresponds to 2^{-1} which translates to PTP Sync message every 0.536 seconds.
6. Program the TSIE bit of *Interrupt enable register (ETH_MACIER)* to enable generation of Timestamp interrupt.
7. Wait for sbd_intr_o interrupt generated by setting TXTSSIS bit in *Timestamp status register (ETH_MACTSSR)*. It indicates that the timestamp for PTP Sync message is captured in *Tx timestamp status seconds register (ETH_MACTXTSSSR)* and *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)*.

Programming guidelines to enable periodic generation of PTP Pdelay_Req messages

Follow these steps to enable automatic periodic generation of PTP Pdelay_Req messages

1. Program SNAPTYPSEL, TSMSTRENA and TSEVNTENA fields of *Timestamp control register (ETH_MACTSCR)* to 1, 0, and 1 respectively to configure the node as Transparent Slave (1, 1, and 1 for Transparent Master OR 3, X, and X for Peer-to-Peer Transparent).
2. Program the PTOEN bit and DN field of *PTP offload control register (ETH_MACPOCR)* to enable PTP Offload feature and domain number to send in egress PTP Pdelay_Req message.
3. Program the APDREQEN bit of *PTP offload control register (ETH_MACPOCR)* to enable periodic generation of PTP Pdelay_Req messages.
4. Program the 80-bit Source Port Identity in *PTP source port identity 0 register (ETH_MACSPI0R)*, *PTP Source port identity 1 register (ETH_MACSPI1R)* and *PTP Source port identity 2 register (ETH_MACSPI2R)* to send in egress PTP Pdelay_Req message.
5. Program the LMPDRI field of *Log message interval register (ETH_MACLMIR)* to program the periodicity of the PTP Pdelay_Req messages.

   For example, a value of 1 corresponds to 2^1 which translates to PTP Pdelay_Req message every 2 seconds, and a value of 0xFF (twos complement of -1) corresponds to 2^{-1} which translates to PTP Pdelay_Req message every 0.536 seconds.
6. Program the TSIE bit of *Interrupt enable register (ETH_MACIER)* to enable generation of Timestamp interrupt.
7. Wait for sbd_intr_o interrupt generated by setting TXTSSIS bit in *Timestamp status register (ETH_MACTSSR)*. It indicates that the timestamp for PTP Sync message is captured in *Tx timestamp status seconds register (ETH_MACTXTSSSR)* and *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)*.

Programming guidelines to enable the generation of PTP response messages for Ordinary or Boundary Master mode

Follow these steps to enable the generation of PTP response messages for Ordinary or Boundary Master mode (Periodic PTP Sync messages generated and PTP Delay_Resp message generated in response to PTP Delay_Req message):

1. Program SNAPTYPSEL, TSMSTRENA and TSEVNTENA fields of *Timestamp control register (ETH_MACTSCR)* to 0, 1, and 1 respectively.
2. Program the PTOEN bit and DN field of *PTP offload control register (ETH_MACPOCR)* to enable PTP Offload feature and domain number to match with ingress PTP Delay_Req message and send in egress PTP Delay_Resp message.
3. Program the 80-bit Source Port Identity in *PTP source port identity 0 register (ETH_MACSPI0R)*, *PTP Source port identity 1 register (ETH_MACSPI1R)* and *PTP Source port identity 2 register (ETH_MACSPI2R)* to match with ingress PTP Delay_Req message and send in egress PTP Delay_Resp message.
4. Program the DRSYNCR and LSI fields in *Log message interval register (ETH_MACLMIR)*. The sum of both fields is updated in logMinDelayReqInterval field of PTP Delay_Resp message.

Programming guidelines to enable the generation of PTP response messages for Ordinary or Boundary Slave mode

Follow these steps to enable generation of PTP response messages for Ordinary or Boundary Slave mode (PTP Delay_Req message generated in response to PTP Sync message):

1. Program SNAPTYPSEL, TSMSTRENA and TSEVNTENA fields of *Timestamp control register (ETH_MACTSCR)* to 0, 0, and 1 respectively.
2. Program the PTOEN bit and DN field of *PTP offload control register (ETH_MACPOCR)* to enable PTP Offload feature and domain Number to match with ingress PTP Sync message and send in egress PTP Delay_Req message.
3. Program the 80-bit Source Port Identity in *PTP source port identity 0 register (ETH_MACSPI0R)*, *PTP Source port identity 1 register (ETH_MACSPI1R)* and *PTP Source port identity 2 register (ETH_MACSPI2R)* to match with ingress PTP Sync message and send in egress PTP Delay_Req message.
4. Program the DRSYNCR field in *Log message interval register (ETH_MACLMIR)* to indicate one PTP Delay_Req message is generated in response to how many received PTP Sync messages.

Programming guidelines to enable the generation of PTP response messages for Transparent Slave mode

Follow these steps to enable generation of PTP response messages for Transparent Slave mode (PTP Delay_Req message generated in response to PTP Sync message, PTP Pdelay_Resp message generated in response to PTP Pdelay_Req message and Periodic PTP Pdelay_Req messages generated)

1. Program SNAPTYPSEL, TSMSTRENA and TSEVNTENA fields of *Timestamp control register (ETH_MACTSCR)* to 1, 0, and 1 respectively.
2. Program the PTOEN bit and DN field of *PTP offload control register (ETH_MACPOCR)* to enable PTP Offload feature and domain Number to match with ingress PTP Sync or Pdelay_Req message and send in egress PTP Delay_Req or Pdelay_Resp or Pdelay_Req message.
3. Program the 80-bit Source Port Identity in *PTP source port identity 0 register (ETH_MACSPI0R)*, *PTP Source port identity 1 register (ETH_MACSPI1R)* and *PTP Source port identity 2 register (ETH_MACSPI2R)* to match with ingress PTP Sync or Pdelay_Req message and send in egress PTP Delay_Req or Pdelay_Resp or Pdelay_Req message.
4. Program the DRSYNCR and LMPDRI fields in *Log message interval register (ETH_MACLMIR)* to indicate one PTP Delay_Req message is generated in response to how many received PTP Sync messages and periodicity of the PTP Pdelay_Req messages.
5. Program the TSIE bit of *Interrupt enable register (ETH_MACIER)* to enable generation of Timestamp interrupt.
6. Wait for sbd_intr_o interrupt generated by setting TXTSSIS bit in *Timestamp status register (ETH_MACTSSR)*. It indicates that the timestamp for PTP Sync message is captured in *Tx timestamp status seconds register (ETH_MACTXTSSSR)* and *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* for egress PTP Pdelay_Req and Pdelay_Resp messages.

Programming guidelines to enable the generation of PTP response messages for Transparent Master mode

Follow these steps to enable generation of PTP response messages for Transparent Master mode (PTP Delay_Resp message generated in response to PTP Delay_Req message, PTP Pdelay_Resp message generated in response to PTP Pdelay_Req message and Periodic PTP Pdelay_Req or Sync messages generated):

1. Program SNAPTYPSEL, TSMSTRENA and TSEVNTENA fields of *Timestamp control register (ETH_MACTSCR)* to 1, 1, and 1 respectively.
2. Program the PTOEN bit and DN field of *PTP offload control register (ETH_MACPOCR)* to enable PTP Offload feature and domain number to match with ingress PTP Delay_Req or Pdelay_Req message and send in egress PTP Delay_Resp or Pdelay_Resp or Pdelay_Req or Sync message.
3. Program the 80-bit Source Port Identity in *PTP source port identity 0 register (ETH_MACSPI0R)*, *PTP Source port identity 1 register (ETH_MACSPI1R)* and *PTP Source port identity 2 register (ETH_MACSPI2R)* to match with ingress PTP Delay_Req or Pdelay_Req message and send in egress PTP Delay_Resp or Pdelay_Resp or Pdelay_Req or Sync message.
4. Program the DRSYNCR, LSI and LMPDRI fields in *Log message interval register (ETH_MACLMIR)*, the sum of DRSYNCR and LSI is updated in logMinDelayReqInterval field of PTP Delay_Resp message and periodicity of the PTP Sync or Pdelay_Req messages.
5. Program the TSIE bit of *Interrupt enable register (ETH_MACIER)* to enable generation of Timestamp interrupt.
6. Wait for sbd_intr_o interrupt generated by setting TXTSSIS bit in *Timestamp status register (ETH_MACTSSR)*. It indicates that the timestamp for PTP Sync message is captured in *Tx timestamp status seconds register (ETH_MACTXTSSSR)* and *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* for egress PTP Sync, Pdelay_Req and Pdelay_Resp messages.

Programming guidelines to enable the generation of PTP response messages for Peer-to-Peer Transparent mode

Follow these steps to enable generation of PTP response messages for Peer-to-Peer Transparent mode (PTP Pdelay_Resp message generated in response to PTP Pdelay_Req message and Periodic PTP Pdelay_Req messages generated):

1. Program the SNAPTYPSEL, TSMSTRENA and TSEVNTENA fields of *Timestamp control register (ETH_MACTSCR)* to 3, X, and X respectively.
2. Program the PTOEN bit and DN field of *PTP offload control register (ETH_MACPOCR)* to enable PTP Offload feature and domain Number to match with ingress PTP Pdelay_Req message and send in egress PTP Pdelay_Resp message.
3. Program the 80-bit Source Port Identity in *PTP source port identity 0 register (ETH_MACSPI0R)*, *PTP Source port identity 1 register (ETH_MACSPI1R)* and *PTP Source port identity 2 register (ETH_MACSPI2R)* to match with ingress PTP Pdelay_Req message and send in egress PTP Pdelay_Resp message.
4. Program the LMPDRI field in *Log message interval register (ETH_MACLMIR)* to indicate periodicity of the PTP Pdelay_Req messages

5. Program the TSIE bit of *Interrupt enable register (ETH_MACIER)* to enable generation of Timestamp interrupt
6. Wait for sbd_intr_o interrupt generated by setting TXTSSIS bit in *Timestamp status register (ETH_MACTSSR)*. It indicates that the timestamp for PTP Sync message is captured in *Tx timestamp status seconds register (ETH_MACTXTSSSR)* and *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* for egress PTP Pdelay_Req and Pdelay_Resp messages.

76.9.12 Programming guidelines for AV feature

Initializing the DMA

Complete the following steps to initialize the DMA:

1. Provide a software reset to reset all registers and logic (bit 0 in ETH_DMAMR).
2. Wait for the completion of the reset process. Poll bit 0 of the ETH_DMAMR, which is cleared only when the reset operation is completed.
3. Program the fields to initialize the DMA register by setting the values in ETH_DMAMR register.
4. Create a proper descriptor list for transmit and receive. In addition, ensure that the DMA owns the transmit and receive descriptors. When OSF mode is used, at least two TX descriptors are required. For more information about descriptors, refer to *Section 76.10: Descriptors*.
5. Make sure that your software creates three or more different transmit or receive descriptors in the list before reusing any of the descriptors.
6. Program the transmit and receive Ring length registers (*Channel x Tx descriptor ring length register (ETH_DMACxTXRLR)* and *Channel x Rx descriptor ring length register (ETH_DMACxRXRLR)*). The ring length programmed must be at least 4.
7. Initialize the receive and transmit descriptor list address with the base address of the transmit and receive descriptor (*Channel x Tx descriptor list address register (ETH_DMACxTXDLAR)*, *Channel x Rx descriptor list address register (ETH_DMACxRXDLAR)*). In addition, program the transmit and receive tail pointer registers indicating the available descriptors to the DMA (*Channel x Tx descriptor tail pointer register (ETH_DMACxTXDTPR)* and *Channel x Rx descriptor tail pointer register (ETH_DMACxRXDTPR)*).
8. Program the following fields to initialize the operating mode in the ETH_MTLTXQ0OMR:
   a) Transmit Store And Forward (TSF)
   b) Transmit Threshold Control (TTC)
   c) Transmit Queue Enable (TXQEN) to value 10 to enable Transmit Queue0
   d) Transmit Queue Size (TQS)
9. Enable the interrupts by programming the EH_DMAC0IER.
10. Repeat steps 4 to 9 for all the Tx DMA and RX DMA channels.
11. Program the CBS control register, idleSlope, sendSlope, hiCredit, and loCredit registers of the AV queues.
12. Start the Receive and Transmit DMA by setting bit 0 of the ETH_DMAC0TXCR and bit 0 of the ETH_DMAC0RXCR (see *Channel x transmit control register (ETH_DMACxTXCR)* and *Channel x receive control register (ETH_DMACxRXCR)*).

Enabling slot number checking

You can use the slot number check feature to specify the intervals at which the DMA Channels mapped to AV queues fetches the frames from the AHB/AXI system bus. This feature is useful to ensure a uniform and periodic transfer of the AV traffic from the host memory. It is available only when timestamping is enabled and the *Subsecond increment register (ETH_MACSSIR)* is programmed. Complete the following steps to enable the slot number checking:

Note: The slot number checking steps should be complete after programming the CBS control register, idleSlope, sendSlope, hiCredit, and loCredit registers of the AV queues (step 11 of Section : Initializing the DMA). and before starting the Receive and Transmit DMA (step 12).

1. Enable timestamping by following the steps described in *Section : Initializing the System time generation*.
2. Make sure that the SLOTNUM field (bits 22 to 19) of TDES3 Normal Descriptor (Read Format) contains a valid slot number. To do this, read the current reference slot number from the *Channel x slot function control status register (ETH_DMACxSFCSR)*.
3. Set bit 0 (ESC) of *Channel x slot function control status register (ETH_DMACxSFCSR)* to enable the slot number checking.

Enabling average bits per slot reporting

The CBS Status register of the additional AV channels provides information about the average bits that are transmitted in a slot. The software can asynchronously read this register to retrieve information about the average bits transmitted per slot. Complete the following steps to enable average bits per slot reporting:

1. Enable timestamping by following the steps described in *Section : Initializing the System time generation*.
2. Program SLC bits of the *Tx queue 1 ETS control register (ETH_MTLTXQ1ECR)* of a channel with number of slots over which the average transmitted bits per slot need to be computed.
3. Enable ABPSIE of the *Queue x interrupt control status register (ETH_MTLQxICSR)* of a channel to generate the average bits per slot interrupt.

Note: The frequency of this interrupt depends on the value programmed during previous step. For example, when you program value 0 in the SLC field, the interrupt is generated at every 125 microsecond.

When it is not required, you can disable this interrupt to stop the interrupt flooding.

4. Read ABS bits from *Tx queue x ETS status register (ETH_MTLTXQxESR)* of a channel on each interrupt.

   The software can read the ABS bits in polling mode even if the ABPSIE bit is not enabled. When high, bit 1 (ABPSIS) of *Queue x interrupt control status register (ETH_MTLQxICSR)* indicates that a new value is updated in the ABS field.

Disabling flow control for AV-enabled queues

Transmit flow

Program the EHFC (Enable Hardware Flow Control) bit of corresponding *Rx queue x operating mode register (ETH_MTLRXQxOMR)* to 0.

76.9.13 Programming guidelines for Energy Efficient Ethernet (EEE)

Entering and exiting Tx LPI mode

EEE enables the IEEE 802.3 Media Access Control (MAC) sublayer along with a family of physical layers to operate in the Low-power idle (LPI) mode. In the Transmit path, the software must set the LPIEN bit of the *LPI control and status register (ETH_MACLCSR)* to indicate to the MAC to stop transmission and initiate the LPI protocol.

Complete the following steps during MAC initialization:

1. Read the PHY register through the MDIO interface and check if the remote end has the EEE capability. Then negotiate the timer values.
2. Program the PHY registers through the MDIO interface (including the RX_CLK_stoppable bit that indicates to the PHY whether to stop Rx clock in LPI mode or not).
3. Program bits 25 to 16 and bits 15 to 0 in *LPI timers control register (ETH_MACLTCR)*.
4. Read the PHY link status by using the MDIO interface and update bit 17 of *LPI control and status register (ETH_MACLCSR)*.

   Update *LPI control and status register (ETH_MACLCSR)* accordingly. This update should be done whenever the link status in the PHY chip changes.
5. Program *One-microsecond-tick counter register (ETH_MAC1USTCR)* as per the frequency of the clock used for accessing the CSR slave port.
6. Program the LPIET bit in the *LPI entry timer register (ETH_MACLETR)* with the IDLE time for which the MAC should wait before entering the LPI state on its own.
7. Set LPITE and LPITXA (bits 20 to 19) of *LPI control and status register (ETH_MACLCSR)* to enable LPI auto-entry and MAC auto-exit from LPI state.
8. Set bit 16 of *LPI control and status register (ETH_MACLCSR)* to put the MAC transmitter in LPI state.

   The MAC enters the LPI state when all scheduled packets are completed. It remains IDLE for the time indicated by LPIET bits. It sets the TLPIEN (bit 0) after entering LPI state.
9. When a packet transmission is scheduled (when the TxDMA exits IDLE state or when a packet is presented at ATI or MTI interface), the MAC transmitter automatically exits LPI state. It waits for TWT time before setting the TLPIEX interrupt status bit and then resume the packet transmission.
10. The MAC transmitter enters again LPI state if it remains IDLE for LPIET time. It then sets the TLPIEN bit and the entry-exit cycle continues.
11. Reset LPIEN bit if the application needs to override the auto-entry/exit modes and directly exit the MAC transmitter from LPI state.

Note: To make sure the MAC enters the LPI state only after the transmission of all the queued frames in the Tx FIFO is complete, set LPITXA bit in LPI control and status register (ETH_MACLCSR).

To switch off the GMII Transmit clock during the LPI state, use the sbd_tx_clk_gating_ctrl_o signal to gate the clock input.

To switch off the CSR clock or power to the rest of the system during the LPI state, wait for the TLPIEN interrupt of LPI control and status register (ETH_MACLCSR) to be generated. Restore the clocks before performing step 6 when you want to come out of the LPI state.

Gating Off the CSR Clock in the LPI mode

You can gate off the CSR clock to save the power when the MAC is in the Low-Power Idle (LPI) mode.

Gating off the CSR clock in the Rx LPI mode

The following operations are performed when the MAC receives the LPI pattern from the PHY:

1. The MAC RX enters the LPI mode and the Rx LPI entry interrupt status (RLPIEN interrupt of *LPI control and status register (ETH_MACLCSR)*) is set.
2. The interrupt pin (sbd_intr_o) is asserted. The sbd_intr_o interrupt is cleared when the host reads the *LPI control and status register (ETH_MACLCSR)*.

After the sbd_intr_o interrupt is asserted and the MAC Tx is also in the LPI mode, the CSR clock can be gated off. If the MAC TX is not in LPI mode when the CSR clock is gated off, the events on the MAC transmitter do not get reported or updated in the CSR. To restore the CSR clock, wait for the LPI exit indication from the PHY after which the MAC asserts the LPI exit interrupt on lpi_intr_o (synchronous to clk_rx_i). The lpi_intr_o interrupt is cleared when *LPI control and status register (ETH_MACLCSR)* is read.

Gating off the CSR clock in the Tx LPI mode

The following operations are performed when bit 16 (LPIEN) of *LPI control and status register (ETH_MACLCSR)* is set:

1. The Transmit LPI Entry interrupt (TLPIEN bit of *LPI control and status register (ETH_MACLCSR)*) is set.
2. The interrupt pin (sbd_intr_o) is asserted. The sbd_intr_o interrupt is cleared when the host reads the *LPI control and status register (ETH_MACLCSR)*.

After the sbd_intr_o interrupt is asserted and the MAC RX is also in the LPI mode, the CSR clock can be gated off. If the MAC RX is not in LPI mode when the CSR clock is gated off, the events on the MAC receiver do not get reported or updated in the CSR. To restore the CSR clock, switch on the CSR clock when the MAC has exited TX LPI mode. After the CSR clock is resumed, reset bit 16 (LPIEN) of *LPI control and status register (ETH_MACLCSR)* to exit the MAC from LPI mode.

76.9.14 Programming guidelines for flexible pulse-per-second (PPS) output

Generating a single pulse on PPS

To generate a single pulse on PPS:

1. Program TRGTMODSEL[1:0] bit to 11 or 10 (for interrupt) in *PPS control register (ETH_MACPPSCR)*. This instructs the MAC to use the Target Time registers (*PPS x target time seconds register (ETH_MACPPSTTSxR)* and *PPS x target time nanoseconds register (ETH_MACPPSTTNxR)*) as start time of PPS signal output.
2. Program the start time value in the target time registers (register *PPS x target time seconds register (ETH_MACPPSTTSxR)* and *PPS x target time nanoseconds register (ETH_MACPPSTTNxR)*).
3. Program the width of the PPS signal output in *PPS x width register (ETH_MACPPSWxR)* Register.
4. Program PPSCMD[3:0] of *PPS control register (ETH_MACPPSCR)* to 0001. This instructs the MAC to generate a single pulse on the PPS signal output at the time programmed in the Target Time registers.

Generating next pulse on PPS

When the PPSCMD is executed (PPSCMD bits = 0), you can cancel the pulse generation by giving the Cancel Start Command (PPSCMD=0011) before the programmed start time has elapsed. You can also program the behavior of the next pulse in advance. To program the next pulse:

1. Program the start time for the next pulse in the Target Time registers. This time should be higher than the time at which the falling edge occurs for the previous pulse.
2. Program the width of the next PPS signal output in *PPS x width register (ETH_MACPPSWxR)*.
3. Program PPSCMD[3:0] bits of *PPS control register (ETH_MACPPSCR)* to generate a single pulse after the previous pulse is deasserted. This instructs the MAC to generate a single pulse on the PPS signal output at the time programmed in Target Time registers.

   If this command is given before the previous pulse becomes low, then the new command overwrites the previous command and the peripheral may generate only 1 extended pulse.

Generating a pulse train on PPS

To generate a pulse train on PPS:

1. Program TRGTMODSEL[1:0] bits to 11 or 10 (for interrupt) in *PPS control register (ETH_MACPPSCR)*. This instructs the MAC to use the Target Time registers (*PPS x target time seconds register (ETH_MACPPSTTSxR)* and *PPS x target time nanoseconds register (ETH_MACPPSTTNxR)*) for start time of the PPS signal output.
2. Program the start time value in the Target Time registers (register *PPS x target time seconds register (ETH_MACPPSTTSxR)* and *PPS x target time nanoseconds register (ETH_MACPPSTTNxR)*).
3. Program the interval value between the train of pulses on the PPS signal output in *PPS x interval register (ETH_MACPPSIxR)*.
4. Program the width of the PPS signal output in *PPS x width register (ETH_MACPPSWxR)*.
5. Program PPSCMD[3:0] bits in *PPS control register (ETH_MACPPSCR)* to 0010. This instructs the MAC to generate a train of pulses on the PPS signal output at the start time programmed in Target Time registers.

   By default, the PPS pulse train is free-running unless it is stopped by issuing a ‘STOP Pulse train at time’ or ‘STOP Pulse Train immediately’ commands.
6. Program the stop value in the Target Time registers. Ensure that TSTRBUSY bit in *PPS x target time nanoseconds register (ETH_MACPPSTTNxR)* is reset before programming the Target Time registers again.
7. Program the PPSCMD[3:0] bits in *PPS control register (ETH_MACPPSCR)* to 0100 to stop the train of pulses on PPS signal output after the programmed stop time specified at step 6 has elapsed.

The pulse train can be stopped at any time by programming 0101 in the PPSCMD[3:0] field.

Similarly, the Stop Pulse train command (given in Step 7) can be canceled by programming PPSCMD[3:0] bits to 0110 before the time (programmed at step 6) has elapsed.

The pulse train generation can be stopped by programming PPSCMD[3:0] to 0011 before the start time programmed at step 2) has elapsed.

Generating an interrupt without affecting the PPS

TRGTMODSEL[1:0] bits in *PPS control register (ETH_MACPPSCR)* enable you to program the Target Time registers (*PPS x target time seconds register (ETH_MACPPSTTSxR)* and *PPS x target time nanoseconds register (ETH_MACPPSTTNxR)*) to do any one of the following:

- Generate only interrupts.
- Generate interrupts and the PPS start and stop time.
- Generate only PPS start and stop time.

To program the Target Time registers to generate only interrupt event:

1. Program TRGTMODSEL[1:0] bits of *PPS control register (ETH_MACPPSCR)* to 00 (for interrupt). This instructs the MAC to use the Target Time registers for target time interrupt.
2. Program a target time value in the Target Time registers. This instructs the MAC to generate an interrupt when the target time elapses.

   If TRGTMODSEL[1:0] bits are changed (for example, to control the PPS), then the interrupt generation is overwritten with the new mode and new programmed Target Time register value.

Note: The TSTRGTERR0 bit in Timestamp status register (ETH_MACTSSR) is set when the programmed target time is smaller (that is corresponds to a time in the past) compared to the system time in the System time seconds register (ETH_MACSTSR) and System time nanoseconds register (ETH_MACSTNR).

An interrupt is generated (sbd_intr_o) if the TSIE bit in the Interrupt enable register (ETH_MACIER) is set.

Therefore, to avoid unwanted interrupts, the correct writing order is as follows:

1. *PPS x target time nanoseconds register (ETH_MACPPSTTNxR).*
2. *PPS x target time seconds register (ETH_MACPPSTTSxR).*
3. *PPS x interval register (ETH_MACPPSIxR).*
4. *PPS x width register (ETH_MACPPSWxR).*
5. *PPSCTRL[3:0] and PPSCTRL[3:0] and PPSEN0 bitfields of PPS control register (ETH_MACPPSCR).*

76.9.15 Programming guidelines for IEEE 1588 auxiliary snapshot

The IEEE 1588 auxiliary snapshot function is used to capture the auxiliary snapshot for a trigger event received on the ptp_aux_ts_trig_i input.

To program the IEEE 1588 auxiliary snapshot function, complete the following steps:

1. Program the TSIE bit in the *Interrupt enable register (ETH_MACIER)* to enable interrupt generation when the auxiliary snapshot is captured in the FIFO.
2. Program the ATSENx bit of the *Auxiliary control register (ETH_MACACR)* to enable the capture of the auxiliary snapshot based on the respective trigger input.
3. If the software does not want the captured auxiliary snapshot to be read, program the ATSFC bit of *Auxiliary control register (ETH_MACACR)* anytime to clear the auxiliary snapshot FIFO.
4. When TSIS is set or an interrupt is generated (due to TSIE programmed to 1), read the *Timestamp status register (ETH_MACTSSR)*, *Tx timestamp status seconds register*

(ETH_MACTXTSSSR), and *Auxiliary timestamp nanoseconds register (ETH_MACATSNR)* to access the captured auxiliary snapshot.

Note: When the auxiliary snapshot is captured in the FIFO for a trigger event, the following bitfields are updated:

- *AUXTSTRIG (status bit indicating that an auxiliary snapshot has been captured), ATSSTN (trigger event input source identifying the first available entry in the FIFO), and ATSNS (indicating the number of auxiliary snapshots available in the FIFO) bitfields of the Timestamp status register (ETH_MACTSSR).*
- *TSIS (indicating that an interesting event occurred in the IEEE 1588 timestamp function) bitfield in the Interrupt status register (ETH_DMAISR).*

The AUXTSTRIG bit is cleared on read (or set when the RCWE bit of the CSR software control register (ETH_MACCSRSWCR) is set).

Note: When the captured auxiliary snapshot is ignored because that the FIFO is full, the ATSSTM bit, which indicates that a trigger has been missed, is set.

76.9.16 Programming guidelines for TSO

The TCP segmentation offload (TSO) engine is used to offload the TCP segmentation functions to the hardware. To program the TSO, set the TSE bit to enable TCP packet segmentation, and program descriptor fields to enable TSO for the current packet.

Follow the steps below to program TSO:

1. Program TSE bit of the corresponding *Channel x transmit control register (ETH_DMACxTXCR)* to enable TCP packet segmentation in that DMA.
2. In addition to the normal transfer descriptor setting, the following descriptor fields must be programmed to enable TSO for the current packet:
   a) Enable TSE of TDES3 (bit 18).
   b) Program the length of the unsegmented TCP/IP packet payload in bits 17 to 0 of TDES3, and the TCP header in bits 22 to 19 of TDES3.
   c) Program the maximum size of the segment in:
   – MSS[13:0] of *Channel x control register (ETH_DMACxCR)*
   – or MSS in the context descriptor

   If MSS[13:0] field is programmed in both *Channel x control register (ETH_DMACxCR)* and in the context descriptor, the latest software programmed sequence is considered.
3. The unsegmented TCP/IP packet header should be stored in Buffer 1 of the first descriptor. This buffer must not hold any payload bytes. The payload is allocated to Buffer 2 and the buffers of the subsequent descriptors.

Caution: If TSE is enabled in TDES3 for a non-TCP-IP packet, the result is unpredictable.

76.9.17 Programming guidelines to perform VLAN filtering on the receiver

Follow the sequence below to perform VLAN filtering on the receiver:

1. Program *VLAN tag Control register (ETH_MACVTCR)* for the following bit to select the filtering method:
   - ETV: Enable 12-bit VLAN Tag Comparison or 16-bit VLAN Tag comparison.
   - VTHM: VLAN Tag Hash Table Match Enable.
   - ERIVLT: Enable inner VLAN Tag or outer VLAN Tag (to enable the inner or outer VLAN Tag filtering, Double VLAN Processing should enabled by setting EDVLP)
   - ERSVLM: Enable Receive S-VLAN Match or C-VLAN match (for S-VLAN processing to be enabled, set ESVL)
   - DOVLTC: Ignores VLAN Type for Tag Match
   - VTIM: to enable VLAN Tag Inverse Match instead of the normal VLAN Tag matching
2. Program VL bit in *VLAN tag Control register (ETH_MACVTCR)* for the 12-bit or 16-bit VLAN tag.
3. If VLAN tag Hash filtering is enabled, program *VLAN Hash table register (ETH_MACVHTR)*.
   - When the ETV bit is reset, the upper 4 bits of the calculated CRC-32 of VLAN tag are inverted and used to index the content of the *VLAN Hash table register (ETH_MACVHTR)*.
   - When ETV bit is set, the upper 4 bits of the calculated CRC-32 of VLAN tag are used to index the content of *VLAN Hash table register (ETH_MACVHTR)*.

   For example, when ETV bit is set, a hash value of 0b1000 selects bit 8 of the VLAN Hash table. When ETV bit is reset, a hash value of 0b1000 selects bit 7 of the VLAN Hash table.

76.9.18 Programming guidelines for extended VLAN filtering and routing on reception

For the indirect access to the per VLAN Tag registers, follow these steps:

- Write access
  a) Write the required data into the *VLAN tag data register (ETH_MACVTDR)*.
  b) Program the OFS field of *VLAN tag Control register (ETH_MACVTCR)* with the required filter register offset and command type to the CT field. For a write command, set this bit to 0.
  c) Write 1 to the OB field and wait until the OB bit is reset before performing the next write. This is to ensure that the appropriate VLAN Tag Filter Register has been programmed.
- Read access
  a) Program the OFS field of the *VLAN tag Control register (ETH_MACVTCR)* with the required register offset and command type to the CT field. For a read command, set this bit to 1.
  b) Write 1 to the OB field and wait till the OB bit is reset. The appropriate VLAN Tag Filter register value is available in the *VLAN tag data register (ETH_MACVTDR)*.

76.9.19 Programming sequence for queue/channel-based VLAN inclusion register

Writing to the per queue/channel VLAN inclusion indirect registers

To write to the queue/channel-based VLAN inclusion register, complete the following steps.

Note: Indirect VLAN include registers are not accessed while setting the CBTI bit.

1. Set the CBTI bit of *VLAN inclusion register (ETH_MACVIR)* to 1, to enable queue/channel based VLAN tag insertion on all transmitted packets. This bit must be set before any indirect access to the queue/channel specific *VLAN inclusion register [alternate] (ETH_MACVIR)*.
2. Set the RDWR field of *VLAN inclusion register (ETH_MACVIR)* to 1, to indicate write access.
3. Program the VLAN tag and VLAN type to be inserted in the packets from a particular queue/channel in VLT[15:0] and CSVL fields of *VLAN inclusion register (ETH_MACVIR)*. The corresponding offset address in ADDR field (0 for queue/channel 0, 1 for queue/channel 1, and so on) must be set. Set the RDWR bit to 1 to indicate write access. The write to byte 0 of *VLAN inclusion register (ETH_MACVIR)* initiates access to indirect access *VLAN inclusion register [alternate] (ETH_MACVIR)*.
4. The BUSY bit of *VLAN inclusion register (ETH_MACVIR)* is set by the peripheral to indicate the progress of access to indirect access *VLAN inclusion register [alternate] (ETH_MACVIR)*. On completion of the access, the BUSY bit is cleared. The application must not attempt subsequent access to *VLAN inclusion register [alternate] (ETH_MACVIR)* when the BUSY bit is 1.
5. Repeat step 2 and step 3 to program VLAN tag and VLAN type to be inserted in packets from the remaining queues/channels. The application must ensure that the required VLAN tag and VLAN type for all the queues/channels are programmed; otherwise unintended VLAN tag and VLAN type might be inserted.

Read the per queue/channel VLAN inclusion indirect registers

To read the queue/channel-based VLAN inclusion register, complete the following steps:

1. Set the CBTI field of *VLAN inclusion register (ETH_MACVIR)* to 1, to enable queue/channel based VLAN tag insertion on all the transmitted packets. This bit must be set before indirect access to the queue/channel specific *VLAN inclusion register [alternate] (ETH_MACVIR)*.
2. Program the read offset address in the ADDR field (0 for queue/channel 0, 1 for queue/channel 1, and so on).

   Set the RDWR bit to 0 to indicate read access.
3. The BUSY field of *VLAN inclusion register (ETH_MACVIR)* is set by the peripheral to indicate the progress of access to the indirect access *VLAN inclusion register [alternate] (ETH_MACVIR)*. On completion of the access, the BUSY field is cleared. Bits[15:0] and bit [19] of *VLAN inclusion register (ETH_MACVIR)* contain the VLAN tag and VLAN type respectively, from the corresponding queue/channel/offset address.
4. Repeat step 2 and step 3 to read VLAN tag and VLAN type from the remaining queues/channels.

76.9.20 Programming guidelines for EST

Program the gate control values and time intervals in the software owned gate control list (SWOL) along with the other EST related registers that are described in *Section 76.11.3: Ethernet MTL registers*, to appropriate values. The following subsections provide step-by-step details for programming the GCL and the other EST related registers.

Programming the GCL and GCL-linked registers

Follow these steps to program the gate control list (GCL) and the four other registers implemented per GCL.

1. The GCL and the four other GCL-linked registers should be accessed through indirect addressing using the *EST Gate Control List register (ETH_MTLESTGCLCR)* and *EST Gate Control List Data register (ETH_MTLESTGCLDR)*. The SWOL bitfield of the *EST Status register (ETH_MTLESTSR)* indicates if GCL0 or GCL1 is owned by software.
2. To program the GCL, write the 32-bit write data to *EST Gate Control List Data register (ETH_MTLESTGCLDR)*. Then program *EST Gate Control List Data register (ETH_MTLESTGCLDR)* register to write the write address and other control information.
3. In the *EST Gate Control List Data register (ETH_MTLESTGCLDR)*, writing data consists of 8 bits of gate controls plus 24 bits of time interval. Gate close is indicated by programming a 0 and gate open is indicated by programming a 1. The data should be written in the following format:

   {TC7, TC6, TC5, TC4, TC3, TC2, TC1, TC0, 24-bit time interval} where TCx = 0 or 1.
4. In the *EST Gate Control List Data register (ETH_MTLESTGCLDR)*, program the SRWO bit to 1 (to start a write operation) and program the address and r/w bitfields appropriately.
5. Poll and check for the SRWO bit to be cleared by hardware to indicate the completion of the previous operation before initiating a new read/write operation using the same indirect addressing mode.
6. Repeat steps 3, 4, 5 until the programming of the GCL is complete.
7. Using the indirect addressing method, program the BTR, CTR, TER and LLR registers. Set the GCRR bit in the *EST Gate Control List Data register (ETH_MTLESTGCLDR)* appropriately. The GCRR bit interprets the address field as belonging to these registers (instead of the GCL).

After programming the GCL and the related registers, program the *EST Control register (ETH_MTLESTCR)* to allow the hardware to own and process the GCL. When the List length (as indicated in LLR) is 1, the associated time interval should be smaller than the value of the cycle time register. Otherwise, an error is reported (as detailed in the Error handling section) as a single set of gate controls adds no value in the TSN context.

Note: The time unit in all the GCL related registers is seconds and nanoseconds. In cases where internally generated PTP system time is used, the nanoseconds field must be programmed to use the Digital rollover mode (TSCTRLSSR bitfield of Timestamp control register (ETH_MACTSCR) set to 1).

Programming the EST registers

After completing the steps mentioned in *Section : Programming the GCL and GCL-linked registers*, program the *EST Control register (ETH_MTLESTCR)*.

1. Set the current time offset value and time internal left shift bitfields of the *EST Control register (ETH_MTLESTCR)* appropriately. In addition, set the enable EST (EEST) and switch to SWOL (SWOL) bits.
2. This enables the Ethernet controller to own and process the new GCL and switch to the new GCL at the BTR value. If enabled, the Ethernet controller generates an interrupt when the switch to the new list occurs.
3. Software must address any other interrupts received during the hardware execution of the GCL.

Set the SSWL field in the *EST Control register (ETH_MTLESTCR)* to hand off to the controller.

The SSWL bit is reset/cleared by the controller when it successfully switches to the new list. The controller also flips the SWOL bit to indicate the new GCL that the software owns.

To install a new GCL, program the GCL it owns (indicated by SWOL bit) as described in *Section : Programming the GCL and GCL-linked registers* and then program the *EST Control register (ETH_MTLESTCR)* as described above. Ensure that the new BTR is set to an appropriate value to avoid BTR error that might need software intervention in some cases.

To avoid transmission overruns, the packet length (frame size) information should be available at all times. Therefore, in the DMA configurations, program the packet length in the first descriptor of every Tx frame. Similarly, in the MTL configuration, provide the packet length in the control word.

76.9.21 Programming the launch time in time-based scheduling

The launch time is programmed in the enhanced normal transmit descriptors.

The OSTC and launch time features are mutually exclusive and must not be used together; in case of a conflict (if OSTC = LTV = 1 in MTL configuration), priority is given to OSTC and the launch time is ignored.

If a context descriptor is received with a valid OSTC value immediately before receiving a first normal descriptor with LTV bit set, then the LTV is ignored.

76.9.22 Enabling the frame preemption function

Receive-side programming

1. Enable at least two Rx queues by programming the appropriate values in the *Rx queue control 0 register (ETH_MACRXQC0R)*; one queue for express and the other for preemption when preemption receive traffic is expected.
2. Program the *Rx queue control 1 register (ETH_MACRXQC1R)*, *Rx queue control 2 register (ETH_MACRXQC2R)* and *Rx Queue control register (ETH_MACRXQCR)* to

route the received packets to MTL Rx queues. Ensure that the preemptable traffic and express traffic are not routed to the same Rx queue.

As the queue mapping for tagged packets is based on VLAN user-priority field, the priority of preemptable and express packets must be exclusive to each other. In other words, packets of a certain priority (traffic class) are either express or preemptable but cannot be both.

3. Program the preemption residue queue (FPRQ[2:0] bitfield of the *Rx queue control 1 register (ETH_MACRXQC1R)*) to route all non-classified received preemptive packets, so that these packets are not routed to the default or residue queue for express packets, which is Rx Queue 0.

Transmit-side programming

1. Enable at least two Tx queues by programming appropriate values in the TXQEN[1:0] bitfield of the *Tx queue x operating mode register (ETH_MTLTXQxOMR)*, one for express and the other for preemption (by setting the corresponding bit in the PEC bitfield of *FPE Frame Preemption Control Status register (ETH_MTLFPECSR)*) when preemption receive traffic is expected.
2. Set the EFPE bit of the *FPE control and status register (ETH_MACFPECSR)* to enable the preemption function in the transmitter.
3. Program the AFSZ[1:0] bitfield of the *FPE Frame Preemption Control Status register (ETH_MTLFPECSR)* to set the minimum size of the non-final fragments.
4. When 802.1Qbv/EST (Enhancement to Scheduled Traffic) is also enabled, follow the programming guidelines for EST and GCL.
5. Program the Hold Advance (HADV[15:0]) and Release Advance (RADV[15:0]) bitfields of *FPE Frame Preemption Advance register (ETH_MTLFPEAR)* with how much early the hold and release, respectively, must be initiated, so that the hold and release happen at exactly the boundaries of GCL rows. The value of HADV[15:0] must be equal to the time required to transmit minimum_fragment_size + IPG/EIPG + Preamble (in nanosecond), so that the packet can be preempted before the hold window starts.

76.10 Descriptors

76.10.1 Descriptor overview

In the Ethernet peripheral, the DMA transfers data based on a linked list of descriptors. The application creates the descriptors in the system memory (SRAM). The following two types of descriptors are supported:

- **Normal descriptors**

  The normal descriptors are used for packet data and to provide control information applicable to the packets to be transmitted.

- **Context descriptors**

  The context descriptors are used to provide control information applicable to the packet to be transmitted.

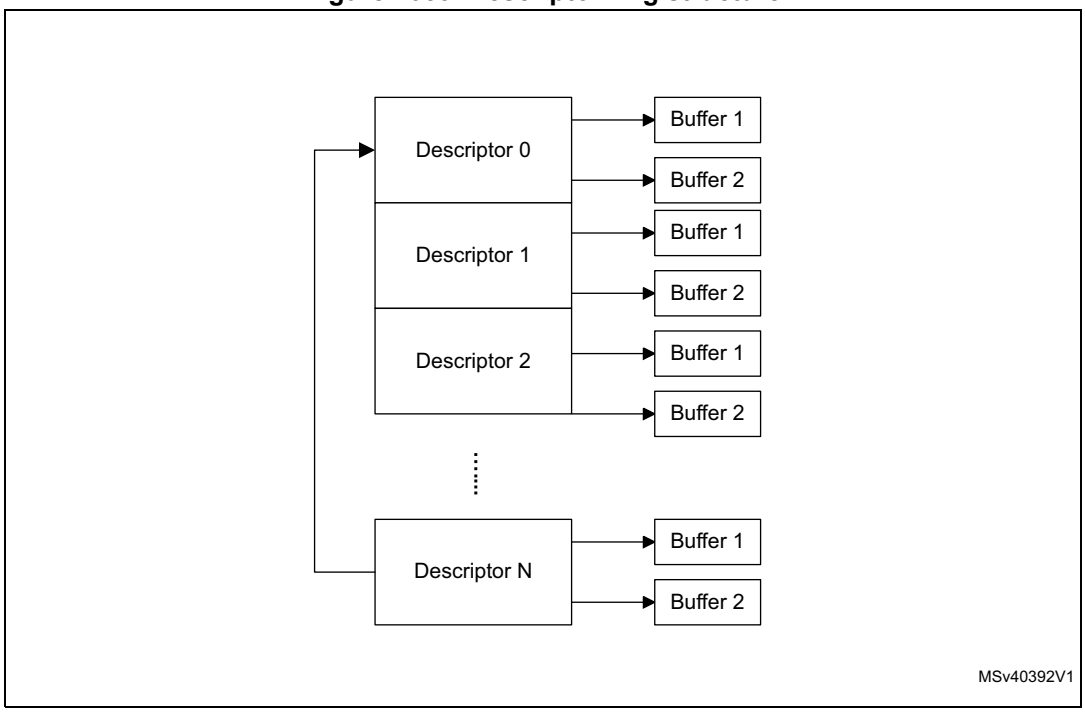
Each normal descriptor contains two buffers and two address pointers. These buffers enable the adapter port to be compatible with various types of memory management schemes.

There is no limit to the number of descriptors that can be used for a single packet.

76.10.2 Descriptor structure

The Ethernet peripheral supports the ring structure for DMA descriptors.

Figure 1059. Descriptor ring structure



In a ring structure, descriptors are separated by the 64-bit word number programmed in the DSL field of the *Channel x control register (ETH_DMACxCR)*). The application needs to program the total ring length, that is the total number of descriptors in ring span, in the following registers of a DMA channel:

- *Channel x Tx descriptor ring length register (ETH_DMACxTXRLR)*)
- *Channel x Rx descriptor ring length register (ETH_DMACxRXRLR)*)

The *Channel x Tx descriptor tail pointer register (ETH_DMACxTXDTPR)* or *Channel x Rx descriptor tail pointer register (ETH_DMACxRXDTPR)* contains the pointer to the descriptor address (*N*). The base address and the current descriptor pointer decide the address of the current descriptor that the DMA can process. The descriptors up to one location less than the one indicated by the descriptor tail pointer (*N* – 1) are owned by the DMA. The DMA continues to process the descriptors until the following condition occurs:

```
Current Descriptor Pointer == Descriptor Tail Pointer;
```

The DMA enters the Suspend state when this condition occurs. The application must perform a write operation to the descriptor tail pointer register and update the tail pointer so that the following condition is met:

```
Descriptor Tail Pointer > Current Descriptor Pointer;
```

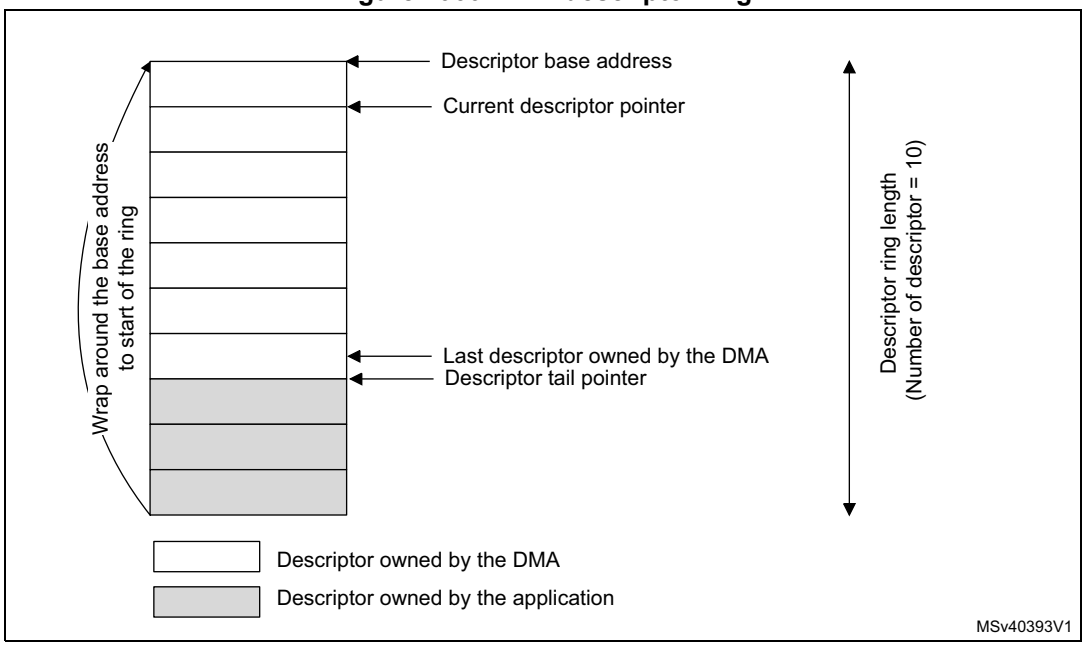
Note: When the application sets the descriptors into the end-of-descriptor ring, continues setting from the start of the descriptor list and updating the descriptor tail pointer, the above condition is not satisfied. The DMA detects such a case, processes the descriptors until the

end of the ring and then continues processing from the start of the descriptor list until the current descriptor pointer is equal to the descriptor tail pointer.

When the application programs the descriptor tail pointer beyond the end of the descriptor ring, the DMA interprets this as the availability of one more complete ring of descriptors, and after processing the last descriptor in the ring continues to process descriptors from the start of the descriptor list. If this is not intended, the application must program the descriptor tail pointer to the start of the descriptor list after setting the last descriptor in the ring.

The DMA automatically wraps around the base address when the end of ring is reached, as shown in *Figure 1060: DMA descriptor ring*.

Figure 1060. DMA descriptor ring



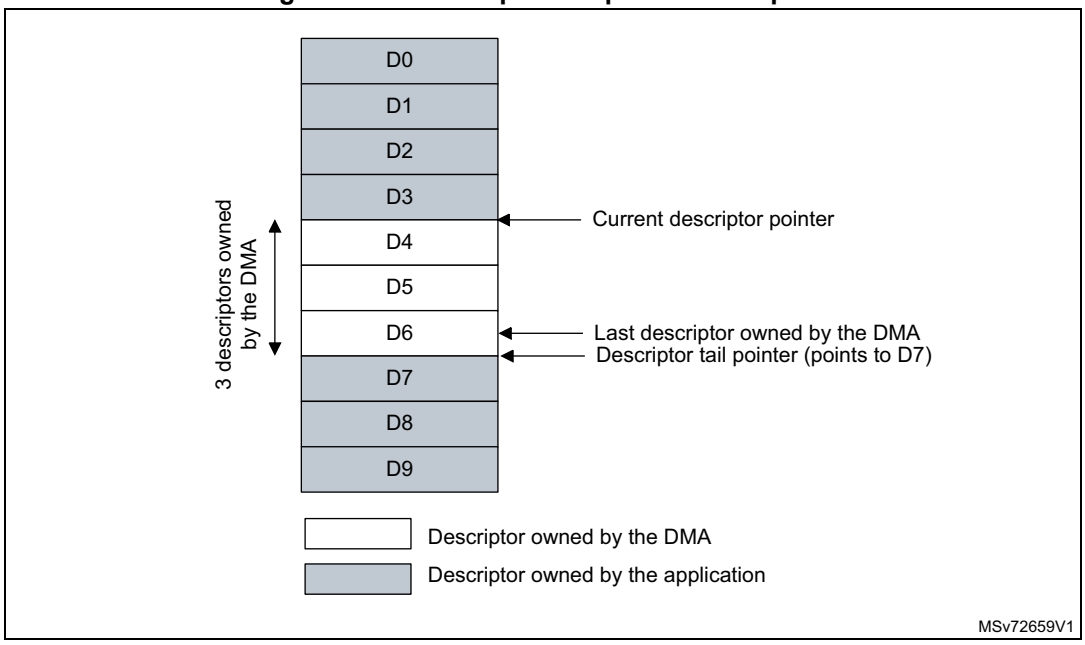
For descriptors owned by the application, the OWN bit of DES3 is reset to 0.

For descriptors owned by the DMA, the OWN bit is set to 1.

At the beginning, if the application has only one descriptor, it sets the last descriptor address (tail pointer) to Descriptor Base Address + 1. The DMA then processes the first descriptor and waits for the application to increment the tail pointer.

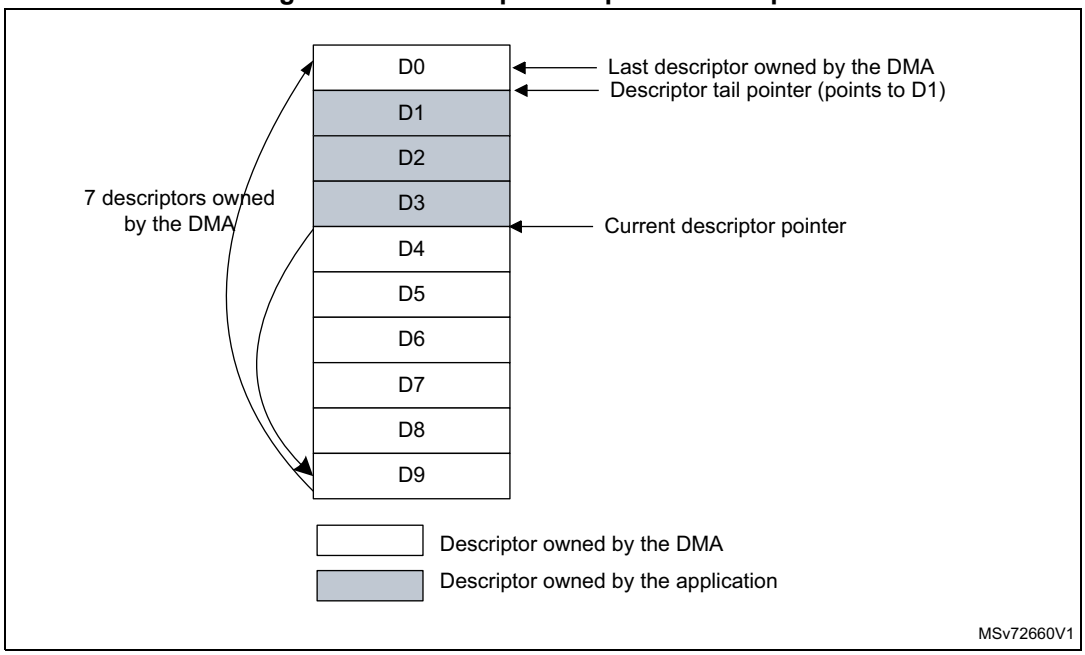
Descriptor tail pointer handling examples

Figure 1061. Descriptor tail pointer example 1



When the current descriptor pointer points to descriptor 4 (D4) and the descriptor tail pointer points to descriptor 7 (D7), the last descriptor owned by the DMA is descriptor 6 (D6) and there are three descriptors (D4, D5, D6) available for the DMA, as shown in *Figure 1061: Descriptor tail pointer example 1*.

Figure 1062. Descriptor tail pointer example 2



As shown in *Figure 1062: Descriptor tail pointer example 2*, the application frees four descriptors (D7, D8, D9 and D0) and updates the descriptor tail pointer to point to descriptor

1 (D1). The last descriptor owned by the DMA is descriptor 0 (D0). The descriptors referenced between the current descriptor pointer (D4) and the descriptor tail pointer (D1) are available to the DMA.

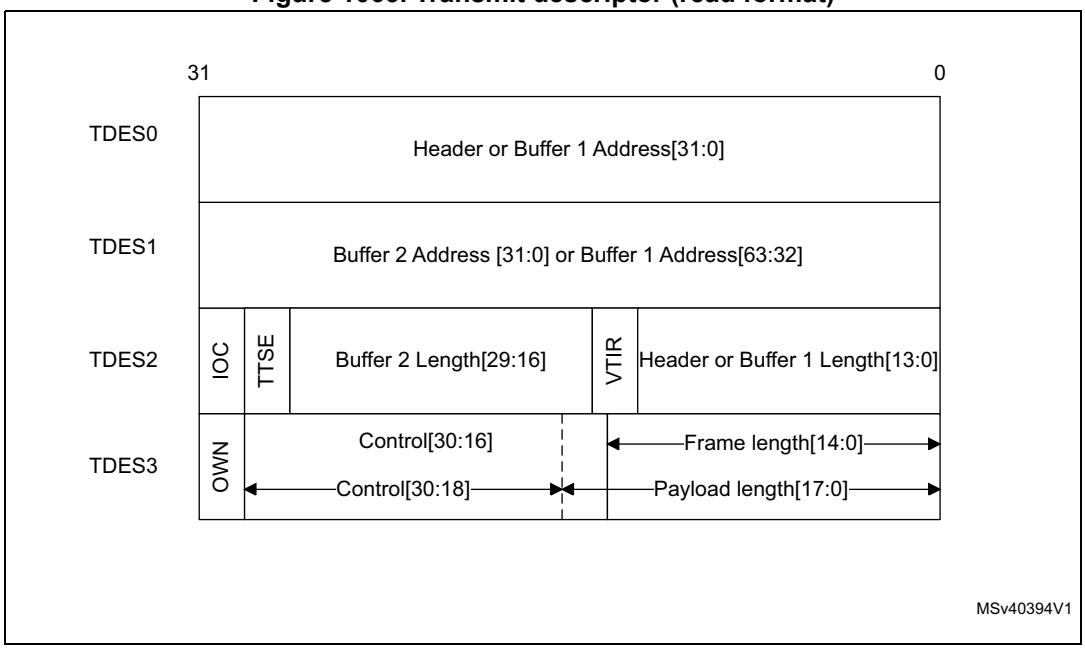
76.10.3 Transmit descriptor

The Ethernet peripheral DMA requires at least one descriptor for a transmit packet. In addition to two buffers, two byte-count buffers, and two address pointers, the transmit descriptor features control fields which can be used to manage the MAC operation on per-transmit packet basis. The Transmit normal descriptor has the following two formats: Read format and Write-back format.

Transmit normal descriptor (read format)

Figure 1063 shows the Read format for Transmit normal descriptor. *Table 821* to *Table 824* provide a detailed description of all Transmit normal descriptors (read format).

Figure 1063. Transmit descriptor (read format)



- TDES0 normal descriptor (read format)

Table 821. TDES0 normal descriptor (read format)

Bit	Name	Description
31:0	BUF1AP	**Buffer 1 Address Pointer or TSO Header Address Pointer** These bits indicate either the physical address of Buffer 1 or the TSO Header Address pointer when the following bits are set: – TSE bit of TDES3 – FD bit of TDES3

- TDES1 normal descriptor (read format)

Table 822. TDES1 normal descriptor (read format)

Bit	Name	Description
31:0	BUF2AP	**Buffer 2 or Buffer 1 Address Pointer:** These bits indicate the physical address of Buffer 2 when a descriptor ring structure is used. There is no limitation to the buffer address alignment.

- TDES2 normal descriptor (read format)

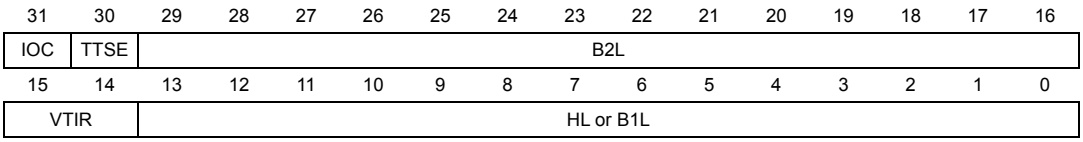


Table 823. TDES2 normal descriptor (read format)

Bits	Name	Description
31	IOC	**Interrupt on completion:** This bit sets the TI bit in the *Channel x status register (ETH_DMACxSR)*) when the present packet transmission is complete.
30	TTSE	**Transmit Timestamp Enable** This bit enables the IEEE1588 timestamping for Transmit packet referenced by the descriptor.
29:16	B2L	**Buffer 2 Length** The driver sets this field. When set, this field indicates Buffer 2 length.
15:14	VTIR	**VLAN Tag Insertion or Replacement:** These bits request the MAC to perform VLAN tagging or untagging before transmitting the packets. The application must set the CRC Pad Control bits appropriately when VLAN tag insertion, replacement, or deletion is enabled for the packet. The values of these bits are as follows: 00: Do not add a VLAN tag. 01: Remove the VLAN tag from the packets before transmission. This option should be used only with the VLAN packets. 10: Insert a VLAN tag with the tag value programmed in the *VLAN inclusion register (ETH_MACVIR)* or context descriptor. 11: Replace the VLAN tag in packets with the tag value programmed in the *VLAN inclusion register (ETH_MACVIR)* or context descriptor. This option should be used only with the VLAN packets.

Table 823. TDES2 normal descriptor (read format) (continued)

Bits	Name	Description
13:0	HL or B1L	**Header length or buffer 1 length** For Header length, only bits [9:0] are taken into account. Bits 13 to 0 are applicable only to buffer 1 length. If the TCP Segmentation Offload feature is enabled through the TSE bit of TDES3, this field is equal to the header length. When the TSE bit is set in TDES3, the header length includes the length (expressed in bytes) from Ethernet Source address till the end of the TCP header. The maximum header length supported for TSO feature is 1023 bytes. If the TCP Segmentation Offload feature is not enabled, this field is equal to Buffer 1 length.

- TDES3 normal descriptor (read format)

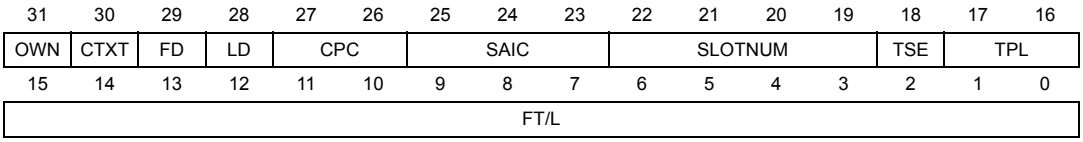


Table 824. TDES3 normal descriptor (read format)

Bits	Name	Description
31	OWN	**Own bit** 1: the DMA owns the descriptor. 0: the application owns the descriptor. The DMA clears this bit after it completes the transfer of data given in the associated buffer(s).
30	CTXT	**Context Type** This bit should be set to 0 for normal descriptor.
29	FD	**First Descriptor** When this bit is set, it indicates that the buffer contains the first segment of a packet.
28	LD	**Last Descriptor** When this bit is set, it indicates that the buffer contains the last segment of the packet. B1L or B2L field should have a non-zero value.

Table 824. TDES3 normal descriptor (read format) (continued)

Bits	Name	Description
27:26	CPC	**CRC Pad Control** This field controls the CRC and Pad Insertion for Tx packet. It is valid only when the first descriptor bit (TDES3[29]) is set. The values of bits[27:26] are the following: 00: CRC and Pad Insertion The MAC appends the cyclic redundancy check (CRC) at the end of the transmitted packets whose length greater than or equal to 60 bytes. The MAC automatically appends padding and CRC to a packet with length less than 60 bytes. 01: CRC Insertion (Disable Pad Insertion) The MAC appends the CRC at the end of the transmitted packet but it does not append padding. The application should ensure that the padding bytes are present in the packet being transferred from the Transmit buffer, that is, the packet being transferred from the Transmit Buffer is of length greater than or equal to 60 bytes. 10: Disable CRC Insertion The MAC does not append the CRC at the end of the transmitted packet. The application should ensure that the padding and CRC bytes are present in the packet being transferred from the Transmit Buffer. 11: CRC Replacement The MAC replaces the last four bytes of the transmitted packet with recalculated CRC bytes. The application should ensure that the padding and CRC bytes are present in the packet being transferred from the Transmit Buffer. When the TSE bit is set, the MAC ignores this field because the CRC and pad insertion is always done for segmentation.
25:23	SAIC	**SA Insertion Control** These bits request the MAC to add or replace the Source Address field in the Ethernet packet with the value given in the MAC Address 0 register. The application must appropriately set the CRC Pad Control bits when SA Insertion Control is enabled for the packet. Bit 25 specifies the MAC Address Register (1 or 0) value that is used for Source Address insertion or replacement. The following list describes the values of Bits[24:23]: 00: Do not include the source address 01: Include or insert the source address. For reliable transmission, the application must provide frames without source addresses. 10: Replace the source address. For reliable transmission, the application must provide frames with source addresses. 11: Reserved These bits are valid when the First Segment control bit (TDES3 [29]) is set.

Table 824. TDES3 normal descriptor (read format) (continued)

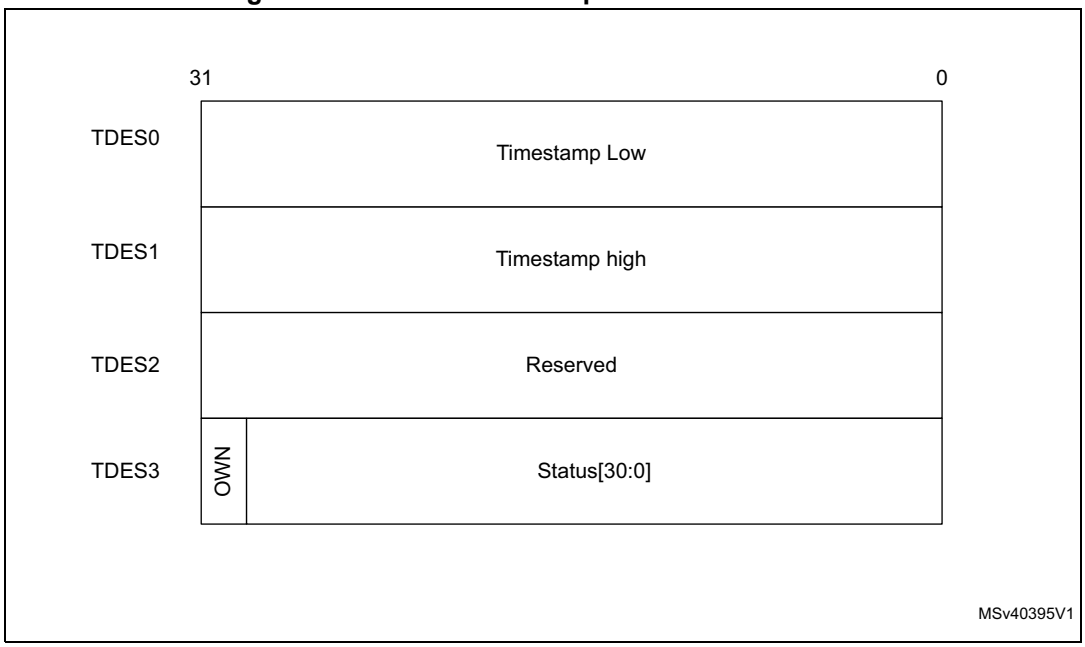
Bits	Name	Description
22:19	SLOTNUM or THL	**SLOTNUM**: Slot Number Control Bits in AV mode These bits indicate the slot interval in which the data should be fetched from the corresponding buffers addressed by TDES0 or TDES1. When the Transmit descriptor is fetched, the DMA compares the slot number value in this field with the slot interval maintained in the RSN field of *Channel x slot function control status register (ETH_DMACxSFCSR)*. It fetches the data from the buffers only if a value matches. These bits are valid only for the AV channels. **THL**: TCP Header Length If the TSE bit is set, this field contains the length of the TCP/UDP header. The minimum value of this field must be 5 for TCP header. THL value must be equal to 2 for UDP header. This field is valid only for the first descriptor
18	TSE	**TCP Segmentation Enable** When this bit is set, the DMA performs the TCP/UDP segmentation for a packet. This bit is valid only if the FD bit is set.
17:16	CIC/TPL	**Checksum Insertion Control or TCP Payload Length** These bits control the checksum calculation and insertion. They can take the following values: 00: Checksum insertion disabled. 01: Only IP header checksum calculation and insertion are enabled. 10: IP header checksum and payload checksum calculation and insertion are enabled, but pseudo-header checksum is not calculated in hardware. 11: IP header checksum and payload checksum calculation and insertion are enabled, and pseudo-header checksum is calculated in hardware. This field is valid when the TSE bit is reset.When the TSE bit is set, it contains the upper bits [17:16] of the TCP Payload length. This allows the TCP packet length field to be spanned across TDES3[17:0] to provide 256 Kbyte packet length support.
15	TPL	**Reserved or TCP Payload Length** When the TSE bit is reset, this bit is reserved. When the TSE bit is set, this is bit 15 of the TCP payload length [17:0].
14:0	FL/TPL	**Reserved or TCP Payload Length** When the TSE bit is set, this field is equal to the lower 15 bits of the TCP payload length. This length does not include Ethernet header or TCP/UDP/IP header length. When the TSE bit is reset, this bit is reserved.

Transmit normal descriptor (write-back format)

The write-back format is applicable only for the last descriptor of the corresponding packet. The LD bit (TDES3[28]) is set in the descriptor where the DMA writes back the status and timestamp information for the corresponding Transmit packet.

Figure 1064 shows the write-back format for Transmit normal descriptors. *Table 825* to *Table 828* provide a detailed description of all Transmit Normal descriptors (Write-Back Format).

Figure 1064. Transmit descriptor write-back format



- TDES0 normal descriptor (write-back format)

Table 825. TDES0 normal descriptor (write-back format)(1)

Bit	Name	Description
31:0	TTSL	**Transmit Packet Timestamp Low** The DMA updates this field with least significant 32 bits of the timestamp captured for the corresponding Transmit packet. The DMA writes the timestamp only if TTSE bit of TDES2 is set in the first descriptor of the packet. This field holds the timestamp only if the Last Segment bit (LS) in the descriptor is set and the Timestamp status (TTSS) bit is set.

1. This format is only applicable to the last descriptor of a packet.

- TDES1 normal descriptor (write-back format)

Table 826. TDES1 normal descriptor (write-back format)(1)

Bit	Name	Description
31:0	TTSH	**Transmit Packet Timestamp High** The DMA updates this field with the most significant 32 bits of the timestamp captured for corresponding receive packet. The DMA writes the timestamp only if the TTSE bit of TDES2 is set in the first descriptor of the packet. This field has the timestamp only if the Last Segment bit (LS) in the descriptor is set and Timestamp status (TTSS) bit is set.

1. This format is only applicable to the last descriptor of a packet.

- TDES2 normal descriptor (write-back format)

Table 827. TDES2 normal descriptor (write-back format)(1)

Bit	Description
31:0	Reserved

1. This format is only applicable to the last descriptor of a packet.

- TDES3 normal descriptor (write-back format)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OWN	CTXT	FD	LD	Reserved										TTSS	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ES	JT	FF	PCE	LoC	NC	LC	EC	CC				ED	UF	DB	IHE

Table 828. TDES3 normal descriptor (write-back format)(1)

Bit	Name	Description
31	OWN	**Own bit** When this bit is set, it indicates that the DMA owns the descriptor. The DMA clears this bit when it completes the packet transmission. After the write-back is complete, this bit is set to 0.
30	CTXT	**Context Type** This bit should be set to 0 for normal descriptors.
29	FD	**First Descriptor** This bit indicates that the buffer contains the first segment of a packet.
28	LD	**Last Descriptor** This bit is set 1 for last descriptor of a packet. The DMA writes the status fields only in the last descriptor of the packet.
27:24	Reserved	
23	DE	**Descriptor Error** When this bit is set, it indicates that the descriptor content is incorrect. The DMA sets this bit during write-back while closing the descriptor. Descriptor errors can be: – Incorrect sequence from the context descriptor, such as a location after the first descriptor for a packet. – All 1s. – CTXT is set to 1 along with LD or FD bits set to 1. *Note: When a Descriptor error occurs due to All 1s or CTXT, LD, and FD bits set to 1, the Transmit DMA closes the transmit descriptor with DE and LD bits set to 1. When IOC bit in TDES2 of corresponding first descriptor is set to 1, Transmit DMA sets the TI bit in the Channel x status register (ETH_DMACxSR).* *Based on CTXT, LD, and FD bits of the transmit descriptor, the subsequent descriptor might be considered as the First Descriptor (even if FD bit is not set) and partial packet is sent.*
22:18	Reserved	

Table 828. TDES3 normal descriptor (write-back format)(1) (continued)

Bit	Name	Description
17	TTSS	**Tx Timestamp Status** This status bit indicates that a timestamp has been captured for the corresponding transmit packet. When this bit is set, TDES0 and TDES1 have timestamp values that were captured for the Transmit packet. This field is valid only when the Last Segment control bit (TDES3 [28]) in a descriptor is set.
16	Reserved	
15	ES	**Error Summary** This bit indicates the logical OR of the following bits: TDES3[0]: IP Header Error TDES3[14]: Jabber timeout TDES3[13]: Packet Flush TDES3[12]: Payload Checksum Error TDES3[11]: Loss of Carrier TDES3[10]: No Carrier TDES3[9]: Late Collision TDES3[8]: Excessive Collision TDES3[3]: Excessive Deferral TDES3[2]: Underflow Error
14	JT	**Jabber timeout** This bit indicates that the MAC transmitter has experienced a jabber timeout. This bit is set only when the JD bit of the *Operating mode configuration register (ETH_MACCR)* is not set.
13	FF	**Packet Flushed** This bit indicates that the DMA or MTL flushed the packet because of a software flush command given by the CPU.
12	PCE	**Payload Checksum Error** This bit indicates that the Checksum Offload engine had a failure and did not insert any checksum into the encapsulated TCP, UDP, or ICMP payload. This failure can be either caused by insufficient bytes, as indicated by the Payload Length field of the IP Header, or by the MTL starting to forward the packet to the MAC transmitter in Store-and-Forward mode without the checksum having been calculated yet. This second error condition only occurs when the Transmit FIFO depth is less than the length of the Ethernet packet being transmitted to avoid deadlock, the MTL starts forwarding the packet when the FIFO is full, even in the store-and-forward mode. This error can also occur when a Bus error is detected during packet transfer.
11	LoC	**Loss of Carrier** This bit indicates that Loss of Carrier occurred during packet transmission (that is, the ETH_CRS signal was inactive for one or more transmit clock periods during packet transmission). This is valid only for the packets transmitted without collision and when the MAC operates in the Half-duplex mode.

Table 828. TDES3 normal descriptor (write-back format)(1) (continued)

Bit	Name	Description
10	NC	**No Carrier** This bit indicates that the carrier sense signal form the PHY was not asserted during transmission.
9	LC	**Late Collision** This bit indicates that packet transmission was aborted because a collision occurred after the collision window (64 byte times including Preamble in MII mode and 512 byte times including Preamble and Carrier Extension in GMII mode). This bit is not valid if Underflow Error is set.
8	EC	**Excessive Collision** This bit indicates that the transmission was aborted after 16 successive collisions while attempting to transmit the current packet. If the DR bit is set in the *Operating mode configuration register (ETH_MACCR)*, this bit is set after first collision and the transmission of the packet is aborted.
7:4	CC	**Collision Count** This 4-bit counter value indicates the number of collisions occurred before the packet was transmitted. The count is not valid when the EC bit is set.
3	ED	**Excessive Deferral** This bit indicates that the transmission ended because of excessive deferral of over 24,288 bit times (155,680 bits times in 1000-Mbps mode or Jumbo Packet enabled mode) if DC bit is set in the *Operating mode configuration register (ETH_MACCR)*. When TBS is enabled in Full duplex mode and this bit is set, it indicates that the frame has been dropped after the expiry time has elapsed.
2	UF	**Underflow Error** This bit indicates that the MAC aborted the packet because the data arrived late from the system memory. The underflow error can occur because of either of the following conditions: The DMA encountered an empty Transmit Buffer while transmitting the packet The application filled the MTL Tx FIFO slower than the MAC transmit rate The transmission process enters the Suspend state and sets the underflow bit corresponding to a queue in the ETH_MTLISR register.

Table 828. TDES3 normal descriptor (write-back format)(1) (continued)

Bit	Name	Description
1	DB	**Deferred Bit** This bit indicates that the MAC deferred before transmitting because of presence of carrier. This bit is valid only in the Half-duplex mode.
0	IHE	**IP Header Error** When IP Header Error is set, this bit indicates that the Checksum Offload engine detected an IP header error. If COE detects an IP header error, it still inserts an IPv4 header checksum if the Ethernet Type field indicates an IPv4 payload. In Full-duplex mode, when EST/Qbv is enabled and this bit is set, it indicates the frame drop status due to Frame Size error or Schedule Error.

1. This format is only applicable to the last descriptor of a packet.

Transmit context descriptor

The Transmit context descriptor can be provided any time before a packet descriptor. The context is valid for the current packet and subsequent packets. The context descriptor is used to provide the timestamps for one-step timestamp correction, and VLAN Tag ID for VLAN insertion feature. Write-back is only done on a context descriptor to reset the OWN bit.

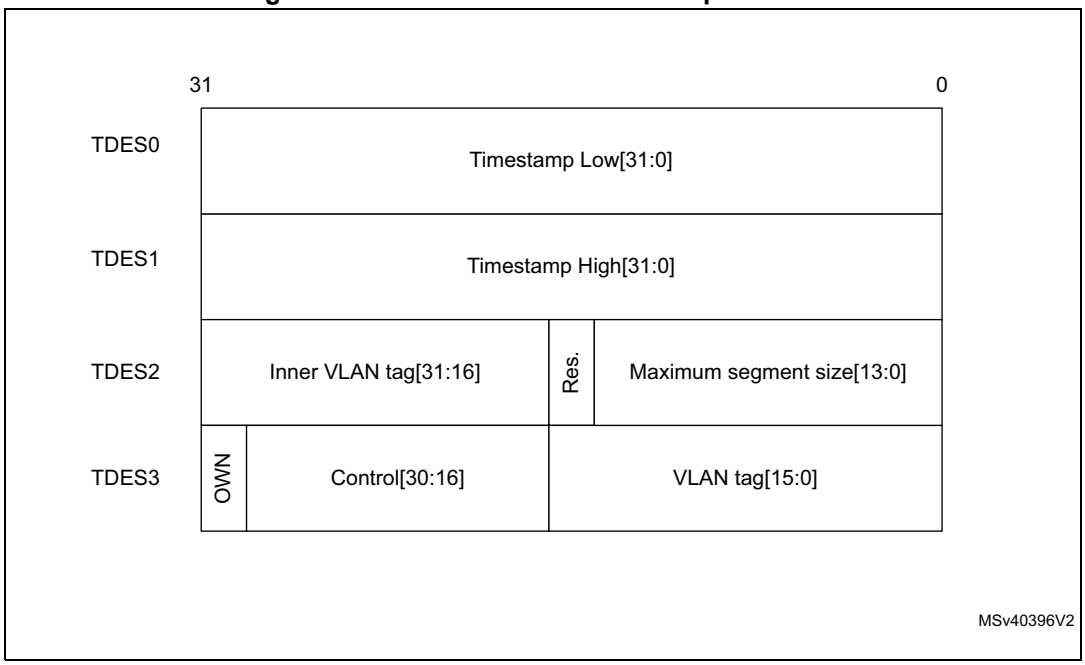
Note: The VLAN tag IDs and MSS values, which are provided by the application in a context descriptor with their corresponding Valid bits set, are stored internally by the DMA.

When the outer or inner VLAN tag is provided with the Valid bit set, the DMA always passes the last valid VLAN tag to the MTL. The application cannot invalidate the valid VLAN tag stored by the DMA. The VLAN tag is inserted or replaced based on the control inputs provided for the packet.

The Inner VLAN Tag Control input is used only for the packet that immediately follows the context descriptor. The application must provide a context descriptor before the normal descriptor of each packet for which the DMA should use the inner VLAN tag control input.

Figure 1065 shows the format for transmit context descriptors. *Table 829* to *Table 832* provide a detailed description of all transmit context descriptors.

Figure 1065. Transmit context descriptor format



- TDES0 context descriptor (read format)

Table 829. TDES0 context descriptor

Bit	Name	Description
31:0	TTSL	**Transmit Packet Timestamp Low** For one-step correction, the driver can provide the lower 32 bits of timestamp in this descriptor word. The DMA uses this value as the low word for doing one-step timestamp correction. This field is valid only if the OSTC and TCMSSV bits of TDES3 context descriptor are set.

- TDES1 context descriptor (read format)

Table 830. TDES1 context descriptor

Bit	Name	Description
31:0	TTSH	**Transmit Packet Timestamp High** For one-step correction, the driver can provide the upper 32 bits of timestamp in this descriptor. The DMA uses this value as the high word for doing one-step timestamp correction. This field is valid only if the OSTC and TCMSSV bits of TDES3 context descriptor are set.

- TDES2 context descriptor (read format)

Table 831. TDES2 context descriptor

Bit	Name	Description
31:16	IVT	**Inner VLAN Tag** When the IVLTV bit of TDES3 context descriptor is set and the TCMSSV and OSTC bits of TDES3 context descriptor are reset, TDES2[31:16] contains the inner VLAN Tag to be inserted in the subsequent Transmit packets.
15:14	Reserved	
13:0	MSS	**Maximum Segment Size** This segment size is used while segmenting the TCP/IP payload. This field is valid only if the TCMSSV bit of TDES3 context descriptor is set and the OSTC bit of the TDES3 context descriptor is reset.

- TDES3 context descriptor (read format)

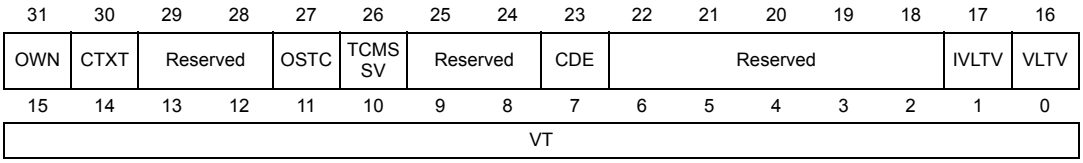


Table 832. TDES3 context descriptor

Bit	Name	Description
31	OWN	**Own bit** 1: the DMA owns the descriptor. 0: the application owns the descriptor. The DMA clears this bit immediately after a read operation.
30	CTXT	**Context Type** This bit should be set to 1 for context descriptor.
29:28	Reserved	
27	OSTC	**One-Step Timestamp Correction Enable** When this bit is set, the DMA performs a one-step timestamp correction with reference to the timestamp values provided in TDES0 and TDES1.
26	TCMSSV	**One-Step Timestamp Correction Input or MSS Valid** When this bit and the OSTC bit are set, it indicates that the Timestamp Correction input provided in TDES0 and TDES1 is valid. When the OSTC bit is reset and this bit and the TSE bit of TDES3 are set in subsequent normal descriptor, it indicates that the MSS input in TDES2 is valid.
25:24	Reserved	

Table 832. TDES3 context descriptor (continued)

Bit	Name	Description
23	CDE	**Context Descriptor Error** When this bit is set, it indicates that the context descriptor is incorrect. The DMA sets this bit during write-back while closing the context descriptor. The Context Descriptor errors can be: – Incorrect sequence from the context descriptor. For example, a location before the first descriptor for a packet. – All 1s. – CD, LD, and FD bits set to 1. *Note: When a Context Descriptor error occurs due to All 1s or CTXT, LD, and FD bits set to 1, the Transmit DMA closes the transmit descriptor with DE and LD bits set to 1. When IOC bit in TDES2 of corresponding first descriptor is set to 1, Transmit DMA sets the TI bit in the Channel x status register (ETH_DMACxSR).* *Based on CTXT, LD, and FD bits of the transmit descriptor, the subsequent descriptor might be considered as the First Descriptor (even if FD bit is not set) and partial packet is sent.* *This error is categorized as an abnormal event; recovery is only by issuing a software reset (DMA stopping-reconfiguring-restarting recovery mechanism is not supported)*
22:20	Reserved	
19:18	IVTIR	Inner VLAN Tag Insert or Replace When these bits are set, they request the MAC to perform Inner VLAN tagging or untagging before transmitting the packets. If the packet is modified for VLAN tags, the MAC automatically recalculates and replaces the CRC bytes. This bitfield has the following values: 00: Do not add the inner VLAN tag. 01: Remove the inner VLAN tag from the packets before transmission. This option should be used only with the VLAN frames. 10: Insert an inner VLAN tag with the tag value programmed in the *Inner VLAN inclusion register (ETH_MACIVIR)* or context descriptor. 11: Replace the inner VLAN tag in packets with the tag value programmed in the *Inner VLAN inclusion register (ETH_MACIVIR)* or context descriptor. This option should be used only with the VLAN frames.
17	IVLTV	Inner VLAN Tag Valid When this bit is set, it indicates that the IVT field of TDES2 is valid.
16	VLTV	VLAN Tag Valid When this bit is set, it indicates that the VT field of TDES3 is valid.
15:0	VT	VLAN Tag This field contains the VLAN Tag to be inserted or replaced in the packet. This field is used as VLAN Tag only when the VLTI bit of the *VLAN inclusion register (ETH_MACVIR)* is reset.

76.10.4 Receive descriptor

The DMA in the Ethernet peripheral attempts to read a descriptor only if the Tail pointer is different from the Base pointer or current pointer. It is recommended to have a descriptor ring with a length that can accommodate at least two complete packets received by the MAC; otherwise, the performance of the DMA is greatly impacted because of the unavailability of the descriptors. In such a situation, the MTL RxFIFO becomes full and starts dropping packets.

The following receive descriptors are present:

- Normal descriptors with read and write-back formats
- Context descriptors

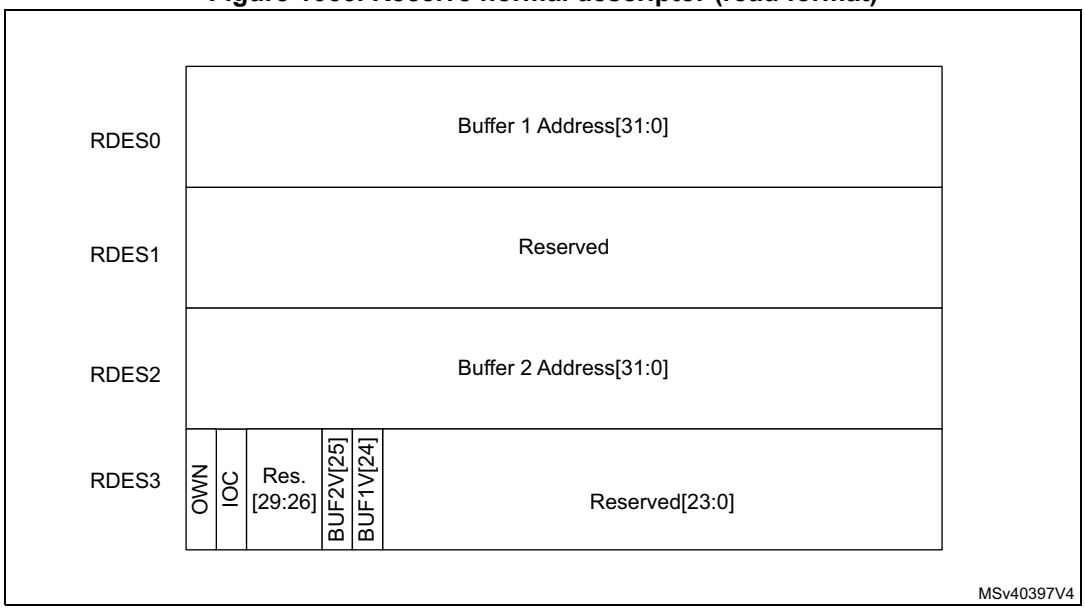
All received descriptors are prepared by the software and given to the DMA as “normal” descriptors (see *Figure 1066: Receive normal descriptor (read format)* for a description of their content). The DMA reads this descriptor and, after transferring a received packet (or part of it) to the buffers indicated by the descriptor, the Rx DMA closes the descriptor with the corresponding packet status. The status format is given in *Figure 1067: Receive normal descriptor (write-back format)*.

For some packets, the normal descriptor bits are not sufficient to write the complete status. For such packets, the Rx DMA writes the extended status to the next descriptor (without processing or using the Buffers pointers embedded in that descriptor). The format and content of this write-back descriptor is described in *Figure 1068: Receive context descriptor*.

Receive normal descriptor (read format)

Figure 1066 shows the read format for receive normal descriptors. *Table 833* to *Table 836* provide a detailed description of all receive normal descriptors (read format).

Figure 1066. Receive normal descriptor (read format)



Note: In the receive descriptor (read format), if the Buffer Address field contains only 0s, the MAC does not transfer data to this buffer and skips to the next buffer or next descriptor.

- RDES0 normal descriptor (read format)

Table 833. RDES0 normal descriptor (read format)

Bit	Name	Description
31:0	BUF1AP	**Buffer 1 Address Pointer** These bits indicate the physical address of Buffer 1. The application can program a byte-aligned address for this buffer, which means that the LS bits of this field can be non-zero. However, while transferring the start of packet, the DMA performs a write operation with RDES2[2:0]=0 in case of 64-/128-bit configuration) as zero. However, the packet data is shifted by the actual offset as given in the buffer address pointer. If the address pointer points to a buffer where the middle or last part of the packet is stored, the DMA ignores the offset address and writes to the full location as indicated by the data-width.

- RDES1 normal descriptor (read format)

Table 834. RDES1 normal descriptor (read format)

Bit	Name	Description
31:0	Reserved	Field reserved.

- RDES2 normal descriptor (read format)

Table 835. RDES2 normal descriptor (read format)

Bit	Name	Description
31:0	BUF2AP	**Buffer 2 Address Pointer** These bits indicate Buffer 2 physical address. The RxDMA uses the LS bits of the pointer address only while transferring the start bytes of a packet. If the BUF2AP is giving the address of a buffer in which the middle or last part of a packet is stored, the DMA ignores RDES2[2:0]=0 and writes to the complete location.

- RDES3 normal descriptor (read format)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OWN	IOC	Reserved				BUF2V	BUF1V	Reserved							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved															

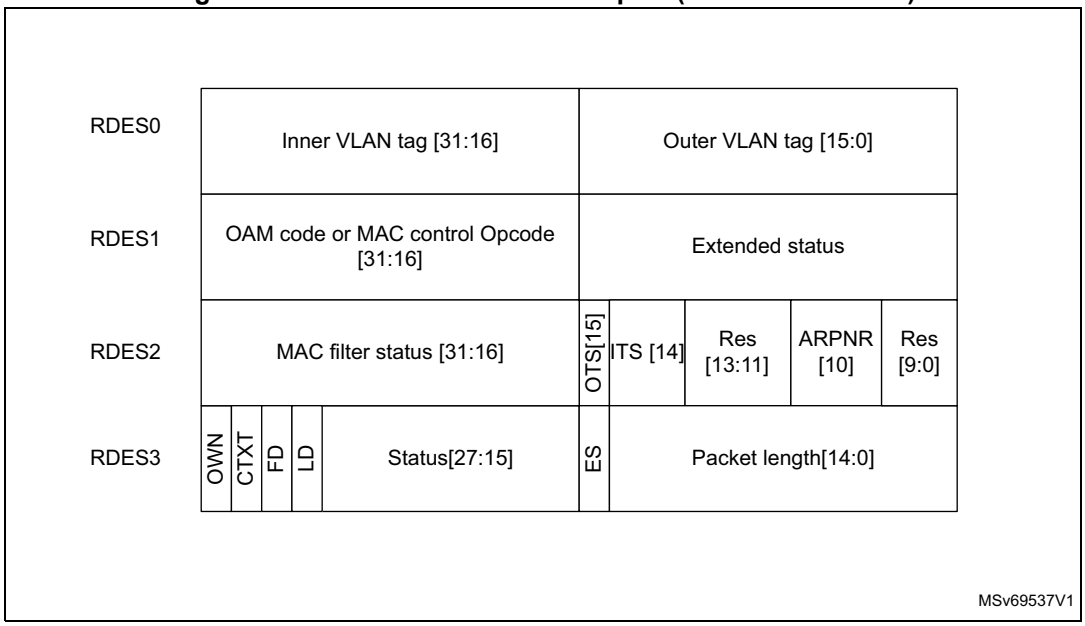
Table 836. RDES3 normal descriptor (read format)

Bit	Name	Description
31	OWN	**Own bit** When this bit is set, it indicates that the DMA owns the descriptor. When this bit is reset, it indicates that the application owns the descriptor. The DMA clears this bit when either of the following conditions is true: – The DMA completes the packet reception – The buffers associated with the descriptor are full
30	IOC	**Interrupt Enabled on Completion** When this bit is set, an interrupt is issued to the application when the DMA closes this descriptor.
29:26	Reserved	
25	BUF2V	**Buffer 2 Address Valid** When this bit is set, it indicates to the DMA that the buffer 2 address specified in RDES2 is valid. The application must set this bit so that the DMA can use the address to which the Buffer 2 address in RDES0 is pointing, to write received packet data.
24	BUF1V	**Buffer 1 Address Valid** When set, this indicates to the DMA that the buffer 1 address specified in RDES0 is valid. The application must set this value if the address to which Buffer 1 address points in RDES0, can be used by the DMA to write received packet data.
23:0	Reserved	

Receive normal descriptor (write-back format)

Figure 1067 shows the write-back format for receive normal descriptors. *Table 837* to *Table 840* provide a detailed description of all receive normal descriptors (write-back format).

Figure 1067. Receive normal descriptor (write-back format)



- RDES0 normal descriptor (write-back format)

Table 837. RDES0 normal descriptor (write-back format)

Bit	Name	Description
31:16	IVT	**Inner VLAN Tag** This field contains the Inner VLAN tag of the received packet if the RS0V bit of RDES3 is set. This is valid only when Double VLAN tag processing and VLAN tag stripping are enabled.
15:0	OVT	**Outer VLAN Tag** This field contains the Outer VLAN tag of the received packet if the RS0V bit of RDES3 is set.

- RDES1 normal descriptor (write-back format)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OPC															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TD	TSA	PV	PFT	PMT				IPCE	IPCB	IPV6	IPV4	IPHE	PT		

Table 838. RDES1 normal descriptor (write-back format)(1)

Bit	Name	Description
31:16	OPC	**OAM Subtype Code, or MAC Control Packet opcode** OAM Subtype Code If bits[18:16] of RDES3 are set to 111, this field contains the OAM subtype and code fields. MAC Control Packet opcode If bits[18:16] of RDES3 are set to 110, this field contains the MAC Control packet opcode field.
15	TD	**Timestamp Dropped** This bit indicates that the timestamp was captured for this packet but got dropped in the MTL Rx FIFO because of overflow.
14	TSA	**Timestamp Available** When Timestamp is present, this bit indicates that the timestamp value is available in a context descriptor word 2 (RDES2) and word 1(RDES1). This is valid only when the Last Descriptor bit (RDES3 [28]) is set. The context descriptor is written in the next descriptor just after the last normal descriptor for a packet.
13	PV	**PTP Version** 1: Received PTP message in IEEE 1588 version 2 format 0: Received PTP message in IEEE 1588 version 1 format
12	PFT	**PTP Packet Type** This bit indicates that the PTP message is sent directly over Ethernet.
11:8	PMT	**PTP Message Type** These bits are encoded to give the type of the message received: 0000: No PTP message received 0001: SYNC (all clock types) 0010: Follow_Up (all clock types) 0011: Delay_Req (all clock types) 0100: Delay_Resp (all clock types) 0101: Pdelay_Req (in peer-to-peer transparent clock) 0110: Pdelay_Resp (in peer-to-peer transparent clock) 0111: Pdelay_Resp_Follow_Up (in peer-to-peer transparent clock) 1000: Announce 1001: Management 1010: Signaling 1011–1110: Reserved 1111: PTP packet with Reserved message type
7	IPCE	**IP Payload Error** When this bit is set, it indicates either of the following: – The 16-bit IP payload checksum (that is, the TCP, UDP, or ICMP checksum) calculated by the MAC does not match the corresponding checksum field in the received segment. – The TCP, UDP, or ICMP segment length does not match the payload length value in the IP Header field. – The TCP, UDP, or ICMP segment length is less than minimum allowed segment length for TCP, UDP, or ICMP. Bit 15 (ES) of RDES3 is not set when this bit is set.

Table 838. RDES1 normal descriptor (write-back format)(1) (continued)

Bit	Name	Description
6	IPCB	**IP Checksum Bypassed** This bit indicates that the checksum offload engine is bypassed.
5	IPV6	**IPv6 header Present** This bit indicates that an IPV6 header is detected.
4	IPV4	**IPV4 Header Present** This bit indicates that an IPV4 header is detected.
3	IPHE	**IP Header Error** – When this bit is set, it indicates either of the following: – The 16-bit IPv4 header checksum calculated by the MAC does not match the received checksum bytes. – The IP datagram version is not consistent with the Ethernet Type value. – Ethernet packet does not have the expected number of IP header bytes. This bit is valid when either bit 5 or bit 4 is set.
2:0	PT	**Payload Type** These bits indicate the type of payload encapsulated in the IP datagram processed by the receive checksum offload engine (COE): 000: Unknown type or IP/AV payload not processed 001: UDP 010: TCP 011: ICMP 100: IGMP if IPV4 Header Present bit is set 110: AV Tagged Data Packet 111: AV Tagged Control Packet 101: AV Untagged Control Packet Others: reserved. If the COE does not process the payload of an IP datagram because there is an IP header error or fragmented IP, it sets these bits to 3'b000.

1. The Status fields in write-back format are valid only for the last descriptor (RDES3[28] is set).

- RDES2 normal descriptor (write-back format)

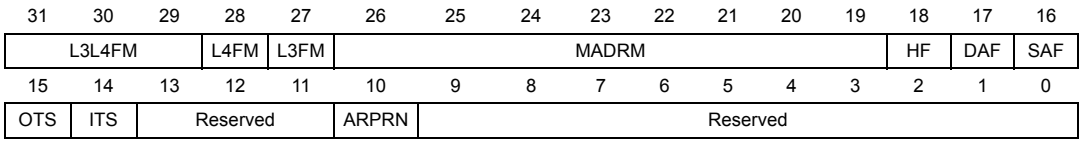


Table 839. RDES2 normal descriptor (write-back format)

Bit	Name	Description
31:29	L3L4FM	**Layer 3 and Layer 4 Filter Number Matched** These bits indicate the number of the Layer 3 and Layer 4 Filter that matched the received packet: – 000: Filter 0 – 001: Filter 1 – 010: Filter 2 – 011: Filter 3 – 100: Filter 4 – 101: Filter 5 – 110: Filter 6 – 111: Filter 7 This field is valid only when bit 28 or bit 27 is set high. When more than one filter matches, these bits give the number of lowest filter.
28	L4FM	**Layer 4 Filter Match** When this bit is set, it indicates that the received packet matches one of the enabled Layer 4 Port Number fields. This status is given only when one of the following conditions is true: – Layer 3 fields are not enabled and all enabled Layer 4 fields match – All enabled Layer 3 and Layer 4 filter fields match When more than one filter matches, this bit gives the layer 4 filter status of filter indicated by bits[31:29].
27	L3FM	**Layer 3 Filter Match** When this bit is set, it indicates that the received packet matches one of the enabled Layer 3 IP Address fields. This status is given only when one of the following conditions is true: – All enabled Layer 3 fields match and all enabled Layer 4 fields are bypassed – All enabled filter fields match When more than one filter matches, this bit gives the layer 3 filter status of filter indicated by bits[31:29].
26:19	MADRM	**MAC Address Match or Hash Value** When the HF bit is reset, this field contains the MAC address register number that matched the Destination address of the received packet. This field is valid only if the DAF bit is reset. When the HF bit is set, this field contains the Hash value computed by the MAC. A packet passes the Hash filter when the bit corresponding to the Hash value is set in the Hash filter register.
18	HF	**Hash Filter Status** When this bit is set, it indicates that the packet passed the MAC address Hash filter. its[26:19] indicate the Hash value.
17	DAF	**Destination Address Filter Fail** When this bit is set, it indicates that the packet failed the DA Filter in the MAC.
16	SAF	**SA Address Filter Fail** When this bit is set, it indicates that the packet failed the SA Filter in the MAC.

Table 839. RDES2 normal descriptor (write-back format) (continued)

Bit	Name	Description
15	OTS	**Outer VLAN Tag Filter Status** This bit is valid for both Single and Double VLAN tagged frames.
14	ITS	**Inner VLAN Tag Filter Status** This bit is valid only for Double VLAN tagged frames, when Double VLAN Processing is enabled. For more information, see the VLAN Filter Status section.
13:11	Reserved	
10	ARPNR	**ARP Reply Not Generated** When this bit is set, it indicates that the MAC did not generate the ARP Reply for received ARP Request packet. This bit is set when the MAC is busy transmitting ARP reply to earlier ARP request (only one ARP request is processed at a time).
9:0	Reserved	

- RDES3 normal descriptor (write-back format)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OWN	CTXT	FD	LD	RS2V	RS1V	RS0V	CE	GP	RWT	OE	RE	DE	LT		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ES	PL														

Table 840. RDES3 normal descriptor (write-back format)

Bit	Name	Description
31	OWN	**Own bit** 1: The DMA owns the descriptor. 0: The application owns the descriptor. The DMA clears this bit when either of the following conditions is true: The DMA completes the packet reception The buffers associated with the descriptor are full
30	CTXT	**Receive Context Descriptor** When this bit is set, it indicates that the current descriptor is a context type descriptor. The DMA writes 0 to this bit for normal receive descriptor. When CTXT and FD bits are used together, {CTXT, FD} possible values are: 00: Intermediate Descriptor 01: First Descriptor 10: Reserved 11: Descriptor error (due to all 1s) *Note: When a Descriptor error occurs, the receive DMA closes the receive descriptor indicating a Descriptor error. This receive descriptor is skipped and the buffer addresses are not used to write the packet data. The receive DMA sets the CDE field of the Channel x status register (ETH_DMACxSR) but does not set the RI field, even when IOC field is set, as this is not marked as last receive descriptor for the packet. The subsequent valid receive descriptor is used to write the packet data.*
29	FD	**First Descriptor** When this bit is set, it indicates that this descriptor contains the first buffer of the packet. If the size of the first buffer is 0, the second buffer contains the beginning of the packet. If the size of the second buffer is also 0, the next descriptor contains the beginning of the packet. Refer to the CTXT bit description for details on how to use the CTXT bit and FD bit together.
28	LD	**Last Descriptor** When this bit is set, it indicates that the buffers to which this descriptor is pointing are the last buffers of the packet.
27	RS2V	**Receive Status RDES2 Valid** When this bit is set, it indicates that the status in RDES2 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.
26	RS1V	**Receive Status RDES1 Valid** When this bit is set, it indicates that the status in RDES1 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.
25	RS0V	**Receive Status RDES0 Valid** When this bit is set, it indicates that the status in RDES0 is valid and it is written by the DMA. This bit is valid only when the LD bit of RDES3 is set.

Table 840. RDES3 normal descriptor (write-back format) (continued)

Bit	Name	Description
24	CE	**CRC error** When this bit is set, it indicates that a Cyclic Redundancy Check (CRC) Error occurred on the received packet. This field is valid only when the LD bit of RDES3 is set.
23	GP	**Giant packet** When this bit is set, it indicates that the packet length exceeds the specified maximum Ethernet size of 1518, 1522, or 2000 bytes (9018 or 9022 bytes if jumbo packet enable is set). Giant packet indicates only the packet length. It does not cause any packet truncation.
22	RWT	**Receive watchdog timeout** When this bit is set, it indicates that the receive watchdog timer has expired while receiving the current packet. The current packet is truncated after watchdog timeout.
21	OE	**Overflow error** When this bit is set, it indicates that the received packet is damaged because of buffer overflow in Rx FIFO. This bit is set only when the DMA transfers a partial packet to the application. This happens only when the Rx FIFO is operating in the threshold mode. In the store-and-forward mode, all partial packets are dropped completely in Rx FIFO.
20	RE	**Receive error** When this bit is set, it indicates that the ETH_RX_ER signal is asserted while the ETH_RX_DV signal is asserted during packet reception. This error also includes carrier extension error in the GMII and Half-duplex mode. Error can be of less or no extension, or error (rxd!= 0f) during extension.
19	DE	**Dribble bit error** When this bit is set, it indicates that the received packet has a non-integer multiple of bytes (odd nibbles). This bit is valid only in the MII Mode.
18:16	LT	**Length/type field** This field indicates if the packet received is a length packet or a type packet. The encoding of the 3 bits is as follows: 000: The packet is a length packet 001: The packet is a type packet. 011: The packet is a ARP Request packet type 100: The packet is a type packet with VLAN Tag 101: The packet is a type packet with Double VLAN tag 110: The packet is a MAC Control packet type 111: The packet is a OAM packet type 010: Reserved

Table 840. RDES3 normal descriptor (write-back format) (continued)

Bit	Name	Description
15	ES	**Error Summary** When this bit is set, it indicates the logical OR of the following bits: RDES3[19]: Dribble Error RDES3[20]: Receive Error RDES3[21]: Overflow Error RDES3[22]: Watchdog timeout RDES3[23]: Giant Packet RDES3[24]: CRC Error This field is valid only when the LD bit of RDES3 is set.
14:0	PL	**Packet Length** These bits indicate the byte length of the received packet that was transferred to system memory (including CRC). This field is valid when both the LD bit of RDES3 is set and the Overflow Error bit is reset. The packet length also includes the two bytes appended to the Ethernet packet when IP checksum calculation is enabled and the received packet is not a MAC control packet. When LD bit of RDES3 is reset, this field contains the accumulated number of bytes (partial) that have been transferred for the current packet.

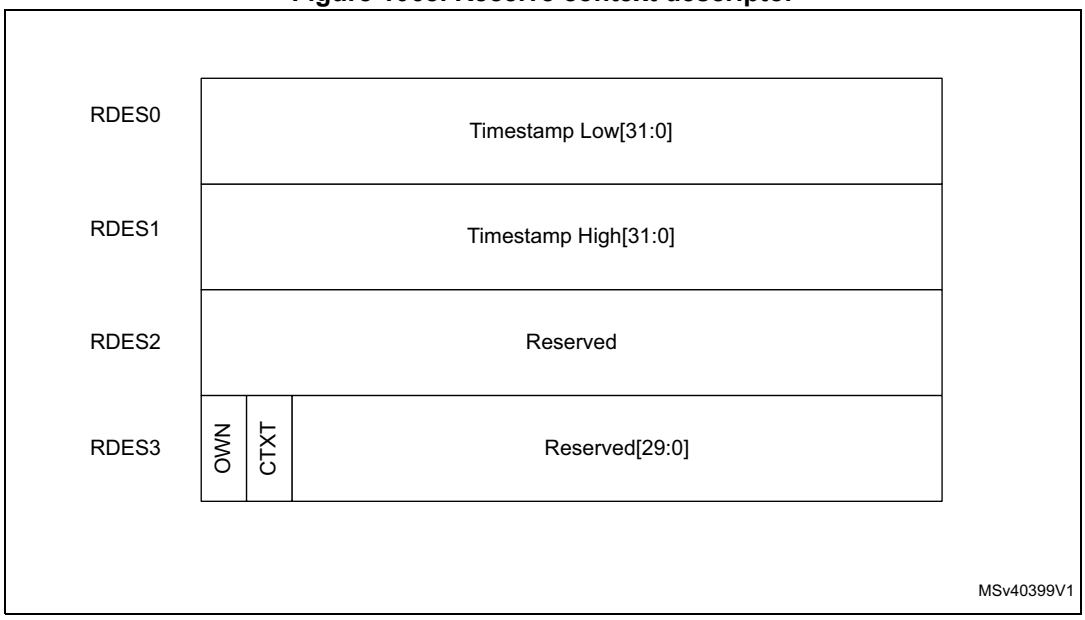
Receive context descriptor

This descriptor is read-only for the application. This descriptor can be written only by the DMA.

The context descriptor provides information about the extended status related to the last received packet. Bit 30 of RDES3 indicates the context type descriptor.

Figure 1068 shows the format for receive context descriptors. *Table 841* to *Table 844* provide a detailed description of all receive context descriptors.

Figure 1068. Receive context descriptor



- RDES0 context descriptor

Table 841. RDES0 context descriptor

Bit	Name	Description
31:0	RTSL	**Receive Packet Timestamp Low** The DMA updates this field with least significant 32 bits of the timestamp captured for corresponding receive packet. When this field and the RTSH field of RDES1 show all-ones value, the timestamp must be considered as corrupt.

- RDES1 context descriptor

Table 842. RDES1 context descriptor

Bit	Field	Description
31:0	RTSH	**Receive Packet Timestamp High** The DMA updates this field with most significant 32 bits of the timestamp captured for corresponding receive packet. When this field and the RTSL field of RDES0 show all-ones value, the timestamp must be considered as corrupt.

- RDES2 context descriptor

Table 843. RDES2 context descriptor

Bit	Description
31:0	Reserved

- RDES3 context descriptor

Table 844. RDES3 context descriptor

Bit	Name	Description
31	OWN	**Own Bit** 1: The DMA owns the descriptor 0: The application owns the descriptor. The DMA clears this bit when either of the following conditions is true: The DMA completes the packet reception The buffers associated with the descriptor are full
30	CTXT	**Receive Context Descriptor** When this bit is set, it indicates that the current descriptor is a context descriptor. The DMA writes 1'b1 to this bit for context descriptor. When CTXT and DE bits are used together, {CTXT, DE} possible values are: 00: Reserved 01: Reserved 10: Context Descriptor 11: Descriptor error When a Descriptor error occurs, the receive DMA closes the receive descriptor indicating a Descriptor error. This receive descriptor is skipped and the buffer addresses are not used to write the packet data The receive DMA sets the CDE bit in *Channel x status register (ETH_DMACxSR)* but does not set the RI field even when IOC is set, as this is not marked as last receive descriptor for the packet. The subsequent valid receive descriptor is used to write the packet data.
29	DE	**Descriptor Error** Refer to the CTXT bit description for details on how to use the E bit along with CTXT bit.
28:0	Reserved	

76.10.5 Enhanced descriptor for time-based scheduling

The time-based scheduling feature needs 32-byte enhanced descriptors to be enabled on all the DMA channels that intend to use the feature. This is done by setting the EDSE bit of *Channel x transmit control register (ETH_DMACxTXCR)*.

The structure of the 32-byte descriptor for the context and the normal descriptor in read and write formats are described in the following sections.

Enhanced normal descriptor (read format)

Figure 1069. Enhanced normal descriptor (read format)

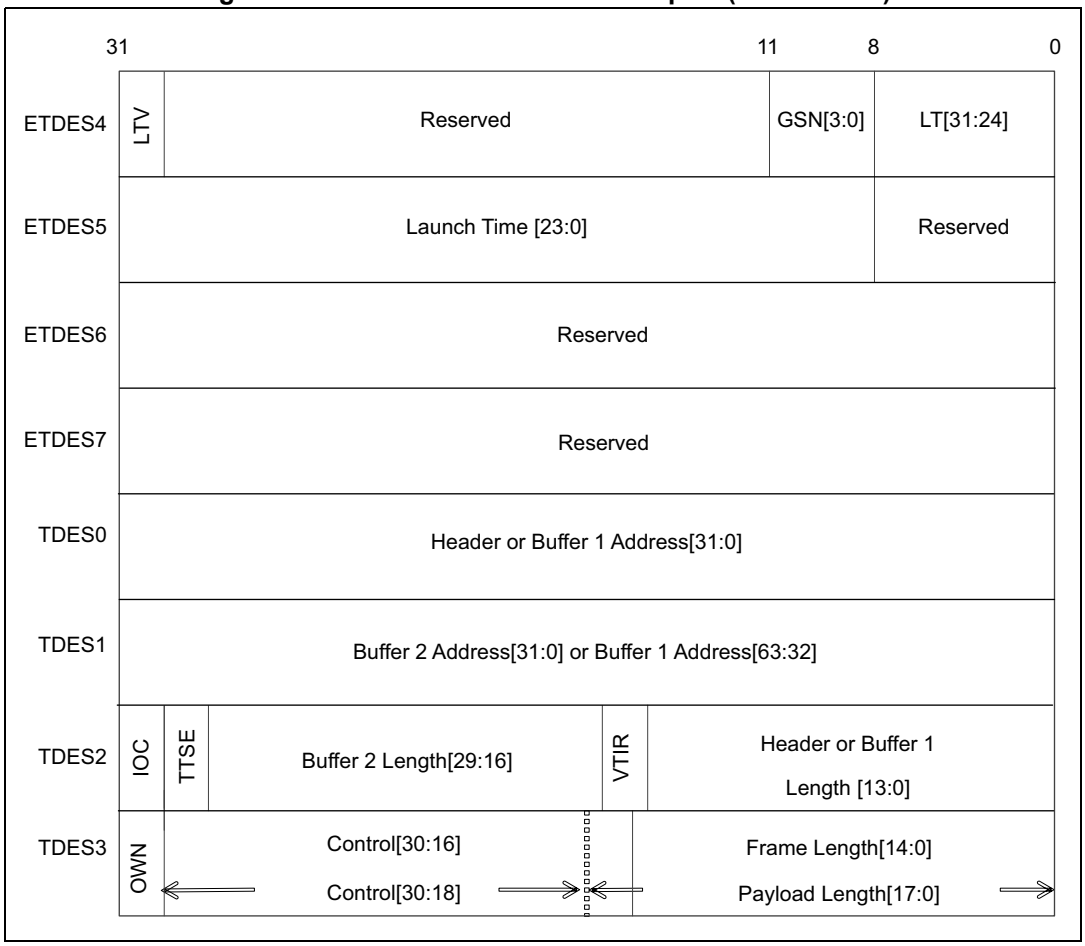


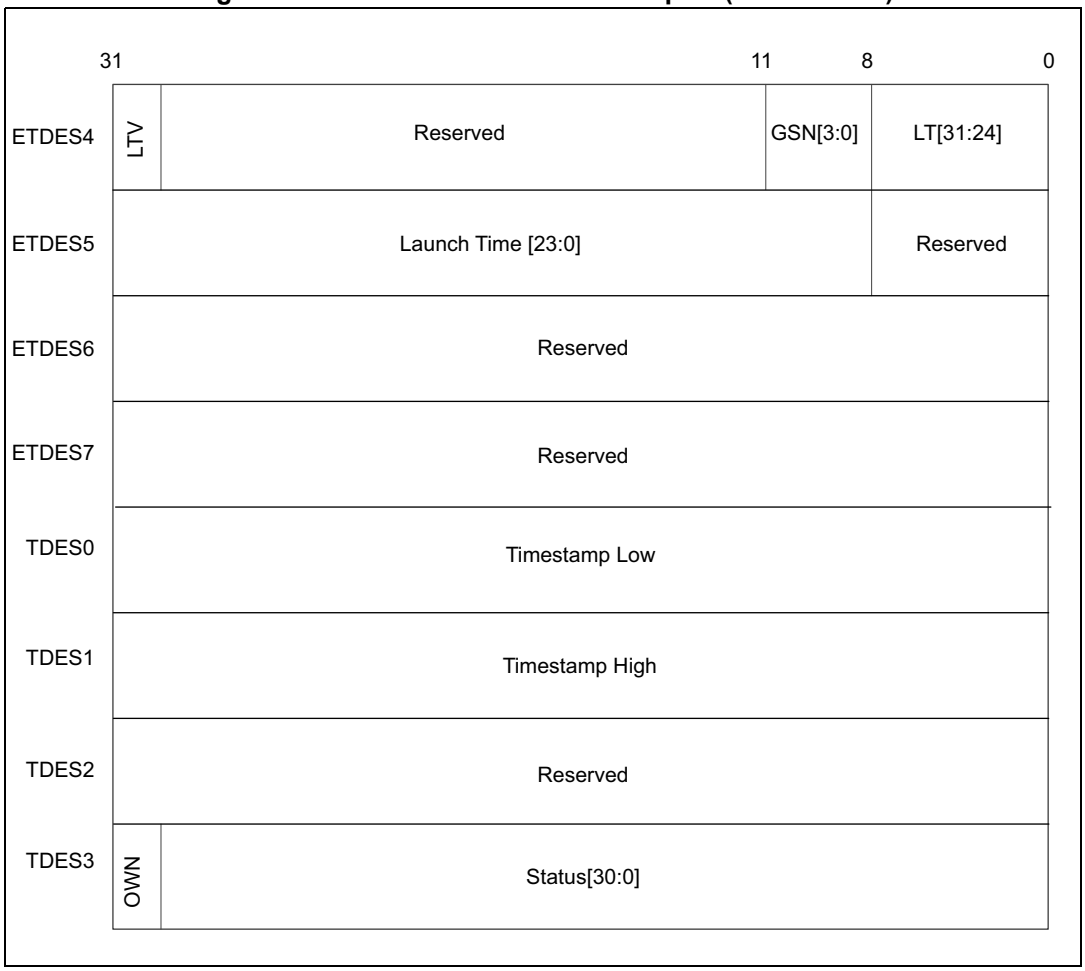
Table 845. Enhanced normal descriptor (read format)

Bit	Name	Description
31	LTV	Launch time valid indicates that the launch time (LT) and GSN fields present in the descriptor are valid. The LTV must be set only if the FD bit of the descriptor is not set.
30:12	Reserved.	
11:8	GSN	GCL slot number associated with the packet
7:0	LT	Launch time associated with the packet.

For details of the other fields, see *Table 821: TDES0 normal descriptor (read format)*.

Enhanced normal descriptor (write format)

Figure 1070. Enhanced normal descriptor (write format)



For details on LTV, GSN, and LT fields, see *Enhanced normal descriptor (read format)*. For details on the other fields, see *Table 825: TDES0 normal descriptor (write-back format)*.

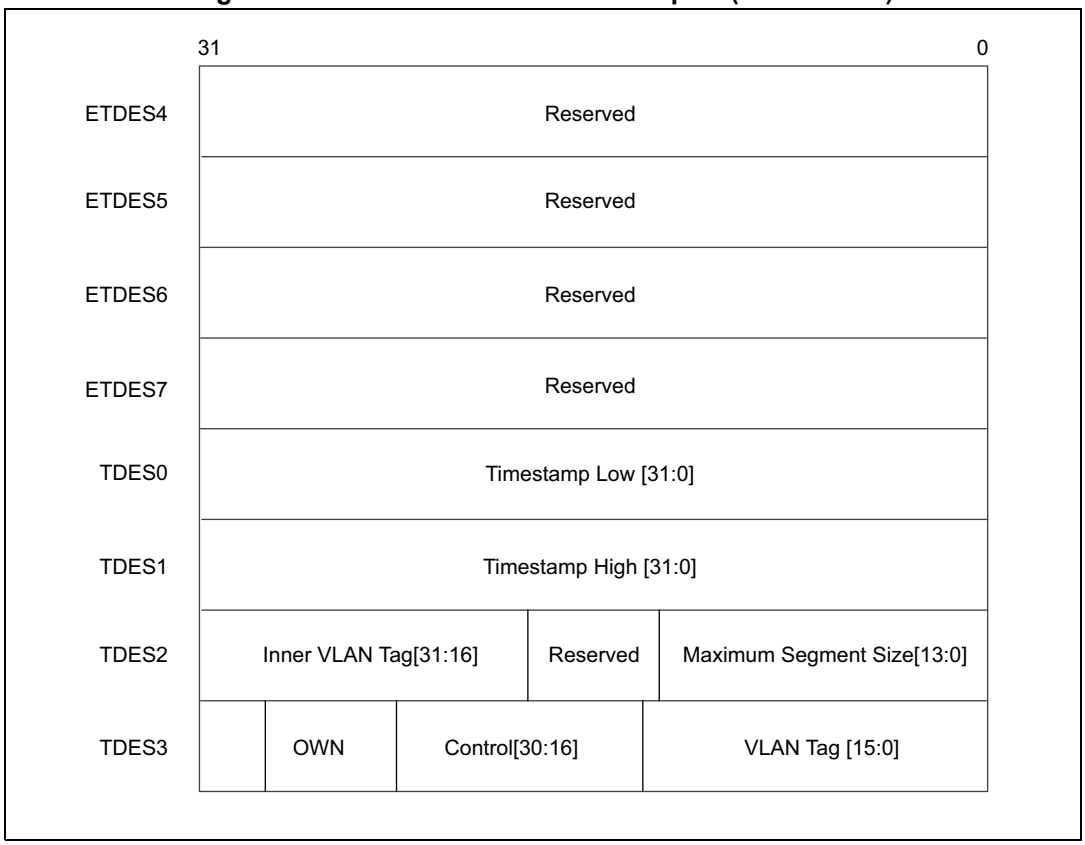
Note: In both write-back formats, no modifications can be done to the extended 16 bytes. The rest of the 16 bytes (TDESC0 to TDESC3) are written back as per the previous 16 bytes descriptor format.

The fetch time, when enabled, overrides the AV slot function.

When an unaligned new GCL list (with a different CTR) is installed, it is recommended not to have traffic during the installation of the new list. Any traffic during the switching of the lists might result in unpredictable behavior of fetch, launch, and launch expiry, as the CTR and BTR values get updated while the frame is being processed.

Enhanced context descriptor (read format)

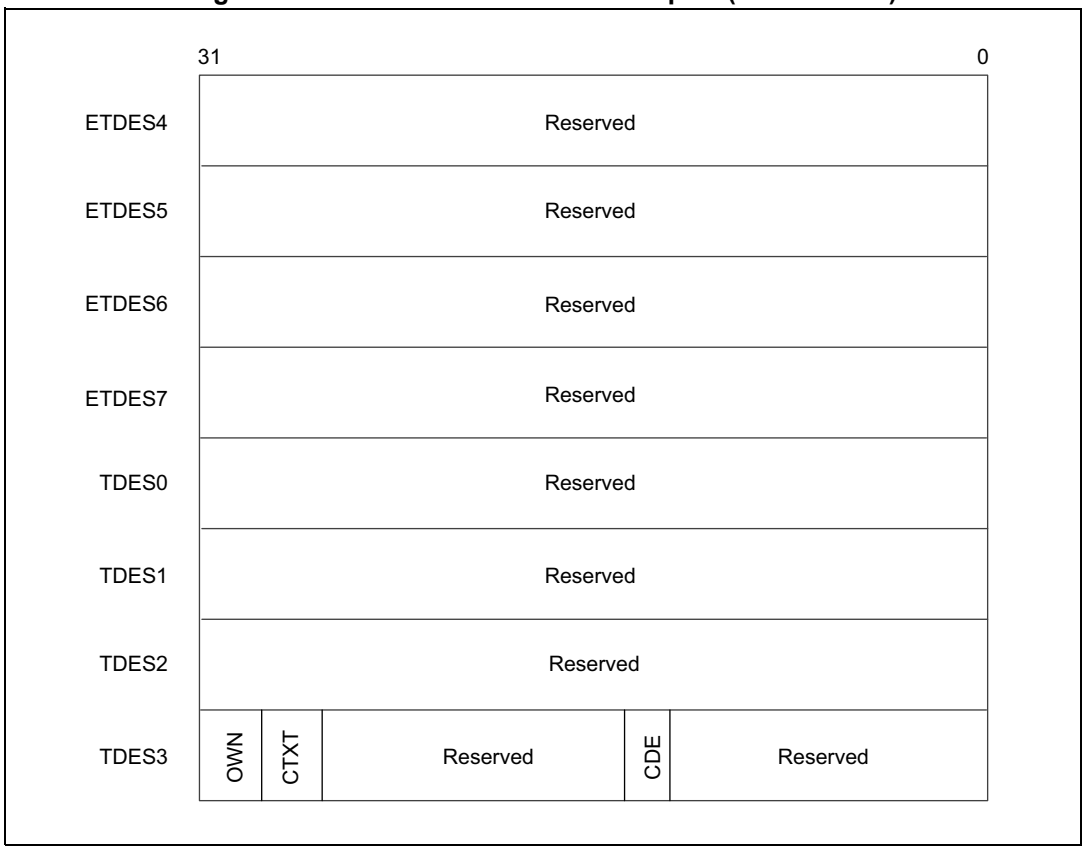
Figure 1071. Enhanced context descriptor (read format)



For details of the various fields, see *Transmit context descriptor*.

Enhanced context descriptor (write format)

Figure 1072. Enhanced context descriptor (write format)



For details of the various fields, see *Receive context descriptor*.

76.11 Ethernet registers

76.11.1 Ethernet register maps

This section provides the following register maps:

- DMA registers (see *Section 76.11.2: Ethernet DMA registers*)
- MTL registers (see *Section 76.11.3: Ethernet MTL registers*)
- MAC registers including MMC registers (see *Section 76.11.4: Ethernet MAC and MMC registers*)

76.11.2 Ethernet DMA registers

DMA mode register (ETH_DMAMR)

Address offset: 0x1000

Reset value: 0x0000 0000

The DMA mode register establishes the bus operating modes for the DMA.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTM[1:0]	
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TXPR	Res.	Res.	DSPW	Res.	Res.	Res.	TAA2	TAA1	TAA0	Res.	SWR
				rw			rw				r	r	r		rw

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:16 **INTM[1:0]**: Interrupt Mode

This field defines the interrupt mode of the Ethernet peripheral.
The behavior of the interrupt signal and of the RI/TI bits in the ETH_DMACSR register changes depending on the INTM value (refer to *Table 820: Transfer complete interrupt behavior*).

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **TXPR**: Transmit priority

When set, this bit indicates that the Tx DMA has higher priority than the Rx DMA during arbitration for the system-side bus.

Bits 10:9 Reserved, must be kept at reset value.

Bit 8 **DSPW**: Descriptor Posted Write

When this bit is set to
0: The descriptor writes are always non-posted.
1: The descriptor writes are non-posted only when IOC (Interrupt on completion) is set in last descriptor. Otherwise, the descriptor writes are always posted.

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:2 **TAA[2:0]**: Transmit Arbitration Algorithm

This field is used to select the arbitration algorithm for the Transmit side when multiple Tx DMAs are selected.

000: Fixed priority. In fixed priority, Channel 0 has the lowest priority and the last channel has the highest priority.

001: Weighted Strict priority (WSP)

010: Weighted Round-Robin (WRR)

011 to 111: Reserved, must not be used.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **SWR**: Software Reset

When this bit is set, the MAC and the DMA controller reset the logic and all internal registers of the DMA, MTL, and MAC. This bit is automatically cleared after the reset operation is complete in all clock domains. Before reprogramming any register, a value of zero should be read in this bit.

Note: The reset operation is complete only when all resets in all active clock domains are deasserted. Therefore, it is essential that all PHY inputs clocks (applicable for the selected PHY interface) are present for software reset completion. The time to complete the software reset operation depends on the frequency of the slowest active clock.

System bus mode register (ETH_DMASBMR)

Address offset: 0x1004

Reset value: 0x0101 0000

The System bus mode register controls the behavior of the AXI master. It mainly controls burst splitting and number of outstanding requests.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EN_LPI	LPI_XIT_PKT	Res.	Res.	Res.	Res.	WR_OSR_LMT[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	RD_OSR_LMT[1:0]	
rw	rw					rw	rw							rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ONEKBBE	AAL	Res.	AALE	Res.	Res.	BLEN256	BLEN128	BLEN64	BLEN32	BLEN16	BLEN8	BLEN4	FB
		rw	rw		rw			rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **EN_LPI**: Enable Low Power Interface (LPI)

When set to 1, this bit enables the LPI mode and accepts the LPI request from the AXI System Clock controller.

When set to 0, this bit disables the LPI mode and always denies the LPI request from the AXI System Clock controller.

Bit 30 **LPI_XIT_PKT**: Unlock on Magic Packet or Remote wake-up Packet

When set to 1, this bit enables the AXI master to come out of the LPI mode only when the magic packet or remote wake-up packet is received. When set to 0, this bit enables the AXI master to come out of the LPI mode when any packet is received.

Bits 29:26 Reserved, must be kept at reset value.

Bits 25:24 **WR_OSR_LMT[1:0]**: AXI Maximum Write Outstanding Request Limit

This value limits the maximum outstanding request on the AXI write interface. Maximum outstanding requests = WR_OSR_LMT + 1.

Bits 23:18 Reserved, must be kept at reset value.

Bits 17:16 **RD_OSR_LMT[1:0]**: AXI Maximum Read Outstanding Request Limit
This value limits the maximum outstanding request on the AXI read interface. Maximum outstanding requests = RD_OSR_LMT + 1

Bits 15:14 Reserved, must be kept at reset value.

Bit 13 **ONEKBBE**: 1 Kbyte Boundary Crossing Enable for the AXI Master
When set, the burst transfers performed by the AXI master do not cross 1 Kbyte boundary. When reset, the burst transfers performed by the AXI master do not cross 4 Kbyte boundary.

Bit 12 **AAL**: Address-Aligned Beats
When this bit is set to 1, the master performs address-aligned burst transfers on Read and Write channels.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **AALE**: Automatic AXI LPI enable
When set to 1, enables the AXI master to enter into LPI state when there is no activity in the Ethernet peripheral for a number of system clock cycles programmed in the LPIEI field of ETH_DMALPIEI register.

Bits 9:8 Reserved, must be kept at reset value.

Bit 7 **BLEN256**: AXI Burst Length 256
When this bit is set to 1, the AXI master can select a burst length of 256 on the AXI interface.

Bit 6 **BLEN128**: AXI Burst Length 128
When this bit is set to 1, the AXI master can select a burst length of 128 on the AXI interface.

Bit 5 **BLEN64**: AXI Burst Length 64
When this bit is set to 1, the AXI master can select a burst length of 64 on the AXI interface.

Bit 4 **BLEN32**: AXI Burst Length 32
When this bit is set to 1, the AXI master can select a burst length of 32 on the AXI interface.

Bit 3 **BLEN16**: AXI Burst Length 16
When this bit is set to 1 or the FB bit is set to 0, the AXI master can select a burst length of 16 on the AXI interface.
When the FB bit is set to 0, setting this bit has no effect.

Bit 2 **BLEN8**: AXI Burst Length 8
When this bit is set to 1 or the FB bit is set to 0, the AXI master can select a burst length of 8 on the AXI interface.
When the FB bit is set to 0, setting this bit has no effect.

Bit 1 **BLEN4**: AXI Burst Length 4
When this bit is set to 1 or the FB bit is set to 0, the AXI master can select a burst length of 4 on the AXI interface.
When the FB bit is set to 0, setting this bit has no effect.

Bit 0 **FB**: Fixed Burst Length
When this bit is set to 1, the AXI master initiates burst transfers of specified lengths as given below.
- Burst transfers of fixed burst lengths as indicated by the BLEN256, BLEN128, BLEN64, BLEN32, BLEN16, BLEN8, or BLEN4 field
- Burst transfers of length 1: when this bit is set to 0, the AXI master initiates burst transfers that are equal to or less than the maximum allowed burst length programmed in Bits[7:1].

Interrupt status register (ETH_DMAISR)

Address offset: 0x1008

Reset value: 0x0000 0000

The application reads this interrupt status register during interrupt service routine or polling to determine the interrupt status of DMA channels, MTL queues, and the MAC.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MACIS	MTLIS
														r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DC1IS	DC0IS
														r	r

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **MACIS**: MAC interrupt status

This bit indicates an interrupt event in the MAC. To reset this bit to 1'b0, the software must read the corresponding register in the MAC to get the exact cause of the interrupt and clear its source.

Bit 16 **MTLIS**: MTL interrupt status

This bit indicates an interrupt event in the MTL. To reset this bit to 1'b0, the software must read the corresponding register in the MTL to get the exact cause of the interrupt and clear its source.

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **DC1IS**: DMA Channel 1 interrupt status

This bit indicates an interrupt event in DMA Channel 1. To reset this bit to 0, the software must read the corresponding register in DMA Channel 1 to get the exact cause of the interrupt and clear its source.

Bit 0 **DC0IS**: DMA Channel 0 interrupt status

This bit indicates an interrupt event in DMA Channel 0. To reset this bit to 0, the software must read the corresponding register in DMA Channel 0 to get the exact cause of the interrupt and clear its source.

Debug status register (ETH_DMADSR)

Address offset: 0x100C

Reset value: 0x0000 0000

The Debug status register gives the receive and transmit process status for DMA Channel 0 to 1 for debugging purpose.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TPS1[3:0]				RPS1[3:0]			
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TPS0[3:0]				RPS0[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	AXRHSTS	AXWHSTS
r	r	r	r	r	r	r	r							r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:20 **TPS1[3:0]**: DMA Channel 1 Transmit Process State
This field indicates the Tx DMA FSM state for Channel 1. This field is similar to the TPS0 field.

Bits 19:16 **RPS1[3:0]**: DMA Channel 1 receive Process State
This field indicates the Rx DMA FSM state for Channel 1. This field is similar to the RPS0 field.

Bits 15:12 **TPS0[3:0]**: DMA Channel 0 Transmit Process State
This field indicates the Tx DMA FSM state for Channel 0:
000: Stopped (Reset or Stop Transmit Command issued)
001: Running (Fetching Tx Transfer Descriptor)
010: Running (Waiting for status)
011: Running (Reading Data from system memory buffer and queuing it to the Tx buffer (Tx FIFO))
100: Timestamp write state
101: Reserved for future use
110: Suspended (Tx Descriptor Unavailable or Tx Buffer Underflow)
111: Running (Closing Tx Descriptor)
The MSB of this field always returns 0. This field does not generate an interrupt.

Bits 11:8 **RPS0[3:0]**: DMA Channel 0 receive Process State
This field indicates the Rx DMA FSM state for Channel 0:
000: Stopped (Reset or Stop receive Command issued)
001: Running (Fetching Rx Transfer Descriptor)
010: Reserved for future use
011: Running (Waiting for Rx packet)
100: Suspended (Rx Descriptor Unavailable)
101: Running (Closing the Rx Descriptor)
110: Timestamp write state
111: Running (Transferring the received packet data from the Rx buffer to the system memory)
The MSB of this field always returns 0. This field does not generate an interrupt.

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **AXRHSTS**: AXI Master Read Channel Status
When high, this bit indicates that the read channel of the AXI master is active, and it is transferring the data.

Bit 0 **AXWHSTS**: AXI Master Write Channel
When high, this bit indicates that the write channel of the AXI master is active, and it is transferring data.

AXI4 transmit channel ACE control register (ETH_DMAA4TXACR)

Address offset: 0x1020

Reset value: 0x0000 0000

This register is used to control the AXI4 Cache Coherency Signals for read transactions by all the Transmit DMA channels. The following signals of the AXI4 interface are driven with different values as programmed for corresponding type (descriptor, buffer1, buffer2) of access:

- arcache_m_o[3:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	THC[3:0]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TEC[3:0]				Res.	Res.	Res.	Res.	TDRC[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **THC[3:0]**: Transmit DMA First Packet Buffer or TSO Header Cache Control

When TSO is NOT enabled, This field is used to drive arcache_o[3:0] signal when Transmit DMA is accessing First Buffer of the Packet (First valid buffer with FD being set in the TDES3 of the Descriptor).

When TSO is enabled, This field is used to drive arcache_o[3:0] signal when the Transmit DMA is accessing the TSO Header data.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **TEC[3:0]**: Transmit DMA Extended Packet Buffer or TSO Payload Cache Control

When TSO is NOT enabled, This field is used to drive arcache o[3:0] signal when Transmit DMA is accessing the extended buffers (when packet is distributed across multiple buffers).

When TSO is enabled, This field is used to drive arcache_o[3:0] signal when the Transmit DMA is accessing the TSO payload data.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **TDRC[3:0]**: Transmit DMA Read Descriptor Cache Control

This field is used to drive arcache_o[3:0] signal when Transmit DMA engines access the Descriptor.

AXI4 receive channel ACE control register (ETH_DMAA4RXACR)

Address offset: 0x1024

Reset value: 0x0000 0000

This register is used to control the AXI4 Cache Coherency Signals for write transactions by all the receive DMA channels. The following signals of the AXI4 interface are driven with different values as programmed for corresponding type (descriptor, buffer1, buffer2) of access:

- awcache_m_o[3:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	RDC[3:0]				Res.	Res.	Res.	Res.	RHC[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	RPC[3:0]				Res.	Res.	Res.	Res.	RDWC[3:0]			
				rw	rw	rw	rw					rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:24 **RDC[3:0]**: Receive DMA Buffer Cache Control
This field is used to drive awcache_o[3:0] signal when receive DMA is accessing the Buffer when Header and payload are NOT separated.

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **RHC[3:0]**: Receive DMA Header Cache Control
This field is used to drive awcache_o[3:0] and signal when receive DMA is accessing the header Buffer when Header and payload are separated.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:8 **RPC[3:0]**: Receive DMA Payload Cache Control
This field is used to drive awcache_o[3:0] signal when receive DMA is accessing the Payload Buffer when Header and payload are separated.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **RDWC[3:0]**: Receive DMA Write Descriptor Cache Control
This field is used to drive awcache_o[3:0] signal when receive DMA accesses the Descriptor.

AXI4 descriptor ACE control register (ETH_DMAA4DACR)

Address offset: 0x1028

Reset value: 0x0000 0000

This register is used to control the AXI4 Cache Coherency Signals for Descriptor write transactions by all the TxDMA channels and Descriptor read transactions by all the RxDMA channels. It also controls the values to be driven on awprot_m_o and arprot_m_o.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	RDRC[3:0]				Res.	Res.	TDWD[1:0]		TDWC[3:0]			
				rw	rw	rw	rw			rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **RDRC[3:0]**: Receive DMA Read Descriptor Cache control
This field is used to drive arcache_o[3:0] signal when receive DMA engines read the Descriptor.

Bits 7:6 Reserved, must be kept at reset value.

Bits 5:4 **TDWD[1:0]**: Transmit DMA Write Descriptor Domain control
This field is used to drive awdomain_o[1:0] signal when Transmit DMA write to the Descriptor.

Bits 3:0 **TDWC[3:0]**: Transmit DMA Write Descriptor Cache control
This field is used to drive awcache_o[3:0] signal when Transmit DMA writes to the Descriptor.

AXI4 LPI Entry Interval register (ETH_DMALPIEI)

Address offset: 0x1040

Reset value: 0x0000 0000

This register is used to control the AXI LPI entry interval.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPIEI[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **LPIEI[3:0]**: LPI Entry Interval

Contains the number of system clock cycles, multiplied by 64, to wait for an activity in the peripheral to enter into the AXI low power state.
0 indicates 64 clock cycles.

DMA TBS control register 0 (ETH_DMATBSCTRL0R)

Address offset: 0x1050

Reset value: 0x0000 0000

This register controls time-based scheduling attributes.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FTOS[23:8]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FTOS[7:0]								Res.	FGOS[2:0]			Res.	Res.	Res.	FTOV
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw				rw

Bits 31:8 **FTOS[23:0]**: Fetch time offset

The value in units of 256 nanoseconds, that has to be deducted from the launch time to compute the fetch time.
The maximum value is 999,999,999 ns. It must be smaller than CTR - 1 when the ESTM mode is set since this value is a modulo CTR value.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **FGOS[2:0]**: Fetch GSN offset

The number of GSN slots that must be deducted from the launch GSN to compute the fetch GSN. This value is valid only when FTOV is set.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **FTOV**: Fetch time offset valid

When set, this bit indicates that the FTOS field is valid. When not set, it indicates that the fetch offset is not valid and the DMA engine can fetch the frames from host memory without any time restrictions.

0: Fetch time offset invalid

1: Fetch time offset valid

Channel x control register (ETH_DMACxCR)

Address offset: 0x1100 + 0x80 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The DMA Channel x control register specifies the MSS value for segmentation, length to skip between two descriptors, and also the features such as header splitting and 8xPBL mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DSL[2:0]			Res.	PBLX8
											rw	rw	rw		rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	MSS[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bits 20:18 **DSL[2:0]**: Descriptor Skip Length

This bit specifies the 64-bit double-word number to skip between two unchained descriptors. The address skipping starts from the end of the current descriptor to the start of the next descriptor.

When the DSL value is equal to zero, the DMA takes the descriptor table as contiguous.

Bit 17 Reserved, must be kept at reset value.

Bit 16 **PBLX8**: 8xPBL mode

When this bit is set, the PBL value programmed in Bits[21:16] in *Channel x transmit control register (ETH_DMACxTXCR)* and in Bits[21:16] in *Channel x receive control register (ETH_DMACxRXCR)* is multiplied eight times. Therefore, the DMA transfers the data in 8, 16, 32, 64, 128, and 256 beats depending on the PBL value.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:0 **MSS[13:0]**: Maximum Segment Size

This field specifies the maximum segment size that should be used while segmenting the packet. This field is valid only if the TSE bit of *Channel x transmit control register (ETH_DMACxTXCR)* is set.

The value programmed in this field must be more than the configured Data width in bytes. It is recommended to use a MSS value of 64 bytes or more.

Channel x transmit control register (ETH_DMACxTXCR)

Address offset: 0x1104 + 0x80 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The DMA Channel x Transmit Control register controls the Tx features such as PBL, TCP segmentation, and Tx Channel weights.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	EDSE	TQOS[3:0]				Res.	Res.	TXPBL[5:0]					
			rw	rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IPBL	Res.	Res.	TSE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSF	TCW[2:0]			ST
rw			rw								rw	r	r	r	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **EDSE**: Enhanced Descriptor Enable

When this bit is set, the corresponding channel uses 32-byte enhanced descriptors both for normal and context descriptors.

When reset, the corresponding channel uses the 16-byte descriptors.

Bits 27:24 **TQOS[3:0]**: Transmit QOS

This field is used to drive arqos_m_o[3:0] or awqos_m_o[3:0] output signals for all transactions of DMA Tx Channel x.

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **TXPBL[5:0]**: Transmit Programmable Burst Length

These bits indicate the maximum number of beats to be transferred in one DMA data transfer. This is the maximum value that is used in a single block Read or Write. The DMA always attempts to burst as specified in PBL each time it starts a burst transfer on the application bus. You can program PBL with any of the following values: 1, 2, 4, 8, 16, or 32. Any other value results in undefined behavior.

To transfer more than 32 beats, perform the following steps:

a) Set the PBLx8 mode in ETH_DMACxCR.

b) Set the TXPBL[5:0].

Note: The maximum value of TXPBL must be less than or equal to half the Tx Queue size (TQS field of *Tx queue x operating mode register (ETH_MTLTXQxOMR)*) in terms of beats. This is required so that the Tx Queue has space to store at least another Tx PBL worth of data while the MTL Tx Queue Controller is transferring data to MAC. The total locations in Tx Queue of size 4096 bytes is 512, TXPBL and 8xPBL needs to be programmed to less than or equal to 512/2.

Bit 15 **IPBL**: Ignore PBL Requirement

When this bit is set, the DMA does not check for PBL number of locations in the MTL before initiating a transfer. If space is not available, the MTL might use handshaking to slow the DMA.

Note: This bit/mode must not be used when multiple Transmit DMA Channels are enabled as it might block other transmit and receive DMA Channels from accessing the Read Data Channel of AXI bus until space is available in Transmit Queue for current transfer.

Bits 14:13 Reserved, must be kept at reset value.

Bit 12 **TSE**: TCP Segmentation Enabled

When this bit is set, the DMA performs the TCP segmentation for packets in Channel x. The TCP segmentation is done only for those packets for which the TSE bit (TDES0[19]) is set in the Tx Normal descriptor. When this bit is set, the TxPBL value must be greater than or equal to 4.

Bits 11:5 Reserved, must be kept at reset value.

Bit 4 **OSF**: Operate on Second Packet

When this bit is set, it instructs the DMA to process the second packet of the Transmit data even before the status for the first packet is obtained.

Bits 3:1 **TCW[2:0]**: Transmit Channel Weight

This field indicates the weight assigned to the corresponding Transmit channel. When reset is complete, this field is set to 0 for all channels by default, resulting in equal weights to all channels.

Bit 0 **ST**: Start or Stop Transmission Command

When this bit is set, transmission is placed in the Running state. The DMA checks the Transmit list at the current position for a packet to be transmitted.

The DMA tries to acquire descriptor from either of the following positions:

- The current position in the list: this is the base address of the Transmit list set by the ETH_DMACxTXDLAR register.
- The position at which the transmission was previously stopped

If the DMA does not own the current descriptor, the transmission enters the Suspended state and the TBU bit of the ETH_DMACxSR is set. The Start Transmission command is effective only when the transmission is stopped. If the command is issued before setting the ETH_DMACxTXDLAR register, the DMA behavior is unpredictable.

When this bit is reset, the transmission process is placed in the Stopped state after completing the transmission of the current packet. The Next Descriptor position in the Transmit list is saved, and it becomes the current position when the transmission is restarted. To change the list address, you need to program ETH_DMACxTXDLAR register with a new value when this bit is reset. The new value is considered when this bit is set again. The stop transmission command is effective only when the transmission of the current packet is complete or the transmission is in the Suspended state.

Channel x receive control register (ETH_DMACxRXCR)

Address offset: 0x1108 + 0x80 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The DMA channel x receive control register controls the Rx features such as PBL, buffer size, and extended status.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RPF	Res.	Res.	Res.	RQOS[3:0]				Res.	Res.	RXPBL[5:0]					
rw				rw	rw	rw	rw			rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RBSZ[13:0]														SR
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RPF**: DMA Rx Channel x Packet Flush

When this bit is set to 1, the Ethernet peripheral automatically flushes the packet from the Rx queues destined to DMA Rx Channel x when the DMA Rx Channel x is stopped after a system bus error has occurred. When this bit remains set and the DMA is re-started by the software driver, the packets residing in the Rx Queues that were received when this RxDMA was stopped, are flushed out. The packets that are received by the MAC after the RxDMA is re-started are routed to the RxDMA. The flushing happens on the Read side of the Rx queue.

When this bit is set to 0 the Ethernet peripheral does not flush the packet in the Rx queue destined to DMA Rx Channel x after the DMA is stopped due to a system bus error.

This might cause head-of-line blocking in the corresponding RxQueue.

Note: The stopping of packet flow from a Rx DMA Channel to the application by setting RPF works only when there is one-to-one mapping of Rx Queue to Rx DMA channels. In Dynamic mapping mode, setting RPF bit in ETH_DMACxRXCR register might flush packets from unintended Rx Queues which are destined to the stopped Rx DMA Channel.

Bits 30:28 Reserved, must be kept at reset value.

Bits 27:24 **RQOS[3:0]**: Rx AXI4 QOS.

This field is used to drive arqos_m_o[3:0] or awqos_m_o[3:0] output signals for all transactions of DMA Rx Channel0.

Bits 23:22 Reserved, must be kept at reset value.

Bits 21:16 **RXPBL[5:0]**: Receive Programmable Burst Length

These bits indicate the maximum number of beats to be transferred in one DMA data transfer. This is the maximum value that is used in a single block Read or Write. The DMA always attempts to burst as specified in PBL each time it starts a burst transfer on the application bus. You can program PBL with any of the following values: 1, 2, 4, 8, 16, or 32. Any other value results in undefined behavior.

To transfer more than 32 beats, perform the following steps:

a) Set the PBLx8 mode in the ETH_DMAC0CR.

b) Set the RXPBL[5:0].

Note: The maximum value of RXPBL must be less than or equal to half the Rx Queue size (RQS field of Rx queue x operating mode register (ETH_MTLRXQxOMR)) in terms of beats. This is required so that the Rx Queue has space to store at least another Rx PBL worth of data while the MTL Rx Queue Controller is transferring data to MAC.The total locations in Rx Queue of size 4096 bytes is 512, RXPBL and 8xPBL needs to be programmed to less than or equal to 512/2.

Bit 15 Reserved, must be kept at reset value.

Bits 14:1 **RBSZ[13:0]**: Receive Buffer size

This field indicates the size of the Rx buffers specified in bytes. The maximum buffer size is limited to 16 Kbytes.

Note: The buffer size must be a multiple of 8. This is required even if the value of buffer address pointer is not aligned to bus width. If the buffer size is not a multiple of 8, it may result into undefined behavior.

The LSB bits (2:0) are ignored and the DMA internally takes the LSB bits as all-zero. Therefore, these LSB bits are read-only (RO).

Bit 0 **SR**: Start or Stop receive

When this bit is set, the DMA tries to acquire the descriptor from the receive list and processes the incoming packets.

The DMA tries to acquire descriptor from either of the following positions:

– The current position in the list: this is the address set by the *Channel x Rx descriptor list address register (ETH_DMACxRXDLAR)*.

– The position at which the Rx process was previously stopped

If the DMA does not own the current descriptor, the reception is suspended and the RBU bit of the ETH_DMAC0SR is set. The Start Receive command is effective only when the reception is stopped. If the command is issued before setting the *Channel x Rx descriptor list address register (ETH_DMACxRXDLAR)*, the DMA behavior is unpredictable.

When this bit is reset, the Rx DMA operation is stopped after the transfer of the current packet. The next descriptor position in the receive list is saved, and it becomes the current position after the Rx process is restarted. The Stop Receive command is effective only when the Rx process is in the Running (waiting for Rx packet) or Suspended state.

Channel x Tx descriptor list address register (ETH_DMACxTXDLAR)

Address offset: 0x1114 +0x80 * x, (x = 0 to 1)

Reset value: 0x0000 0000

Channel x Tx descriptor list address register points the DMA to the start of the transmit descriptor list. The descriptor lists reside in the physical memory space of the application and must be Dword-aligned. The DMA internally converts it to bus width aligned address by making the corresponding LSB to low.

Writing to this register is permitted only when the Tx DMA has stopped, that is, the ST bit is cleared in ETH_DMACxTXCR register. When stopped, this register can be written with a new descriptor list address. When the ST bit is set, the DMA takes the newly-programmed descriptor base address. If this register is not changed when ST bit is cleared, the DMA takes the descriptor address where it was stopped earlier.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TDESLA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TDESLA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r

Bits 31:0 **TDESLA[31:0]**: Start of Transmit List

This field contains the base address of the first descriptor in the Transmit descriptor list. The DMA ignores the LSB bits (2:0) for 64-bit bus width and internally takes these bits as all-zero. Therefore, these LSB bits are read-only (RO).

Channel x Rx descriptor list address register (ETH_DMACxRXDLAR)

Address offset: 0x111C + 0x80 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The Channel x Rx descriptor list address register points the DMA to the start of receive descriptor list.

This register points to the start of the receive descriptor list. The descriptor lists reside in the physical memory space of the application and must be Dword-aligned. The DMA internally converts it to bus width aligned address by making the corresponding LS bits low. Writing to this register is permitted only when reception is stopped. When stopped, this register must be written to before the receive Start command is given. You can write to this register only when Rx DMA has stopped, that is, SR bit is set to zero in ETH_DMACxRXCR register. When stopped, this register can be written with a new descriptor list address.

When you set the SR bit to 1, the DMA takes the newly programmed descriptor base address.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDESLA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDESLA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r

Bits 31:0 **RDESLA[31:0]**: Start of receive list

This field contains the base address of the first descriptor in the Rx Descriptor list. The DMA ignores the LSB bits (2:0) for 64-bit bus width and internally takes these bits as all-zero. Therefore, these LSB bits are read-only (RO).

Channel x Tx descriptor tail pointer register (ETH_DMACxTXDTPR)

Address offset: 0x1120 + 0x80 * x (x = 0 to 1)

Reset value: 0x0000 0000

The channel x Tx descriptor tail pointer register points to an offset from the base and indicates the location of the last valid descriptor.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TDT[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TDT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r

Bits 31:0 **TDT[31:0]**: transmit descriptor tail pointer

This field contains the tail pointer for the Tx descriptor ring. The software writes the tail pointer to add more descriptors to the Tx channel. The hardware tries to transmit all packets referenced by the descriptors between the head and the tail pointer registers.

Channel x Rx descriptor tail pointer register (ETH_DMACxRXDTPR)

Address offset: 0x1128 + 0x80 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The channel x Rx descriptor tail pointer points to an offset from the base and indicates the location of the last valid descriptor.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDT[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r

Bits 31:0 **RDT[31:0]**: Receive Descriptor Tail Pointer

This field contains the tail pointer for the Rx descriptor ring. The software writes the tail pointer to add more descriptors to the Rx channel. The hardware tries to write all received packets to the descriptors referenced between the head and the tail pointer registers.

Channel x Tx descriptor ring length register (ETH_DMACxTXRLR)

Address offset: 0x112C + 0x80 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The Tx descriptor ring length register contains the length of the transmit descriptor ring.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TDRL[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:0 **TDRL[9:0]**: Transmit descriptor ring length

This 10-bit field sets the maximum number of Tx descriptors in the circular descriptor ring.
TDRL[9:0] corresponds to the number of Tx descriptors minus one.
It can be programmed to any value up to 0x3FF (1024 descriptors).
A minimum ring descriptor length of four is recommended.
For example, program 0x9 to configure 10 descriptors.

Channel x Rx descriptor ring length register (ETH_DMACxRXRLR)

Address offset: 0x1130 + 0x80 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The channel x Rx descriptor ring length register contains the length of the receive descriptor circular ring.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARBS[6:0]							Res.
								rw	rw	rw	rw	rw	rw	rw	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	RDRL[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:17 **ARBS[6:0]**: Alternate receive buffer size

Indicates size in bytes for Buffer 1 when ARBS[6:0] is programmed to a non-zero value.
When ARBS[6:0] = 0, Rx Buffer1 and Rx Buffer2 sizes are based on RBSZ[13:0] field of *Channel x receive control register (ETH_DMACxRXCR)*.

Bits 16:10 Reserved, must be kept at reset value.

Bits 9:0 **RDRL[9:0]**: Receive Descriptor Ring Length

This 10-bit field sets the maximum number of Rx descriptors in the circular descriptor ring.
RDRL[9:0] corresponds to the number of Rx descriptors minus one. It can be programmed to any value up to 0x3FF (1024 descriptors).
For example, program 0x9 to configure 10 descriptors.

Channel x interrupt enable register (ETH_DMACxIER)

Address offset: 0x1134 + 0x80 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The channel x interrupt enable register enables the interrupts reported by the Status register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NIE	AIE	CDEE	FBEE	ERIE	ETIE	RWTE	RSE	RBUE	RIE	Res.	Res.	Res.	TBUE	TXSE	TIE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **NIE**: Normal Interrupt Summary Enable

When this bit is set, the normal interrupt summary is enabled. This bit enables the following interrupts in the *Channel x status register (ETH_DMACxSR)*:
Bit 0: Transmit interrupt
Bit 2: Transmit buffer unavailable
Bit 6: Receive interrupt
Bit 11: Early receive interrupt
When this bit is reset, the normal interrupt summary is disabled.

Bit 14 **AIE**: Abnormal Interrupt Summary Enable

When this bit is set, the abnormal interrupt summary is enabled. This bit enables the following interrupts in the *Channel x status register (ETH_DMACxSR)*:
Bit 1: Transmit process stopped
Bit 7: Rx buffer unavailable
Bit 8: Receive process stopped
Bit 9: Receive watchdog timeout
Bit 10: Early transmit interrupt
Bit 12: Fatal bus error
Bit 13: Context descriptor error
When this bit is reset, the abnormal interrupt summary is disabled.

Bit 13 **CDEE**: Context descriptor error enable

When this bit is set along with the AIE bit, the Context Descriptor error interrupt is enabled. When this bit is reset, the Context Descriptor error interrupt is disabled.

Bit 12 **FBEE**: Fatal Bus Error enable

When this bit is set along with the AIE bit, the Fatal Bus error interrupt is enabled. When this bit is reset, the Fatal Bus Error error interrupt is disabled.

Bit 11 **ERIE**: Early receive interrupt enable

When this bit is set along with the NIE bit, the Early receive interrupt is enabled. When this bit is reset, the Early receive interrupt is disabled.

Bit 10 **ETIE**: Early Transmit Interrupt enable

When this bit is set along with the AIE bit, the Early Transmit interrupt is enabled. When this bit is reset, the Early Transmit interrupt is disabled.

Bit 9 **RWTE**: Receive watchdog timeout enable

When this bit is set along with the AIE bit, the receive watchdog timeout interrupt is enabled. When this bit is reset, the receive watchdog timeout interrupt is disabled.

Bit 8 **RSE**: Receive Stopped Enable

When this bit is set along with the AIE bit, the Receive Stopped Interrupt is enabled. When this bit is reset, the Receive Stopped interrupt is disabled.

Bit 7 **RBUE**: Receive Buffer Unavailable enable

When this bit is set along with the AIE bit, the Receive Buffer Unavailable interrupt is enabled. When this bit is reset, the Receive Buffer Unavailable interrupt is disabled.

Bit 6 **RIE**: Receive Interrupt enable

When this bit is set along with the NIE bit, the receive interrupt is enabled. When this bit is reset, the receive interrupt is disabled.

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **TBUE**: Transmit Buffer Unavailable enable

When this bit is set along with the NIE bit, the Transmit Buffer Unavailable interrupt is enabled. When this bit is reset, the Transmit Buffer Unavailable interrupt is disabled.

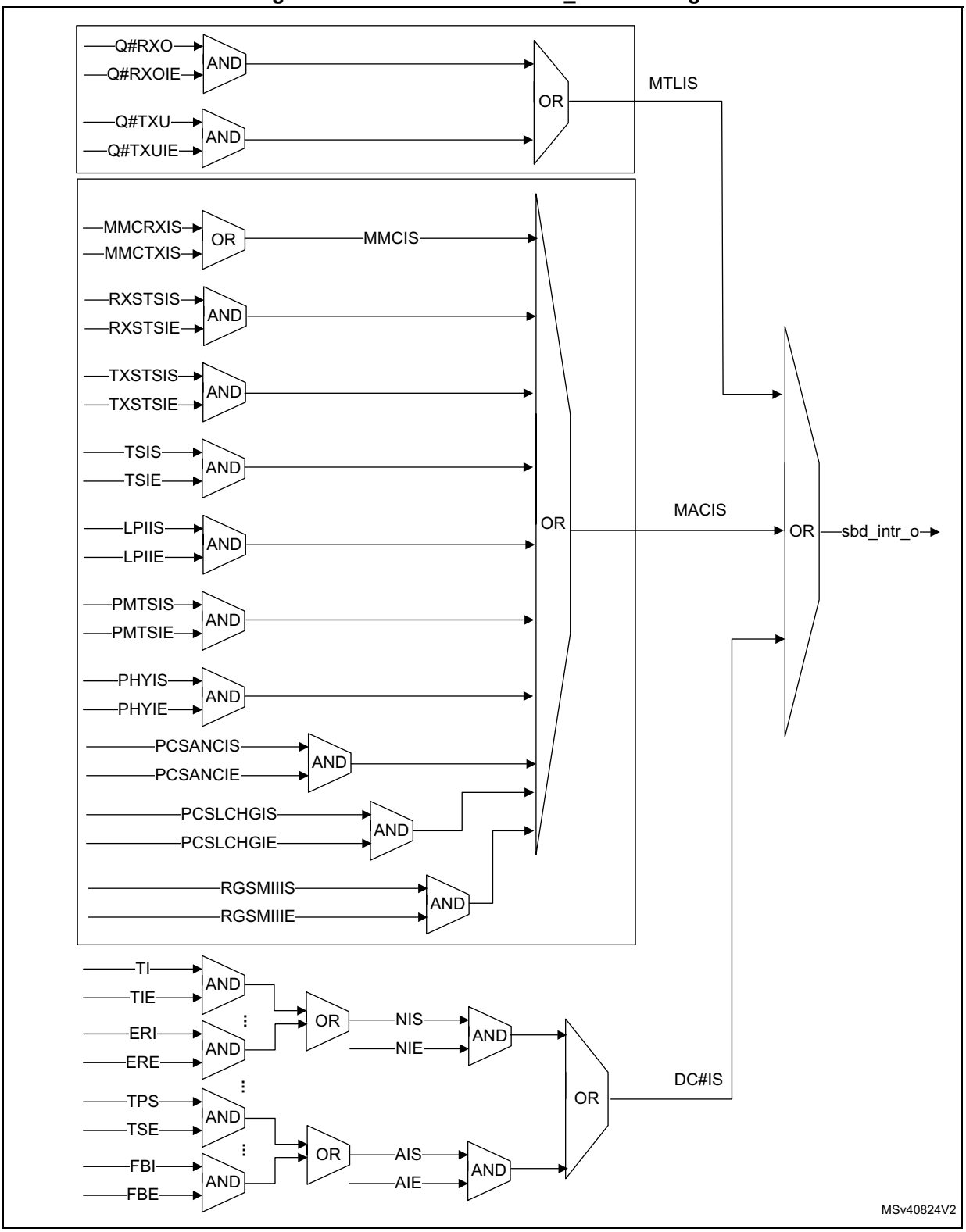
Bit 1 **TXSE**: Transmit Stopped enable

When this bit is set along with the AIE bit, the Transmission Stopped interrupt is enabled. When this bit is reset, the Transmission Stopped interrupt is disabled.

Bit 0 **TIE**: Transmit Interrupt enable

When this bit is set along with the NIE bit, the transmit interrupt is enabled. When this bit is reset, the transmit interrupt is disabled.

Figure 1073. Generation of ETH_DMAISR flags



Channel x Rx interrupt watchdog timer register (ETH_DMACxRXIWTR)

Address offset: 0x1138 + 0x80 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The receive interrupt watchdog timer register indicates the watchdog timeout for receive interrupt (RI) from the DMA. When this register is written with a non-zero value, it enables the watchdog timer for the RI bit of the *Channel x status register (ETH_DMACxSR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RWTU[1:0]	
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RWT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:16 **RWTU[1:0]**: Receive Interrupt Watchdog Timer Count Units

This bitfield indicates the number of system clock cycles corresponding to one unit in RWT[7:0] bitfield.
00: 256
01: 512
10: 1024
11: 2048
For example, when RWT[7:0] = 2 and RWTU[1:0] = 1, the watchdog timer is set for 2 × 512 = 1024 system clock cycles.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **RWT[7:0]**: Receive Interrupt Watchdog Timer Count

This bitfield indicates the number of system clock cycles, multiplied by factor indicated in RWTU field, for which the watchdog timer is set.
The watchdog timer is triggered with the programmed value after the Rx DMA completes the transfer of a packet for which the RI bit is not set in the ETH_DMACxSR, because of the setting of Interrupt Enable bit in the corresponding descriptor RDES3[30].
When the watchdog timer runs out, the RI bit is set and the timer is stopped. The watchdog timer is reset when the RI bit is set high because of automatic setting of RI as per the Interrupt Enable bit RDES3[30] of any received packet.

Channel x slot function control status register (ETH_DMACxSFCSR)

Address offset: 0x113C + 0x80 * x, (x = 0 to 1)

Reset value: 0x0000 07C0

The Slot Function Control and Status register contains the control bits for slot function and the status for Transmit path.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSN[3:0]			
												r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SIV[11:0]												Res.	Res.	ASC	ESC
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **RSN[3:0]**: Reference Slot Number

This field gives the current value of the reference slot number in the DMA. It is used for slot comparison.

Bits 15:4 **SIV[11:0]**: Slot Interval Value

This field controls the period of the slot interval in which the TxDMA fetches the scheduled packets. A value of 0 specifies the slot interval of 1 µs while the maximum value 4095 specifies the slot interval of 4096 µs. The default/reset value is 0x07C which corresponds to slot interval of 125 µs.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **ASC**: Advance Slot Check

When set, this bit enables the DMA to fetch the data from the buffer when the slot number (SLOTNUM) programmed in the Tx descriptor is

– equal to the reference slot number given in the RSN field

or

– ahead of the reference slot number by up to two slots

This bit is applicable only when the ESC bit is set.

Bit 0 **ESC**: Enable Slot Comparison

When set, this bit enables the checking of the slot numbers programmed in the Tx descriptor with the current reference given in the RSN field. The DMA fetches the data from the corresponding buffer only when the slot number is

– equal to the reference slot number

or

– ahead of the reference slot number by one slot

When reset, this bit disables the checking of the slot numbers. The DMA fetches the data immediately after the descriptor is processed.

Note: The UFO (UDP Fragmentation over IPv4)/TSO/USO should not be enabled along with TBS/AVB Slot number check. The UFO/TSO/USO involves multiple packets/segments/fragments transmission for single packet received from application and the slot number check are applicable for fetching of only first segment/fragment. As a result it might be difficult for the software to specify the slot number for subsequent packets.

Channel x current application transmit descriptor register (ETH_DMACxCATXDR)

Address offset: 0x1144 + 0x80 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The channel x current application transmit descriptor register points to the current transmit descriptor read by the DMA.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CURTDESAPTR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CURTDESAPTR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CURTDESAPTR[31:0]**: Application transmit descriptor address pointer
The DMA updates this pointer during Tx operation. This pointer is cleared on reset.

Channel x current application receive descriptor register (ETH_DMACxCARXDR)

Address offset: 0x114C + 0x80 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The channel x current application receive descriptor register points to the current receive descriptor read by the DMA.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CURRDESAPTR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CURRDESAPTR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CURRDESAPTR[31:0]**: Application receive descriptor address pointer
The DMA updates this pointer during Rx operation. This pointer is cleared on reset.

Channel x current application transmit buffer register (ETH_DMACxCATXBR)

Address offset: 0x1154 + 0x80 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The channel x current application transmit buffer address register points to the current Tx buffer address read by the DMA.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CURTBUFAPTR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CURTBUFAPTR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CURTBUFAPTR[31:0]**: Application Transmit Buffer Address Pointer
The DMA updates this pointer during Tx operation. This pointer is cleared on reset.

Channel x current application receive buffer register (ETH_DMACxCARXBR)

Address offset: 0x115C + 0x80 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The channel x current application receive buffer address register points to the current Rx buffer address read by the DMA.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CURRBUFAPTR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CURRBUFAPTR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CURRBUFAPTR[31:0]**: Application receive buffer address pointer
The DMA updates this pointer during Rx operation. This pointer is cleared on reset.

Channel x status register (ETH_DMACxSR)

Address offset: 0x1160 + 0x80 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The software driver (application) reads the status register during interrupt service routine or polling to determine the status of the DMA.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REB[2:0]			TEB[2:0]		
										r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NIS	AIS	CDE	FBE	ERI	ETI	RWT	RPS	RBU	RI	Res.	Res.	Res.	TBU	TPS	TI
rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rc_w1	rw	rc_w1	rc_w1	rc_w1				rc_w1	rc_w1	rc_w1

Bits 31:22 Reserved, must be kept at reset value.

Bits 21:19 **REB[2:0]**: Rx DMA Error Bits

This field indicates the type of error that caused a bus error. For example, error response on the AXI interface.
Bit [2]: Error during data transfer by Rx DMA when 1, no rror during data transfer by Rx DMA when 0.
Bit[1]: Error during descriptor access when 1, error during data buffer access when 0
Bit[0]: Error during read transfer when 1, error during write transfer when 0
This field is valid only when the FBE bit is set. This field does not generate an interrupt.

Bits 18:16 **TEB[2:0]**: Tx DMA Error Bits

This field indicates the type of error that caused a bus error. For example, error response on the AXI interface.
Bit[2]: Error during data transfer by Tx DMA when 1, no error during data transfer by Tx DMA when 0
Bit[1]: Error during descriptor access when 1, error during data buffer access when 0
Bit[0]: Error during read transfer when 1, Error during write transfer when 0
This field is valid only when the FBE bit is set. This field does not generate an interrupt.

Bit 15 **NIS**: Normal Interrupt Summary

Normal Interrupt Summary bit value is the logical OR of the following bits when the corresponding interrupt bits are enabled in the ETH_DMACxIER register:
Bit 0: Transmit interrupt
Bit 2: Transmit buffer unavailable
Bit 6: Receive interrupt
Bit 11: Early receive interrupt
Only unmasked bits (interrupts for which interrupt enable is set in ETH_DMACxIER register) affect the Normal Interrupt Summary bit.
This is a sticky bit. You must clear this bit (by writing 1 to this bit) each time a corresponding bit which causes NIS to be set is cleared.

Bit 14 **AIS**: Abnormal Interrupt Summary

Abnormal Interrupt Summary bit value is the logical OR of the following when the corresponding interrupt bits are enabled in the ETH_DMACxIER register:
Bit 1: Transmit process stopped
Bit 7: Receive buffer unavailable
Bit 8: Receive process stopped
Bit 10: Early transmit interrupt
Bit 12: Fatal bus error
Bit 13: Context descriptor error
Only unmasked bits affect the Abnormal Interrupt Summary bit.
This is a sticky bit. You must clear this bit (by writing 1 to this bit) each time a corresponding bit, which causes AIS to be set, is cleared.

Bit 13 **CDE**: Context descriptor error

This bit indicates that the DMA Tx/Rx engine received a descriptor error, which indicates invalid context in the middle of packet flow (intermediate descriptor) or all one‘s descriptor in Tx case and on Rx side it indicates DMA has read a descriptor with either of the buffer address as ones which is considered to be invalid.

Bit 12 **FBE**: Fatal bus error

This bit indicates that a bus error occurred (as described in the EB field). When this bit is set, the corresponding DMA channel engine disables all bus accesses.

Bit 11 **ERI**: Early receive interrupt

When set, this bit indicates that the RxDMA has completed the transfer of packet data to the memory. The RI bit of this register automatically clears this bit.

Bit 10 **ETI**: Early transmit interrupt

When set, this bit indicates that the TxDMA has completed the transfer of packet data to the MTL TXFIFO memory.

Bit 9 **RWT**: Receive watchdog timeout

This bit is asserted when a packet with length greater than 2,048 bytes (10,240 bytes when Jumbo Packet mode is enabled) is received.

Bit 8 **RPS**: Receive process stopped

This bit is asserted when the Rx process enters the Stopped state.

Bit 7 **RBU**: Receive buffer unavailable

This bit indicates that the application owns the next descriptor in the receive list, and the DMA cannot acquire it. The Rx process is suspended. To resume processing Rx descriptors, the application should change the ownership of the descriptor and issue a Receive Poll Demand command. If this command is not issued, the Rx process resumes when the next recognized incoming packet is received. In ring mode, the application should advance the receive descriptor tail pointer register of a channel. This bit is set only when the DMA owns the previous Rx descriptor.

Bit 6 **RI**: Receive interrupt

This bit indicates that the packet reception is complete. When packet reception is complete, Bit 31 of RDES1 is reset in the last descriptor, and the specific packet status information is updated in the descriptor.

The reception remains in the Running state.

Bits 5:3 Reserved, must be kept at reset value.

Bit 2 **TBU**: Transmit buffer unavailable

This bit indicates that the application owns the next descriptor in the Transmit list, and the DMA cannot acquire it. Transmission is suspended. The TPSi field of the *Debug status register (ETH_DMADSR)* register explains the Transmit Process state transitions.

To resume processing the Transmit descriptors, the application should do the following:

1. Change the ownership of the descriptor by setting Bit 31 of TDES3.
2. Issue a Transmit Poll Demand command.

For ring mode, the application should advance the transmit descriptor tail pointer register of a channel.

Bit 1 **TPS**: Transmit process stopped

This bit is set when the transmission is stopped.

Bit 0 **TI**: Transmit interrupt

This bit indicates that the packet transmission is complete. When transmission is complete, Bit 31 of TDES3 is reset in the last descriptor, and the specific packet status information is updated in the descriptor.

Channel x missed frame count register (ETH_DMACxMFCR)

Address offset: 0x1164 + 0x80 * x, (x = 0 to 1)

Reset value: 0x0000 0000

This register has the number of packet counter that got dropped by the DMA either due to bus error or due to programing RPF field in *Channel x receive control register (ETH_DMACxRXCR)* register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MFCO	Res.	Res.	Res.	Res.	MFC[10:0]										
rc_r					rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r

Bits 31:16 Reserved, must be kept at reset value.

Bit 15 **MFCO**: Overflow status of the MFC Counter

When this bit is set then the MFC counter does not get incremented further. The bit gets cleared when this register is read.

Bits 14:11 Reserved, must be kept at reset value.

Bits 10:0 **MFC[10:0]**: Dropped packet counters

This counter indicates the number of packet counters that are dropped by the DMA either because of bus error or because of programing RPF field in *Channel x receive control register (ETH_DMACxRXCR)*. The counter gets cleared when this register is read.

Ethernet DMA register map and reset values

Table 846. ETH_DMA common register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1000	ETH_DMAMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTM[1:0]		Res.	Res.	Res.	Res.	TXPR	Res.	Res.	DSPW	Res.	Res.	Res.	TAA[2:0]			Res.	SWR
	Reset value															0	0					0			0				0	0	0		0
0x1004	ETH_DMASBMR	EN_LPI	LPI_XIT_PKT	Res.	Res.	Res.	Res.	WR_OSR_LMT[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	RD_OSR_LMT[1:0]		Res.	Res.	ONEKBBE	AAL	Res.	AALE	Res.	Res.	BLEN256	BLEN128	BLEN64	BLEN32	BLEN16	BLEN8	BLEN4	FB
	Reset value	0	0					0	1							0	1			0	0		0			0	0	0	0	0	0	0	0
0x1008	ETH_DMAISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MACIS	MTLIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DC1IS	DC0IS
	Reset value															0	0															0	0

Table 846. ETH_DMA common register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x100C	ETH_DMADSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TPS1[3:0]				RPS1[3:0]				TPS0[3:0]				RPS0[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	AXRHSTS	AXWHSTS
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							0	0
0x1010 - 0x101C	Reserved																																
0x1020	ETH_DMAA4TXACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	THC[3:0]				Res.	Res.	Res.	Res.	TEC[3:0]				Res.	Res.	Res.	Res.	TDRC[3:0]			
	Reset value													0	0	0	0					0	0	0	0					0	0	0	0
0x1024	ETH_DMAA4RXACR	Res.	Res.	Res.	Res.	RDC[3:0]				Res.	Res.	Res.	Res.	RHC[3:0]				Res.	Res.	Res.	Res.	RPC[3:0]				Res.	Res.	Res.	Res.	RDWC[3:0]			
	Reset value					0	0	0	0					0	0	0	0					0	0	0	0					0	0	0	0
0x1028	ETH_DMAA4DACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDRC[3:0]				Res.	Res.	TDWD[1:0]		TDWC[3:0]			
	Reset value																					0	0	0	0			0	0	0	0	0	0
0x102C - 0x103C	Reserved																																
0x1040	ETH_DMALPIEI	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPIEI[3:0]			
	Reset value																													0	0	0	0
0x1044 - 0x104C	Reserved																																
0x1050	ETH_DMATBSCTRL0R	FTOS[23:0]																								Res.	FGOS[2:0]			Res.	Res.	Res.	FTOV
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0				0

Table 847. ETH_DMA_CH0 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1100	ETH_DMAC0CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DSL[2:0]			Res.	PBLX8	Res.	Res.	MSS[13:0]													
	Reset value												0	0	0		0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1104	ETH_DMAC0TXCR	Res.	Res.		EDSE	TQOS[3:0]				Res.	Res.	TXPBL[5:0]						IPBL	Res.	Res.	TSE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSF	TCW[2:0]		ST
	Reset value				0	0	0	0	0			0	0	0	0	0	0	0			0									0	0	0	0	0

Table 847. ETH_DMA_CH0 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1108	ETH_DMAC0RXCR	RPF	Res.	Res.	Res.	RQOS[3:0]				Res.	Res.	RXPBL[5:0]						Res.	RBSZ[13:0]											Res.	Res.	Res.	SR
	Reset value	0				0	0	0	0			0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0				0
0x110C - 0x1110	Reserved																																
0x1114	ETH_DMAC0TXDLAR	TDESLA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1118	Reserved																																
0x111C	ETH_DMAC0RXDLAR	RDESLA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1120	ETH_DMAC0TXDTPR	TDT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1124	Reserved																																
0x1128	ETH_DMAC0RXDTPR	RDT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x112C	ETH_DMAC0TXRLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDRL[9:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x1130	ETH_DMAC0RXRLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARBS[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDRL[9:0]									
	Reset value									0	0	0	0	0	0	0								0	0	0	0	0	0	0	0	0	0
0x1134	ETH_DMAC0IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NIE	AIE	CDEE	FBEE	ERIE	ETIE	RWTE	RSE	RBUE	RIE	Res.	Res.	Res.	TBUE	TXSE	TIE
	Reset value																	0	0	0	0	0	0	0	0	0	0				0	0	0
0x1138	ETH_DMAC0RXIWTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RWTU[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RWT[7:0]							
	Reset value															0	0									0	0	0	0	0	0	0	0
0x113C	ETH_DMAC0SFCSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSN[3:0]				SIV[11:0]												Res.	Res.	ASC	ESC
	Reset value													0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0			0	0
0x1140	Reserved																																
0x1144	ETH_DMAC0CATXDR	CURTDESAPTR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1148	Reserved																																
0x114C	ETH_DMAC0CARXDR	CURRDESAPTR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1150	Reserved																																

Table 847. ETH_DMA_CH0 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1154	ETH_DMAC0CATXBR	CURTBUFAPTR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1158	Reserved																																
0x115C	ETH_DMAC0CARXBR	CURRBUFAPTR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1160	ETH_DMAC0SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REB[2:0]			TEB[2:0]			NIS	AIS	CDE	FBE	ERI	ETI	RWT	RPS	RBU	RI	Res.	Res.	Res.	TBU	TPS	TI
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0
0x1164	ETH_DMAC0MFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MFCO	Res.	Res.	Res.	Res.	MFC[10:0]										
	Reset value																	0					0	0	0	0	0	0	0	0	0	0	0

Table 848. ETH_DMA_CH1 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1180	ETH_DMAC1CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DSL[2:0]			Res.	PBLX8	Res.	Res.	MSS[13:0]													
	Reset value												0	0	0		0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1184	ETH_DMAC1TXCR	Res.	Res.		EDSE	TQOS[3:0]				Res.	Res.	TXPBL[5:0]						IPBL		Res.	TSE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OSF	TCW[2:0]			ST
	Reset value				0	0	0	0	0			0	0	0	0	0	0	0			0								0	0	0	0	0
0x1188	ETH_DMAC1RXCR	RPF	Res.	Res.	Res.	RQOS[3:0				Res.	Res.	RXPBL[5:0]						Res.	RBSZ[13:0]												Res.	Res.	Res.	SR
	Reset value	0				0	0	0	0			0	0	0	0	0	0		0	0	0	0	0	0	0	0	0	0	0				0
0x118C - 0x1190	Reserved																																
0x1194	ETH_DMAC1TXDLAR	TDESLA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x1198	Reserved																																
0x119C	ETH_DMAC1RXDLAR	RDESLA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x11A0	ETH_DMAC1TXDTPR	TDT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x11A4	Reserved																																
0x11A8	ETH_DMAC1RXDTPR	RDT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 848. ETH_DMA_CH1 register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x11AC	ETH_DMAC1TXRLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TDRL[9:0]									
	Reset value																							0	0	0	0	0	0	0	0	0	0
0x11B0	ETH_DMAC1RXRLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ARBS[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	RDRL[9:0]									
	Reset value									0	0	0	0	0	0	0								0	0	0	0	0	0	0	0	0	0
0x11B4	ETH_DMAC1IER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NIE	AIE	CDEE	FBEE	ERIE	ETIE	RWTE	RSE	RBUE	RIE	Res.	Res.	Res.	TBUE	TXSE	TIE
	Reset value																	0	0	0	0	0	0	0	0	0	0				0	0	0
0x11B8	ETH_DMAC0RXIWTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RWTU[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RWT[7:0]							
	Reset value															0	0									0	0	0	0	0	0	0	0
0x11BC	ETH_DMAC1SFCSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RSN[3:0]				SIV[11:0]												Res.	Res.	ASC	ESC
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			0	0
0x11C0	Reserved																																
0x11C4	ETH_DMAC1CATXDR	CURTDESAPTR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x11C8	Reserved																																
0x11CC	ETH_DMAC1CARXDR	CURRDESAPTR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x11D0	Reserved																																
0x11D4	ETH_DMAC1CATXBR	CURTBUFAPTR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
011D8	Reserved																																
0x11DC	ETH_DMAC1CARXBR	CURRBUFAPTR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x11E0	ETH_DMAC1SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REB[2:0]			TEB[2:0]			NIS	AIS	CDE	FBE	ERI	ETI	RWT	RPS	RBU	RI	Res.	Res.	Res.	TBU	TPS	TI
	Reset value											0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0
0x11E4	ETH_DMAC1MFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MFCO	Res.	Res.	Res.	Res.	MFC[10:0]										
	Reset value																	0					0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

76.11.3 Ethernet MTL registers

Operating mode register (ETH_MTLOMR)

Address offset: 0x0C00

Reset value: 0x0000 0000

The operating mode register establishes the transmit and receive operating modes and commands.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	CNT CLR	CNT PRST	Res.	SCHALG[1:0]		Res.	Res.	RAA	DTX STS	Res.
						rw	rw		rw	rw			rw	rw	

Bits 31:10 Reserved, must be kept at reset value.

Bit 9 **CNTCLR**: Counters reset

When this bit is set, all counters are reset. This bit is cleared automatically after 1 clock cycle. If this bit is set along with CNTPRST bit, CNTPRST has precedence.

Bit 8 **CNTPRST**: Counters preset

When this bit is set:

- *Tx queue x underflow register (ETH_MTLTXQxUR)* is initialized/preset to 0x7F0.
- Missed Packet and Overflow Packet counters in *Rx queue x missed packet and overflow counter register (ETH_MTLRXQxMPOCR)* is initialized/preset to 0x7F0

This bit is cleared automatically.

Bit 7 Reserved, must be kept at reset value.

Bits 6:5 **SCHALG[1:0]**: Tx scheduling algorithm

This field indicates the algorithm for Tx scheduling:
00: Weighted round robin (WRR) algorithm
01 to 10: Reserved, must not be used.
11: Strict priority (SP) algorithm.

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **RAA**: Receive arbitration algorithm

This field is used to select the arbitration algorithm for the Rx side.
0: Strict priority (SP). Queue 0 has the lowest priority and the last queue has the highest priority.
1: Weighted strict priority (WSP)

Bit 1 **DTXSTS**: Drop transmit status

When this bit is set, the Tx packet status received from the MAC is dropped in the MTL. When this bit is reset, the Tx packet status received from the MAC is forwarded to the application.

Bit 0 Reserved, must be kept at reset value.

Interrupt status register (ETH_MTLISR)

Address offset: 0x0C20

Reset value: 0x0000 0000

The software driver (application) reads this register during interrupt service routine or polling to determine the interrupt status of MTL queues and the MAC.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ESTIS	Res.	Res.
													r		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Q1IS	Q0IS
														r	r

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **ESTIS**: EST (TAS- 802.1Qbv) Interrupt Status
This bit indicates an interrupt event during the operation of 802.1Qbv. To reset this bit, the application must clear the error/event that has caused the Interrupt.

Bits 17:2 Reserved, must be kept at reset value.

Bit 1 **Q1IS**: Queue 1 interrupt status
This bit indicates that an interrupt has been generated by Queue 1. To reset this bit, read ETH_MTLQ1ICSR register to identify the interrupt cause and clear the source.

Bit 0 **Q0IS**: Queue 0 interrupt status
This bit indicates that an interrupt has been generated by Queue 0. To reset this bit, read ETH_MTLQ0ICSR register to identify the interrupt cause and clear the source.

Rx queue and DMA channel mapping register (ETH_MTLRXQDMAMR)

Address offset: 0x0C30

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Q1DDMACH	Res.	Res.	Res.	Q1MDMACH	Res.	Res.	Res.	Q0DDMACH	Res.	Res.	Res.	Q0MDMACH
			rw				rw				rw				rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **Q1DDMACH**: Queue 1 Enabled for DA-based DMA Channel Selection
When set, this bit indicates that the packets received in Queue 1 are routed to a particular DMA channel as decided in the MAC receiver based on the DMA channel number programmed in the L3-L4 filter registers, or the Ethernet DA address.
When reset, this bit indicates that the packets received in Queue 1 are routed to the DMA Channel programmed in the Q1MDMACH bit.

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **Q1MDMACH**: Queue 1 Mapped to DMA Channel

This field controls the routing of the received packet in Queue 1 to the DMA channel:
0: DMA Channel 0
1: DMA Channel 1
This field is valid when the Q1DDMACH field is reset.

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **Q0DDMACH**: Queue 0 Enabled for DA-based DMA Channel Selection

When set, this bit indicates that the packets received in Queue 0 are routed to a particular DMA channel as decided in the MAC receiver based on the DMA channel number programmed in the L3-L4 filter registers, or the Ethernet DA address.
When reset, this bit indicates that the packets received in Queue 0 are routed to the DMA Channel programmed in the Q0MDMACH bit.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **Q0MDMACH**: Queue 0 Mapped to DMA Channel

This field controls the routing of the received packet in Queue 0 to the DMA channel:
0: DMA Channel 0
1: DMA Channel 1
This field is valid when the Q0DDMACH field is reset.

TBS control register (ETH_MTLTBSCR)

Address offset: 0x0C40

Reset value: 0x0000 0000

This register controls the time-based scheduling operation.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LEOS[23:8]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LEOS[7:0]								Res.	LEGOS[2:0]			Res.	Res.	LEOV	ESTM
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw			rw	rw

Bits 31:8 **LEOS[23:0]**: Launch Expiry Offset

The value in units of 256 nanoseconds that has to be added to the launch time to compute the launch expiry time. The value is valid only when LEOV is set.

The maximum value is 999,999,999 ns. It must be smaller than CTR - 1 when the ESTM mode is set since this value is a modulo CTR value.

Bit 7 Reserved, must be kept at reset value.

Bits 6:4 **LEGOS[2:0]**: Launch Expiry GSN Offset

The number of GSN slots that has to be added to the launch GSN to compute the launch expiry time. This value is valid only when LEOV is set.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **LEOV**: Launch expiry offset valid

When set, this bit indicates that the LEOS field is valid. When not set, it indicates that the launch expiry offset is not valid and that the MTL must not check for launch expiry time.

0: LEOS field invalid

1: LEOS field valid

Bit 0 **ESTM**: EST offset mode

When this bit is set, the launch time value used in time-based scheduling is interpreted as an EST offset value and is added to the base time register (BTR) of the current list.

When reset, the launch time value is used as an absolute value that must be compared with the system time [39:8].

0: EST offset mode disabled

1: EST offset mode enabled

EST Control register (ETH_MTLESTCR)

Address offset: 0x0C50

Reset value: 0x0000 0000

This register controls the operation of Enhancements to Scheduled Transmission (IEEE802.1Qbv).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PTOV[7:0]								CTOV[11:4]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTOV[3:0]				Res.	TILS[2:0]			LCSE[1:0]		DFBS	DDBF	Res.	Res.	SSWL	EEST
rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw			rw	rw

Bits 31:24 **PTOV[7:0]**: PTP Time Offset Value

The value of PTP Clock period multiplied by 6 in nanoseconds. This value is needed to avoid transmission overruns at the beginning of the installation of a new GCL.

Bits 23:12 **CTOV[11:0]**: Current Time Offset Value

Provides a 12 bit time offset value in nanosecond that is added to the current time to compensate for all the implementation pipeline delays such as the CDC sync delay, buffering delays, data path delays and so on. This offset helps to ensure that the impact of gate controls is visible on the line exactly at the predetermined schedule (or as close to the schedule as possible).

Bit 11 Reserved, must be kept at reset value.

Bits 10:8 **TILS[2:0]**: Time Interval Left Shift Amount

This field provides the left shift amount for the programmed Time Interval values used in the Gate Control Lists.
000: No left shift needed (equal to x1ns)
001: Left shift TI by 1 bit (equal to x2ns)
010: Left shift TI by 2 bits (equal to x4ns)
...
100: Left shift TI by 7 bits (equal to x128ns)

Bits 7:6 **LCSE[1:0]**: Loop Count to report Scheduling Error

Programmable number of GCL list iterations before reporting an HLBS error defined in *EST Status register (ETH_MTLESTSR)*.
00: 4 iterations
01: 8 iterations
10: 16 iterations
11: 32 iterations

Bit 5 **DFBS**: Drop Frames causing Scheduling Error

When set frames reported to cause HOL Blocking due to not getting scheduled (HLBS field of *EST Status register (ETH_MTLESTSR)*) after 4,8,16,32 (based on LCSE field of this register) GCL iterations are dropped.
0: Do not drop
1: Drop

Bit 4 **DDBF**: Do not Drop frames during Frame Size Error

When set, frames are not be dropped during Head-of-Line blocking due to Frame Size Error (HLBF field of *EST Status register (ETH_MTLESTSR)*).

0: Drop

1: Do not drop

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **SSWL**: Switch to S/W owned list

When set, indicates that the software has programmed that list that it currently owns (SWOL) and the hardware should switch to the new list based on the new BTR. Hardware clears this bit when the switch to the SWOL happens to indicate the completion of the switch or when an BTR error (BTRE in *EST Status register (ETH_MTLESTSR)*) is set. When BTRE is set this bit is cleared but SWOL is not updated as the switch was not successful.

Bit 0 **EEST**: Enable EST

When reset, the gate control list processing is halted and all gates are assumed to be in Open state. Should be set for the hardware to start processing the gate control lists. During the toggle from 0 to 1, the gate control list processing starts only after the SSWL bit it set.

0: EST disabled

1: EST enabled

EST Extended Control register (ETH_MTLESTECR)

Address offset: 0x0C54

Reset value: 0x0000 0000

This register indicates the number of Overhead bytes for EST related scheduling.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVHD[5:0]					
										rw	rw	rw	rw	rw	rw

Bits 31:6 Reserved, must be kept at reset value.

Bits 5:0 **OVHD[5:0]**: Overhead Bytes Value

This field indicates the fixed overhead for every packet to account for EST Scheduler Delay, IPG or EIPG, and Preamble bytes.

EST Status register (ETH_MTLESTSR)

Address offset: 0x0C58

Reset value: 0x0000 0000

This register provides the Status related to Enhancements to Scheduled Transmission (IEEE802.1Qbv).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CGSN[3:0]			
												r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BTRL[7:0]								SWOL	Res.	Res.	CGCE	HLBS	HLBF	BTRE	SWLC
r	r	r	r	r	r	r	r	r			rc_w1	r	r	rc_w1	rc_w1

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **CGSN[3:0]**: Current GCL slot number

Indicates the slot number of the GCL list. The slot number is a modulo 16 count of the GCL list loops executed so far. Even if a new GCL list is installed, the count is incremental.

Bits 15:8 **BTRL[7:0]**: BTR Error Loop Count

Provides the minimum count (N) for which the equation Current Time =< New BTR + (N × New Cycle Time) becomes true. N = 0b11111111 indicates the iterations exceeded the value of 128 and the hardware was not able to update New BTR to be equal to or greater than Current Time. Software intervention is needed to update the New BTR. Value cleared when BTRE field of this register is cleared.

Bit 7 **SWOL**: S/W owned list

When 0, indicates Gate Control list number “0” is owned by software and when 1, indicates Gate Control list “1” is owned by the software. Any reads/writes by the software (using indirect access through *EST Gate Control List register (ETH_MTLESTGCLCR)*) are directed to the list indicated by this value by default. The inverse of this value is treated as HWOL. R/W operations performed by hardware are directed to the list pointed by HWOL by default.

0: Inactive

1: Active

Bits 6:5 Reserved, must be kept at reset value.

Bit 4 **CGCE**: Constant Gate Control Error

This error occurs when the list length (LLR) is 1 and the Cycle Time (CTR) is less than or equal to the programmed Time Interval (TI) value after the optional Left Shifting. This implies Gates are either always Closed or always Open based on the Gate Control values; the same effect can be achieved by other simpler (non TSN) programming mechanisms. Since the implementation does not support such a programming an error is reported.

Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.

0: Inactive

1: Active

Bit 3 **HLBS**: Head-Of-Line Blocking due to Scheduling

Set when the frame is not able to win arbitration and get scheduled even after 4 iterations of the GCL. Indicates to software a potential programming error. The one hot encoded values of the Queue Numbers that are not able to make progress are indicated in the *EST Schedule Error register (ETH_MTLESTSCHER)*. Bit cleared when *EST Schedule Error register (ETH_MTLESTSCHER)* is all zeros.

0: Inactive

1: Active

Bit 2 **HLBF**: Head-Of-Line Blocking due to Frame Size

Set when HOL Blocking is noticed on one or more Queues as a result of none of the Time Intervals of gate open in the GCL being greater than or equal to the duration needed for frame size (or frame fragment size when preemption is enabled) transmission. The one hot encoded Queue numbers that are experiencing HLBF are indicated in the *EST frame size error register (ETH_MTLESTFSER)*. Additionally, the first Queue number that experienced HLBF along with the frame size is captured in *EST Frame size Capture register (ETH_MTLESTFSCR)*. HLBF bit is cleared when *EST frame size error register (ETH_MTLESTFSER)* is all zeros.

Bit 1 **BTRE**: BTR Error

When 1, indicates a programming error in the BTR of SWOL where the programmed value is less than current time. If the BTRL = 0b11111111, SWOL is not updated and the software should reprogram the BTR to a value greater than the current time and then set SSWL to reinitiate the switch to SWOL. Else if the value of BTRL < 0b11111111, SWOL is updated and this field indicates the number of iterations (of + CycleTime) taken by hardware to update the BTR to a value greater than the Current Time.
Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.

0: Inactive

1: Active

Bit 0 **SWLC**: Switch to S/W owned list Complete

When 1, indicates the hardware has successfully switched to the SWOL, and the SWOL bit has been updated to that effect. Cleared when the SSWL of *EST Control register (ETH_MTLESTCR)* transitions from 0 to 1, or on a software write.
Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.

0: Inactive

1: Active

EST Schedule Error register (ETH_MTLESTSCHER)

Address offset: 0x0C60

Reset value: 0x0000 0000

This register provides the one hot encoded queue numbers that are having the scheduling related error (timeout)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEQN[1:0]	
														rc_w1	rc_w1

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **SEQN[1:0]**: Schedule Error Queue Number
The One Hot Encoded Queue Numbers that have experienced error/timeout described in HLBS field of *EST Status register (ETH_MTLESTSR)*.
Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.

EST frame size error register (ETH_MTLESTFSER)

Address offset: 0x0C64

Reset value: 0x0000 0000

This register provides the one hot encoded queue numbers that are having the Frame Size related error.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FEQN[1:0]	
														rc_w1	rc_w1

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **FEQN[1:0]**: Frame Size Error Queue Number
The One Hot Encoded Queue Numbers that have experienced error described in HLBF field of status register.
Self-set to 1 on internal event. Setting 1 clears. Setting 0 has no effect.

EST Frame size Capture register (ETH_MTLESTFSCR)

Address offset: 0x0C68

Reset value: 0x0000 0000

This register captures the Frame Size and Queue Number of the first occurrence of the Frame Size related error. Up on clearing it captures the data of immediate next occurrence of a similar error.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HBFQ
															r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	HBFS[14:0]														
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **HBFQ**: Queue Number of HLBF

Captures the binary value of the of the first Queue (number) experiencing HLBF error (see HLBF field of *EST Status register (ETH_MTLESTSR)*). Value once written is not altered by any subsequent queue errors of similar nature. Once cleared the queue number of the next occurring HLBF error is captured.

Cleared when *EST frame size error register (ETH_MTLESTFSER)* is all zeros.

Bit 15 Reserved, must be kept at reset value.

Bits 14:0 **HBFS[14:0]**: Frame Size of HLBF

Captures the Frame Size of the dropped frame related to queue number indicated in HBFQ field of this register. The content of this register should be considered invalid if this field is zero.

Cleared when *EST frame size error register (ETH_MTLESTFSER)* is all zeros.

EST Interrupt Enable register (ETH_MTLESTIER)

Address offset: 0x0C70

Reset value: 0x0000 0000

This register implements the Interrupt Enable bits for the various events that generate an interrupt. Bit positions have a 1 to 1 correlation with the status bit positions in *EST Status register (ETH_MTLESTSR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CGCE	IEHS	IEHF	IEBE	IECC
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **CGCE**: Interrupt Enable for CGCE

When set, this bit generates an interrupt when the Constant Gate Control Error occurs and is indicated in the status. When reset, this event does not generate an interrupt.

Bit 3 **IEHS**: Interrupt Enable for HLBS

When set, this bit generates an interrupt when the head-of-line blocking due to a scheduling issue and is indicated in the status. When reset, this event does not generate an interrupt.

Bit 2 **IEHF**: Interrupt Enable for HLBF

When set, this bit generates an interrupt when the head-of-line blocking due to Frame Size error occurs and is indicated in the status. When reset, this event does not generate an interrupt.

Bit 1 **IEBE**: Interrupt Enable for BTR Error

When set, this bit generates an interrupt when the BTR error occurs and is indicated in the status. When reset, this event does not generate an interrupt.

Bit 0 **IECC**: Interrupt Enable for Switch List

When set, this bit generates interrupt when the configuration change is successful and the hardware has switched to the new list. When reset, this event does not generate an interrupt.

EST Gate Control List register (ETH_MTLESTGCLCR)

Address offset: 0x0C80

Reset value: 0x0000 0000

This register provides the control information for reading/writing to the Gate Control lists.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ADDR[5:0]						Res.	Res.	DBGB	DBGM	Res.	GCRR	R1W0	SRWO
		rw	rw	rw	rw	rw	rw			rw	rw		rw	rw	rc_w1

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:8 **ADDR[5:0]**: Gate Control List Address:

Condition: GCRR = 0

This bitfield provides the address (row number) of the gate control list at which the R/W operation has to be performed. By default the gate control list pointed by SWOL of *EST Status register (ETH_MTLESTSR)* is selected for R/W, however if the DBGM bit of this register is set, a debug mode access is given to read/write from DBGB.

Condition: GCRR = 1

By default the GCL related register set pointed by SWOL of *EST Status register (ETH_MTLESTSR)* is selected for R/W, however if the DBGM bit of this register is set, a debug mode access is given to R/W from DBGB. Lower 3 bits are only used in this mode, higher order bits are treated as don't care.

000: BTR Low (31:0)
001: BTR High (63:31)
010: CTR Low (31:0)
011: CTR High (39:32)
100: TER (30:0)
101: LLR (6:0)
Others: Reserved, must not be used

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **DBGB**: Debug Mode Bank Select

When set to 0, this bit indicates that R/W in Debug mode should be directed to Bank 0 (GCL0 and corresponding Time-related registers). When set to 1, it indicates that R/W in Debug mode should be directed to Bank 1 (GCL1 and corresponding Time-related registers). This value is used when DBGM is set and overrides by value of SWOL which is normally used.

0: Bank 0
1: Bank 1

Bit 4 **DBGM**: Debug Mode

When set to 1, this bit indicates R/W in Debug mode where the memory bank (for GCL and Time related registers) is explicitly provided by DBGB value, when set. When set to 0, SWOL bit is used to determine which bank to use.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **GCRR**: Gate Control Related registers

When set to 1, this bit indicates the R/W access is directed to the GCL related registers (BTR, CTR, TER, LLR) whose address is provided by GCRA. When 0, it indicates R/W operations should be directed to GCL from the address provided by GCLA.

Bit 1 **R1W0**: Read 1, Write 0

1: Read operation

0: Write operation.

Bit 0 **SRWO**: Start read/write Operation

1: Indicates the start/progress of a read/write operation

Reset by hardware: Indicates the R/W operation has completed or an error has occurred (when bit 20 is set)

- Reads: Data can be read from *EST Gate Control List Data register (ETH_MTLESTGCLDR)* after this bit is reset.
- Writes: *EST Gate Control List Data register (ETH_MTLESTGCLDR)* should be programmed with write data before setting SRWO.

EST Gate Control List Data register (ETH_MTLESTGCLDR)

Address offset: 0x0C84

Reset value: 0x0000 0000

This register holds the read data or write data in case of reads and writes respectively.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GCD[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GCD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **GCD[31:0]**: Gate Control Data

The data corresponding to the address selected in *EST Gate Control List register (ETH_MTLESTGCLCR)*. Used both for Read and Write operations.

FPE Frame Preemption Control Status register (ETH_MTLFPECSR)

Address offset: 0x0C90

Reset value: 0x0000 0000

This register provides the control information for reading/writing to the Gate Control lists.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	HRS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			rw												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	PEC[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	AFSZ[1:0]	
						rw	rw							rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **HRS**: Hold/Release Status

1: Indicates a Set-and-Hold-MAC operation was last executed and the pMAC is in Hold State.
0: Indicates a Set-and-Release-MAC operation was last executed and the pMAC is in Release State.

Bits 27:10 Reserved, must be kept at reset value.

Bits 9:8 **PEC[1:0]**: Preemption Classification

When set, this bit indicates the corresponding Queue must be classified as preemptable, when 0, the Queue is classified as express. When both EST (Qbv) and Preemption are enabled, Queue 0 is always assumed to be preemptable. When EST (Qbv) is enabled Queues categorized as preemptable here are always assumed to be in “Open” state in the Gate Control List.

Bits 7:2 Reserved, must be kept at reset value.

Bits 1:0 **AFSZ[1:0]**: Additional Fragment Size

Used to indicate, in units of 64 bytes, the minimum number of bytes over 64 bytes required in non-final fragments of preempted frames. The minimum non-final fragment size is (AFSZ+1) × 64 bytes

FPE Frame Preemption Advance register (ETH_MTLFPEAR)

Address offset: 0x0C94

Reset value: 0x0000 0000

This register holds the Hold and Release Advance time.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RADV[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HADV[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **RADV[15:0]**: Release Advance

The maximum time in nanoseconds that can elapse between issuing a RELEASE to the MAC and the MAC being ready to resume transmission of preemptable frames, in the absence of there being any express frames available for transmission.

Bits 15:0 **HADV[15:0]**: Hold Advance

The maximum time in nanoseconds that can elapse between issuing a HOLD to the MAC and the MAC ceasing to transmit any preemptable frame that is in the process of transmission or any preemptable frames that are queued for transmission.

Tx queue x operating mode register (ETH_MTLTXQxOMR)

Address offset: 0x0D00 + 0x40 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The queue x transmit operating mode register establishes the transmit queue operating modes and commands.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TQS[3:0]			
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TTC[2:0]			TXQEN[1:0]		TSF	FTQ
									rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **TQS[3:0]**: Transmit queue size

This field indicates the size of the allocated transmit queues in blocks of 256 bytes. Queue size range from 256 bytes (TQS = 0b0000) to 4096 bytes (TQS = 0b1111).

Bits 15:7 Reserved, must be kept at reset value.

Bits 6:4 **TTC[2:0]**: Transmit threshold control

These bits control the threshold level of the MTL Tx queue. The transmission starts when the packet size within the MTL Tx queue is larger than the threshold. In addition, full packets with length less than the threshold are also transmitted. These bits are used only when the TSF bit is reset.

000: 32
001: 64
010: 96
011: 128
100: 192
101: 256
110: 384
111: 512

Bits 3:2 **TXQEN[1:0]**: Transmit queue enable

This field is used to enable/disable the transmit queue 0.

00: Not enabled
01: Enable in AV mode
10: Enabled
Others: Reserved, must not be used.

Note: In multiple Tx queues configuration, all the queues are disabled by default. Enable the Tx queue by programming this field.

Bit 1 **TSF**: Transmit store and forward

When this bit is set, the transmission starts when a full packet resides in the MTL Tx queue. When this bit is set, the TTC values specified in Bits[6:4] of this register are ignored. This bit should be changed only when the transmission is stopped.

Bit 0 **FTQ**: Flush transmit queue

When this bit is set, the Tx queue controller logic is reset to its default values. Therefore, all the data in the Tx queue is lost or flushed. This bit is internally reset when the flushing operation is complete. Until this bit is reset, you should not write to the ETH_MTLTXQ1OMR register. The data which is already accepted by the MAC transmitter is not flushed. It is scheduled for transmission and results in underflow and runt packet transmission.

Note: The flush operation is complete only when the Tx queue is empty and the application has accepted the pending Tx Status of all transmitted packets. To complete this flush operation, the PHY Tx clock (eth_tx_clk) should be active.

Tx queue x underflow register (ETH_MTLTXQxUR)

Address offset: 0x0D04 + 0x40 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The queue x underflow counter register contains the counter for packets aborted because of transmit queue underflow and packets missed because of receive queue packet flush

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	UFCNT OVF	UFFRMCNT[10:0]										
				rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **UFCNTOVF**: Overflow Bit for Underflow packet counter

This bit is set every time the Tx queue Underflow packet counter field overflows, that is, it has crossed the maximum count. In such a scenario, the overflow packet counter is reset to all-zeros and this bit indicates that the rollover happened.

Bits 10:0 **UFFRMCNT[10:0]**: Underflow packet counter

This field indicates the number of packets aborted by the controller because of Tx queue Underflow. This counter is incremented each time the MAC aborts outgoing packet because of underflow. The counter is cleared when this register is read.

Tx queue x debug register (ETH_MTLTXQxDR)

Address offset: 0x0D08 + 0x40 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The queue x transmit debug register gives the debug status of various blocks related to the transmit queue.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STXSTSF[2:0]			Res.	PTXQ[2:0]		
									r	r	r		r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXSTS FSTS	TXQ STS	TWC STS	TRCSTS[1:0]		TXQPA USED
										r	r	r	r	r	r

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:20 **STXSTSF[2:0]**: Number of status words in Tx status FIFO of queue

This field indicates the current number of status in the Tx Status FIFO of this queue.
When the DTXSTS bit of ETH_MTLOMR register is set to 1, this field does not reflect the number of status words in Tx Status FIFO.

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **PTXQ[2:0]**: Number of packets in the transmit queue

This field indicates the current number of packets in the Tx queue.
When the DTXSTS bit of *Operating mode register (ETH_MTLOMR)* register is set to 1, this field does not reflect the number of packets in the transmit queue.

Bits 15:6 Reserved, must be kept at reset value.

Bit 5 **TXSTSFSTS**: MTL Tx status FIFO Full status

When high, this bit indicates that the MTL Tx Status FIFO is full. Therefore, the MTL cannot accept any more packets for transmission.

Bit 4 **TXQSTS**: MTL Tx queue not empty status

When this bit is high, it indicates that the MTL Tx queue is not empty and some data is left for transmission.

Bit 3 **TWCSTS**: MTL Tx queue write controller status

When high, this bit indicates that the MTL Tx queue Write Controller is active, and it is transferring the data to the Tx queue.

Bits 2:1 **TRCSTS[1:0]**: MTL Tx queue read controller status
This field indicates the state of the Tx Queue Read Controller:
00: Idle state
01: Read state (transferring data to the MAC transmitter)
10: Waiting for pending Tx Status from the MAC transmitter
11: Flushing the Tx queue because of the Packet Abort request from the MAC

Bit 0 **TXQPAUSED**: Transmit queue in pause
When this bit is high and the Rx flow control is enabled, it indicates that the Tx queue is in the Pause condition (in the Full-duplex only mode) because of the following:
– Reception of the PFC packet for the priorities assigned to the Tx queue when PFC is enabled
– Reception of 802.3x Pause packet when PFC is disabled

Tx queue x ETS status register (ETH_MTLTXQxESR)

Address offset: 0x0D14 + 0x40 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The Queue x ETS Status register provides the average traffic transmitted in Queue x.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ABS[23:16]							
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ABS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:0 **ABS[23:0]**: Average Bits per Slot
This field contains the average transmitted bits per slot.

Queue x interrupt control status register (ETH_MTLQxICSR)

Address offset: 0x0D2C + 0x40 * x, (x = 0 to 1)

Reset value: 0x0000 0000

This register contains the interrupt enable and status bits for the queue x interrupts.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXOIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXOVFIS
							rw								rc_w1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	ABPSIE	TXUIE	Res.	Res.	Res.	Res.	Res.	Res.	ABPSIS	TXUNFIS
						rw	rw							rw	rc_w1

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **RXOIE**: Receive queue overflow interrupt enable

When this bit is set, the receive queue overflow interrupt is enabled. When this bit is reset, the Receive Queue Overflow interrupt is disabled.

Bits 23:17 Reserved, must be kept at reset value.

Bit 16 **RXOVFIS**: Receive queue overflow interrupt status

This bit indicates that the receive queue had an overflow while receiving the packet. If a partial packet is transferred to the application, the overflow status is set in RDES3[21]. This bit is cleared when the application writes 1 to this bit.

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **ABPSIE**: Average bits per slot interrupt enable

When this bit is set, the MAC asserts the eth_sbd_intr_it interrupt when the average bits per slot status is updated.

When this bit is cleared, the interrupt is not asserted for such an event.

Bit 8 **TXUIE**: Transmit queue underflow interrupt enable

When this bit is set, the transmit queue underflow interrupt is enabled. When this bit is reset, the Transmit Queue Underflow interrupt is disabled.

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **ABPSIS**: Average bits per slot interrupt status

When set, this bit indicates that the MAC has updated the ABS value. This bit is cleared when the application writes 1 to this bit.

Note: This bit is automatically set after 10 ms in 1000 Mbps mode (100 ms in 100 Mbps), once the reset is released, regardless of the Ethernet controller configuration.

Bit 0 **TXUNFIS**: Transmit queue underflow interrupt status

This bit indicates that the transmit queue had an underflow while transmitting the packet. Transmission is suspended and an Underflow Error TDES3[2] is set. This bit is cleared when the application writes 1 to this bit.

Rx queue x operating mode register (ETH_MTLRXQxOMR)

Address offset: 0x0D30 + 0x40 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The queue x receive operating mode register establishes the receive queue operating modes and command.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RQS[3:0]				Res.	Res.	Res.	RFD [2]
								rw	rw	rw	rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RFD[1:0]		Res.	Res.	Res.	RFA[2:0]			EHFC	DIS_ TCP_EF	RSF	FEP	FUP	Res.	RTC[1:0]	
rw	rw				rw	rw	rw	rw	rw	rw	rw	rw		rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:20 **RQS[3:0]**: Receive Queue Size

This field indicates the size of the allocated receive queues in blocks of 256 bytes. The size of the queues is (RQS + 1) × 256 bytes.

Bits 19:17 Reserved, must be kept at reset value.

Bits 16:14 **RFD[2:0]**: Threshold for Deactivating Flow Control (in Half-duplex and Full-duplex modes)

These bits control the threshold (fill-level of Rx queue) at which the flow control is deasserted after activation:

0: Full minus 1 Kbyte

1: Full minus 1.5 Kbyte

2: Full minus 2 Kbytes

3: Full minus 2.5 Kbytes

4: Full minus 3 Kbytes

5: Full minus 3.5 Kbytes

Other values: reserved

The deassertion is effective only after flow control is asserted.

Note: The value must be programmed in such a way to make sure that the threshold is a positive number.

To have the flow control activated for a fill level of 3 Kbytes and deactivated when the fill level is 1 Kbyte, program:

RFA[2:0] = 0 (full – 1 Kbyte)

RFD[2:0] = 4 (full – 3 Kbytes)

Bits 13:11 Reserved, must be kept at reset value.

Bits 10:8 **RFA[2:0]**: Threshold for Activating Flow Control (in Half-duplex and Full-duplex)

These bits control the threshold (fill-level of Rx queue) at which the flow control is activated: For more information on encoding for this field, see RFD[2:0].

Bit 7 **EHFC**: Enable Hardware Flow Control

When this bit is set, the flow control signal operation, based on the fill-level of Rx queue, is enabled. When reset, the flow control operation is disabled.

Bit 6 **DIS_TCP_EF**: Disable Dropping of TCP/IP Checksum Error Packets

When this bit is set, the MAC does not drop the packets which only have the errors detected by the receive checksum offload engine. Such packets have errors only in the encapsulated payload. There are no errors (including FCS error) in the Ethernet packet received by the MAC.

When this bit is reset, all error packets are dropped if the FEP bit is reset.

Bit 5 **RSF**: Receive Queue Store and Forward

When this bit is set, the Ethernet peripheral reads a packet from the Rx queue only after the complete packet has been written to it, ignoring the RTC field of this register. When this bit is reset, the Rx queue operates in the Threshold (cut-through) mode, subject to the threshold specified by the RTC field of this register.

Bit 4 **FEP**: Forward Error Packets

When this bit is reset, the Rx queue drops packets with error status (CRC error, receive error, watchdog timeout, or overflow). However, if the start byte (write) pointer of a packet is already transferred to the read controller side (in Threshold mode), the packet is not dropped.

When this bit is set, all packets except the runt error packets are forwarded to the application or DMA. If the RSF bit is set and the Rx queue overflows when a partial packet is written, the packet is dropped irrespective of the setting of this bit. However, if the RSF bit is reset and the Rx queue overflows when a partial packet is written, a partial packet may be forwarded to the application or DMA.

Bit 3 **FUP**: Forward Undersized Good Packets

When this bit is set, the Rx queue forwards the undersized good packets (packets with no error and length less than 64 bytes), including pad-bytes and CRC. When this bit is reset, the Rx queue drops all packets of less than 64 bytes, unless a packet is already transferred because of the lower value of Rx Threshold, for example, RTC = 01.

Bit 2 Reserved, must be kept at reset value.

Bits 1:0 **RTC[1:0]**: Receive Queue Threshold Control

These bits control the threshold level of the MTL Rx queue (in bytes):
00: 64
01: 32
10: 96
11: 128
The received packet is transferred to the application or DMA when the packet size within the MTL Rx queue is larger than the threshold. In addition, full packets with length less than the threshold are automatically transferred.
This field is valid only when the RSF bit is zero. This field is ignored when the RSF bit is set to 1.

Rx queue x missed packet and overflow counter register (ETH_MTLRXQxMPOCR)

Address offset: 0x0D34 + 0x40 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The Queue x missed packet and overflow counter registers contain the counter for packets missed because of Receive queue packet flush and packets discarded because of Receive queue overflow.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	MISCNTOVF	MISPKTCNT[10:0]										
				rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	OVFCNTOVF	OVFPKTCNT[10:0]										
				rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **MISCNTOVF**: Missed packet counter Overflow Bit

When set, this bit indicates that the Rx Queue Missed packet counter crossed the maximum limit.

Bits 26:16 **MISPKTCNT[10:0]**: Missed packet counter

This field indicates the number of packets missed by the Ethernet peripheral because the application requested to flush the packets for this queue. This counter is reset when this register is read.
This counter is incremented by 1 when the DMA discards the packet because of buffer unavailability.

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **OVFCNTOVF**: Overflow Counter Overflow Bit

When set, this bit indicates that the Rx Queue Overflow packet counter field crossed the maximum limit.

Bits 10:0 **OVFPKTCNT[10:0]**: Overflow packet counter

This field indicates the number of packets discarded by the Ethernet peripheral because of Receive queue overflow. This counter is incremented each time the Ethernet peripheral discards an incoming packet because of overflow. This counter is reset when this register is read.

Rx queue x debug register (ETH_MTLRXQxDR)

Address offset: 0x0D38 + 0x40 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The Queue x Receive Debug register gives the debug status of various blocks related to the Receive queue.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	PRXQ[13:0]													
		r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXQSTS[1:0]		Res.	RRCSTS[1:0]		RWCSTS
										r	r		r	r	r

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:16 **PRXQ[13:0]**: Number of Packets in Receive Queue

This field indicates the current number of packets in the Rx queue. The theoretical maximum value for this field is 256 Kbyte/16 bytes = 16K Packets, that is, Max_Queue_Size/Min_Packet_Size.

Bits 15:6 Reserved, must be kept at reset value.

Bits 5:4 **RXQSTS[1:0]**: MTL Rx Queue Fill-Level Status

This field gives the status of the fill-level of the Rx queue:
00: Rx queue empty
01: Rx queue fill-level below flow-control deactivate threshold
10: Rx queue fill-level above flow-control activate threshold
11: Rx queue full

Bit 3 Reserved, must be kept at reset value.

Bits 2:1 **RRCSTS[1:0]**: MTL Rx Queue Read Controller State

This field gives the state of the Rx queue Read controller:
00: Idle state
01: Reading packet data
10: Reading packet status (or timestamp)
11: Flushing the packet data and status

Bit 0 **RWCSTS**: MTL Rx Queue Write Controller Active Status

When high, this bit indicates that the MTL Rx queue Write controller is active, and it is transferring a received packet to the Rx queue.

Rx queue x control register (ETH_MTLRXQxCR)

Address offset: 0x0D3C + 0x40 * x, (x = 0 to 1)

Reset value: 0x0000 0000

The Queue Receive Control register controls the receive arbitration and passing of received packets to the application.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXQ_FRM_ARBIT	RXQ_WEGT[2:0]		
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **RXQ_FRM_ARBIT**: Receive Queue Packet Arbitration

When this bit is set, the Ethernet peripheral drives the packet data to the DMA such that the entire packet data of currently-selected queue is transmitted before switching to other queue.

When this bit is reset, the Ethernet peripheral drives the packet data to the DMA such that the following amount of data of currently-selected queue is transmitted before switching to other queue:

– PBL amount of data

or

– complete data of a packet

The status and the timestamp are not a part of the PBL data. Therefore, the Ethernet peripheral drives the complete status (including timestamp status) during first PBL request for the packet (in store-and-forward mode) or the last PBL request for the packet (in Threshold mode).

Bits 2:0 **RXQ_WEGT[2:0]**: Receive Queue Weight

This field indicates the weight assigned to the Rx queue x. The weight is used as the number of continuous PBL requests or contiguous packets (depending on the RXQ_FRM_ARBIT) allocated to the queue in one arbitration cycle.

This field needs to be programmed with one value less than the required weight, that is, reset value of 0 indicates weight of 1, value of 1 indicates weight of 2, and so on.

Note: The change in value of RXQ_WEGT takes effect only after the completion of current service round or when there is change from RAA=SP to RAA=WSP algorithm. This approach is taken so that there is smooth transition. For the RXQ_WEGT value to take effect at the start, the Rx queue x control register (ETH_MTLRXQxCR) must be programmed before the Rx queue x operating mode register (ETH_MTLRXQxOMR).

Tx queue 1 ETS control register (ETH_MTLTXQ1ECR)

Address offset: 0x0D50

Reset value: 0x0000 0000

The Queue ETS Control register controls the enhanced transmission selection operation.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLC[2:0]			CC	AVALG	Res.	Res.
									rw	rw	rw	rw	rw		

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:4 **SLC[2:0]**: Slot Count

If the credit-based shaper algorithm is enabled, the software can program the number of slots (of 125 us duration) over which the average transmitted bits per slot, provided in *Tx queue x ETS status register (ETH_MTLTXQxESR)*, need to be computed for Queue 1. The encoding is as follows:
000: 1 Slot
001: 2 Slots
010: 4 Slots
011: 8 Slots
100: 16 Slots
101 to 111: Reserved, must not be used.

Bit 3 **CC**: Credit Control

When this bit is set, the accumulated credit parameter in the credit-based shaper algorithm logic is not reset to zero when there is positive credit and no packet to transmit in Channel 1. The credit accumulates even when there is no packet waiting in Channel 1 and another channel is transmitting.
When this bit is reset, the accumulated credit parameter in the credit-based shaper algorithm logic is set to zero when there is positive credit and no packet to transmit in Channel 1. When there is no packet waiting in Channel 1 and other channel is transmitting, no credit is accumulated.

Bit 2 **AVALG**: AV Algorithm

When Queue 1 is programmed for AV, this field configures the scheduling algorithm for this queue:
This bit when set, indicates credit based shaper algorithm (CBS) is selected for Queue 1 traffic. When reset, strict priority is selected.

Bits 1:0 Reserved, must be kept at reset value.

Tx queue x quantum weight register (ETH_MTLTXQxQWR)

Address offset: 0x0D18 + 0x40 * x, (x = 0 to 1)

Reset value: 0x0000 0000

This register contains the weights for the Weighted Round Robin (WRR) or IdleSlopeCredit for the queue.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	ISCQW[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:0 **ISCQW[13:0]**: IdleSlopeCredit or Weights

idleSlopeCredit

When AV feature is enabled, this field contains the idleSlopeCredit value required for the credit-based shaper algorithm for queue 1. This is the rate of change of credit in bits per cycle (40 ns for 100 Mbps; 8 ns for 1000 Mbps) when the credit is increasing. The software should program this field with computed credit in bits per cycle scaled by 1,024. The maximum value is portTransmitRate, that is, 0x2000 in 1000 Mbps mode and 0x1000 in 100 Mbps mode.

Weights

If WRR algorithm is enabled for queue x generic traffic (not AVB), this field contains the weight for queue x. The maximum value is 0x64. Bits [13:7] must be written to zero.

Note: In multiple Queue configuration this field must be programmed to some non-zero value when multiple queues are enabled, or a single queue other than Queue 0 is enabled. This field need not be programmed when only Q0 is enabled. In general, when WRR algorithm is selected a non-zero value must be programmed on both receive and transmit. In receive, the register is Operating mode register (ETH_MTLOMR).

The weights programmed do not correspond to the number of packets but the fraction of bandwidth or time allocated for particular queue w.r.t. total BW or time.

Tx queue 1 send slope credit register (ETH_MTLTXQ1SSCR)

Address offset: 0x0D5C

Reset value: 0x0000 0000

The sendSlopeCredit register contains the sendSlope credit value required for the credit-based shaper algorithm for the Queue.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	SSC[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:0 **SSC[13:0]**: sendSlopeCredit Value

When AV operation is enabled, this field contains the sendSlopeCredit value required for credit-based shaper algorithm for queue 1. This is the rate of change of credit in bits per cycle (40 ns and 8 ns for 100 Mbps and 1000 Mbps respectively) when the credit decreases. The software should program this field with computed credit in bits per cycle scaled by 1 024. The maximum value is portTransmitRate, that is 0x2000 in 1000 Mbps mode and 0x1000 in 100 Mbps mode. This field should be programmed with absolute sendSlopeCredit value. The credit-based shaper logic subtracts it from the accumulated credit when queue 1 is selected for transmission.

Tx Queue 1 hiCredit register (ETH_MTLTXQ1HCR)

Address offset: 0x0D60

Reset value: 0x0000 0000

The hiCredit register contains the hiCredit value required for the credit-based shaper algorithm for the Queue.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	HC[28:16]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:0 **HC[28:0]**: hiCredit Value

This field contains the hiCredit value required for the credit-based shaper algorithm when the AV feature is enabled. This is the maximum value that can be accumulated in the credit parameter. This is specified in bits scaled by 1 024.

The maximum value is maxInterferenceSize, that is, best-effort maximum packet size (16 384 bytes or 131 072 bits). The value to be specified is 131 072 ×1 024 = 134 217 728 or 0x0800 0000.

Tx queue 1 loCredit register (ETH_MTLTXQ1LCR)

Address offset: 0x0D64

Reset value: 0x0000 0000

The loCredit register contains the loCredit value required for the credit-based shaper algorithm for the Queue.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	LC[28:16]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:0 **LC[28:0]**: loCredit Value

When AV operation is enabled, this field contains the loCredit value required for the credit-based shaper algorithm. This is the minimum value that can be accumulated in the credit parameter. This is specified in bits scaled by 1 024. The maximum value is maxFrameSize transmitted from this queue, that is, 16 384 × 8 × 1 024 = 134 217 728 or 0x0800 0000. Because it is a negative value, the programmed value is 2's complement of the value, that is, 0x1800 0000.

Ethernet MTL register map and reset values

Table 849. ETH_MTL register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0C00	ETH_MTLOMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CNTCLR	CNTPRST	Res.	SCHALG[1:0]		Res.	Res.	RAA	DTXSTS	Res.
	Reset value																							0	0		0	0			0	0	
0x0C04 - 0x0C1C	Reserved																																
0x0C20	ETH_MTLISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ESTIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Q1IS	Q0IS
	Reset value														0																	0	0
0x0C24 - 0x0C2C	Reserved																																
0x0C30	ETH_MTLRXQDMAMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Q1DDMACH	Res.	Res.	Res.	Q1MDMACH	Res.	Res.	Res.	Q0DDMACH	Res.	Res.	Res.	Q0MDMACH
	Reset value																				0				0				0				0
0x0C34 - 0x0C3C	Reserved																																
0x0C40	ETH_MTLTBSCR	LEOS[23:0]																								LEGOS[2:0]				Res.	Res.	LEOV	ESTM
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0			0	0
0x0C50	ETH_MTLESTCR	PTOV[7:0]								CTOV[11:0]												Res.	TILS[2:0]			LCSE[1:0]		DFBS	DDBF	Res.	Res.	SSWL	EEST
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0	0			0	0
0x0C54	ETH_MTLESTECR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVHD[5:0]				
	Reset value																											0	0	0	0	0	0
0x0C58	ETH_MTLESTSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CGSN[3:0]				BTRL[7:0]								SWOL	Res.	Res.	CGCE	HLBS	HLBF	BTRE	SWLC
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0			0	0	0	0	0
0x0C5C	Reserved																																
0x0C60	ETH_MTLESTSCHER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEQN[1:0]	
	Reset value																															0	0

Table 849. ETH_MTL register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0C64	ETH_ MTLESTFSER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FEQN[1:0]	
	Reset value																															0	0
0x0C68	ETH_ MTLESTFSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HBFQ	Res.	HBFS[14:0]														
	Reset value																0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C6C	Reserved																																
0x0C70	ETH_MTLESTIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CGCE	IEHS	IEHF	IEBE	IECC
	Reset value																												0	0	0	0	0
0x0C74 - 0x0C7C	Reserved																																
0x0C80	ETH_ MTLESTGCLCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ADDR[5:0]						Res.	Res.	DBGB	DBGM	Res.	GCRR	R1W0	SRWO
	Reset value																			0	0	0	0	0	0			0	0		0	0	0
0x0C84	ETH_ MTLESTGCLDR	GCD[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C88- 0x0C8C	Reserved																																
0x0C90	ETH_ MTLFPECSR	Res.	Res.	Res.	HRS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PEC[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	AFSZ[1:0]	
	Reset value				0																			0	0							0	0
0x0C94	ETH_MTLFPEAR	RADV[15:0]																HADV[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0C98 - 0x0CFC	Reserved																																
0x0D00	ETH_ MTLTXQ0OMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TQS[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TTC[2:0]			TXQEN[1:0]		TSF	FTQ
	Reset value													0	0	0	0										0	0	0	0	0	0	0
0x0D04	ETH_ MTLTXQ0UR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UFCNTOVF	UFFRMCNT[10:0]										
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x0D08	ETH_ MTLTXQ0DR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STXSTSF[2:0]			Res.	PTXQ[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXSTSFSTS	TXQSTS	TWCSTS	TRCSTS[1:0]		TXQPAUSED
	Reset value										0	0	0		0	0	0											0	0	0	0	0	0

Table 849. ETH_MTL register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0D0C - 0x0D10	Reserved																																
0x0D14	ETH_ MTLTXQ0ESR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ABS[23:0]																							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0D18	ETH_ MTLTXQ0QWR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ISCQW[6:0]						
	Reset value																										0	0	0	0	0	0	0
0x0D28	Reserved																																
0x0D2C	ETH_MTLQ0ICSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXOIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXOVFIS	Res.	Res.	Res.	Res.	Res.	Res.	ABPSIE	TXUIE	Res.	Res.	Res.	Res.	Res.	Res.	ABPSIS	TXUNFIS
	Reset value								0								0							0	0							0	0
0x0D30	ETH_ MTLRXQ0OMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RQS[3:0]				Res.	Res.	Res.	RFD[2:0]			Res.	Res.	Res.	RFA[2:0]			EHFC	DIS_TCP_EF	RSF	FEP	FUP	Res.	RTC[1:0]	
	Reset value									0	0	0	0				0	0	0				0	0	0	0	0	0	0	0		0	0
0x0D34	ETH_ MTLRXQ0MPOCR	Res.	Res.	Res.	Res.	MISCNTOVF	MISPKTCNT[10:0]											Res.	Res.	Res.	Res.	OVFCNTOVF	OVFPKTCNT[10:0]										
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x0D38	ETH_ MTLRXQ0DR	Res.	Res.	PRXQ[13:0]														Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXQSTS[1:0]		Res.	RRCSTS[1:0]		RWCSTS
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0										0	0		0	0	0
0x0D3C	ETH_ MTLRXQ0CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXQ_FRM_ARBIT	RXQ_WEGT[2:0]		
	Reset value																													0	0	0	0
0x0D40	ETH_ MTLTXQ1OMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TQS[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TTC[2:0]			TXQEN[1:0]		TSF	FTQ
	Reset value													0	0	0	0										0	0	0	0	0	0	0
0x0D44	ETH_ MTLTXQ1UR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UFCNTOVF	UFFRMCNT[10:0]										
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0

Table 849. ETH_MTL register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0D48	ETH_ MTLTXQ1DR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STXSTSF[2:0]			Res.	PTXQ[2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TXSTSFSTS	TXQSTS	TWCSTS	TRCSTS[1:0]		TXQPAUSED
	Reset value										0	0	0		0	0	0											0	0	0	0	0	0
0x0D4C	Reserved																																
0x0D50	ETH_ MTLTXQ1ECR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLC[2:0]			CC	AVALG	Res.	Res.
	Reset value																										0	0	0	0	0		
0x0D54	ETH_ MTLTXQ1ESR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ABS[23:0]																							
	Reset value									0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0D58	ETH_ MTLTXQ1QWR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ISCQW[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0D5C	ETH_ MTLTXQ1SSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSC[13:0]													
	Reset value																			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0D60	ETH_ MTLTXQ1HCR	Res.	Res.	Res.	HC[28:0]																												
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0D64	ETH_ MTLTXQ1LCR	Res.	Res.	Res.	LC[28:0]																												
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0D68	Reserved																																
0x0D6C	ETH_MTLQ1ICSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXOIE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXOVFIS	Res.	Res.	Res.	Res.	Res.	Res.	ABPSIE	TXUIE	Res.	Res.	Res.	Res.	Res.	Res.	ABPSIS	TXUNFIS
	Reset value								0								0							0	0							0	0
0x0D70	ETH_ MTLRXQ1OMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RQS[3:0]				Res.	Res.	Res.	RFD[2:0]			Res.	Res.	Res.	RFA[2:0]			EHFC	DIS_TCP_EF	RSF	FEP	FUP	Res.	RTC[1:0]	
	Reset value									0	0	0	0				0	0	0				0	0	0	0	0	0	0	0		0	0
0x0D74	ETH_MTLRXQ1MP OCR	Res.	Res.	Res.	Res.	MISCNTOVF	MISPKTCNT[10:0]											Res.	Res.	Res.	Res.	OVFCNTOVF	OVFPKTCNT[10:0]										
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0					0	0	0	0	0	0	0	0	0	0	0	0
0x0D78	ETH_ MTLRXQ1DR	Res.	Res.	PRXQ[13:0]														Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXQSTS[1:0]		Res.	RRCSTS[1:0]		RWCSTS
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0											0	0		0	0	0

Table 849. ETH_MTL register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0D7C	ETH_ MTLRXQ1CR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXQ_FRM_ARBIT	RXQ_WEGT[2:0]		
	Reset value																													0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

76.11.4 Ethernet MAC and MMC registers

Operating mode configuration register (ETH_MACCR)

Address offset: 0x0000

Reset value: 0x0000 0000

The MAC Configuration register establishes the operating mode of the MAC.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARPEN	SARC[2:0]			IPC	IPG[2:0]			GPSLCE	S2KP	CST	ACS	WD	BE	JD	JE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PS	FES	DM	LM	ECRSFD	DO	DCRS	DR	Res.	BL[1:0]		DC	PRELEN[1:0]		TE	RE
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw

Bit 31 **ARPEN**: ARP Offload Enable

When this bit is set, the MAC can recognize an incoming ARP request packet and schedules the ARP packet for transmission. It forwards the ARP packet to the application and also indicate the events in the RxStatus.

When this bit is reset, the MAC receiver does not recognize any ARP packet and indicates them as Type frame in the RxStatus.

Bits 30:28 **SARC[2:0]**: Source Address Insertion or Replacement Control

This field controls the source address insertion or replacement for all transmitted packets. Bit 30 specifies which MAC Address register (0 or 1) is used for source address insertion or replacement based on the values of Bits[29:28]:

010:the MAC inserts the content of the MAC Address 0 registers (*MAC address 0 high register (ETH_MACA0HR)* and *MAC address x low register (ETH_MACAxLR)*) in the SA field of all transmitted packets.

011: the MAC replaces the content of the MAC Address 0 registers (*MAC address 0 high register (ETH_MACA0HR)* and *MAC address x low register (ETH_MACAxLR)*) in the SA field of all transmitted packets.

110: the MAC inserts the content of the MAC Address 1 registers (*MAC address x high register (ETH_MACAxHR)* and *MAC address x low register (ETH_MACAxLR)*) in the SA field of all transmitted packets

111: the MAC replaces the content of the MAC Address 1 registers (*MAC address x high register (ETH_MACAxHR)* and *MAC address x low register (ETH_MACAxLR)*) in the SA field of all transmitted packets.

Others: Reserved, must not be used.

Note: Changes to this field take effect only on the start of a packet. If you write to this register field when a packet is being transmitted, only the subsequent packet can use the updated value, that is, the current packet does not use the updated value.

Bit 27 **IPC**: Checksum Offload

When set, this bit enables the IPv4 header checksum checking and IPv4 or IPv6 TCP, UDP, or ICMP payload checksum checking. When this bit is reset, the COE function in the receiver is disabled.

The Layer 3 and Layer 4 Packet Filter feature automatically selects the IPC Full Checksum Offload Engine on the Receive side. When this feature is enabled, you must set the IPC bit.

Bits 26:24 **IPG[2:0]**: Inter-Packet Gap

These bits control the minimum IPG between packets during transmission.

000: 96 bit times

001: 88 bit times

010: 80 bit times

...

111: 40 bit times

This range of minimum IPG is valid in Full-duplex mode.

In the Half-duplex mode, the minimum IPG can be configured only for 64-bit times (IPG = 100). Lower values are not considered.

When a JAM pattern is being transmitted because of backpressure activation, the MAC does not consider the minimum IPG.

The above function (IPG less than 96 bit times) is valid only when EIPGEN bit in ETH_MACECR register is reset. When EIPGEN is set, then the minimum IPG (greater than 96 bit times) is controlled as per the description given in EIPG field in ETH_MACECR register.

Bit 23 **GPSLCE**: Giant Packet Size Limit Control Enable

When this bit is set, the MAC considers the value in GPSL field in ETH_MACECR register to declare a received packet as Giant packet. This field must be programmed to more than 1,518 bytes. Otherwise, the MAC considers 1,518 bytes as giant packet limit.

When this bit is reset, the MAC considers a received packet as Giant packet when its size is greater than 1,518 bytes (1522 bytes for tagged packet).

The watchdog timeout limit, Jumbo Packet Enable and 2K Packet Enable have higher precedence over this bit, that is the MAC considers a received packet as Giant packet when its size is greater than 9,018 bytes (9,022 bytes for tagged packet) with Jumbo Packet Enabled and greater than 2,000 bytes with 2K Packet Enabled. The watchdog timeout, if enabled, terminates the received packet when watchdog limit is reached. Therefore, the programmed giant packet limit should be less than the watchdog limit to get the giant packet status.

Bit 22 **S2KP**: IEEE 802.3as Support for 2K Packets

When this bit is set, the MAC considers all packets with up to 2,000 bytes length as normal packets. When the JE bit is not set, the MAC considers all received packets of size more than 2K bytes as Giant packets.

When this bit is reset and the JE bit is not set, the MAC considers all received packets of size more than 1,518 bytes (1,522 bytes for tagged) as giant packets. For more information about how the setting of this bit and the JE bit impact the Giant packet status, see *Table 850: Giant Packet Status based on S2KP and JE Bits*.

Note: When the JE bit is set, setting this bit has no effect on the giant packet status.

Bit 21 **CST**: CRC stripping for Type packets

When this bit is set, the last four bytes (FCS) of all packets of Ether type (type field greater than 1,536) are stripped and dropped before forwarding the packet to the application. This function is not valid when the IP Checksum Engine (Type 1) is enabled in the MAC receiver. This function is valid when Type 2 Checksum Offload Engine is enabled.

Note: For information about how the settings of the ACS bit and this bit impact the packet length, see Table 851: Packet Length based on the CST and ACS bits.

Bit 20 **ACS**: Automatic Pad or CRC Stripping

When this bit is set, the MAC strips the Pad or FCS field on the incoming packets only if the value of the length field is less than 1,536 bytes. All received packets with length field greater than or equal to 1,536 bytes are passed to the application without stripping the Pad or FCS field.

When this bit is reset, the MAC passes all incoming packets to the application, without any modification.

Note: For information about how the settings of CST bit and this bit impact the packet length, see Table 851: Packet Length based on the CST and ACS bits.

Bit 19 **WD**: Watchdog Disable

When this bit is set, the MAC disables the watchdog timer on the receiver. The MAC can receive packets of up to 16,383 bytes.

When this bit is reset, the MAC does not allow more than 2,048 bytes (10,240 if JE is set high) of the packet being received. The MAC cuts off any bytes received after 2,048 bytes.

Bit 18 **BE**: Packet Burst Enable

When this bit is set, the MAC allows packet bursting during transmission in the GMII Half-duplex mode.

Bit 17 **JD**: Jabber Disable

When this bit is set, the MAC disables the jabber timer on the transmitter. The MAC can transfer packets of up to 16,383 bytes.

When this bit is reset, if the application sends more than 2,048 bytes of data (10,240 if JE is set high) during transmission, the MAC does not send rest of the bytes in that packet.

Bit 16 **JE**: Jumbo Packet Enable

When this bit is set, the MAC allows jumbo packets of 9,018 bytes (9,022 bytes for VLAN tagged packets) without reporting a giant packet error in the Rx packet status.

For more information about how the setting of this bit and the JE bit impact the Giant packet status, see *Table 850: Giant Packet Status based on S2KP and JE Bits*.

Bit 15 **PS**: Port Select

This bit selects the Ethernet line speed.

0: For 1000 Mbps operations

1: For 10 or 100 Mbps operations

In 10 or 100 Mbps operations, this bit, along with Bit FES, selects the exact line speed.

Bit 14 **FES**: MAC Speed

This bit selects the speed in the 10/100 Mbit/s mode:

0: 10 Mbit/s

1: 100 Mbit/s

Bit 13 **DM**: Duplex Mode

When this bit is set, the MAC operates in the Full-duplex mode in which it can transmit and receive simultaneously.

Bit 12 **LM**: Loopback Mode

When this bit is set, the MAC operates in the loopback mode at GMII or MII. The GMII Rx clock input (eth_rx_clk) is required for the loopback to work properly. This is because the Tx clock is not internally looped back.

Bit 11 **ECRSFD**: Enable Carrier Sense Before Transmission in Full-duplex mode

When this bit is set, the MAC transmitter checks the CRS signal before packet transmission in the Full-duplex mode. The MAC starts the transmission only when the CRS signal is low.

When this bit is reset, the MAC transmitter ignores the status of the CRS signal.

Bit 10 **DO**: Disable Receive Own

When this bit is set, the MAC disables the reception of packets when the ETH_TX_EN is asserted in the Half-duplex mode. When this bit is reset, the MAC receives all packets given by the PHY.

This bit is not applicable in the Full-duplex mode. This bit is reserved and read-only (RO) with default value in the Full-duplex-only configurations.

Bit 9 **DCRS**: Disable Carrier Sense During Transmission

When this bit is set, the MAC transmitter ignores the GMII CRS signal during packet transmission in the Half-duplex mode. As a result, no errors are generated because of Loss of Carrier or No Carrier during transmission.

When this bit is reset, the MAC transmitter generates errors because of Carrier Sense. The MAC can even abort the transmission.

Bit 8 **DR**: Disable Retry

When this bit is set, the MAC attempts only one transmission. When a collision occurs on the GMII or MII interface, the MAC ignores the current packet transmission and reports a Packet Abort with excessive collision error in the Tx packet status.

When this bit is reset, the MAC retries based on the settings of the BL field. This bit is applicable only in the Half-duplex mode.

Bit 7 Reserved, must be kept at reset value.

Bits 6:5 **BL[1:0]**: Back-Off Limit

The back-off limit determines the random integer number (r) of slot time delays (4,096 bit times for 1000 Mbps; 512 bit times for 10/100 Mbit/s) for which the MAC waits before rescheduling a transmission attempt during retries after a collision:

00: k= min (n, 10)

01: k = min (n, 8)

10: k = min (n, 4)

11: k = min (n, 1)

where n = retransmission attempt

The random integer r takes the value in the range $0 <= r < 2^k$.

This bit is applicable only in the Half-duplex mode.

Bit 4 **DC**: Deferral check

When this bit is set, the deferral check function is enabled in the MAC. The MAC issues a Packet Abort status, along with the excessive deferral error bit set in the Tx packet status, when the Tx state machine is deferred for more than 24,288 bit times in 10 or 100 Mbit/s mode.

If the MAC is configured for 1000 Mbps operation, the threshold for deferral is 155,680 bits times. Deferral begins when the transmitter is ready to transmit, but it is prevented because of an active carrier sense signal (CRS) on GMII or MII.

The defer time is not cumulative. For example, if the transmitter defers for 10,000 bit times because the CRS signal is active and the CRS signal becomes inactive, the transmitter transmits and collision happens. Because of collision, the transmitter needs to back off and then defer again after back off completion. In such a scenario, the deferral timer is reset to 0, and it is restarted.

When this bit is reset, the deferral check function is disabled and the MAC defers until the CRS signal goes inactive.

This bit is applicable only in the Half-duplex mode.

Bits 3:2 **PRELEN[1:0]**: Preamble length for transmit packets

These bits control the number of preamble bytes that are added to the beginning of every Tx packet. The preamble reduction occurs only when the MAC is operating in the Full-duplex mode.

00: 7 bytes of preamble

01: 5 bytes of preamble

10: 3 bytes of preamble

11: Reserved, must not be used

Bit 1 **TE**: Transmitter enable

When this bit is set, the Tx state machine of the MAC is enabled for transmission on the GMII or MII interface. When this bit is reset, the MAC Tx state machine is disabled after it completes the transmission of the current packet. The Tx state machine does not transmit any more packets.

Bit 0 **RE**: Receiver enable

When this bit is set, the Rx state machine of the MAC is enabled for receiving packets from the GMII or MII interface. When this bit is reset, the MAC Rx state machine is disabled after it completes the reception of the current packet. The Rx state machine does not receive any more packets from the GMII or MII interface.

Table 850 shows how the settings of S2KP and JE bits of the ETH_MACCR register impact the giant packet status.

Table 850. Giant Packet Status based on S2KP and JE Bits(1)

Length/Type Field	Received Packet Length	S2KP	JE	Giant Packet Status
Untagged packet	> 1,518	0	0	1
	> 2,000	1	0	1
	> 9,018	x	1	1
VLAN tagged packet	> 1,522	0	0	1
	> 2,000	1	0	1
	> 9,022	x	1	1

1. For all other combinations, the Giant Packet status is 0.

Table 851 shows how the settings of the CST and ACS bits of the ETH_MACCR register impact whether CRC length is included in the packet length.

Table 851. Packet Length based on the CST and ACS bits

Received Packet Length	CST	ACS	FCS Stripping Done
< 1,536	x	0	No
	x	1	Yes (for Ethernet packets)
≥ 1,536	0	x	No
	1	x	Yes (for Type packets)

Extended operating mode configuration register (ETH_MACECR)

Address offset: 0x0004

Reset value: 0x0000 0000

The MAC Extended Configuration register establishes the operating mode of the MAC.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	APDIM	EIPG[4:0]					EIPGEN	Res.	Res.	Res.	Res.	Res.	USP	SPEN	DCRCC
	rw	rw	rw	rw	rw	rw	rw						rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	GPSL[13:0]													
		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **APDIM**: ARP Packet Drop if IP Address Mismatch

When this bit is set, Packet for which Target Protocol Address does not match IPv4 address is dropped in the MTL layer.
When this bit is reset, when target Protocol Address does not match, the packet is forwarded to MTL maintaining backward compatibility.

Bits 29:25 **EIPG[4:0]**: Extended Inter-Packet Gap

The value in this field is applicable when the EIPGEN bit is set. This field (as Most Significant bits) along with IPG field in *Operating mode configuration register (ETH_MACCR)*, gives the minimum IPG greater than 96 bit times in steps of 8 bit times. For example:

- EIPG = 0 and IPG = 0 give 104 bit times
- EIPG = 0 and IPG = 1 give 112 bit times
- EIPG = 0 and IPG = 2 give 120 bit times
- ..
- EIPG = 7 and IPG = 31 give 2144 bit times

Bit 24 **EIPGEN**: Extended Inter-Packet Gap Enable

When this bit is set, the MAC interprets EIPG field and IPG field in *Operating mode configuration register (ETH_MACCR)* together as minimum IPG greater than 96 bit times in steps of 8 bit times.
When this bit is reset, the MAC ignores EIPG field and interprets IPG field in *Operating mode configuration register (ETH_MACCR)* as minimum IPG less than or equal to 96 bit times in steps of 8 bit times.

Note: The extended Inter-Packet Gap feature must be enabled when operating in Full-duplex mode only. There may be undesirable effects on back-pressure function and frame transmission if it is enabled in Half-duplex mode.

Bits 23:19 Reserved, must be kept at reset value.

Bit 18 **USP**: Unicast Slow Protocol Packet Detect

When this bit is set, the MAC detects the Slow Protocol packets with unicast address of the station specified in the *MAC address 0 high register (ETH_MACA0HR)* and MAC Address 0 low register *MAC address x low register (ETH_MACAxLR)*. The MAC also detects the Slow Protocol packets with the Slow Protocols multicast address (01-80-C2-00-00-02).
When this bit is reset, the MAC detects only Slow Protocol packets with the Slow Protocol multicast address specified in the IEEE 802.3-2008, Section 5.

Bit 17 **SPEN**: Slow Protocol Detection Enable

When this bit is set, MAC processes the Slow Protocol packets (Ether Type 0x8809) and provides the Rx status. The MAC discards the Slow Protocol packets with invalid subtypes.
When this bit is reset, the MAC forwards all error-free Slow Protocol packets to the application. The MAC considers such packets as normal Type packets.

Bit 16 **DCRCC**: Disable CRC Checking for Received Packets

When this bit is set, the MAC receiver does not check the CRC field in the received packets.
When this bit is reset, the MAC receiver always checks the CRC field in the received packets.

Bits 15:14 Reserved, must be kept at reset value.

Bits 13:0 **GPSL[13:0]**: Giant Packet Size Limit

If the received packet size is greater than the value programmed in this field in units of bytes, the MAC declares the received packet as Giant packet. The value programmed in this field must be greater than or equal to 1,518 bytes. Any other programmed value is considered as 1,518 bytes.

For VLAN tagged packets, the MAC adds 4 bytes to the programmed value. For double VLAN tagged packets, the MAC adds 8 bytes to the programmed value. The value in this field is applicable when the GPSLCE bit is set in ETH_MACCR register.

Packet filtering control register (ETH_MACPFR)

Address offset: 0x0008

Reset value: 0x0000 0000

The MAC Packet Filter register contains the filter controls for receiving packets. Some of the controls from this register go to the address check block of the MAC which performs the first level of address filtering. The second level of filtering is performed on the incoming packet based on other controls such as Pass Bad Packets and Pass Control Packets.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DNTU	IPFE	Res.	Res.	Res.	VTFE
rw										rw	rw				rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	HPF	SAF	SAIF	PCF[1:0]		DBF	PM	DAIF	HMC	HUC	PR
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **RA**: Receive All

When this bit is set, the MAC receiver module passes all received packets to the application, irrespective of whether they pass the address filter or not. The result of the SA or DA filtering is updated (pass or fail) in the corresponding bit in the Rx Status Word.

When this bit is reset, the receiver module passes only those packets to the application that pass the SA or DA address filter.

Bits 30:22 Reserved, must be kept at reset value.

Bit 21 **DNTU**: Drop Non-TCP/UDP over IP Packets

When this bit is set, the MAC drops the non-TCP or UDP over IP packets. The MAC forward only those packets that are processed by the Layer 4 filter. When this bit is reset, the MAC forwards all non-TCP or UDP over IP packets.

Bit 20 **IPFE**: Layer 3 and Layer 4 Filter Enable

When this bit is set, the MAC drops packets that do not match the enabled Layer 3 and Layer 4 filters. If Layer 3 or Layer 4 filters are not enabled for matching, this bit does not have any effect.

When this bit is reset, the MAC forwards all packets irrespective of the match status of the Layer 3 and Layer 4 fields.

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **VTFE**: VLAN Tag Filter Enable

When this bit is set, the MAC drops the VLAN tagged packets that do not match the VLAN Tag. When this bit is reset, the MAC forwards all packets irrespective of the match status of the VLAN Tag.

Bits 15:11 Reserved, must be kept at reset value.

Bit 10 **HPF**: Hash or Perfect Filter

When this bit is set, the address filter passes a packet if it matches either the perfect filtering or Hash filtering as set by the HMC or HUC bit.

When this bit is reset and the HUC or HMC bit is set, the packet is passed only if it matches the Hash filter.

Bit 9 **SAF**: Source Address Filter Enable

When this bit is set, the MAC compares the SA field of the received packets with the values programmed in the enabled SA registers. If the comparison fails, the MAC drops the packet.

When this bit is reset, the MAC forwards the received packet to the application with updated SAF bit of the Rx Status depending on the SA address comparison.

Note: According to the IEEE specification, Bit 47 of the SA is reserved. However, the MAC compares all 48 bits. The software driver should take this into consideration while programming the MAC address registers for SA.

Bit 8 **SAIF**: SA Inverse Filtering

When this bit is set, the Address Check block operates in the inverse filtering mode for SA address comparison. If the SA of a packet matches the values programmed in the SA registers, it is marked as failing the SA Address filter.

When this bit is reset, if the SA of a packet does not match the values programmed in the SA registers, it is marked as failing the SA Address filter.

Bits 7:6 **PCF[1:0]**: Pass Control Packets

These bits control the forwarding of all control packets (including unicast and multicast Pause packets).

00: The MAC filters all control packets from reaching the application.

01: The MAC forwards all control packets except Pause packets to the application even if they fail the Address filter.

10: The MAC forwards all control packets to the application even if they fail the Address filter.

11: The MAC forwards the control packets that pass the Address filter.

Bit 5 **DBF**: Disable Broadcast Packets

When this bit is set, the AFM module blocks all incoming broadcast packets. In addition, it overrides all other filter settings.

When this bit is reset, the AFM module passes all received broadcast packets.

Bit 4 **PM**: Pass All Multicast

When this bit is set, it indicates that all received packets with a multicast destination address (first bit in the destination address field is '1') are passed. When this bit is reset, filtering of multicast packet depends on HMC bit.

Bit 3 **DAIF**: DA Inverse Filtering

When this bit is set, the Address Check block operates in inverse filtering mode for the DA address comparison for both unicast and multicast packets. When this bit is reset, normal filtering of packets is performed.

Bit 2 **HMC**: Hash Multicast

When this bit is set, the MAC performs the destination address filtering of received multicast packets according to the Hash table.

When this bit is reset, the MAC performs the perfect destination address filtering for multicast packets, that is, it compares the DA field with the values programmed in DA registers.

Bit 1 **HUC**: Hash Unicast

When this bit is set, the MAC performs the destination address filtering of unicast packets according to the Hash table.

When this bit is reset, the MAC performs a perfect destination address filtering for unicast packets, that is, it compares the DA field with the values programmed in DA registers.

Bit 0 **PR**: Promiscuous Mode

When this bit is set, the Address Filtering module passes all incoming packets irrespective of the destination or source address. The SA or DA Filter Fails status bits of the Rx Status Word are always cleared when PR is set.

Watchdog timeout register (ETH_MACWTR)

Address offset: 0x000C

Reset value: 0x0000 0000

The Watchdog timeout register controls the watchdog timeout for received packets.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	PWE	Res.	Res.	Res.	Res.	WTO[3:0]			
							rw					rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **PWE**: Programmable Watchdog Enable

When this bit is set and the WD bit of the *Operating mode configuration register (ETH_MACCR)* register is reset, the WTO field is used as watchdog timeout for a received packet. When this bit is cleared, the watchdog timeout for a received packet is controlled by setting of WD and JE bits in *Operating mode configuration register (ETH_MACCR)* register.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:0 **WTO[3:0]**: Watchdog timeout

When the PWE bit is set and the WD bit of the *Operating mode configuration register (ETH_MACCR)* register is reset, this field is used as watchdog timeout for a received packet. If the length of a received packet exceeds the value of this field, such packet is terminated and declared as an error packet.

Encoding is as follows:

0x0: 2 Kbytes

0x1: 3 Kbytes

0x2: 4 Kbytes

0x3: 5 Kbytes

..

0xC: 14 Kbytes

0xD: 15 Kbytes

0xE: 16383 Bytes

0xF: Reserved, must not be used

Note: When the PWE bit is set, the value in this field should be more than 1,522 (0x05F2). Otherwise, the IEEE 802.3-specified valid tagged packets are declared as error packets and then dropped.

Hash Table 0 register (ETH_MACHT0R)

Address offset: 0x0010

Reset value: 0x0000 0000

The Hash Table Register 0 contains the first lower 32 bits of the Hash table (64 bits).

The Hash table is used for group address filtering.

For Hash filtering, the content of the destination address in the incoming packet is passed through the CRC logic and the upper six bits of the CRC register are used to index the content of the Hash table. The most significant bits determines the register to be used (Hash Table Register 0 or 1) and the least significant five bits determine the bit within the register. For example, a hash value of 0b100000 selects Bit 0 of the Hash Table Register 1.

The Hash value of the destination address is calculated in the following way:

1. Calculate the 32-bit CRC for the DA (See IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32).
2. Perform bitwise reversal for the value obtained in Step 1.
3. Take the upper 7 or 8 bits from the value obtained in Step 2.

If the corresponding bit value of the register is 1, the packet is accepted. Otherwise, it is rejected. If the PM bit is set in ETH_MACPFR, all multicast packets are accepted regardless of the multicast Hash values.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HT31T0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HT31T0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **HT31T0[31:0]**: MAC Hash Table First 32 Bits

This field contains the first 32 Bits [31:0] of the Hash table.

Hash Table 1 register (ETH_MACHT1R)

Address offset: 0x0014

Reset value: 0x0000 0000

The Hash Table 1 register contains the upper 32 bits of the Hash table (64 bits).

The Hash table is used for group address filtering.

For Hash filtering, the content of the destination address in the incoming packet is passed through the CRC logic and the upper six bits of the CRC register are used to index the content of the Hash table. The most significant bits determines the register to be used (Hash Table Register 0 or 1) and the least significant five bits determine the bit within the register. For example, a hash value of 6'b100000 selects Bit 0 of the Hash Table Register 1.

The Hash value of the destination address is calculated in the following way:

1. Calculate the 32-bit CRC for the DA (See IEEE 802.3, Section 3.2.8 for the steps to calculate CRC32).
2. Perform bitwise reversal for the value obtained in Step 1.
3. Take the upper 7 or 8 bits from the value obtained in Step 2.

If the corresponding bit value of the register is 1, the packet is accepted. Otherwise, it is rejected. If the PM bit is set in ETH_MACPFR, all multicast packets are accepted regardless of the multicast Hash values.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HT63T32[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HT63T32[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **HT63T32[31:0]**: MAC Hash Table Second 32 Bits

This field contains the second 32 Bits [63:32] of the Hash table.

VLAN tag Control register (ETH_MACVTCR)

Address offset: 0x0050

Reset value: 0x0000 0000

This register is the redefined format of the MAC VLAN Tag Register. It is used for indirect addressing. It contains the address offset, command type and Busy Bit for CSR access of the Per VLAN Tag registers.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EIVLRXS	Res.	EIVLS[1:0]		ERIVLT	EDVLP	VTHM	EVLRXS	Res.	EVLS[1:0]		DOVLTC	ERSVLM	ESVL	VTIM	ETV
rw		rw	rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFS[1:0]		CT	OB
												rw	rw	rw	rw

Bit 31 **EIVLRXS**: Enable Inner VLAN Tag in Rx Status

When this bit is set, the MAC provides the inner VLAN Tag in the Rx status. When this bit is reset, the MAC does not provide the inner VLAN Tag in Rx status.

Bit 30 Reserved, must be kept at reset value.

Bits 29:28 **EIVLS[1:0]**: Enable Inner VLAN Tag Stripping on Receive

This field indicates the stripping operation on inner VLAN Tag in received packet:
00: Do not strip
01: Strip if VLAN filter passes
10: Strip if VLAN filter fails
11: Always strip

Bit 27 **ERIVLT**: Enable Inner VLAN Tag

When this bit and the EDVLP field are set, the MAC receiver enables operation on the inner VLAN Tag (if present). When this bit is reset, the MAC receiver enables operation on the outer VLAN Tag (if present). The ERSVLM bit determines which VLAN type is enabled for filtering or matching.The ERSVLM bit and DOVLTC bit determines which VLAN type is enabled for filtering.

Bit 26 **EDVLP**: Enable Double VLAN Processing

When this bit is set, the MAC enables processing of up to two VLAN Tags on Tx and Rx (if present). When this bit is reset, the MAC enables processing of up to one VLAN Tag on Tx and Rx (if present).

Bit 25 **VTHM**: VLAN Tag Hash Table Match Enable

When this bit is set, the most significant four bits of CRC of VLAN Tag are used to index the content of the ETH_MACVLANHTR register. A value of 1 in the VLAN Hash Table register, corresponding to the index, indicates that the packet matched the VLAN Hash table.
When the ETV bit is set, the CRC of the 12-bit VLAN Identifier (VID) is used for comparison.
When the ETV bit is reset, the CRC of the 16-bit VLAN tag is used for comparison.
When this bit is reset, the VLAN Hash Match operation is not performed.

Bit 24 **EVLRXS**: Enable VLAN Tag in Rx status

When this bit is set, MAC provides the outer VLAN Tag in the Rx status. When this bit is reset, the MAC does not provide the outer VLAN Tag in Rx status.

Bit 23 Reserved, must be kept at reset value.

Bits 22:21 **EVLS[1:0]**: Enable VLAN Tag Stripping on Receive
This field indicates the stripping operation on the outer VLAN Tag in received packet:
00: Do not strip
01: Strip if VLAN filter passes
10: Strip if VLAN filter fails
11: Always strip

Bit 20 **DOVLTC**: Disable VLAN Type Check
When this bit is set, the MAC does not check whether the VLAN Tag specified by the ERIVLT bit is of type S-VLAN or C-VLAN.
When this bit is reset, the MAC filters or matches the VLAN Tag specified by the ERIVLT bit only when VLAN Tag type is similar to the one specified by the ERSVLM bit.

Bit 19 **ERSVLM**: Enable Receive S-VLAN Match
When this bit is set, the MAC receiver enables filtering or matching for S-VLAN (Type = 0x88A8) packets. When this bit is reset, the MAC receiver enables filtering or matching for C-VLAN (Type = 0x8100) packets.
The ERIVLT bit determines the VLAN tag position considered for filtering or matching.

Bit 18 **ESVL**: Enable S-VLAN
When this bit is set, the MAC transmitter and receiver consider the S-VLAN packets (Type = 0x88A8) as valid VLAN tagged packets.

Bit 17 **VTIM**: VLAN Tag Inverse Match Enable
When this bit is set, this bit enables the VLAN Tag inverse matching. The packets without matching VLAN Tag are marked as matched. When reset, this bit enables the VLAN Tag perfect matching. The packets with matched VLAN Tag are marked as matched.

Bit 16 **ETV**: Enable 12-Bit VLAN Tag Comparison
When this bit is set, a 12-bit VLAN identifier is used for comparing and filtering instead of the complete 16-bit VLAN tag. Bits[11:0] of VLAN tag are compared with the corresponding field in the received VLAN-tagged packet. Similarly, when enabled, only 12 bits of the VLAN tag in the received packet are used for Hash-based VLAN filtering.
When this bit is reset, all 16 bits of the 15th and 16th bytes of the received VLAN packet are used for comparison and VLAN Hash filtering.

Bits 15:4 Reserved, must be kept at reset value.

Bits 3:2 **OFS[1:0]**: Offset
This field holds the address offset of the MAC VLAN Tag Filter Register which the application is trying to access.
00:ETH_MACVTDR holds MAC VLAN Tag Filter0 content
01:ETH_MACVTDR holds MAC VLAN Tag Filter1 content
10:ETH_MACVTDR holds MAC VLAN Tag Filter2 content
11:ETH_MACVTDR holds MAC VLAN Tag Filter3 content

Bit 1 **CT**: Command Type
This bit indicates if the current register access is a read or a write.
When set, it indicate a read operation. When reset, it indicates a write operation.

Bit 0 **OB**: Operation Busy

This bit is set along with a read or write command for initiating the indirect access to per VLAN Tag Filter register. This bit is reset when the read or write command to per VLAN Tag Filter indirect access register is complete. The next indirect register access can be initiated only after this bit is reset.
During a write operation, the bit is reset only after the data has been written into the Per VLAN Tag register.
During a read operation, the data should be read from the MAC_VLAN_Tag_Data register only after this bit is reset.

VLAN tag data register (ETH_MACVTDR)

Address offset: 0x0054

Reset value: 0x0000 0000

This register contains VLAN Tag filter 0..3 control information, as per OFS[1:0] bitfield setting in ETH_MACVTCR register. This register holds the read/write data for Indirect Access of the Per VLAN Tag registers. During the read access, this field contains valid read data only after the OB bit is reset.
During the write access, this field should be valid prior to setting the OB bit in the MAC_VLAN_Tag_Ctrl Register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	DMACHN	DMACHEN	Res.	Res.	Res.	ERIVLT	ERSVLM	DOVLTC	ETV	VEN
						rw	rw				rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VID[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **DMACHN**: DMA Channel Number

The DMA Channel number to which the VLAN Tagged Frame is to be routed if it passes this VLAN Tag Filter is programmed in this field.
If the Routing based on VLAN Tag Filter is not necessary, this field need not be programmed.

Bit 24 **DMACHEN**: DMA Channel Number Enable

This bit is the Enable for the DMA Channel Number value programmed in the DMACH field. When this bit is reset, the Routing does not occur based on VLAN Filter result. The frame is routed based on DA-based DMA Channel Routing.

Bits 23:21 Reserved, must be kept at reset value.

Bit 20 **ERIVLT**: Enable Inner VLAN Tag Comparison

This bit is valid only when Double VLAN Tag enable of the Filter is set.
When this bit and the EDVLP field are set, the MAC receiver enables operation on the inner VLAN Tag (if present).
When this bit is reset, the MAC receiver enables operation on the outer VLAN Tag (if present).

Bit 19 **ERSVLM**: Enable S-VLAN Match for received Frames

This bit is valid only when VLAN Tag Enable of the Filter is set.
When this bit is set, the MAC receiver enables filtering or matching for S-VLAN (Type = 0x88A8) packets.
When this bit is reset, the MAC receiver enables filtering or matching for C-VLAN (Type =0x8100) packets.

Bit 18 **DOVLTC**: Disable VLAN Type Comparison

This bit is valid only when VLAN Tag Enable of the Filter is set.
When this bit is set, the MAC does not check whether the VLAN Tag specified by the Enable Inner VLAN Tag Comparison bit is of type S-VLAN or C-VLAN.
When this bit is reset, the MAC filters or matches the VLAN Tag specified by the Enable Inner VLAN Tag Comparison bit only when the VLAN Tag type is similar to the one specified by the Enable S-VLAN Match for received frames bit.

Bit 17 **ETV**: 12-bit or 16-bit VLAN comparison

This bit is valid only when VEN of the Filter is set.
When this bit is set, a 12-bit VLAN identifier is used for comparing and filtering instead of the complete 16-bit VLAN tag. Bits[11:0] of VLAN tag are compared to the corresponding field in the received VLAN-tagged packet.
0: 16-bit VLAN
1: 12-bit VLAN

Bit 16 **VEN**: VLAN Tag Enable

This bit is used to enable or disable the VLAN Tag.
When this bit is set, the MAC compares the VLAN Tag of received packet with the VLAN Tag ID.
When this bit is reset, no comparison is performed irrespective of the programming of the other fields.

Bits 15:0 **VID[15:0]**: VLAN Tag ID

This field holds the VLAN Tag value which is used by the MAC for perfect comparison. It is valid when VLAN Tag Enable is set.

VLAN Hash table register (ETH_MACVHTR)

Address offset: 0x0058

Reset value: 0x0000 0000

When the VTHM bit of *VLAN tag Control register (ETH_MACVTCR)* register is set, the 16-bit VLAN Hash Table register is used for group address filtering based on the VLAN tag. For Hash filtering, the content of the 16-bit VLAN tag or 12-bit VLAN ID (based on the ETV bit of *VLAN tag Control register (ETH_MACVTCR)* register) in the incoming packet is passed through the CRC logic. The upper four bits of the calculated CRC are used to index the contents of the VLAN Hash table. For example, a Hash value of 1000 selects Bit 8 of the VLAN Hash table.

The Hash value of the destination address is calculated in the following way:

1. Calculate the 32-bit CRC for the VLAN tag or ID (For steps to calculate CRC32, see Section 3.2.8 of IEEE 802.3).
2. Perform bitwise reversal for the value obtained in step 1.
3. Take the upper four bits from the value obtained in step 2.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VLHT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **VLHT[15:0]**: VLAN Hash Table
This field contains the 16-bit VLAN Hash Table.

VLAN inclusion register (ETH_MACVIR)

Address offset: 0x0060

Reset value: 0x0000 0000

The VLAN Tag Inclusion or Replacement register contains the VLAN tag for insertion or replacement in the transmit packets. It also contains the VLAN tag insertion controls.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BUSY	RDWR	Res.	Res.	Res.	Res.	Res.	ADDR	Res.	Res.	CBTI	VLTI	CSVL	VLP	VLC[1:0]	
r	rw						rw			rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VLT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **BUSY**: Busy

This bit indicates the status of the read/write operation of indirect access to the queue/channel specific VLAN inclusion register.

For write operation write to a register is complete when this bit is reset. For read operation the read data is valid when the bit is reset.

No further writes are allowed to this register when this bit is set. This does not have any effect when CBTI is reset

Bit 30 **RDWR**: Read write control

This bit controls the read or write operation for indirectly accessing the queue/channel specific VLAN Inclusion register.

When set indicates write operation and when reset indicates read operation

This has no effect when CBTI is reset

Bits 29:25 Reserved, must be kept at reset value.

Bit 24 **ADDR**: Address

This bit selects one of the queue/channel specific VLAN Inclusion register for read/write access

This does not have any effect when CBTI is reset

0: VLAN tag for insertion in the transmit packets from Tx Queue 0

1: VLAN tag for insertion in the transmit packets from Tx Queue 1

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **CBTI**: Channel based tag insertion

When this bit is set, outer VLAN tag is inserted for every packets transmitted by the MAC. The tag value is taken from the queue/channel specific VLAN tag register. This overrides VLTI, VLP, VLC and VLT fields of this register

When this bit is set, a write operation to byte 3 of this register initiates the read/write access to the indirect register.

When reset outer VLAN operation is based on the setting of VLTI, VLP, VLC and VLT fields of this register

Bit 20 **VLTI**: VLAN tag input

When this bit is set, it indicates that the VLAN tag to be inserted or replaced in Tx packet should be taken from the Tx descriptor.

Bit 19 **CSVL**: C-VLAN or S-VLAN

When this bit is set, S-VLAN type (0x88A8) is inserted or replaced in the 13th and 14th bytes of transmitted packets. When this bit is reset, C-VLAN type (0x8100) is inserted or replaced in the 13th and 14th bytes of transmitted packets.

0: C-LAN

1: S-LAN

Bit 18 **VLP**: VLAN priority control

When this bit is set, the control bits[17:16] are used for VLAN deletion, insertion, or replacement. When this bit is reset, bits[17:16] are ignored.

Bits 17:16 **VLC[1:0]**: VLAN tag control in transmit packets

00: No VLAN tag deletion, insertion, or replacement

01: VLAN tag deletion. The MAC removes the VLAN type (bytes 13 and 14) and VLAN tag (bytes 15 and 16) of all transmitted packets with VLAN tags.

10: VLAN tag insertion. The MAC inserts VLT in bytes 15 and 16 of the packet after inserting the Type value (0x8100 or 0x88a8) in bytes 13 and 14. This operation is performed on all transmitted packets, irrespective of whether they already have a VLAN tag.

11: VLAN tag replacement. The MAC replaces VLT in bytes 15 and 16 of all VLAN-type transmitted packets (Bytes 13 and 14 are 0x8100 or 0x88a8).

Note: Changes to this field take effect only on the start of a packet. If you write this register field when a packet is being transmitted, only the subsequent packet can use the updated value, that is, the current packet does not use the updated value.

Bits 15:0 **VLT[15:0]**: VLAN tag for transmit packets

This field contains the value of the VLAN tag to be inserted or replaced. The value must only be changed when the transmit lines are inactive or during the initialization phase.

The following list describes the bits of this field:

Bits[15:13]: User Priority

Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI)

Bits[11:0]: VLAN Identifier (VID) field of VLAN tag

VLAN inclusion register [alternate] (ETH_MACVIR)

Address offset: 0x0060

Reset value: 0x0000 0000

The Tx Queue x VLAN Tag Inclusion register contains the VLAN tag for insertion in the transmit packets from Tx Queue x. It also contains the VLAN tag insertion controls. This is an indirect register accessible when the CBTI bit is set in *VLAN inclusion register (ETH_MACVIR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSVL	Res.	Res.	Res.
												rw			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VLT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **CSVL**: C-VLAN or S-VLAN

When this bit is set, S-VLAN type (0x88A8) is inserted or replaced in the 13th and 14th bytes of transmitted packets. When this bit is reset, C-VLAN type (0x8100) is inserted or replaced in the 13th and 14th bytes of transmitted packets.

0: C-LAN

1: S-LAN

Bits 18:16 Reserved, must be kept at reset value.

Bits 15:0 **VLT[15:0]**: VLAN tag for transmit packets

This field contains the value of the VLAN tag to be inserted or replaced. The value must only be changed when the transmit lines are inactive or during the initialization phase.

The following list describes the bits of this field:

Bits[15:13]: User Priority

Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI)

Bits[11:0]: VLAN Identifier (VID) field of VLAN tag

Inner VLAN inclusion register (ETH_MACIVIR)

Address offset: 0x0064

Reset value: 0x0000 0000

The Inner VLAN Tag Inclusion or Replacement register contains the inner VLAN tag to be inserted or replaced in the transmit packet. It also contains the inner VLAN tag insertion controls.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VLTI	CSVL	VLP	VLC[1:0]	
											rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VLT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **VLTI**: VLAN tag input

When this bit is set, it indicates that the VLAN tag to be inserted or replaced in Tx packet should be taken from the Tx descriptor

Bit 19 **CSVL**: C-VLAN or S-VLAN

When this bit is set, S-VLAN type (0x88A8) is inserted or replaced in the 13th and 14th bytes of transmitted packets. When this bit is reset, C-VLAN type (0x8100) is inserted or replaced in the 13th and 14th bytes of transmitted packets.

0: C-LAN

1: S-LAN

Bit 18 **VLP**: VLAN priority control

When this bit is set, the VLC field is used for VLAN deletion, insertion, or replacement. When this bit is reset, the VLC field is ignored.

Bits 17:16 **VLC[1:0]**: VLAN tag control in transmit packets

00: No VLAN tag deletion, insertion, or replacement

01: VLAN tag deletion

The MAC removes the VLAN type (bytes 17 and 18) and VLAN tag (bytes 19 and 20) of all transmitted packets with VLAN tags.

10: VLAN tag insertion

The MAC inserts VLT in bytes 19 and 20 of the packet after inserting the Type value (0x8100 or 0x88a8) in bytes 17 and 18. This operation is performed on all transmitted packets, irrespective of whether they already have a VLAN tag.

11: VLAN tag replacement

The MAC replaces VLT in bytes 19 and 20 of all VLAN-type transmitted packets (Bytes 17 and 18 are 0x8100 or 0x88a8).

Note: Changes to this field take effect only on the start of a packet. If you write to this register field when a packet is being transmitted, only the subsequent packet can use the updated value, that is, the current packet does not use the updated value.

Bits 15:0 **VLT[15:0]**: VLAN tag for transmit packets

This field contains the value of the VLAN tag to be inserted or replaced. The value must only be changed when the transmit lines are inactive or during the initialization phase.
The following list describes the bits of this field:
Bits[15:13]: User Priority
Bit 12: Canonical Format Indicator (CFI) or Drop Eligible Indicator (DEI)
Bits[11:0]: VLAN Identifier (VID) field of VLAN tag

Tx Queue 0 flow control register (ETH_MACQ0TXFCR)

Address offset: 0x0070

Reset value: 0x0000 0000

The flow control register controls the generation and reception of the control (Pause Command) packets by the flow control module of the MAC. A Write to a register with the Busy bit set to 1 triggers the Flow Control block to generate a Pause packet. The fields of the control packet are selected as specified in the 802.3x specification, and the Pause Time value from this register is used in the Pause Time field of the control packet. The Busy bit remains set until the control packet is transferred onto the cable. The application must make sure that the Busy bit is cleared before writing to the register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DZPQ	PLT[2:0]			Res.	Res.	TFE	FCB_BPA
								rw	rw	rw	rw			rw	rw

Bits 31:16 **PT[15:0]**: Pause Time

This field holds the value to be used in the Pause Time field in the Tx control packet. I

Bits 15:8 Reserved, must be kept at reset value.

Bit 7 **DZPQ**: Disable Zero-Quanta Pause

When this bit is set, it disables the automatic generation of the zero-quanta Pause packets.
When this bit is reset, normal operation with automatic zero-quanta Pause packet generation is enabled.

Bits 6:4 **PLT[2:0]**: Pause Low Threshold

This field configures the threshold of the Pause timer at which the input flow is checked for automatic retransmission of the Pause packet.

The threshold values should be always less than the Pause Time configured in Bits[31:16]. For example, if PT = 100H (256 slot times), and PLT = 001, a second Pause packet is automatically transmitted at 228 (256-28) slot times after the first Pause packet is transmitted.

The following list provides the threshold values for different values:

000: Pause Time minus 4 Slot Times (PT -4 slot times)

001: Pause Time minus 28 Slot Times (PT -28 slot times)

010: Pause Time minus 36 Slot Times (PT -36 slot times)

011: Pause Time minus 144 Slot Times (PT -144 slot times)

100: Pause Time minus 256 Slot Times (PT -256 slot times)

101: Pause Time minus 512 Slot Times (PT -512 slot times)

110 to 111: Reserved, must not be used

The slot time is defined as the time taken to transmit 512 bits (64 bytes) on the GMII or MII interface.

This (approximate) computation is based on the packet size (64, 1518, 2000, 9018, 16384, or 32768) + 2 Pause Packet Size + IPG in Slot Times.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **TFE**: Transmit Flow Control Enable

Full-duplex mode: when this bit is set, the MAC enables the flow control operation to Tx Pause packets. When this bit is reset, the flow control operation in the MAC is disabled, and the MAC does not transmit any Pause packets.

Half-duplex mode: when this bit is set, the MAC enables the backpressure operation. When this bit is reset, the backpressure feature is disabled.

Bit 0 **FCB_BPA**: Flow Control Busy or Backpressure Activate

This bit initiates a Pause packet in the Full-duplex mode and activates the backpressure function in the Half-duplex mode if the TFE bit is set.

Full-Duplex mode: this bit should be read as 0 before writing to this register. To initiate a Pause packet, the application must set this bit to 1. During Control packet transfer, this bit continues to be set to indicate that a packet transmission is in progress. When Pause packet transmission is complete, the MAC resets this bit to 0. You should not write to this register until this bit is cleared.

Half-duplex mode: When this bit is set (and TFE bit is set) in the Half-duplex mode, the MAC asserts the backpressure. During backpressure, when the MAC receives a new packet, the transmitter starts sending a JAM pattern resulting in a collision. When the MAC is configured for the Full-duplex mode, the BPA is automatically disabled.

Rx flow control register (ETH_MACRXFCR)

Address offset: 0x0090

Reset value: 0x0000 0000

The Receive Flow Control register controls the pausing of MAC transmit based on the received Pause packet.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UP	RFE
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **UP**: Unicast Pause Packet Detect

A pause packet is processed when it has the unique multicast address specified in the IEEE 802.3. When this bit is set, the MAC can also detect Pause packets with unicast address of the station. This unicast address should be as specified in *MAC address 0 high register (ETH_MACA0HR)* and MAC Address 0 low register *MAC address x low register (ETH_MACAxLR)*.

When this bit is reset, the MAC only detects Pause packets with unique multicast address.

Note: The MAC does not process a Pause packet if the multicast address is different from the unique multicast address. This is also applicable to the received PFC packet when the Priority Flow Control (PFC) is enabled. The unique multicast address (0x01 80 C2 00 00 01) is as specified in IEEE 802.1 Qbb-2011.

Bit 0 **RFE**: Receive Flow Control Enable

When this bit is set and the MAC is operating in Full-duplex mode, the MAC decodes the received Pause packet and disables its transmitter for a specified (Pause) time. When this bit is reset or the MAC is operating in Half-duplex mode, the decode function of the Pause packet is disabled.

When PFC is enabled, flow control is enabled for PFC packets. The MAC decodes the received PFC packet and disables the transmit queue, with matching priorities, for a duration of received Pause time.

Rx Queue control register (ETH_MACRXQCR)

Address offset: 0x0094

Reset value: 0x0000 0000

The Receive Queue Control register controls the routing of unicast and multicast packets that fail the Destination or Source address filter to the Rx queues

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VFFQ	VFFQE
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	MFFQ	MFFQE	Res.	Res.	Res.	Res.	Res.	Res.	UFFQ	UFFQE
						rw	rw							rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **VFFQ**: VLAN Tag Filter Fail Packets Queue

This field holds the Rx queue number to which the tagged packets failing the Destination or Source Address filter (and UFFQE/MFFQE not enabled) or failing the VLAN tag filter must be routed to. This field is valid only when the VFFQE bit is set.

0: Queue 0

1: Queue 1

Bit 16 **VFFQE**: VLAN Tag Filter Fail Packets Queuing Enable

When this bit is set, the tagged packets which fail the Destination or Source address filter or fail the VLAN tag filter are routed to the Rx Queue Number programmed in the VFFQ. When this bit is reset, the tagged packets which fail the Destination or Source Address filter or fail the VLAN tag filter are routed based on other routing options. This bit is valid only when the RA bit of the *Packet filtering control register (ETH_MACPFR)* is set.

0: Disabled

1: Enabled

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **MFFQ**: Multicast Address Filter Fail Packets Queue.

This field holds the Rx queue number to which the Multicast packets failing the Destination or Source Address filter are routed to. This field is valid only when the MFFQE bit is set.

0: Queue 0

1: Queue 1

Bit 8 **MFFQE**: Multicast Address Filter Fail Packets Queuing Enable.

When this bit is set, the Multicast packets which fail the Destination or Source address filter is routed to the Rx Queue Number programmed in the MFFQ.

When this bit is reset, the Multicast packets which fail the Destination or Source address filter is routed based on other routing options.

This bit is valid only when the RA bit of the *Packet filtering control register (ETH_MACPFR)* is set.

Bits 7:2 Reserved, must be kept at reset value.

Bit 1 **UFFQ**: Unicast Address Filter Fail Packets Queue.

This field holds the Rx queue number to which the Unicast packets failing the Destination or Source Address filter are routed to. This field is valid only when the UFFQE bit is set.

0: Queue 0

1: Queue 1

Bit 0 **UFFQE**: Unicast Address Filter Fail Packets Queuing Enable.

When this bit is set, the Unicast packets which fail the Destination or Source address filter is routed to the Rx Queue Number programmed in the UFFQ.

When this bit is reset, the Unicast packets which fail the Destination or Source Address filter is routed based on other routing options.

This bit is valid only when the RA bit of the *Packet filtering control register (ETH_MACPFR)* is set.

Rx queue control 0 register (ETH_MACRXQC0R)

Address offset: 0x00A0

Reset value: 0x0000 0000

The Receive Queue Control 0 register controls the queue management in the MAC receiver.

Note: In multiple Rx queues configuration, all the queues are disabled by default. Enable the Rx queue by programming the corresponding field in this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXQ1EN[1:0]		RXQ0EN[1:0]	
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:2 **RXQ1EN[1:0]**: Receive Queue 1 Enable
This field indicates whether Rx queue 1 is enabled for AV or Generic traffic.
00: Not enabled
01: Queue 1 enabled for AV
10: Queue 1 enabled for Generic traffic
11: Reserved, must not be used

Bits 1:0 **RXQ0EN[1:0]**: Receive Queue 0 Enable
This field indicates whether Rx queue 0 is enabled for AV or Generic traffic.
00: Not enabled
01: Queue 0 enabled for AV
10: Queue 0 enabled for Generic traffic
11: Reserved, must not be used

Rx queue control 1 register (ETH_MACRXQC1R)

Address offset: 0x00A4

Reset value: 0x0000 0000

The Receive Queue Control 1 register controls the routing of multicast, broadcast, AV and untagged packets to the Rx queues.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	TBRQE	OMCBCQ	Res.	FPRQ2	FPRQ1	FPRQ0	TPQC[1:0]		TACPQE	MCBCQEN	Res.	MCBCQ[2:0]		
		rw	rw		rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	UPQ[2:0]			Res.	Res.	Res.	Res.	Res.	PTPQ[2:0]			Res.	AVCPQ2	AVCPQ1	AVCPQ0
	rw	rw	rw						rw	rw	rw		rw	rw	rw

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **TBRQE**: Type Field Based Rx Queuing Enable

When this bit is set, it enables Type field based Rx queuing where the Type field of received packet is compared with the programmed TYP field in ETH_MAC_TMRQR register and if a match occurs the packet is routed to the corresponding TMRQ field.

Bit 28 **OMCBCQ**: Overriding MC-BC queue priority select

1: Priority of MCBCQ is reduced and the received packet is first routed to PTPQ, AVCPQ, depending on packet type.
0: Received Multicast/Broadcast packet is routed to MCBCQ.

Bit 27 Reserved, must be kept at reset value.

Bits 26:24 **FPRQ[2:0]**: Frame Preemption Residue Queue

This field holds the Rx queue number to which the residual preemption frames must be forwarded. Preemption frames that are tagged and pass the SA/DA/VLAN filtering are routed based on PSRQ and all other frames are treated as residual frames and is routed to the queue number mentioned in this field. Queue 0 is used as a default queue for express frames, so this field cannot be programmed to a value 0.
000: Reserved, must not be used (Rx Queue 0)
001: Rx queue 1
Others: Reserved, must not be used

Bits 23:22 **TPQC[1:0]**: Tagged PTP over Ethernet Packets Queuing Control

This field controls the routing of the VLAN tagged PTPoE packets.
00: VLAN tagged PTPoE packets are routed as generic VLAN tagged packet (based on PSRQ for only non-AV enabled Rx queues).
01: VLAN tagged PTPoE packets are routed to Rx Queue specified by PTPQ field (this Rx Queue can be enabled for AV or non-AV traffic)
10: VLAN tagged PTPoE packets are routed to only AV enabled Rx queues based on PSRQ.
11: Reserved

Bit 21 **TACPQE**: Tagged AV Control Packets Queuing Enable
When set, the MAC routes the received Tagged AV control packets to the Rx queue specified by AVCPQ field.
When reset, the MAC routes the received Tagged AV control packets based on the tag priority matching the PSRQ fields in ETH_MACRXQC2R register.
This field is reserved if Rx queues are not selected for AV feature.

Bit 20 **MCBCQEN**: Multicast and Broadcast Queue Enable
This bit specifies that Multicast or Broadcast packets routing to the Rx queue is enabled and the Multicast or Broadcast packets must be routed to Rx queue specified in MCBCQ field.

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **MCBCQ[2:0]**: Multicast and Broadcast Queue
This field specifies the Rx queue onto which Multicast or Broadcast packets are routed. Any Rx queue enabled for Generic/AV traffic can be used to route the Multicast or Broadcast packets.
000: Rx queue 0
001: Rx queue 1
Others: Reserved, must not be used

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **UPQ[2:0]**: Untagged Packet Queue
This field indicates the Rx queue to which Untagged Packets are to be routed. Any Rx queue enabled for Generic/AV traffic can be used to route the Untagged Packets.
000: Rx queue 0
001: Rx queue 1
Others: Reserved, must not be used

Bits 11:7 Reserved, must be kept at reset value.

Bits 6:4 **PTPQ[2:0]**: PTP Packets Queue
This field specifies the Rx queue on which the PTP packets sent over the Ethernet payload (not over IPv4 or IPv6) are routed.
000: Rx queue 0
001: Rx queue 1
Others: Reserved, must not be used
When the AV8021ASMEN bit of *Timestamp control register (ETH_MACTSCR)* is set, only untagged PTP over Ethernet packets are routed on an Rx queue. If the bit is not set, then based on programming of TPQC field, both tagged and untagged PTPoE packets can be routed to this Rx Queue.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **AVCPQ[2:0]**: AV Untagged Control Packets Queue
This field specifies the Receive queue on which the received AV untagged control packets are routed:
000: Receive Queue 0
001: Receive Queue 1
Others: Reserved, must not be used
The AV tagged control and data packets are routed based on PSRQ field in the transmit flow control register of the corresponding queue.
This field is reserved if the receive queues are not selected for AV feature.

Rx queue control 2 register (ETH_MACRXQC2R)

Address offset: 0x00A8

Reset value: 0x0000 0000

This register controls the routing of tagged packets based on the USP (user priority) field of the received packets to the Rx queue 0 and 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSRQ1[7:0]								PSRQ0[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **PSRQ1[7:0]**: Priorities Selected in the Receive Queue 1

This field defines the priorities assigned to Rx queue 1. All packets with priorities that match the values set in this field are routed to Rx queue 1.

For example, if PSRQ1[4] is set, packets with USP field equal to 4 are routed to Rx queue 1. The software must ensure that the content of this field is mutually exclusive to the PSRQ fields for other queues, that is, the same priority is not mapped to multiple Rx queues.

Bits 7:0 **PSRQ0[7:0]**: Priorities Selected in the Receive Queue 0

This field defines the priorities assigned to Rx queue 0. All packets with priorities that match the values set in this field are routed to Rx queue 0.

For example, if PSRQ0[5] is set, packets with USP field equal to 5 are routed to Rx queue 0. The software must ensure that the content of this field is mutually exclusive to the PSRQ fields for other queues, that is, the same priority is not mapped to multiple Rx queues.

Interrupt status register (ETH_MACISR)

Address offset: 0x00B0

Reset value: 0x0000 0000

The interrupt status register contains the status of interrupts.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MFRIS	MFTIS	MDIOIS	FPEIS	Res.
											r	r	rc_r	r	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RXSTSIS	TXSTSIS	TSIS	Res.	MMCTXIS	MMCRXIS	MMCIS	Res.	Res.	LPIIS	PMTIS	PHYIS	Res.	Res.	RGSMIIIS
	rc_r	rc_r	rc_r		r	r	r			r	r	r			r

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **MFRIS**: MMC FPE Receive interrupt status

This bit is set high when an interrupt is generated in the *MMC FPE Rx interrupt status register (ETH_MMC_FPE_RX_ISR)*. This bit is cleared when all bits in this interrupt register are cleared.

Bit 19 **MFTIS**: MMC FPE transmit interrupt status

This bit is set high when an interrupt is generated in the *MMC FPE Tx interrupt status register (ETH_MMC_FPE_TX_ISR)*. This bit is cleared when all bits in this interrupt register are cleared.

Bit 18 **MDIOIS**: MDIO interrupt status

This bit indicates an interrupt event after the completion of MDIO operation. To reset this bit, the application has to read this bit (or write it to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* register is set) register is set.

Bit 17 **FPEIS**: Frame preemption interrupt status

This bit indicates an interrupt event during the operation of Frame Preemption (Bits[19:16] of *FPE control and status register (ETH_MACFPECSR)* is set). To reset this bit, the application must clear the event in *FPE control and status register (ETH_MACFPECSR)* that has caused the Interrupt.

Bits 16:15 Reserved, must be kept at reset value.

Bit 14 **RXSTSIS**: Receive status interrupt

This bit indicates the status of received packets. This bit is set when the RWT bit is set in the *Rx Tx status register (ETH_MACRXTXSR)*. This bit is cleared when the corresponding interrupt source bit is read (or corresponding interrupt source bit is written to 1 when RCWE bit of *CSR software control register (ETH_MACCSRSWCR)* is set) in the ETH_MACISR register.

Bit 13 **TXSTSIS**: Transmit status interrupt

This bit indicates the status of transmitted packets. This bit is set when any of the following bits is set in the *Rx Tx status register (ETH_MACRXTXSR)*:

- Excessive Collision (EXCOL)
- Late Collision (LCOL)
- Excessive Deferral (EXDEF)
- Loss of Carrier (LCARR)
- No Carrier (NCARR)
- Jabber timeout (TJT)

This bit is cleared when the corresponding interrupt source bit is read (or corresponding interrupt source bit is written to 1 when RCWE bit of *CSR software control register (ETH_MACCSRSWCR)* is set) in the ETH_MACISR register.

Bit 12 **TSIS**: Timestamp interrupt status

If the Timestamp feature is enabled, this bit is set when any of the following conditions is true:

- The system time value is equal to or exceeds the value specified in the Target Time High and Low registers.
- There is an overflow in the Seconds register.
- The Target Time Error occurred, that is, programmed target time already elapsed.

If the Auxiliary Snapshot feature is enabled, this bit is set when the auxiliary snapshot trigger is asserted.

When drop transmit status is enabled in MTL, this bit is set when the captured transmit timestamp is updated in the *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* and *Tx timestamp status seconds register (ETH_MACTXTSSSR)* registers.

When PTP offload feature is enabled, this bit is set when the captured transmit timestamp is updated in the *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* and *Tx timestamp status seconds register (ETH_MACTXTSSSR)* registers, for PTO generated Delay Request and Pdelay request packets.

This bit is cleared when the corresponding interrupt source bit is read (or corresponding interrupt source bit is written to 1 when RCWE bit of *CSR software control register (ETH_MACCSRSWCR)* is set) in the *Timestamp status register (ETH_MACTSSR)*.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **MMCTXIS**: MMC transmit interrupt status

This bit is set high when an interrupt is generated in the *MMC Tx interrupt register (ETH_MMC_TX_INTERRUPT)*. This bit is cleared when all bits in this interrupt register are cleared.

Bit 9 **MMCRXIS**: MMC Receive interrupt status

This bit is set high when an interrupt is generated in the *MMC Rx interrupt register (ETH_MMC_RX_INTERRUPT)*. This bit is cleared when all bits in this interrupt register are cleared.

Bit 8 **MMCIS**: MMC interrupt status

This bit is set high when MMCTXIS or MMCRXIS is set high. This bit is cleared only when all these bits are low.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **LPIIS**: LPI interrupt status

This bit is set for any LPI state entry or exit in the MAC transmitter or receiver. This bit is cleared when the TLPIEN bit of *LPI control and status register (ETH_MACLCSR)* is read.

Bit 4 **PMTIS**: PMT interrupt status
This bit is set when a Magic packet or Wake-on-LAN packet is received in the power-down mode (RWKPRCVD and MGKPRCVD bits in ETH_MACPCSR register). This bit is cleared when Bits[6:5] are cleared because of a Read operation to the *PMT control status register (ETH_MACPCSR)*.

Bit 3 **PHYIS**: PHY Interrupt
This bit is set when rising edge is detected on the ETH_PHY_INTN input. This bit is cleared when this register is read.

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **RGSMIIIS**: RGMII interrupt status
This bit is set by any change in value of the Link Status of RGMII interface (LNKSTS bit in *PHYIF control status register (ETH_MACPHYCSR)* register). This bit is cleared when the ETH_MACPHYCSR register is read.

Interrupt enable register (ETH_MACIER)

Address offset: 0x00B4

Reset value: 0x0000 0000

The Interrupt Enable register contains the masks for generating the interrupts.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MDIOIE	FPEIE	Res.
													rw	rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	RXSTSIE	TXSTSIE	TSIE	Res.	Res.	Res.	Res.	Res.	Res.	LPIIE	PMTIE	PHYIE	Res.	Res.	RGSMIIIE
	rw	rw	rw							rw	rw	rw			rw

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **MDIOIE**: MDIO Interrupt Enable
When this bit is set, it enables the assertion of the interrupt when MDIOIS field is set in the *Interrupt status register (ETH_MACISR)*.

Bit 17 **FPEIE**: Frame Preemption Interrupt Enable
When this bit is set, it enables the assertion of the interrupt when FPEIS field is set in the *Interrupt status register (ETH_MACISR)*.

Bit 16 Reserved, must be kept at reset value.

Bit 15 Reserved, must be kept at reset value.

Bit 14 **RXSTSIE**: Receive Status Interrupt Enable
When this bit is set, it enables the assertion of the interrupt signal because of the setting of RXSTSIS bit in the *Interrupt status register (ETH_MACISR)*.

Bit 13 **TXSTSIE**: Transmit Status Interrupt Enable
When this bit is set, it enables the assertion of the interrupt signal because of the setting of TXSTSIS bit in the *Interrupt status register (ETH_MACISR)*.

Bit 12 **TSIE**: Timestamp Interrupt Enable
When this bit is set, it enables the assertion of the interrupt signal because of the setting of TSIS bit in *Interrupt status register (ETH_MACISR)*.

Bits 11:6 Reserved, must be kept at reset value.

Bit 5 **LPIIE**: LPI Interrupt Enable
When this bit is set, it enables the assertion of the interrupt signal because of the setting of LPIIS bit in *Interrupt status register (ETH_MACISR)*.

Bit 4 **PMTIE**: PMT Interrupt Enable
When this bit is set, it enables the assertion of the interrupt signal because of the setting of PMTIS bit in *Interrupt status register (ETH_MACISR)*.

Bit 3 **PHYIE**: PHY Interrupt Enable
When this bit is set, it enables the assertion of the interrupt signal because of the setting of PHYIS bit in *Interrupt status register (ETH_MACISR)*.

Bits 2:1 Reserved, must be kept at reset value.

Bit 0 **RGSMIIIE**: RGMII Interrupt Enable
When this bit is set, it enables the assertion of the interrupt signal because of the setting of RGSMIIIS bit in *Interrupt status register (ETH_MACISR)*.

Rx Tx status register (ETH_MACRXTXSR)

Address offset: 0x00B8

Reset value: 0x0000 0000

The receive transmit status register contains the receive and transmit error status.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RWT	Res.	Res.	EXCOL	LCOL	EXDEF	LCARR	NCARR	TJT
							rc_r			rc_r	rc_r	rc_r	rc_r	rc_r	rc_r

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **RWT**: Receive watchdog timeout
This bit is set when a packet with length greater than 2,048 bytes is received (10, 240 bytes when Jumbo Packet mode is enabled) and the WD bit is reset in the *Operating mode configuration register (ETH_MACCR)*. This bit is set when a packet with length greater than 16,383 bytes is received and the WD bit is set in the *Operating mode configuration register (ETH_MACCR)*.
Cleared on read (or write of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **EXCOL**: Excessive Collisions

When the DTXSTS bit is set in the *Operating mode register (ETH_MTLOMR)*, this bit indicates that the transmission aborted after 16 successive collisions while attempting to transmit the current packet. If the DR bit is set in the *Operating mode configuration register (ETH_MACCR)*, this bit is set after the first collision and the packet transmission is aborted.
Cleared on read (or write of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 4 **LCOL**: Late Collision

When the DTXSTS bit is set in the *Operating mode register (ETH_MTLOMR)*, this bit indicates that the packet transmission aborted because a collision occurred after the collision window (64 bytes including Preamble in MII mode; 512 bytes including Preamble and Carrier Extension in GMII mode).
This bit is not valid if the Underflow error occurs.
Cleared on read (or write of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 3 **EXDEF**: Excessive Deferral

When the DTXSTS bit is set in the *Operating mode register (ETH_MTLOMR)* and the DC bit is set in the *Operating mode configuration register (ETH_MACCR)*, this bit indicates that the transmission ended because of excessive deferral of over 24,288 bit times (155,680 in 1000 Mbps mode or when Jumbo packet is enabled).
Cleared on read (or write of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 2 **LCARR**: Loss of Carrier

When the DTXSTS bit is set in the *Operating mode register (ETH_MTLOMR)*, this bit indicates that the loss of carrier occurred during packet transmission, that is, the ETH_CRS signal was inactive for one or more transmission clock periods during packet transmission. This bit is valid only for packets transmitted without collision.
Cleared on read (or write of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 1 **NCARR**: No Carrier

When the DTXSTS bit is set in the *Operating mode register (ETH_MTLOMR)*, this bit indicates that the carrier signal from the PHY is not present at the end of preamble transmission.
Cleared on read (or write of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 0 **TJT**: Transmit Jabber timeout

This bit indicates that the transmit jabber timer expired which happens when the packet size exceeds 2,048 bytes (10,240 bytes when the Jumbo packet is enabled) and JD bit is reset in the *Operating mode configuration register (ETH_MACCR)*. This bit is set when the packet size exceeds 16,383 bytes and the JD bit is set in the *Operating mode configuration register (ETH_MACCR)*.
Cleared on read (or write of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

PMT control status register (ETH_MACPCSR)

Address offset: 0x00C0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RWKFILTRST	Res.	Res.	RWKPTR[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw			r	r	r	r	r								

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	RWKPFE	GLBLUCAST	Res.	Res.	RWKPRCVD	MGKPRCVD	Res.	Res.	RWKPKTEN	MGKPKTEN	PWRDWN
					rw	rw			r	rc_r			rw	rw	rw

Bit 31 **RWKFILTRST**: Remote Wake-up Packet Filter Register Pointer Reset

When this bit is set, the remote wake-up packet filter register pointer is reset to 0. It is automatically cleared after 1 clock cycle.

Bits 30:29 Reserved, must be kept at reset value.

Bits 28:24 **RWKPTR[4:0]**: Remote wake-up FIFO Pointer

This field gives the current value (0 to 7) of the Remote wake-up Packet Filter register pointer. When the value of this pointer is equal to 7, the contents of the Remote wake-up Packet Filter Register are transferred to the eth_rx_clk domain when a Write occurs to that register.

Bits 23:11 Reserved, must be kept at reset value.

Bit 10 **RWKPFE**: Remote wake-up Packet Forwarding Enable

When this bit is set along with RWKPKTEN, the MAC receiver drops all received frames until it receives the expected wake-up frame. All frames after that event including the received wake-up frame are forwarded to application. This bit is then self-cleared on receiving the wake-up packet.

The application can also clear this bit before the expected wake-up frame is received. In such cases, the MAC reverts to the default behavior where packets received are forwarded to the application. This bit must only be set when RWKPKTEN is set high and PWRDWN is set low. The setting of this bit has no effect when PWRDWN is set high.

Note: If Magic Packet Enable and wake-up Frame Enable are both set along with setting of this bit and Magic Packet is received prior to wake-up frame, this bit is self-cleared on receiving Magic Packet, the received Magic packet is dropped, and all frames after received Magic Packet are forwarded to application.

Bit 9 **GLBLUCAST**: Global Unicast

When this bit set, any unicast packet filtered by the MAC (DAF) address recognition is detected as a remote wake-up packet.

Bits 8:7 Reserved, must be kept at reset value.

Bit 6 **RWKPRCVD**: Remote wake-up Packet Received

When this bit is set, it indicates that the power management event is generated because of the reception of a remote wake-up packet. This bit is cleared when this register is read.

Bit 5 **MGKPRCVD**: Magic Packet Received

When this bit is set, it indicates that the power management event is generated because of the reception of a magic packet. This bit is cleared when this register is read (or written to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **RWKPKTEN**: Remote wake-up Packet Enable

When this bit is set, a power management event is generated when the MAC receives a remote wake-up packet.

Bit 1 **MGKPKTEN**: Magic Packet Enable

When this bit is set, a power management event is generated when the MAC receives a magic packet.

Bit 0 **PWRDWN**: Power Down

When this bit is set, the MAC receiver drops all received packets until it receives the expected magic packet or remote wake-up packet. This bit is then self-cleared and the power-down mode is disabled. The software can clear this bit before the expected magic packet or remote wake-up packet is received. The packets received by the MAC after this bit is cleared are forwarded to the application. This bit must only be set when the Magic Packet Enable, Global Unicast, or Remote wake-up Packet Enable bit is set high.

Note: You can gate-off the CSR clock during the power-down mode. However, when the CSR clock is gated-off, you cannot perform any read or write operations on this register. Therefore, the Software cannot clear this bit.

Remote wake-up packet filter register (ETH_MACRWKPFR)

Address offset: 0x00C4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MACRWKPFR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MACRWKPFR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MACRWKPFR[31:0]**: Remote wake-up packet filter

Refer to *Table 817*, *Table 818* and *Table 819* for details on register content and programming sequence.

The ETH_MACRWKPFR register at address 0x00C4 loads the wake-up Packet Filter register.

To load values in a wake-up Packet Filter register, the entire register (ETH_MACRWKPFR) must be written. The ETH_MACRWKPFR register is loaded by sequentially loading the eight, sixteen or thirty two register values in address (0x00C4) for ETH_MACRWKPFR value 0 to 7, respectively. The ETH_MACRWKPFR register is read in a similar way. The Ethernet peripheral updates the ETH_MACRWKPFR register current pointer value in Bits[26:24] of ETH_MACPCSR register.

LPI control and status register (ETH_MACLCSR)

Address offset: 0x00D0

Reset value: 0x0000 0000

The LPI Control and Status Register controls the LPI functions and provides the LPI interrupt status. The status bits are cleared when this register is read.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPITCSE	LPITE	LPITXA	PLSEN	PLS	LPIEN
										rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	RLPIST	TLPIST	Res.	Res.	Res.	Res.	RLPIEX	RLPIEN	TLPIEX	TLPIEN
						r	r					r	r	r	r

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **LPITCSE**: LPI Tx Clock Stop Enable

When this bit is set, the MAC asserts sbd_tx_clk_gating_ctrl_o signal high after it enters Tx LPI mode to indicate that the Tx clock to MAC can be stopped. When this bit is reset, the MAC does not assert sbd_tx_clk_gating_ctrl_o signal high after it enters Tx LPI mode.
If RGMII Interface is selected, the Tx clock is required for transmitting the LPI pattern. The Tx Clock cannot be gated and so the LPITCSE bit cannot be programmed.

Bit 20 **LPITE**: LPI Timer Enable

This bit controls the automatic entry of the MAC transmitter into and exit out of the LPI state. When LPITE, LPITXA and LPIEN bits are set, the MAC transmitter enters LPI state only when the complete MAC TX data path is IDLE for a period indicated by the ETH_MACLETR register.

After entering LPI state, if the data path becomes non-IDLE (due to a new packet being accepted for transmission), the transmitter exits LPI state but does not clear LPIEN bit. This enables the re-entry into LPI state when it is IDLE again.

When LPITE is 0, the LPI Auto timer is disabled and MAC transmitter enters LPI state based on the settings of LPITXA and LPIEN bit descriptions.

Bit 19 **LPITXA**: LPI Tx Automate

This bit controls the behavior of the MAC when it is entering or coming out of the LPI mode on the transmit side.

If the LPITXA and LPIEN bits are set to 1, the MAC enters the LPI mode only after all outstanding packets (in the core) and pending packets (in the application interface) have been transmitted. The MAC comes out of the LPI mode when the application sends any packet for transmission or the application issues a Tx FIFO Flush command. In addition, the MAC automatically clears the LPIEN bit when it exits the LPI state. If Tx FIFO Flush is set in the FTQ bit of ETH_MTLTxQxOMR, when the MAC is in the LPI mode, it exits the LPI mode.

When this bit is 0, the LPIEN bit directly controls behavior of the MAC when it is entering or coming out of the LPI mode.

Bit 18 **PLSEN**: PHY Link Status Enable

This bit enables the link status received on the RGMII Receive paths to be used for activating the LPI LS TIMER.

When this bit is set, the MAC uses the link-status bits of the ETH_MACPHYCSR register and the PLS bit for the LPI LS Timer trigger. When this bit is reset, the MAC ignores the link-status bits of the ETH_MACPHYCSR register and takes only the PLS bit.

Bit 17 **PLS**: PHY Link Status

This bit indicates the link status of the PHY. The MAC transmitter asserts the LPI pattern only when the link status is up (OKAY) at least for the time indicated by the LPI LS TIMER.

When this bit is set, the link is considered to be okay (UP) and when this bit is reset, the link is considered to be down.

Bit 16 **LPIEN**: LPI Enable

When this bit is set, it instructs the MAC transmitter to enter the LPI state. When this bit is reset, it instructs the MAC to exit the LPI state and resume normal transmission.

This bit is cleared when the LPITXA bit is set and the MAC exits the LPI state because of the arrival of a new packet for transmission.

Bits 15:10 Reserved, must be kept at reset value.

Bit 9 **RLPIST**: Receive LPI State

When this bit is set, it indicates that the MAC is receiving the LPI pattern on the GMII or MII interface.

Bit 8 **TLPIST**: Transmit LPI State

When this bit is set, it indicates that the MAC is transmitting the LPI pattern on the GMII or MII interface.

Bits 7:4 Reserved, must be kept at reset value.

Bit 3 **RLPIEX**: Receive LPI Exit

When this bit is set, it indicates that the MAC receiver has stopped receiving the LPI pattern on the GMII or MII interface, exited the LPI state, and resumed the normal reception. This bit is cleared by a read into this register (or by writing it to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Note: This bit may not be set if the MAC stops receiving the LPI pattern for a very short duration, such as, less than three clock cycles of CSR clock.

Bit 2 **RLPIEN**: Receive LPI Entry

When this bit is set, it indicates that the MAC receiver has received an LPI pattern and entered the LPI state. This bit is cleared by a read into this register (or by writing it to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Note: This bit may not be set if the MAC stops receiving the LPI pattern for a very short duration, such as, less than three clock cycles of CSR clock.

Bit 1 **TLPIEX**: Transmit LPI Exit

When this bit is set, it indicates that the MAC transmitter exited the LPI state after the application cleared the LPIEN bit and the LPI TW Timer has expired. This bit is cleared by a read into this register (or by writing it to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 0 **TLPIEN**: Transmit LPI Entry

When this bit is set, it indicates that the MAC transmitter has entered the LPI state because of the setting of the LPIEN bit. This bit is cleared by a read into this register (or by writing it to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

LPI timers control register (ETH_MACLTCR)

Address offset: 0x00D4

Reset value: 0x03E8 0000

The LPI timers control register controls the timeout values in the LPI states. It specifies the time for which the MAC transmits the LPI pattern and also the time for which the MAC waits before resuming the normal transmission.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	LST[9:0]									
						rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TWT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bits 25:16 **LST[9:0]**: LPI LS Timer

This field specifies the minimum time (in milliseconds) for which the link status from the PHY should be up (OKAY) before the LPI pattern can be transmitted to the PHY. The MAC does not transmit the LPI pattern even when the LPIEN bit is set unless the LPI LS Timer reaches the programmed terminal count. The default value of the LPI LS Timer is 1000 (1 s) as defined in the IEEE standard.

Bits 15:0 **TWT[15:0]**: LPI TW Timer

This field specifies the minimum time (in microseconds) for which the MAC waits after it stops transmitting the LPI pattern to the PHY and before it resumes the normal transmission. The TLPIEX status bit is set after the expiry of this timer.

LPI entry timer register (ETH_MACLETR)

Address offset: 0x00D8

Reset value: 0x0000 0000

This register controls the Tx LPI entry timer. This counter is enabled only when LPITE bit of *LPI control and status register (ETH_MACLCSR)* register is set to 1.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPIET[19:16]			
												rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LPIET[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r	r	r

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:0 **LPIET[19:0]**: LPI Entry Timer

This field specifies the time in microseconds the MAC waits to enter LPI mode, after it has transmitted all the frames. This field is valid and used only when LPITE and LPITXA are set to 1.

Bits [2:0] are read-only so that the granularity of this timer is in steps of 8 micro-seconds.

One-microsecond-tick counter register (ETH_MAC1USTCR)

Address offset: 0x00DC

Reset value: 0x0000 0063

This register controls the generation of the Reference time (one-microsecond tick) for all the LPI timers. This timer has to be programmed by the software initially.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TIC_1US_CNTR[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 TIC_1US_CNTR[11:0]: 1 µs tick Counter

The application must program this counter so that the number of clock cycles of CSR clock is 1 µs (subtract 1 from the value before programming).

For example if the CSR clock is 100 MHz then this field needs to be programmed to 100 - 1 = 99 (which is 0x63).

This is required to generate the 1 µs events that are used to update some of the EEE related counters.

PHYIF control status register (ETH_MACPHYCSR)

Address offset: 0x00F8

Reset value: 0x0000 0000

The PHY Interface Control and Status register indicates the status signals received by the RGMII interface from the PHY.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LNKSTS	LNKSPEED[1:0]		LNKMOD
												r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	LUD	TC
														rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **LNKSTS**: Link Status

This bit indicates whether the link is up or down.

0: Link down

1: Link up

Bits 18:17 **LNKSPEED[1:0]**: Link Speed

This bit indicates the current speed of the link:

00: 2.5 MHz

01: 25 MHz

10: 125 MHz

11: Reserved, must not be used

Bit 16 **LNKMOD**: Link Mode

This bit indicates the current operating mode of the link:

0: Half-duplex mode

1: Full-duplex mode

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **LUD**: Link Up or Down

This bit indicates whether the link is up or down during transmission of configuration in the RGMII interface.

0: Link Down

1: Link Up

Bit 0 **TC**: Transmit Configuration in RGMII

When set, this bit enables the transmission of duplex mode, link speed, and link up or down information to the PHY in the RGMII port. When this bit is reset, no such information is driven to the PHY (see *Section 76.6.3: Reduced media independent interface (RMII)*).

Version register (ETH_MACVR)

Address offset: 0x0110

Reset value: 0x0000 2052

The version register identifies the version of the Ethernet peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
USERVER[7:0]								SNPSVER[7:0]							
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **USERVER[7:0]**: ST-defined version

Bits 7:0 **SNPSVER[7:0]**: IP version

Debug register (ETH_MACDR)

Address offset: 0x0114

Reset value: 0x0000 0000

The Debug register provides the debug status of various MAC blocks.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFCSTS[1:0]		TPESTS
													r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFCFCSTS[1:0]		RPESTS
													r	r	r

Bits 31:19 Reserved, must be kept at reset value.

Bits 18:17 **TFCSTS[1:0]**: MAC transmit packet controller status

This field indicates the state of the MAC transmit packet controller module:

00: Idle state

01: Waiting for one of the following:

– Status of the previous packet

– IPG or backoff period to be over

10: Generating and transmitting a Pause control packet (in Full-duplex mode)

11: Transferring input packet for transmission

Bit 16 **TPESTS**: MAC GMII or MII transmit protocol engine status

When this bit is set, it indicates that the MAC GMII or MII transmit protocol engine is actively transmitting data, and it is not in the Idle state.

Bits 15:3 Reserved, must be kept at reset value.

Bits 2:1 **RFCFCSTS[1:0]**: MAC receive packet controller FIFO status

When this bit is set, this field indicates the active state of the small FIFO Read and Write controllers of the MAC receive packet controller module.

Bit 0 **RPESTS**: MAC GMII or MII receive protocol engine status

When this bit is set, it indicates that the MAC GMII or MII receive protocol engine is actively receiving data, and it is not in the Idle state.

HW feature 0 register (ETH_MACHWF0R)

Address offset: 0x011C

Reset value: 0x0A0D 73F7

This register indicates the presence of first set of the optional features or functions of the Ethernet peripheral. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACTPHYSEL[3:0]				SAVLANINS	TSSTSSEL[1:0]		MACADR64SEL	MACADR32SEL	ADDMACADRSEL[4:0]					Res.	RXCOESEL
r	r	r	r	r	r	r	r	r	r	r	r	r	r		r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	TXCOESEL	EEESEL	TSSEL	Res.	Res.	ARPOFFSEL	MMCSEL	MGKSEL	RWKSEL	SMASEL	VLHASH	PCSSEL	HDSEL	GMIISEL	MIISEL
	r	r	r			r	r	r	r	r	r	r	r	r	r

Bits 31:28 **ACTPHYSEL[3:0]**: Active PHY selected
When you have multiple PHY interfaces in your configuration, this field indicates the sampled value of phy_intf_sel_i during reset de-assertion:
0x0: GMII or MII
0x1: RGMII
0x2: SGMII
0x3: TBI
0x4: RMII
0x5: RTBI
0x6: SMII
Others: Reserved, must not be used

Bit 27 **SAVLANINS**: Source Address or VLAN Insertion Enable
This bit is set to 1 when the Enable SA and VLAN Insertion on Tx option is selected

Bits 26:25 **TSSTSSEL[1:0]**: Timestamp System Time Source
This bit indicates the source of the Timestamp system time:
01: Internal
10: External
11: Both
00: Reserved, must not be used
This bit is set to 1 when the Enable IEEE 1588 Timestamp Support option is selected

Bit 24 **MACADR64SEL**: MAC Addresses 64-127 Selected
This bit is set to 1 when the Enable Additional 64 MAC Address Registers (64-127) option is selected

Bit 23 **MACADR32SEL**: MAC Addresses 32-63 Selected
This bit is set to 1 when the Enable Additional 32 MAC Address Registers (32-63) option is selected

Bits 22:18 **ADDMACADRSEL[4:0]**: MAC Addresses 1-31 Selected
This bit is set to 1 when the Enable Additional 1-31 MAC Address Registers option is selected

Bit 17 Reserved, must be kept at reset value.

Bit 16 **RXCOESEL**: Receive checksum offload Enabled
This bit is set to 1 when the enable receive TCP/IP checksum check option is selected

Bit 15 Reserved, must be kept at reset value.

Bit 14 **TXCOESEL**: Transmit Checksum Offload Enabled
This bit is set to 1 when the enable transmit TCP/IP checksum insertion option is selected

Bit 13 **EEESEL**: Energy Efficient Ethernet Enabled
This bit is set to 1 when the Enable Energy Efficient Ethernet (EEE) option is selected

Bit 12 **TSSEL**: IEEE 1588-2008 Timestamp Enabled
This bit is set to 1 when the Enable IEEE 1588 Timestamp Support option is selected

Bits 11:10 Reserved, must be kept at reset value.

Bit 9 **ARPOFFSEL**: ARP Offload Enabled
This bit is set to 1 when the Enable IPv4 ARP Offload option is selected

Bit 8 **MMCSEL**: RMON Module Enable
This bit is set to 1 when the Enable MAC management counters (MMC) option is selected

Bit 7 **MGKSEL**: PMT Magic Packet Enable
This bit is set to 1 when the Enable Magic Packet Detection option is selected

Bit 6 **RWKSEL**: PMT Remote Wake-up Packet Enable
This bit is set to 1 when the Enable Remote Wake-up Packet Detection option is selected

Bit 5 **SMASEL**: SMA (MDIO) Interface
This bit is set to 1 when the Enable Station Management (MDIO Interface) option is selected

Bit 4 **VLHASH**: VLAN Hash Filter Selected
This bit is set to 1 when the Enable VLAN Hash Table Based Filtering option is selected

Bit 3 **PCSSEL**: PCS Registers (TBI, SGMII, or RTBI PHY interface)
This bit is set to 1 when the TBI, SGMII, or RTBI PHY interface option is selected

Bit 2 **HDSEL**: Half-duplex Support
This bit is set to 1 when the Half-duplex mode is selected

Bit 1 **GMIISEL**: 1000 Mbit/s Support
This bit is set to 1 when 1000 Mbit/s is selected as operating mode.

Bit 0 **MIISEL**: 10 or 100 Mbit/s Support
This bit is set to 1 when 10/100 Mbit/s is selected as operating mode.

HW feature 1 register (ETH_MACHWF1R)

Address offset: 0x0120

Reset value: 0x1114 1965

This register indicates the presence of second set of the optional features or functions of the Ethernet peripheral. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	L3L4FNUM[3:0]				Res.	HASHTBLSZ[1:0]		POUOST	Res.	RAVSEL	AVSEL	DBGMEMA	TSOEN	SPHEN	DCBEN
	r	r	r	r		r	r	r		r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR64[1:0]		ADVTHWORD	PTOEN	OSTEN	TXFIFOSIZE[4:0]					SPRAM	RXFIFOSIZE[4:0]				
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bits 30:27 **L3L4FNUM[3:0]**: Total number of L3 or L4 Filters
This field indicates the total number of L3 or L4 filters:
0000: No L3 or L4 Filter
0001: 1 L3 or L4 Filter
0010: 2 L3 or L4 Filters
..
1000: 8 L3 or L4

Bit 26 Reserved, must be kept at reset value.

Bits 25:24 **HASHTBLSZ[1:0]**: Hash Table Size
This field indicates the size of the Hash table:
00: No Hash table
01: 64
10: 128
11: 256

Bit 23 **POUOST**: One Step for PTP over UDP/IP Feature Enable
This bit is set to 1 when the Enable one step timestamp for PTP over UDP/IP feature is selected.

Bit 22 Reserved, must be kept at reset value.

Bit 21 **RAVSEL**: Rx Side Only AV Feature Enable
This bit is set to 1 when the Enable Audio video bridging option on Rx Side Only is selected.

Bit 20 **AVSEL**: AV Feature Enable
This bit is set to 1 when the Enable Audio video bridging option is selected.

Bit 19 **DBGMEMA**: DMA debug registers enable
This bit is set to 1 when the Debug Mode Enable option is selected

Bit 18 **TSOEN**: TCP segmentation offload enable
This bit is set to 1 when the Enable TCP Segmentation Offloading for TCP/IP Packets option is selected

Bit 17 **SPHEN**: Split header feature enable
This bit is set to 1 when the Enable Split Header Structure option is selected

Bit 16 **DCBEN**: DCB feature enable
This bit is set to 1 when the Enable Data Center Bridging option is selected

Bits 15:14 **ADDR64[1:0]**: Address width
This field indicates the configured address width.
00: 32 bits
Others: Reserved, must not be used

Bit 13 **ADVTHWORD**: IEEE 1588 High Word Register enable
This bit is set to 1 when the Add IEEE 1588 Higher Word Register option is selected

Bit 12 **PTOEN**: PTP offload enable
This bit is set to 1 when the Enable PTP Timestamp Offload Feature is selected.

Bit 11 **OSTEN**: One-step timestamping enable
This bit is set to 1 when the Enable One-Step Timestamp Feature is selected.

Bits 10:6 **TXFIFOSIZE[4:0]**: MTL transmit FIFO size
This field contains the configured value of MTL Tx FIFO in bytes expressed as Log to base 2 minus 7, that is, Log2(TXFIFO_SIZE) -7:
00000: 128 bytes
00001: 256 bytes
00010: 512 bytes
00011: 1,024 bytes
00100: 2,048 bytes
00101: 4,096 bytes
00110: 8,192 bytes
00111: 16,384 bytes
01000: 32 Kbytes
01001: 64 Kbytes
01010: 128 Kbytes
01011 to 11111: Reserved, must not be used

Bit 5 **SPRAM**: Single Port RAM Enable
This bit is set to 1 when the Use single port RAM feature is selected.

Bits 4:0 **RXFIFOSIZE[4:0]**: MTL receive FIFO Size
This field contains the configured value of MTL Rx FIFO in bytes expressed as Log to base 2 minus 7, that is, Log2(RXFIFO_SIZE) -7:
00000: 128 bytes
00001: 256 bytes
00010: 512 bytes
00011: 1,024 bytes
00100: 2,048 bytes
00101: 4,096 bytes
00110: 8,192 bytes
00111: 16,384 bytes
01000: 32 Kbytes
01001: 64 Kbytes
01010: 128 Kbytes
01011: 256 Kbytes
01100 to 11111: Reserved, must not be used

HW feature 2 register (ETH_MACHWF2R)

Address offset: 0x0124

Reset value: 0x4204 1041

This register indicates the presence of third set of the optional features or functions of the Ethernet peripheral. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	AUXSNAPNUM[2:0]			Res.	PPSOUTNUM[2:0]			TDCSZ[1:0]		TXCHCNT[3:0]				RDCSZ[1:0]	
	r	r	r		r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXCHCNT[3:0]				Res.	Res.	TXQCNT[3:0]				Res.	Res.	RXQCNT[3:0]			
r	r	r	r			r	r	r	r			r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **AUXSNAPNUM[2:0]**: Number of auxiliary snapshot inputs
This field indicates the number of auxiliary snapshot inputs:
000: No auxiliary input
001: 1 auxiliary input
010: 2 auxiliary inputs
011: 3 auxiliary inputs
100: 4 auxiliary inputs
101 to 111: Reserved, must not be used

Bit 27 Reserved, must be kept at reset value.

Bits 26:24 **PPSOUTNUM[2:0]**: Number of PPS outputs
This field indicates the number of PPS outputs:
000: No PPS output
001: 1 PPS output
010: 2 PPS outputs
011: 3 PPS outputs
100: 4 PPS outputs
101 to 111: Reserved, must not be used

Bits 23:22 **TDCSZ[1:0]**: Tx DMA descriptor cache size in terms of 16-byte descriptors
00: Cache not configured
01: Four 16-byte descriptors
10: Eight 16-byte descriptors
11: Sixteen 16-byte descriptors

Bits 21:18 **TXCHCNT[3:0]**: Number of DMA transmit channels
This field indicates the number of DMA transmit channels:
0000: 1 DMA Tx Channel
0001: 2 DMA Tx Channels
..
0111: 8 DMA Tx

Bits 17:16 **RDCSZ[1:0]**: Rx DMA Descriptor Cache Size in terms of 16-byte descriptors
00: Cache not configured
01: Four 16-byte descriptors
10: Eight 16-byte descriptors
11: Sixteen 16-byte descriptors

Bits 15:12 **RXCHCNT[3:0]**: Number of DMA receive channels
This field indicates the number of DMA receive channels:
0000: 1 DMA Rx Channel
0001: 2 DMA Rx Channels
..
0111: 8 DMA Rx

Bits 11:10 Reserved, must be kept at reset value.

Bits 9:6 **TXQCNT[3:0]**: Number of MTL transmit queues
This field indicates the number of MTL transmit queues:
0000: 1 MTL Tx queue
0001: 2 MTL Tx queues
..
0111: 8 MTL Tx

Bits 5:4 Reserved, must be kept at reset value.

Bits 3:0 **RXQCNT[3:0]**: Number of MTL receive queues

This field indicates the number of MTL receive queues:
0000: 1 MTL Rx queue
0001: 2 MTL Rx queues
..
0111: 8 MTL Rx

HW feature 3 register (ETH_MACHWF3R)

Address offset: 0x0128

Reset value: 0x0C33 0031

This register indicates the presence of fourth set the optional features or functions of the Ethernet peripheral. The software driver can use this register to dynamically enable or disable the programs related to the optional blocks.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	ASP[1:0]		TBSSEL	FPESEL	Res.	Res.	Res.	Res.	ESTWID[1:0]		ESTDEP[2:0]			ESTSEL
		r	r	r	r					r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	FRPES[1:0]		FRPBS[1:0]		FRPSEL	PDUPSEL	Res.	Res.	Res.	DVLAN	CBTISEL	Res.	NRVF[2:0]		
	r	r	r	r	r	r				r	r		r	r	r

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:28 **ASP[1:0]**: Automotive Safety Package
Following are the encoding for the different Safety features
00: None
01: ECC only
10: AS_NPPE
11: AS_PPE

Bit 27 **TBSSEL**: Time-based scheduling Enable
This bit is set to 1 when the time-based scheduling feature is selected.

Bit 26 **FPESEL**: Frame Preemption Enable
This bit is set to 1 when the Enable Frame preemption feature is selected.

Bits 25:22 Reserved, must be kept at reset value.

Bits 21:20 **ESTWID[1:0]**: Width of the Time Interval field in the Gate Control List
This field indicates the width of the Configured Time Interval Field
00: No width
01: 16
10: 20
11: 24

Bits 19:17 **ESTDEP[2:0]**: Depth of the Gate Control List
This field indicates the depth of Gate Control list
000: No depth
001: 64
010: 128
011: 256
100: 512
101: 1024
110: Reserved, must not be used

Bit 16 **ESTSEL**: Enhancements to Scheduled Traffic Enable
This bit is set to 1 when the enable enhancements to scheduling traffic feature is selected.

Bit 15 Reserved, must be kept at reset value.

Bits 14:13 **FRPES[1:0]**: Flexible receive parser table entries size
This field indicates the maximum number of parser entries supported by flexible receive parser.
00: 64 entries
01: 128 entries
10: 256 entries
11: Reserved, must not be used

Bits 12:11 **FRPBS[1:0]**: Flexible receive parser buffer size
This field indicates the supported maximum number of bytes of the packet data to be parsed by flexible receive parser.
00: 64 bytes
01: 128 bytes
10: 256 bytes
11: Reserved, must not be used

Bit 10 **FRPSEL**: Flexible receive parser Selected
This bit is set to 1 when the Enable Flexible Programmable receive parser option is selected.

Bit 9 **PDUPSEL**: Broadcast/Multicast Packet Duplication
This bit is set to 1 when the Broadcast/Multicast Packet Duplication feature is selected.

Bits 8:6 Reserved, must be kept at reset value.

Bit 5 **DVLAN**: Double VLAN processing enable
This bit is set to 1 when Double VLAN processing is enabled.

Bit 4 **CBTISEL**: Queue/Channel based VLAN tag insertion on Tx enable
This bit is set to 1 when the Enable Queue/Channel based VLAN tag insertion on Tx feature is selected.

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **NRVF[2:0]**: Number of Extended VLAN Tag Filters Enabled
This field indicates the Number of Extended VLAN Tag Filters selected:
000: No Extended Rx VLAN Filters
001: 4 Extended Rx VLAN Filters
010: 8 Extended Rx VLAN Filters
011: 16 Extended Rx VLAN Filters
100: 24 Extended Rx VLAN Filters
101: 32 Extended Rx VLAN Filters
110 to 111: Reserved, must not be used

MDIO address register (ETH_MACMDIOAR)

Address offset: 0x0200

Reset value: 0x0000 0000

The MDIO address register controls the management cycles to external PHY through a management interface.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	PSE	BTB	PA[4:0]					RDA[4:0]				
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	NTC[2:0]			CR[3:0]				Res.	Res.	Res.	SKAP	GOC[1:0]		C45E	GB
	rw	rw	rw	rw	rw	rw	rw				rw	rw	rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **PSE**: Preamble suppression enable

When this bit is set, the SMA suppresses the 32-bit preamble and transmit MDIO frames with only 1 preamble bit.

When this bit is 0, the MDIO frame always has 32 bits of preamble as defined in the IEEE specifications.

Bit 26 **BTB**: Back-to-back transactions

When this bit is set and the NTC has value greater than 0, then the MAC informs the completion of a read or write command at the end of frame transfer (before the trailing clocks are transmitted). The software can thus initiate the next command which is executed immediately irrespective of the number trailing clocks generated for the previous frame.

When this bit is reset, then the read/write command completion (GMII busy is cleared) only after the trailing clocks are generated. In this mode, it is ensured that the NTC is always generated after each frame.

This bit must not be set when NTC=0.

Bits 25:21 **PA[4:0]**: Physical layer address

This field indicates which Clause 22 PHY devices (out of 32 devices) the MAC is accessing.
This field indicates which Clause 45 capable PHYs (out of 32 PHYs) the MAC is accessing.

Bits 20:16 **RDA[4:0]**: Register/Device address

These bits select the PHY register in selected Clause 22 PHY device. These bits select the Device (MMD) in selected Clause 45 capable PHY.

Bit 15 Reserved, must be kept at reset value.

Bits 14:12 **NTC[2:0]**: Number of training cocks

This field controls the number of trailing clock cycles generated on ETH_MDC after the end of transmission of MDIO frame. The valid values can be from 0 to 7. Programming the value to 011 indicates that there are additional three clock cycles on the MDC line after the end of MDIO frame transfer.

Bits 11:8 **CR[3:0]**: CSR clock range

The CSR Clock Range selection determines the frequency of the MDC clock according to the CSR clock frequency (eth_hclk):

0000: MDC clock = eth_hclk / 42

0001: MDC clock = eth_hclk / 62

0010: MDC clock = eth_hclk / 16

0011: MDC clock = eth_hclk / 26

0100: MDC clock = eth_hclk / 102

0101: MDC clock = eth_hclk / 124

0110: MDC clock = eth_hclk / 204

0111: MDC clock = eth_hclk / 324

The suggested range of CSR clock frequency applicable for each value (when Bit 11 = 0) ensures that the MDC clock is approximately between 1.0 MHz to 2.5 MHz frequency range.

When Bit 11 is set, you can achieve a higher frequency of the MDC clock than the frequency limit of 2.5 MHz (specified in the IEEE 802.3) and program a clock divider of lower value. For example, when CSR clock is of 100 MHz frequency and you program these bits to 1010, the resultant MDC clock is of 12.5 MHz which is above the range specified in IEEE 802.3. Program the following values only if the interfacing chips support faster MDC clocks:

1000: eth_hclk / 4

1001: eth_hclk / 6

1010: eth_hclk / 8

1011: eth_hclk / 10

1100: eth_hclk / 12

1101: eth_hclk / 14

1110: eth_hclk / 16

1111: eth_hclk / 18

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **SKAP**: Skip address packet

When this bit is set, the SMA does not send the address packets before read, write, or post-read increment address packets. This bit is valid only when C45E is set.

Bits 3:2 **GOC[1:0]**: GMII operation command

This bit indicates the operation command to the PHY.

00: Reserved, must not be used

01: Write

10: Post Read Increment Address for Clause 45 PHY

11: Read

When Clause 22 PHY is enabled, only Write and Read commands are valid.

Bit 1 **C45E**: Clause 45 PHY Enable

When this bit is set, Clause 45 capable PHY is connected to MDIO. When this bit is reset, Clause 22 capable PHY is connected to MDIO.

Bit 0 **GB**: GMII Busy

The application sets this bit to instruct the SMA to initiate a Read or Write access to the MDIOS. The MAC clears this bit after the MDIO frame transfer is completed. Hence the software must not write or change any of the fields in *MDIO address register (ETH_MACMDIOAR)* and *MDIO data register (ETH_MACMDIODR)* as long as this bit is set.

For write transfers, the application must first write 16-bit data in the GD field (and also RA field when C45E is set) in *MDIO data register (ETH_MACMDIODR)* register before setting this bit. When C45E is set, it should also write into the RA field of *MDIO data register (ETH_MACMDIODR)* before initiating a read transfer. When a read transfer is completed (GMII busy=0), the data read from the PHY register is valid in the GD field of the *MDIO data register (ETH_MACMDIODR)*.

Note: Even if the addressed PHY is not present, there is no change in the functionality of this bit.

MDIO data register (ETH_MACMDIODR)

Address offset: 0x0204

Reset value: 0x0000 0000

The MDIO data register stores the write data to be written to the PHY register located at the address specified in the *MDIO address register (ETH_MACMDIOAR)*. This register also stores the read data from the PHY register located at the address specified in the MDIO address register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GD[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **RA[15:0]**: Register address

This field is valid only when C45E is set. It contains the Register Address in the PHY to which the MDIO frame is intended for.

Bits 15:0 **GD[15:0]**: GMII data

This field contains the 16-bit data value read from the PHY after a Management Read operation or the 16-bit data value to be written to the PHY before a Management Write operation.

ARP address register (ETH_MACARPAR)

Address offset: 0x0210

Reset value: 0x0000 0000

The ARP Address register contains the IPv4 Destination Address of the MAC.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARPPA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARPPA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ARPPA[31:0]**: ARP Protocol Address

This field contains the IPv4 Destination Address of the MAC. This address is used for perfect match with the Protocol Address of Target field in the received ARP packet.

CSR software control register (ETH_MACCSRSWCR)

Address offset: 0x0230

Reset value: 0x0000 0000

This register contains software-programmable controls for changing the CSR access response and status bits clearing.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RCWE
							rw								rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **SEEN**: Slave Error Response Enable

When this bit is set, the MAC responds with a Slave Error for accesses to reserved registers in CSR space.

When this bit is reset, the MAC responds with an Okay response to any register accessed from CSR space.

Bits 7:1 Reserved, must be kept at reset value.

Bit 0 **RCWE**: Register Clear on Write 1 Enable

When this bit is set, the access mode to some register fields changes to rc_w1 (clear on write) meaning that the application needs to set that respective bit to 1 to clear it.

When this bit is reset, the access mode to these register fields remains rc_r (clear on read).

FPE control and status register (ETH_MACFPECSR)

Address offset: 0x0234

Reset value: 0x0000 0000

This register controls the operation of Frame Preemption.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRSP	TVER	RRSP	RVER
												rc_r	rc_r	rc_r	rc_r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRSP	SVER	EFPE
													rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **TRSP**: Transmitted respond frame

Set when a Respond mPacket is transmitted (triggered by setting SRSP field). An interrupt can be generated for this event if FPEIE bit of *Interrupt enable register (ETH_MACIER)* is set.
Cleared on read (or write of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 18 **TVER**: Transmitted verify frame

Set when a Verify mPacket is transmitted (triggered by setting SVER field). An interrupt can be generated for this event if FPEIE bit of *Interrupt enable register (ETH_MACIER)* is set.
Cleared on read (or write of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 17 **RRSP**: Received respond frame

Set when a Respond mPacket is received. An interrupt can be generated for this event if FPEIE bit of *Interrupt enable register (ETH_MACIER)* is set.
Cleared on read (or write of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 16 **RVER**: Received verify frame

Set when a Verify mPacket is received. An interrupt can be generated for this event if FPEIE bit of *Interrupt enable register (ETH_MACIER)* is set.
Cleared on read (or write of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bits 15:3 Reserved, must be kept at reset value.

Bit 2 **SRSP**: Send Respond mPacket

When set, this bit indicates to the hardware that a Respond mPacket must be sent.
It is cleared by hardware after sending the Respond mPacket.
Writing 1 sets. Self-cleared.

Bit 1 **SVER**: Send Verify mPacket

When set, this bit indicates to the hardware that a Verify mPacket must be sent. Reset by hardware after sending the Verify mPacket.
Writing 1 sets. Self-cleared.

Bit 0 **EFPE**: Enable Tx Frame Preemption

When set, Frame Preemption Tx functionality is enabled.

MAC presentation time register (ETH_MACPRSTIMR)

Address offset: 0x0240

Reset value: 0x0000 0000

This register contains the 32-bit binary rollover equivalent time of the PTP System Time expressed in ns.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MPTN[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MPTN[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MPTN[31:0]**: MAC 1722 Presentation Time in ns

These bits indicate the value of the 32-bit binary rollover equivalent time of the PTP System Time in ns.

MAC presentation time update register (ETH_MACPRSTIMUR)

Address offset: 0x0244

Reset value: 0x0000 0000

This register holds the 32-bit value of MAC 1722 Presentation Time in ns, that should be added to the current presentation time counter value. The initialization happens when TSINIT is set, and the update when the TSUPDT bit is set (TSINIT or TSUPDT bits in *Timestamp control register (ETH_MACTSCR)*).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MPTU[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MPTU[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MPTU[31:0]**: MAC 1722 Presentation Time Update

This field holds the initialization value or the update value for the presentation time. When used for update, this field holds the 32-bit value, in ns, that should be added to the current presentation time counter value. The initialization happens when TSINIT is set, and the update when the TSUPDT bit is set (TSINIT or TSUPDT bits in *Timestamp control register (ETH_MACTSCR)*). When ADDSUB field of *System time nanoseconds update register (ETH_MACSTNUR)* is set, this value is directly used for subtraction.

MAC address 0 high register (ETH_MACA0HR)

Address offset: 0x0300

Reset value: 0x8000 FFFF

The MAC address0 high register holds the upper 16 bits of the first 6-byte MAC address of the station. The first DA byte that is received on the GMII interface corresponds to the LS byte (Bits [7:0]) of the MAC Address Low register. For example, if 0x11 22 33 44 55 66 is received (0x11 in lane 0 of the first column) on the GMII as the destination address, then the MacAddress0 Register [47:0] is compared with 0x66 55 44 33 22 11.

If the MAC address registers are configured to be double-synchronized to the GMII clock domains, then the synchronization is triggered only when Bits[31:24] (in little-endian mode) or Bits[7:0] (in big-endian mode) of the MAC Address0 Low Register are written. For proper synchronization updates, the consecutive writes to this Address Low Register should be performed after at least four clock cycles in the destination clock domain.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCS
r															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRHI[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **AE**: Address Enable

This bit is always set to 1.

Bits 30:17 Reserved, must be kept at reset value.

Bit 16 **DCS**: DMA Channel Select

This field contains the binary representation of the DMA Channel number to which an Rx packet whose DA matches the MAC Address0 content is routed.
0: DMA Rx channel 0
1: DMA Rx channel 1

Bits 15:0 **ADDRHI[15:0]**: MAC Address0[47:32]

This field contains the upper 16 bits [47:32] of the first 6-byte MAC address. The MAC uses this field for filtering the received packets and inserting the MAC address in the transmit flow control (Pause) packets.

MAC address x low register (ETH_MACAxLR)

Address offset: 0x0304 + 0x8 * x, (x = 0 to 3)

Reset value: 0xFFFF FFFF

The MAC address x low register holds the lower 32 bits of the 6-byte first MAC address of the station.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDRLO[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRLO[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ADDRLO[31:0]**: MAC Address x [31:0] (x = 0 to 3)

This field contains the lower 32 bits of the first 6-byte MAC address. The MAC uses this field for filtering the received packets and inserting the MAC address in the transmit flow control (Pause) packets.

MAC address x high register (ETH_MACAxHR)

Address offset: 0x0308 + 0x8 * (x-1), (x = 1 to 3)

Reset value: 0x0000 FFFF

The MAC address x high register holds the upper 16 bits of the second 6-byte MAC address of the station.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AE	SA	MBC[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCS
rw	rw	rw	rw	rw	rw	rw	rw								rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRHI[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **AE**: Address Enable

When this bit is set, the address filter module uses the second MAC address for perfect filtering. When this bit is reset, the address filter module ignores the address for filtering.

Bit 30 **SA**: Source Address

When this bit is set, the MAC Addressx[47:0] is used to compare with the SA fields of the received packet. When this bit is reset, the MAC Address x[47:0] is used to compare with the DA fields of the received packet.

0: DA

1: SA

Bits 29:24 **MBC[5:0]**: Mask Byte Control

These bits are mask control bits for comparing each of the MAC Address bytes. When set high, the MAC does not compare the corresponding byte of received DA or SA with the contents of MAC Address1 registers. Each bit controls the masking of the bytes as follows:

Bit 29: ETH_MACAxHR[15:8]

Bit 28: ETH_MACAxHR[7:0]

Bit 27: ETH_MACAxLR[31:24]

Bit 26: ETH_MACAxLR[23:16]

Bit 25: ETH_MACAxLR[15:8]

Bit 24: ETH_MACAxLR[7:0]

You can filter a group of addresses (known as group address filtering) by masking one or more bytes of the address.

Bits 23:17 Reserved, must be kept at reset value.

Bit 16 **DCS**: DMA Channel Select

This field contains the binary representation of the DMA Channel number to which an Rx packet whose DA matches the MAC Address0 content is routed.
0: DMA Rx channel 0
1: DMA Rx channel 1

Bits 15:0 **ADDRHI[15:0]**: MAC Address1 [47:32]

This field contains the upper 16 bits[47:32] of the second 6-byte MAC address.

MMC control register (ETH_MMC_CONTROL)

Address offset: 0x0700

Reset value: 0x0000 0000

This register configures the MMC operating mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCDBC	Res.	Res.	CNTPRSTLVL	CNTPRST	CNTFREEZ	RSTONRD	CNTSTOPRO	CNTRST
							rw			rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **UCDBC**: Update MMC Counters for Dropped Broadcast Packets

The CNTRST bit has a higher priority than the CNTPRST bit. Therefore, when the software tries to set both bits in the same write cycle, all counters are cleared and the CNTPRST bit is not set.

When set, the MAC updates all related MMC Counters for Broadcast packets that are dropped because of the setting of the DBF bit of *Packet filtering control register (ETH_MACPFR)*.

When reset, the MMC Counters are not updated for dropped Broadcast packets.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **CNTPRSTLVL**: Full-Half Preset

When this bit is low and the CNTPRST bit is set, all MMC counters get preset to almost-half value. All octet counters get preset to 0x7FFF F800 (Half 2Kbytes) and all packet-counters get preset to 0x7FFF FFF0 (Half 16).

When this bit is high and the CNTPRST bit is set, all MMC counters get preset to almost-full value. All octet counters get preset to 0xFFFF F800 (Full 2Kbytes) and all packet-counters get preset to 0xFFFF FFF0 (Full 16).

For 16-bit counters, the almost-half preset values are 0x7800 and 0x7FF0 for the respective octet and packet counters. Similarly, the almost-full preset values for the 16-bit counters are 0xF800 and 0xFFF0.

Bit 4 **CNTPRST**: Counters Preset

When this bit is set, all counters are initialized or preset to almost full or almost half according to the CNTPRSTLVL bit. This bit is cleared automatically after 1 clock cycle.

This bit, along with the CNTPRSTLVL bit, is useful for debugging and testing the assertion of interrupts because of MMC counter becoming half-full or full.

Bit 3 **CNTFREEZ**: MMC Counter Freeze

When this bit is set, it freezes all MMC counters to their current value.

Until this bit is reset to 0, no MMC counter is updated because of any transmitted or received packet. If any MMC counter is read with the Reset on Read bit set, then that counter is also cleared in this mode.

Bit 2 **RSTONRD**: Reset on Read

When this bit is set, the MMC counters are reset to zero after Read (self-clearing after reset). The counters are cleared when the least significant byte lane (Bits[7:0]) is read.

Bit 1 **CNTSTOPRO**: Counter Stop Rollover
When this bit is set, the counter does not roll over to zero after reaching the maximum value.

Bit 0 **CNTRST**: Counters Reset
When this bit is set, all counters are reset. This bit is cleared automatically after 1 clock cycle.

MMC Rx interrupt register (ETH_MMC_RX_INTERRUPT)

Address offset: 0x0704

Reset value: 0x0000 0000

This register maintains the interrupts generated from all receive statistics counters.

The MMC receive interrupt register maintains the interrupts that are generated when the following occur:

- Receive statistic counters reach half of their maximum values (0x8000 0000 for 32-bit counter and 0x8000 for 16-bit counter).
- Receive statistic counters cross their maximum values (0xFFFF FFFF for 32-bit counter and 0xFFFF for 16-bit counter).

When the CNTSTOPRO is set in *MMC control register (ETH_MMC_CONTROL)*, interrupts are set but the counter remains at all-ones. The MMC receive interrupt register is a 32-bit register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte lane (Bits[7:0]) of the respective counter must be read to clear the interrupt bit.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	RXLPITRCIS	RXLPIUSCIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXUCGPIS	Res.
				rc_r	rc_r									rc_r	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXALGNERPIS	RXCRCERPIS	Res.	Res.	Res.	Res.	Res.
									rc_r	rc_r					

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **RXLPITRCIS**: MMC receive LPI transition counter interrupt status
This bit is set when the *Rx LPI transition counter register (ETH_RX_LPI_TRAN_CNTR)* counter reaches half of the maximum value or the maximum value.

Bit 26 **RXLPIUSCIS**: MMC receive LPI microsecond counter interrupt status
This bit is set when the *Rx LPI microsecond counter register (ETH_RX_LPI_USEC_CNTR)* counter reaches half of the maximum value or the maximum value.

Bits 25:18 Reserved, must be kept at reset value.

Bit 17 **RXUCGPIS**: MMC receive Unicast good packet counter interrupt status
This bit is set when the *Rx unicast packets good register (ETH_RX_UNICAST_PACKETS_GOOD)* counter reaches half of the maximum value or the maximum value.

Bits 16:7 Reserved, must be kept at reset value.

Bit 6 **RXALGNERPIS**: MMC receive alignment error packet counter interrupt status
This bit is set when the *Rx alignment error packets register (ETH_RX_ALIGNMENT_ERROR_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 5 **RXCRCERPIS**: MMC receive CRC error packet counter interrupt status
This bit is set when the *Rx CRC error packets register (ETH_RX_CRC_ERROR_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bits 4:0 Reserved, must be kept at reset value.

MMC Tx interrupt register (ETH_MMC_TX_INTERRUPT)

Address offset: 0x0708

Reset value: 0x0000 0000

This register maintains the interrupts generated from all transmit statistics counters.

The MMC transmit interrupt register maintains the interrupts generated when transmit statistic counters reach half their maximum values (0x8000 0000 for 32-bit counter and 0x8000 for 16 bit counter), and when they cross their maximum values (0xFFFF FFFF for 32-bit counter and 0xFFFF for 16-bit counter).

When CNTSTOPRO is set in *MMC control register (ETH_MMC_CONTROL)*, the interrupts are set but the counter remains at all-ones.

The MMC transmit interrupt register is a 32-bit register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read.

The least significant byte lane (Bits[7:0]) of the respective counter must be read to clear the interrupt bit.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TXLPITRCIS	TXLPIUSCIS	Res.	Res.	Res.	Res.	TXGPKTIS	Res.	Res.	Res.	Res.	Res.
				rc_r	rc_r					rc_r					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXMCOLGPIS	TXSCOLGPIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rc_r	rc_r														

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **TXLPITRCIS**: MMC transmit LPI transition counter interrupt status
This bit is set when the *Tx LPI transition counter register (ETH_TX_LPI_TRAN_CNTR)* counter reaches half of the maximum value or the maximum value.

Bit 26 **TXLPIUSCIS**: MMC transmit LPI microsecond counter interrupt status
This bit is set when the *Tx LPI microsecond timer register (ETH_TX_LPI_USEC_CNTR)* counter reaches half of the maximum value or the maximum value.

Bits 25:22 Reserved, must be kept at reset value.

Bit 21 **TXGPKTIS**: MMC transmit good packet counter interrupt status
This bit is set when the *Tx packet count good register (ETH_TX_PACKET_COUNT_GOOD)* counter reaches half of the maximum value or the maximum value.

Bits 20:16 Reserved, must be kept at reset value.

Bit 15 **TXMCOLGPIS**: MMC transmit multiple collision good packet counter interrupt status
This bit is set when the *Tx multiple collision good packets register (ETH_TX_MULTIPLE_COLLISION_GOOD_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 14 **TXSCOLGPIS**: MMC transmit single collision good packet counter interrupt status
This bit is set when the *Tx single collision good packets register (ETH_TX_SINGLE_COLLISION_GOOD_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bits 13:0 Reserved, must be kept at reset value.

MMC Rx interrupt mask register (ETH_MMC_RX_INTERRUPT_MASK)

Address offset: 0x070C

Reset value: 0x0000 0000

The MMC receive interrupt mask register maintains the masks for the interrupts generated when receive statistic counters reach half of their maximum value or the maximum values.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	RXLPITRCIM	RXLPIUSCIM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXUCGPIM	Res.
				rw	rw									rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXALGNERPIM	RXCRCERPIM	Res.	Res.	Res.	Res.	Res.
									rw	rw					

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **RXLPITRCIM**: MMC receive LPI transition counter interrupt Mask
Setting this bit masks the interrupt when the *Rx LPI transition counter register (ETH_RX_LPI_TRAN_CNTR)* counter reaches half of the maximum value or the maximum value.

Bit 26 **RXLPIUSCIM**: MMC receive LPI microsecond counter interrupt Mask
Setting this bit masks the interrupt when the *Rx LPI microsecond counter register (ETH_RX_LPI_USEC_CNTR)* counter reaches half of the maximum value or the maximum value.

Bits 25:18 Reserved, must be kept at reset value.

Bit 17 **RXUCGPIM**: MMC receive Unicast good packet counter interrupt mask
Setting this bit masks the interrupt when the *Rx unicast packets good register (ETH_RX_UNICAST_PACKETS_GOOD)* counter reaches half of the maximum value or the maximum value.

Bits 16:7 Reserved, must be kept at reset value.

Bit 6 **RXALGNERPIM**: MMC receive Alignment Error packet counter interrupt mask
Setting this bit masks the interrupt when the *Rx alignment error packets register (ETH_RX_ALIGNMENT_ERROR_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 5 **RXCRCERPIM**: MMC receive CRC Error packet counter interrupt mask
Setting this bit masks the interrupt when the *Rx CRC error packets register (ETH_RX_CRC_ERROR_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bits 4:0 Reserved, must be kept at reset value.

MMC Tx interrupt mask register (ETH_MMC_TX_INTERRUPT_MASK)

Address offset: 0x0710

Reset value: 0x0000 0000

The MMC transmit interrupt mask register maintains the masks for the interrupts generated when the transmit statistic counters reach half of their maximum value or the maximum values. This register is 32-bit wide.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	TXLPITRCIM	TXLPIUSCIM	Res.	Res.	Res.	Res.	TXGPKTIM	Res.	Res.	Res.	Res.	Res.
				rw	rw					rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXMCOLGPIM	TXSCOLGPIM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw														

Bits 31:28 Reserved, must be kept at reset value.

Bit 27 **TXLPITRCIM**: MMC transmit LPI transition counter interrupt mask
Setting this bit masks the interrupt when the *Tx LPI transition counter register (ETH_TX_LPI_TRAN_CNTR)* counter reaches half of the maximum value or the maximum value.

Bit 26 **TXLPIUSCIM**: MMC transmit LPI microsecond counter interrupt mask
Setting this bit masks the interrupt when the *Tx LPI microsecond timer register (ETH_TX_LPI_USEC_CNTR)* counter reaches half of the maximum value or the maximum value.

Bits 25:22 Reserved, must be kept at reset value.

Bit 21 **TXGPKTIM**: MMC transmit good packet counter interrupt mask
Setting this bit masks the interrupt when the *Tx packet count good register (ETH_TX_PACKET_COUNT_GOOD)* counter reaches half of the maximum value or the maximum value.

Bits 20:16 Reserved, must be kept at reset value.

Bit 15 **TXMCOLGPIM**: MMC transmit multiple-collision good packet counter interrupt mask
Setting this bit masks the interrupt when the *Tx multiple collision good packets register (ETH_TX_MULTIPLE_COLLISION_GOOD_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bit 14 **TXSCOLGPIM**: MMC transmit single-collision good packet counter interrupt mask
Setting this bit masks the interrupt when the *Tx single collision good packets register (ETH_TX_SINGLE_COLLISION_GOOD_PACKETS)* counter reaches half of the maximum value or the maximum value.

Bits 13:0 Reserved, must be kept at reset value.

Tx single collision good packets register (ETH_TX_SINGLE_COLLISION_GOOD_PACKETS)

Address offset: 0x074C

Reset value: 0x0000 0000

This register provides the number of successfully transmitted packets by Ethernet peripheral after a single collision in the Half-duplex mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXSNGLCOLG[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXSNGLCOLG[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXSNGLCOLG[31:0]**: Tx Single Collision Good Packets
This field indicates the number of successfully transmitted packets after a single collision in the Half-duplex mode.

Tx multiple collision good packets register (ETH_TX_MULTIPLE_COLLISION_GOOD_PACKETS)

Address offset: 0x0750

Reset value: 0x0000 0000

This register provides the number of successfully transmitted packets by Ethernet peripheral after multiple collisions in the Half-duplex mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXMULTCOLG[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXMULTCOLG[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXMULTCOLG[31:0]**: Tx Multiple Collision Good Packets
This field indicates the number of successfully transmitted packets after multiple collisions in the Half-duplex mode.

Tx packet count good register (ETH_TX_PACKET_COUNT_GOOD)

Address offset: 0x0768

Reset value: 0x0000 0000

This register provides the number of good packets transmitted by Ethernet peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXPKTG[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXPKTG[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXPKTG[31:0]**: Tx Packet Count Good
This field indicates the number of good packets transmitted.

Rx CRC error packets register (ETH_RX_CRC_ERROR_PACKETS)

Address offset: 0x0794

Reset value: 0x0000 0000

This register provides the number of packets received by Ethernet peripheral with CRC error.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXCRCERR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXCRCERR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXCRCERR[31:0]**: Rx CRC Error Packets
This field indicates the number of packets received with CRC error.

Rx alignment error packets register (ETH_RX_ALIGNMENT_ERROR_PACKETS)

Address offset: 0x0798

Reset value: 0x0000 0000

This register provides the number of packets received by Ethernet peripheral with alignment (dribble) error. It is valid only in 10/100 mode.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXALGNERR[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXALGNERR[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXALGNERR[31:0]**: Rx Alignment Error Packets
This field indicates the number of packets received with alignment (dribble) error. It is valid only in 10/100 mode.

Rx unicast packets good register (ETH_RX_UNICAST_PACKETS_GOOD)

Address offset: 0x07C4

Reset value: 0x0000 0000

This register provides the number of good unicast packets received by Ethernet peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXUCASTG[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXUCASTG[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXUCASTG[31:0]**: Rx Unicast Packets Good
This field indicates the number of good unicast packets received.

Tx LPI microsecond timer register (ETH_TX_LPI_USEC_CNTR)

Address offset: 0x07EC

Reset value: 0x0000 0000

This register provides the number of microseconds Tx LPI is asserted by Ethernet peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXLPIUSC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXLPIUSC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXLPIUSC[31:0]**: Tx LPI Microseconds Counter

This field indicates the number of microseconds Tx LPI is asserted. For every Tx LPI Entry and Exit, the Timer value can have an error of +/- 1 microsecond.

Tx LPI transition counter register (ETH_TX_LPI_TRAN_CNTR)

Address offset: 0x07F0

Reset value: 0x0000 0000

This register provides the number of times Ethernet peripheral has entered Tx LPI.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXLPITRC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXLPITRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXLPITRC[31:0]**: Tx LPI Transition counter

This field indicates the number of times Tx LPI Entry has occurred. Even if Tx LPI Entry occurs in Automate mode (because of LPITXA bit set in the *LPI control and status register (ETH_MACLCSR)*), the counter is incremented.

Rx LPI microsecond counter register (ETH_RX_LPI_USEC_CNTR)

Address offset: 0x07F4

Reset value: 0x0000 0000

This register provides the number of microseconds Rx LPI is sampled by Ethernet peripheral.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXLPIUSC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXLPIUSC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXLPIUSC[31:0]**: Rx LPI Microseconds Counter

This field indicates the number of microseconds Rx LPI is asserted. For every Rx LPI Entry and Exit, the Timer value can have an error of +/- 1 microsecond.

Rx LPI transition counter register (ETH_RX_LPI_TRAN_CNTR)

Address offset: 0x07F8

Reset value: 0x0000 0000

This register provides the number of times Ethernet peripheral has entered Rx LPI.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RXLPITRC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXLPITRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RXLPITRC[31:0]**: Rx LPI Transition counter

This field indicates the number of times Rx LPI Entry has occurred.

MMC FPE Tx interrupt status register (ETH_MMC_FPE_TX_ISR)

Address offset: 0x08A0

Reset value: 0x0000 0000

This register maintains the interrupts generated from all FPE related transmit statistics counters when transmit statistic counters reach half their maximum values (0x8000 0000 for 32-bit counter and 0x8000 for 16-bit counter), and when they cross their maximum values (0xFFFF FFFF for 32-bit counter and 0xFFFF for 16-bit counter). When CNTSTOPRO bit is set in *MMC control register (ETH_MMC_CONTROL)*, the interrupts are set but the counter remains at all-ones. The MMC FPE transmit interrupt register is a 32-bit register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read. The least significant byte lane (Bits[7:0]) of the respective counter must be read to clear the interrupt bit.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HRCIS	FCIS
														rc_r	rc_r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **HRCIS**: MMC Tx Hold Request Counter interrupt status
This bit is set when the *MMC Tx hold request counter register (ETH_MMC_TX_HRCR)* reaches half of the maximum value or the maximum value.

Bit 0 **FCIS**: MMC Tx FPE Fragment Counter Interrupt status
This bit is set when the *MMC FPE Tx fragment counter register (ETH_MMC_FPE_TX_FCR)* reaches half of the maximum value or the maximum value.

MMC FPE Tx interrupt mask register (ETH_MMC_FPE_TX_IMR)

Address offset: 0x08A4

Reset value: 0x0000 0000

This register maintains the masks for interrupts generated from all FPE related transmit statistics counters when FPE related receive statistic counters reach half of their maximum value or the maximum values.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HRCIM	FCIM
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **HRCIM**: MMC transmit hold request counter interrupt mask

Setting this bit masks the interrupt when the *MMC Tx hold request counter register (ETH_MMC_TX_HRCR)* reaches half of the maximum value or the maximum value.

Bit 0 **FCIM**: MMC transmit fragment counter interrupt mask

Setting this bit masks the interrupt when the *MMC FPE Tx fragment counter register (ETH_MMC_FPE_TX_FCR)* reaches half of the maximum value or the maximum value.

MMC FPE Tx fragment counter register (ETH_MMC_FPE_TX_FCR)

Address offset: 0x08A8

Reset value: 0x0000 0000

This register provides the number of additional mPackets transmitted due to preemption.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXFFC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXFFC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXFFC[31:0]**: Tx FPE Fragment counter

This field indicates the number of additional mPackets that has been transmitted due to preemption

MMC Tx hold request counter register (ETH_MMC_TX_HRCR)

Address offset: 0x08AC

Reset value: 0x0000 0000

This register provides the count of number of times a hold request is issue to the MAC.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXHRC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXHRC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXHRC[31:0]**: Tx hold request counter

This field indicates count of number of a hold requests issued to the MAC.

MMC FPE Rx interrupt status register (ETH_MMC_FPE_RX_ISR)

Address offset: 0x08C0

Reset value: 0x0000 0000

This register maintains the interrupts generated from all FPE related receive statistics counters when transmit statistic counters reach half their maximum values (0x8000 0000 for 32-bit counter and 0x8000 for 16-bit counter), and when they cross their maximum values (0xFFFF FFFF for 32-bit counter and 0xFFFF for 16-bit counter). When CNTSTOPRO bit is set in *MMC control register (ETH_MMC_CONTROL)*, the interrupts are set but the counter remains at all-ones. The MMC FPE receive interrupt register is a 32-bit register. An interrupt bit is cleared when the respective MMC counter that caused the interrupt is read.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCIS	PAOCIS	PSECIS	PAECIS
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **FCIS**: MMC Rx FPE Fragment Counter interrupt status

This bit is set when the *MMC Rx FPE fragments counter register (ETH_RX_FPE_FRAG_CR)* counter reaches half of the maximum value or the maximum value.

Bit 2 **PAOCIS**: MMC Rx Packet Assembly OK counter interrupt status

This bit is set when the *MMC Rx packet assembly OK register (ETH_RX_PACKET_ASM_OKR)* counter reaches half of the maximum value or the maximum value.

Bit 1 **PSECIS**: MMC Rx Packet SMD Error counter interrupt status

This bit is set when the *MMC Rx packet SMD error register (ETH_RX_PACKET_SMD_ERR)* counter reaches half of the maximum value or the maximum value.

Bit 0 **PAECIS**: MMC Rx Packet Assembly Error counter interrupt status
This bit is set when the *MMC Rx packet assembly error register (ETH_RX_PACKET_ASM_ERR)* counter reaches half of the maximum value or the maximum value.

MMC FPE Rx interrupt mask register (ETH_MMC_FPE_RX_IMR)

Address offset: 0x08C4

Reset value: 0x0000 0000

This register maintains the masks for interrupts generated from all FPE related receive statistics counters when FPE related receive statistic counters reach half of their maximum value or the maximum values.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCIM	PAOCIM	PSECIM	PAECIM
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **FCIM**: MMC Rx FPE Fragment Counter interrupt mask
Setting this bit masks the interrupt when the *MMC Rx FPE fragments counter register (ETH_RX_FPE_FRAG_CR)* counter reaches half of the maximum value or the maximum value.

Bit 2 **PAOCIM**: MMC Rx packet Assembly OK Counter interrupt mask
Setting this bit masks the interrupt when the *MMC Rx packet assembly OK register (ETH_RX_PACKET_ASM_OKR)* counter reaches half of the maximum value or the maximum value.

Bit 1 **PSECIM**: MMC Rx Packet SMD Error Counter interrupt mask
Setting this bit masks the interrupt when the *MMC Rx packet SMD error register (ETH_RX_PACKET_SMD_ERR)* counter reaches half of the maximum value or the maximum value.

Bit 0 **PAECIM**: MMC Rx Packet Assembly Error Counter interrupt mask
Setting this bit masks the interrupt when the *MMC Rx packet assembly error register (ETH_RX_PACKET_ASM_ERR)* counter reaches half of the maximum value or the maximum value.

MMC Rx packet assembly error register (ETH_RX_PACKET_ASM_ERR)

Address offset: 0x08C8

Reset value: 0x0000 0000

This register provides the number of MAC frames with reassembly errors on the receiver, due to mismatch in the Fragment Count value.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PAEC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PAEC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PAEC[31:0]**: Rx Packet Assembly Error Counter

This field indicates the number of MAC frames with reassembly errors on the receiver, due to mismatch in the Fragment Count value.

MMC Rx packet SMD error register (ETH_RX_PACKET_SMD_ERR)

Address offset: 0x08CC

Reset value: 0x0000 0000

This register provides the number of received MAC frames rejected due to unknown SMD value and MAC frame fragments rejected due to arriving with an SMD-C when there was no preceding preempted frame.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PSEC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PSEC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PSEC[31:0]**: Rx Packet SMD Error Counter

This field indicates the number of MAC frames rejected due to unknown SMD value and MAC frame fragments rejected due to arriving with an SMD-C when there was no preceding preempted frame

MMC Rx packet assembly OK register (ETH_RX_PACKET_ASM_OKR)

Address offset: 0x08D0

Reset value: 0x0000 0000

This register provides the number of MAC frames that were successfully reassembled and delivered to MAC.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PAOC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PAOC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PAOC[31:0]**: Rx Packet Assembly OK Counter

This field indicates the number of MAC frames that were successfully reassembled and delivered to MAC.

MMC Rx FPE fragments counter register (ETH_RX_FPE_FRAG_CR)

Address offset: 0x08D4

Reset value: 0x0000 0000

This register provides the number of additional mPackets received due to preemption.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FFC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FFC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **FFC[31:0]**: Rx FPE Fragment Counter

This field indicates the number of additional mPackets received due to preemption

L3 and L4 control 0 register (ETH_MACL3L4C0R)

Address offset: 0x0900

Reset value: 0x0000 0000

The Layer 3 and Layer 4 Control register controls the operations of filter 0 of Layer 3 and Layer 4.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	DMCHEN0	Res.	Res.	Res.	DMCHN0	Res.	Res.	L4DPIM0	L4DPM0	L4SPIM0	L4SPM0	Res.	L4PEN0
			rw				rw			rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3HDBM0[4:0]					L3HSBM0[4:0]					L3DAIM0	L3DAM0	L3SAIM0	L3SAM0	Res.	L3PEN0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **DMCHEN0**: DMA Channel Select Enable

When set, this bit enables the selection of the DMA channel number for the packet that is passed by this L3_L4 filter. The DMA channel is indicated by the DMCHN bits. When this bit is reset, the DMA channel is not decided by this filter.

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **DMCHN0**: DMA Channel Number

When DMCHEN is set high, this field selects the DMA Channel number to which the packet passed by this filter is routed. The width of this field depends on the number of the DMA channels present in your configuration.

0: DMA channel 0

1: DMA channel 1

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **L4DPIM0**: Layer 4 Destination Port Inverse Match Enable

When this bit is set, the Layer 4 Destination Port number field is enabled for inverse matching. When this bit is reset, the Layer 4 Destination Port number field is enabled for perfect matching.

This bit is valid and applicable only when the L4DPM0 bit is set high.

Bit 20 **L4DPM0**: Layer 4 Destination Port Match Enable

When this bit is set, the Layer 4 Destination Port number field is enabled for matching. When this bit is reset, the MAC ignores the Layer 4 Destination Port number field for matching.

Bit 19 **L4SPIM0**: Layer 4 Source Port Inverse Match Enable

When this bit is set, the Layer 4 Source Port number field is enabled for inverse matching. When this bit is reset, the Layer 4 Source Port number field is enabled for perfect matching.

This bit is valid and applicable only when the L4SPM0 bit is set high.

Bit 18 **L4SPM0**: Layer 4 Source Port Match Enable

When this bit is set, the Layer 4 Source Port number field is enabled for matching. When this bit is reset, the MAC ignores the Layer 4 Source Port number field for matching.

Bit 17 Reserved, must be kept at reset value.

Bit 16 **L4PEN0**: Layer 4 Protocol Enable

When this bit is set, the Source and Destination Port number fields of UDP packets are used for matching. When this bit is reset, the Source and Destination Port number fields of TCP packets are used for matching.

The Layer 4 matching is done only when the L4SPM0 or L4DPM0 bit is set.

Bits 15:11 **L3HDBM0[4:0]**: Layer 3 IP DA higher bits match

Condition: IPv4 packets

This field contains the number of higher bits of IP Destination Address that are masked in the IPv4 packets:

0: No bits are masked.

1: LSb[0] is masked

2: Two LSbs [1:0] are masked

..

31: All bits except MSb are masked.

Condition: IPv6 packets

Bits[12:11] of this field correspond to Bits[6:5] of L3HSBM0 which indicate the number of lower bits of IP Source or Destination Address that are masked in the IPv6 packets. Number of bits masked is given by concatenated values of the L3HDBM0[1:0] and L3HSBM0 bits:

0: No bits are masked.

1: LSb[0] is masked

2: Two LSbs [1:0] are masked

..

31: All bits except MSb are masked.

This field is valid and applicable only when the L3DAM0 or L3SAM0 bit is set.

Bits 10:6 **L3HSBM0[4:0]**: Layer 3 IP SA higher bits match

Condition: IPv4 packets

This field contains the number of lower bits of IP source address that are masked for matching in the IPv4 packets. The following list describes the values of this field:

0: No bits are masked.

1: LSb[0] is masked

2: Two LSbs [1:0] are masked

..

31: All bits except MSb are masked.

Condition: IPv6 packets:

This field contains Bits[4:0] of L3HSBM0. These bits indicate the number of higher bits of IP source or destination address matched in the IPv6 packets. This field is valid and applicable only when the L3DAM0 or L3SAM0 bit is set high.

Bit 5 **L3DAIM0**: Layer 3 IP DA Inverse Match Enable

When this bit is set, the Layer 3 IP Destination Address field is enabled for inverse matching. When this bit is reset, the Layer 3 IP Destination Address field is enabled for perfect matching.

This bit is valid and applicable only when the L3DAM0 bit is set high.

Bit 4 **L3DAM0**: Layer 3 IP DA Match Enable

When this bit is set, the Layer 3 IP Destination Address field is enabled for matching. When this bit is reset, the MAC ignores the Layer 3 IP Destination Address field for matching.

Note: When the L3PEN0 bit is set, you should set either this bit or the L3SAM0 bit because either IPv6 DA or SA can be checked for filtering.

Bit 3 **L3SAIM0**: Layer 3 IP SA Inverse Match Enable

When this bit is set, the Layer 3 IP Source Address field is enabled for inverse matching. When this bit reset, the Layer 3 IP Source Address field is enabled for perfect matching.

This bit is valid and applicable only when the L3SAM0 bit is set.

Bit 2 **L3SAM0**: Layer 3 IP SA Match Enable

When this bit is set, the Layer 3 IP Source Address field is enabled for matching. When this bit is reset, the MAC ignores the Layer 3 IP Source Address field for matching.

Note: When the L3PEN0 bit is set, you should set either this bit or the L3DAM0 bit because either IPv6 SA or DA can be checked for filtering.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **L3PEN0**: Layer 3 Protocol Enable

When this bit is set, the Layer 3 IP Source or Destination Address matching is enabled for IPv6 packets. When this bit is reset, the Layer 3 IP Source or Destination Address matching is enabled for IPv4 packets.

The Layer 3 matching is done only when the L3SAM0 or L3DAM0 bit is set.

Layer4 address filter 0 register (ETH_MACL4A0R)

Address offset: 0x0904

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L4DP0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L4SP0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **L4DP0[15:0]**: Layer 4 Destination Port Number Field

When the L4PEN0 bit is reset and the L4DPM0 bit is set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with the TCP Destination Port Number field in the IPv4 or IPv6 packets.

When the L4PEN0 and L4DPM0 bits are set in *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with the UDP Destination Port Number field in the IPv4 or IPv6 packets.

Bits 15:0 **L4SP0[15:0]**: Layer 4 Source Port Number Field

When the L4PEN0 bit is reset and the L4DPM0 bit is set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with the TCP Source Port Number field in the IPv4 or IPv6 packets.

When the L4PEN0 and L4DPM0 bits are set in *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with the UDP Source Port Number field in the IPv4 or IPv6 packets.

Layer3 address 0 filter 0 register (ETH_MACL3A00R)

Address offset: 0x0910

Reset value: 0x0000 0000

For IPv4 packets, the Layer 3 Address 0 filter 0 register contains the 32-bit IP Source Address field. For IPv6 packets, it contains Bits[31:0] of the 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A00[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3A00[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A00[31:0]**: Layer 3 Address 0 Field

When the L3PEN0 and L3SAM0 bits are set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with Bits[31:0] of the IP Source Address field in the IPv6 packets.

When the L3PEN0 and L3DAM0 bits are set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with Bits[31:0] of the IP Destination Address field in the IPv6 packets.

When the L3PEN0 bit is reset and the L3SAM0 bit is set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with the IP Source Address field in the IPv4 packets.

Layer3 address 1 filter 0 register (ETH_MACL3A10R)

Address offset: 0x0914

Reset value: 0x0000 0000

For IPv4 packets, the Layer 3 Address 1 filter 0 register contains the 32-bit IP Destination Address field. For IPv6 packets, it contains Bits[63:32] of the 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A10[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3A10[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A10[31:0]**: Layer 3 Address 1 Field

When the L3PEN0 and L3SAM0 bits are set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with Bits[63:32] of the IP Source Address field in the IPv6 packets.

When the L3PEN0 and L3DAM0 bits are set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with Bits[63:32] of the IP Destination Address field in the IPv6 packets.

When the L3PEN0 bit is reset and the L3SAM0 bit is set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with the IP Destination Address field in the IPv4 packets.

Layer3 address 2 filter 0 register (ETH_MACL3A20R)

Address offset: 0x0918

Reset value: 0x0000 0000

The Layer 3 Address 2 filter 0 register is reserved for IPv4 packets. For IPv6 packets, it contains Bits[95:64] of 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A20[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3A20[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A20[31:0]**: Layer 3 Address 2 Field

When the L3PEN0 and L3SAM0 bits are set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with Bits[95:64] of the IP Source Address field in the IPv6 packets.

When the L3PEN0 and L3DAM0 bits are set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with Bits[95:64] of the IP Destination Address field in the IPv6 packets.

When the L3PEN0 bit is reset in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field is not used.

Layer3 address 3 filter 0 register (ETH_MACL3A30R)

Address offset: 0x091C

Reset value: 0x0000 0000

The Layer 3 Address 3 filter 0 register is reserved for IPv4 packets. For IPv6 packets, it contains Bits[127:96] of 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A30[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3A30[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A30[31:0]**: Layer 3 Address 3 Field

When the L3PEN0 and L3SAM0 bits are set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with Bits[127:96] of the IP Source Address field in the IPv6 packets.

When the L3PEN0 and L3DAM0 bits are set in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field contains the value to be matched with Bits[127:96] of the IP Destination Address field in the IPv6 packets.

When the L3PEN0 bit is reset in the *L3 and L4 control 0 register (ETH_MACL3L4C0R)*, this field is not used.

L3 and L4 control 1 register (ETH_MACL3L4C1R)

Address offset: 0x0930

Reset value: 0x0000 0000

The Layer 3 and Layer 4 Control register controls the operations of filter 1 of Layer 3 and Layer 4.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	DMCHEN1	Res.	Res.	Res.	DMCHN1	Res.	Res.	L4DPIM1	L4DPM1	L4SPIM1	L4SPM1	Res.	L4PEN1
			rw				rw			rw	rw	rw	rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3HDBM1[4:0]					L3HSBM1[4:0]					L3DAIM1	L3DAM1	L3SAIM1	L3SAM1	Res.	L3PEN1
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **DMCHEN1**: DMA Channel Select Enable

When set, this bit enables the selection of the DMA channel number for the packet that is passed by this L3_L4 filter. The DMA channel is indicated by the DMCHN bits. When this bit is reset, the DMA channel is not decided by this filter.

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **DMCHN1**: DMA Channel Number
When DMCHEN is set high, this field selects the DMA Channel number to which the packet passed by this filter is routed. The width of this field depends on the number of the DMA channels present in your configuration.

Bits 23:22 Reserved, must be kept at reset value.

Bit 21 **L4DPIM1**: Layer 4 Destination Port Inverse Match Enable
When this bit is set, the Layer 4 Destination Port number field is enabled for inverse matching. When this bit is reset, the Layer 4 Destination Port number field is enabled for perfect matching.
This bit is valid and applicable only when the L4DPM1 bit is set high.

Bit 20 **L4DPM1**: Layer 4 Destination Port Match Enable
When this bit is set, the Layer 4 Destination Port number field is enabled for matching. When this bit is reset, the MAC ignores the Layer 4 Destination Port number field for matching.

Bit 19 **L4SPIM1**: Layer 4 Source Port Inverse Match Enable
When this bit is set, the Layer 4 Source Port number field is enabled for inverse matching. When this bit is reset, the Layer 4 Source Port number field is enabled for perfect matching.
This bit is valid and applicable only when the L4SPM1 bit is set high.

Bit 18 **L4SPM1**: Layer 4 Source Port Match Enable
When this bit is set, the Layer 4 Source Port number field is enabled for matching. When this bit is reset, the MAC ignores the Layer 4 Source Port number field for matching.

Bit 17 Reserved, must be kept at reset value.

Bit 16 **L4PEN1**: Layer 4 Protocol Enable
When this bit is set, the Source and Destination Port number fields of UDP packets are used for matching. When this bit is reset, the Source and Destination Port number fields of TCP packets are used for matching.
The Layer 4 matching is done only when the L4SPM1 or L4DPM1 bit is set.

Bits 15:11 **L3HDBM1[4:0]**: Layer 3 IP DA higher bits match
Condition: IPv4 packets
This field contains the number of lower bits of IP Destination Address that are masked for matching in the IPv4 packets. The following list describes the values of this field:
0: No bits are masked.
1: LSb[0] is masked
2: Two LSbs [1:0] are masked
..
31: All bits except MSb are masked.
Condition: IPv6 packets
Bits[12:11] of this field correspond to Bits[6:5] of L3HSBM1, which indicate the number of lower bits of IP Source or Destination Address that are masked in the IPv6 packets. The following list describes the concatenated values of the L3HDBM1[1:0] and L3HSBM1 bits:
0: No bits are masked
1: LSb[0] is masked
2: Two LSbs [1:0] are masked
..
127: All bits except MSb are masked
This field is valid and applicable only when the L3DAM1 or L3SAM1 bit is set.

Bits 10:6 **L3HSBM1[4:0]**: Layer 3 IP SA Higher Bits Match

Condition: IPv4 packets

This field contains the number of lower bits of IP Source Address that are masked for matching in the IPv4 packets. The following list describes the values of this field:

0: No bits are masked.

1: LSb[0] is masked

2: Two LSbs [1:0] are masked

..

31: All bits except MSb are masked.

Condition: IPv6 packets

This field contains Bits[4:0] of L3HSBM1. These bits indicate the number of higher bits of IP Source or Destination Address matched in the IPv6 packets. This field is valid and applicable only when the L3DAM1 or L3SAM1 bit is set high.

Bit 5 **L3DAIM1**: Layer 3 IP DA Inverse Match Enable

When this bit is set, the Layer 3 IP Destination Address field is enabled for inverse matching. When this bit is reset, the Layer 3 IP Destination Address field is enabled for perfect matching.

This bit is valid and applicable only when the L3DAM1 bit is set high.

Bit 4 **L3DAM1**: Layer 3 IP DA Match Enable

When this bit is set, the Layer 3 IP Destination Address field is enabled for matching. When this bit is reset, the MAC ignores the Layer 3 IP Destination Address field for matching.

Note: When the L3PEN1 bit is set, you should set either this bit or the L3SAM1 bit because either IPv6 DA or SA can be checked for filtering.

Bit 3 **L3SAIM1**: Layer 3 IP SA Inverse Match Enable

When this bit is set, the Layer 3 IP Source Address field is enabled for inverse matching. When this bit reset, the Layer 3 IP Source Address field is enabled for perfect matching.

This bit is valid and applicable only when the L3SAM1 bit is set.

Bit 2 **L3SAM1**: Layer 3 IP SA Match Enable

When this bit is set, the Layer 3 IP Source Address field is enabled for matching. When this bit is reset, the MAC ignores the Layer 3 IP Source Address field for matching.

Note: When the L3PEN01 bit is set, you should set either this bit or the L3DAM1 bit because either IPv6 SA or DA can be checked for filtering.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **L3PEN1**: Layer 3 Protocol Enable

When this bit is set, the Layer 3 IP Source or Destination Address matching is enabled for IPv6 packets. When this bit is reset, the Layer 3 IP Source or Destination Address matching is enabled for IPv4 packets.

The Layer 3 matching is done only when the L3SAM1 or L3DAM1 bit is set.

Layer 4 address filter 1 register (ETH_MACL4A1R)

Address offset: 0x0934

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L4DP1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L4SP1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 **L4DP1[15:0]**: Layer 4 Destination Port Number Field

When the L4PEN1 bit is reset and the L4DPM1 bit is set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with the TCP Destination Port Number field in the IPv4 or IPv6 packets.

When the L4PEN1 and L4DPM1 bits are set in *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with the UDP Destination Port Number field in the IPv4 or IPv6 packets.

Bits 15:0 **L4SP1[15:0]**: Layer 4 Source Port Number Field

When the L4PEN1 bit is reset and the L4DPM1 bit is set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with the TCP Source Port Number field in the IPv4 or IPv6 packets.

When the L4PEN1 and L4DPM1 bits are set in *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with the UDP Source Port Number field in the IPv4 or IPv6 packets.

Layer3 address 0 filter 1 Register (ETH_MACL3A01R)

Address offset: 0x0940

Reset value: 0x0000 0000

For IPv4 packets, the Layer 3 Address 0 filter 1 register contains the 32-bit IP Source Address field. For IPv6 packets, it contains Bits[31:0] of the 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A01[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3A01[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A01[31:0]**: Layer 3 Address 0 Field

When the L3PEN1 and L3SAM1bits are set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with Bits[31:0] of the IP Source Address field in the IPv6 packets.

When the L3PEN1 and L3DAM1 bits are set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with Bits[31:0] of the IP Destination Address field in the IPv6 packets.

When the L3PEN1 bit is reset and the L3SAM1 bit is set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with the IP Source Address field in the IPv4 packets.

Layer3 address 1 filter 1 register (ETH_MACL3A11R)

Address offset: 0x0944

Reset value: 0x0000 0000

For IPv4 packets, the Layer 3 Address 1 filter 1 register contains the 32-bit IP Destination Address field. For IPv6 packets, it contains Bits[63:32] of the 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A11[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3A11[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A11[31:0]**: Layer 3 Address 1 Field

When the L3PEN1 and L3SAM1 bits are set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with Bits[63:32] of the IP Source Address field in the IPv6 packets.

When the L3PEN1 and L3DAM1 bits are set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with Bits[63:32] of the IP Destination Address field in the IPv6 packets.

When the L3PEN1 bit is reset and the L3SAM1 bit is set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with the IP Destination Address field in the IPv4 packets.

Layer3 address 2 filter 1 Register (ETH_MACL3A21R)

Address offset: 0x0948

Reset value: 0x0000 0000

The Layer 3 Address 2 filter 1 register is reserved for IPv4 packets. For IPv6 packets, it contains Bits[95:64] of 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A21[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3A21[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A21[31:0]**: Layer 3 Address 2 Field

When the L3PEN1 and L3SAM1 bits are set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with Bits[95:64] of the IP Source Address field in the IPv6 packets.

When the L3PEN1 and L3DAM1 bits are set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with Bits[95:64] of the IP Destination Address field in the IPv6 packets.

When the L3PEN1 bit is reset in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field is not used.

Layer3 address 3 filter 1 register (ETH_MACL3A31R)

Address offset: 0x94C

Reset value: 0x0000 0000

The Layer 3 Address 3 filter 1 register is reserved for IPv4 packets. For IPv6 packets, it contains Bits[127:96] of 128-bit IP Source Address or Destination Address field.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
L3A31[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3A31[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **L3A31[31:0]**: Layer 3 Address 3 Field

When the L3PEN1 and L3SAM1 bits are set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with Bits[127:96] of the IP Source Address field in the IPv6 packets.

When the L3PEN1 and L3DAM1 bits are set in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field contains the value to be matched with Bits[127:96] of the IP Destination Address field in the IPv6 packets.

When the L3PEN1 bit is reset in the *L3 and L4 control 1 register (ETH_MACL3L4C1R)*, this field is not used.

MAC indirect access control register (ETH_MAC_IACR)

Address offset: 0x0A70

Reset value: 0x0000 0000

This register provides the Indirect Access control and status for the *MAC type-based Rx queue mapping register (ETH_MAC_TMRQR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.			MSEL[3:0]	
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				AOFF[7:0]				Res.	Res.	AUTO	Res.	Res.	Res.	COM	OB
rw	rw	rw	rw	rw	rw	rw	rw			rw				rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **MSEL[3:0]**: Mode Select

This field is used for indirect accesses to the *MAC type-based Rx queue mapping register (ETH_MAC_TMRQR)*. It must be set when initiating a read/write to the *MAC type-based Rx queue mapping register (ETH_MAC_TMRQR)* and should not be changed until the OB bit is reset.
0000:Typ_RXQ_ (Type-based RXQ mapping)
0001-1111: Reserved for future use

Bits 15:8 **AOFF[7:0]**: Address Offset

This field is used for indirect accesses to the *MAC type-based Rx queue mapping register (ETH_MAC_TMRQR)*. It must be set when initiating a read/write to the *MAC type-based Rx queue mapping register (ETH_MAC_TMRQR)* and should not be changed until the OB bit is reset.
00000000: IndReg0 (Indirect register 0)
00000001: IndReg1 (Indirect register 1)
...
00000111: IndReg7 (Indirect register 7)
00001000-11111111: Reserved, must not be used

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **AUTO**: Auto-increment

1: AOFF is incremented by 1. The software should ensure not to cause a wrap condition. Byte-wise read/write is not supported when auto-increment is enabled.
0: AOFF is not automatically incremented. The software should program the correct address offset for each access.

Bits 4:2 Reserved, must be kept at reset value.

Bit 1 **COM**: Command type

This bit indicates the register access type:
1: Indicates a read operation.
0: Indicates a write operation.

Bit 0 **OB**: Operation Busy.

This bit is set along with a read or write command to initiate an indirect access to the *MAC type-based Rx queue mapping register (ETH_MAC_TMRQR)*. This bit is reset when the read or write command to the *MAC type-based Rx queue mapping register (ETH_MAC_TMRQR)* is complete. The next indirect register access can be initiated only after this bit is reset. During a write operation, the bit is reset only after the data has been written to the *MAC type-based Rx queue mapping register (ETH_MAC_TMRQR)* register. During a read operation, the data should be read from the *MAC type-based Rx queue mapping register (ETH_MAC_TMRQR)* only after this bit is reset.

MAC type-based Rx queue mapping register (ETH_MAC_TMRQR)

Address offset: 0x0A74

Reset value: 0x0000 0000

This register contains the type, associated queue number and packet type (Preemption/Express) related to Type-based RXQ mappings. It holds the data for eight indirect registers, defined by the AOFF[7:0] bitfield in the *MAC indirect access control register (ETH_MAC_IACR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PFEX	Res.	TMRQ[2:0]		
											rw		rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TYP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **PFEX**: Preemption or Express Packet

This bit indicates if it is a preemption packet or express packet.

0: Express packet

1: Preemption packet

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **TMRQ[2:0]**: Type Match Rx Queue Number

Indicates the receive queue number to which the packet needs to be forwarded on detecting a type match.

Bits 15:0 **TYP[15:0]**: Type field Value

Indicates the type value of packet that needs to be compared with the received Ethernet packet. This field is valid when programmed to a value greater than or equal to 0x0600.

Timestamp control register (ETH_MACTSCR)

Address offset: 0x0B00

Reset value: 0x0000 2000

This register controls the operation of the System Time generator and processing of PTP packets for timestamping in the receiver.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	AV8021ASMEN	Res.	Res.	Res.	TXTSSTSM	Res.	Res.	Res.	Res.	Res.	TSENMACADDR	SNAPTYPSEL[1:0]	
			rw				rw						rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSMSTRENA	TSEVNTENA	TSIPV4ENA	TSIPV6ENA	TSIPENA	TSVER2ENA	TSCTRLSSR	TSENALL	Res.	PTGE	TSADDREG	Res.	TSUPDT	TSINIT	TSCFUPDT	TSENA
rw	rw	rw	rw	rw	rw	rw	rw		rw	rw		rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **AV8021ASMEN**: AV 802.1AS Mode Enable

When this bit is set, the MAC processes only untagged PTP over Ethernet packets for providing PTP status and capturing timestamp snapshots, that is, IEEE 802.1AS operating mode.

When the PTP offload feature is enabled, for PTP offload, the transport-specific field in the PTP header is generated and checked based on the value of this bit.

Bits 27:25 Reserved, must be kept at reset value.

Bit 24 **TXTSSTSM**: Transmit Timestamp Status Mode

When this bit is set, the MAC overwrites the earlier transmit timestamp status even if it is not read by the software. The MAC indicates this by setting the TXTSSMIS bit of the *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* register.

When this bit is reset, the MAC ignores the timestamp status of current packet if the timestamp status of previous packet is not read by the software. The MAC indicates this by setting the TXTSSMIS bit of the *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)*.

Bits 23:19 Reserved, must be kept at reset value.

Bit 18 **TSENMACADDR**: Enable MAC Address for PTP Packet Filtering

When this bit is set, the DA MAC address (that matches any MAC Address register) is used to filter the PTP packets when PTP is directly sent over Ethernet.

When this bit is set, received PTP packets with DA containing a special multicast or unicast address that matches the one programmed in MAC address registers are considered for processing as indicated below, when PTP is directly sent over Ethernet.

For normal timestamping operation, MAC address registers 0 to 31 is considered for unicast destination address matching.

For PTP offload, only MAC address register 0 is considered for unicast destination address matching.

Bits 17:16 **SNAPTYPSEL[1:0]**: Select PTP packets for Taking Snapshots

These bits, along with Bits 15 and 14, define the set of PTP packet types for which snapshot needs to be taken. The encoding is given in *Table 789: Timestamp Snapshot Dependency on ETH_MACTSCR bits*.

Bit 15 **TSMSTRENA**: Enable Snapshot for Messages Relevant to Master

When this bit is set, the snapshot is taken only for the messages that are relevant to the master node. Otherwise, the snapshot is taken for the messages relevant to the slave node.

Bit 14 **TSEVNTENA**: Enable Timestamp Snapshot for Event Messages

When this bit is set, the timestamp snapshot is taken only for event messages (SYNC, Delay_Req, Pdelay_Req, or Pdelay_Resp). When this bit is reset, the snapshot is taken for all messages except Announce, Management, and Signaling. For more information about the timestamp snapshots, see *Table 789: Timestamp Snapshot Dependency on ETH_MACTSCR bits*.

Bit 13 **TSIPV4ENA**: Enable Processing of PTP Packets Sent over IPv4-UDP

When this bit is set, the MAC receiver processes the PTP packets encapsulated in IPv4-UDP packets. When this bit is reset, the MAC ignores the PTP transported over IPv4-UDP packets. This bit is set by default.

Bit 12 **TSIPV6ENA**: Enable Processing of PTP Packets Sent over IPv6-UDP

When this bit is set, the MAC receiver processes the PTP packets encapsulated in IPv6-UDP packets. When this bit is clear, the MAC ignores the PTP transported over IPv6-UDP packets.

Bit 11 **TSIPENA**: Enable Processing of PTP over Ethernet Packets

When this bit is set, the MAC receiver processes the PTP packets encapsulated directly in the Ethernet packets. When this bit is reset, the MAC ignores the PTP over Ethernet packets.

Bit 10 **TSVER2ENA**: Enable PTP Packet Processing for Version 2 Format

When this bit is set, the IEEE 1588 version 2 format is used to process the PTP packets. When this bit is reset, the IEEE 1588 version 1 format is used to process the PTP packets. The IEEE 1588 formats are described in 'PTP Processing and Control'.

Bit 9 **TSCTRLSSR**: Timestamp Digital or Binary Rollover Control

When this bit is set, the Timestamp Low register rolls over after 0x3B9A C9FF value (that is, 1 nanosecond accuracy) and increments the timestamp (High) seconds. When this bit is reset, the rollover value of subsecond register is 0x7FFF FFFF. The subsecond increment must be programmed correctly depending on the PTP reference clock frequency and the value of this bit.

Bit 8 **TSENALL**: Enable Timestamp for All Packets

When this bit is set, the timestamp snapshot is enabled for all packets received by the MAC.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **PTGE**: Presentation Time Generation Enable

When this bit is set the Presentation Time generation is enabled.

Bit 5 **TSADDREG**: Update Addend Register

When this bit is set, the content of the Timestamp Addend register is updated in the PTP block for fine correction. This bit is cleared when the update is complete. This bit should be zero before it is set.

Bit 4 Reserved, must be kept at reset value.

Bit 3 **TSUPDT**: Update Timestamp

When this bit is set, the system time is updated (added or subtracted) with the value specified in *System time seconds update register (ETH_MACSTSUR)* and *System time nanoseconds update register (ETH_MACSTNUR)*.
When Media Clock Generation and Recovery is enabled, ETH_MACPRSTIMUR should also be updated before setting this field.
This bit should be zero before updating it. This bit is reset when the update is complete in hardware.

Bit 2 **TSINIT**: Initialize Timestamp

When this bit is set, the system time is initialized (overwritten) with the value specified in the *System time seconds update register (ETH_MACSTSUR)* and *System time nanoseconds update register (ETH_MACSTNUR)*.
When Media Clock Generation and Recovery is enabled, *MAC presentation time update register (ETH_MACPRSTIMUR)* should also be updated before setting this field.
This bit should be zero before it is updated. This bit is reset when the initialization is complete.

Bit 1 **TSCFUPDT**: Fine or Coarse Timestamp Update

When this bit is set, the Fine method is used to update system timestamp. When this bit is reset, Coarse method is used to update the system timestamp.

Bit 0 **TSENA**: Enable Timestamp

When this bit is set, the timestamp is added for transmit and receive packets. When disabled, timestamp is not added for transmit and receive packets and the Timestamp Generator is also suspended. You need to initialize the Timestamp (system time) after enabling this mode.
On the receive side, the MAC processes the 1588 packets only if this bit is set.

Subsecond increment register (ETH_MACSSIR)

Address offset: 0x0B04

Reset value: 0x0000 0000

This register specifies the value to be added to the internal system time register every cycle of clk_ptp_ref_i clock.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSINC[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **SSINC[7:0]**: Subsecond Increment Value

The value programmed in this field is accumulated every clock cycle (of clk_ptp_i) with the contents of the subsecond register. For example, when the PTP clock is 50 MHz (period is 20 ns), you should program 20 (0x14) when the System Time Nanoseconds register has an accuracy of 1 ns [TSCTRLSSR bit is set in *Timestamp control register (ETH_MACTSCR)*]. When TSCTRLSSR is cleared, the Nanoseconds register has a resolution of ~0.465 ns. In this case, you should program a value of 43 (0x2B) which is derived by 20 ns/0.465.

Bits 15:0 Reserved, must be kept at reset value.

System time seconds register (ETH_MACSTSR)

Address offset: 0x0B08

Reset value: 0x0000 0000

The System Time Seconds register, along with System Time Nanoseconds register, indicates the current value of the system time maintained by the MAC. Though it is updated on a continuous basis, there is some delay from the actual time because of clock domain transfer latencies (from clk_ptp_ref_i to CSR clock).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSS[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TSS[31:0]**: Timestamp Second

The value in this field indicates the current value in seconds of the System Time maintained by the MAC.

System time nanoseconds register (ETH_MACSTNR)

Address offset: 0x0B0C

Reset value: 0x0000 0000

The System Time Nanoseconds register, along with System Time Seconds register, indicates the current value of the system time maintained by the MAC.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	TSSS[30:16]														
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSSS[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bits 30:0 **TSSS[30:0]**: Timestamp subseconds

The value in this field has the subsecond representation of time, with an accuracy of 0.46 ns. When TSCTRLSSR is set in *Timestamp control register (ETH_MACTSCR)*, each bit represents 1 ns. The maximum value is 0x3B9A C9FF after which it rolls-over to zero.

System time seconds update register (ETH_MACSTSUR)

Address offset: 0x0B10

Reset value: 0x0000 0000

The System Time Seconds Update register, along with the System Time Nanoseconds update register, initializes or updates the system time maintained by the MAC. You must write both registers before setting the TSINIT or TSUPDT bits in *Timestamp control register (ETH_MACTSCR)*.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSS[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSS[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TSS[31:0]**: Timestamp Seconds

The value in this field is the seconds part of the update.
When ADDSUB is reset, this field must be programmed with the seconds part of the update value.
When ADDSUB is set, this field must be programmed with the complement of the seconds part of the update value.
For example, to subtract 2.000000001 seconds from the system time, the TSS field in the ETH_MACSTSUR register must be 0xFFFF FFFE (that is, $2^{32} – 2$).

Caution: When the ADDSUB bit is set, TSSS[30:0] field cannot be set to 0 in *System time nanoseconds update register (ETH_MACSTNUR)*. The TSSS bitfield must be programmed to 0x7FFF FFFF (resulting in –0.46 ns) even if 0 ns must be subtracted.
For example, to subtract 2.000000000 seconds from the system time, the TSS field in the *System time seconds update register (ETH_MACSTSUR)* must be 0xFFFF FFFE (that is, $2^{32} – 1$) and the *System time nanoseconds update register (ETH_MACSTNUR)* must be 0xFFFF FFFF (ADDSUB = 1 and TSSS[30:0] field = 0x7FFF FFFF)

System time nanoseconds update register (ETH_MACSTNUR)

Address offset: 0x0B14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADD SUB	TSSS[30:16]														
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSSS[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **ADDSUB**: Add or Subtract Time

When this bit is set, the time value is subtracted with the contents of the update register.
When this bit is reset, the time value is added with the contents of the update register.

Bits 30:0 **TSSS[30:0]**: Timestamp subseconds

The value in this field is the subseconds part of the update.

– ADDSUB is 1: This field must be programmed with the complement of the subseconds part of the update value as described.

– ADDSUB is 0: This field must be programmed with the subseconds part of the update value, with an accuracy based on the TSCTRLSSR bit of the *Timestamp control register (ETH_MACTSCR)*.

– TSCTRLSSR field in the *Timestamp control register (ETH_MACTSCR)*is 1:
- The programmed value must be 10^9 – <subsecond value>.
- Each bit represents 1 ns and the programmed value should not exceed 0x3B9A C9FF.

– TSCTRLSSR field in the *Timestamp control register (ETH_MACTSCR)* is 0:
- The programmed value must be 2^{31} - <subsecond_value>.
- Each bit represents an accuracy of 0.46 ns.

For example, to subtract 2.000000001 seconds from the system time, then the TSSS field in the ETH_MACSTNUR register must be 0x7FFF FFFF (that is, 2^{31} – 1), when TSCTRLSSR bit in *Timestamp control register (ETH_MACTSCR)* is reset and 0x3B9A C9FF (that is, 10^9 –1), when TSCTRLSSR bit in *Timestamp control register (ETH_MACTSCR)* is set.

Caution: When the ADDSUB bit is set, TSSS[30:0] field cannot be set to 0. The TSSS bitfield must be programmed to 0x7FFF FFFF (resulting in –0.46 ns) even if 0 ns must be subtracted.
For example, to subtract 2.000000000 seconds from the system time, *System time nanoseconds update register (ETH_MACSTNUR)* must be 0xFFFF FFFF (ADDSUB = 1 and TSSS[30:0] = 0) and the TSS field in the *System time seconds update register (ETH_MACSTSUR)* must be 0xFFFF FFFE (that is, 2^{32} – 1).

Timestamp addend register (ETH_MACTSAR)

Address offset: 0x0B18

Reset value: 0x0000 0000

This register value is used only when the system time is configured for Fine Update mode (TSCFUPDT bit in the *Timestamp control register (ETH_MACTSCR)*). The content of this register is added to a 32-bit accumulator in every clock cycle of clk_ptp_ref_i and the system time is updated whenever the accumulator overflows.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSAR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSAR[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TSAR[31:0]**: Timestamp Addend Register

This field indicates the 32-bit time value to be added to the Accumulator register to achieve time synchronization.

Timestamp status register (ETH_MACTSSR)

Address offset: 0x0B20

Reset value: 0x0000 0000

All bits except Bits[27:25] gets cleared when the application reads this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	ATSNS[4:0]					ATSSTM	Res.	Res.	Res.	Res.	ATSSTN[3:0]			
		r	r	r	r	r	rc_r					rc_r	rc_r	rc_r	rc_r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXTSSIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSTRGTERR1	TSTARGT1	TSTRGTERR0	AUXTSTRIG	TSTARGT0	TSSOVF
rc_r										rc_r	rc_r	rc_r	rc_r	rc_r	rc_r

Bits 31:30 Reserved, must be kept at reset value.

Bits 29:25 **ATSNS[4:0]**: Number of Auxiliary Timestamp Snapshots

This field indicates the number of Snapshots available in the FIFO. A value equal to the depth of FIFO (4) indicates that the Auxiliary Snapshot FIFO is full. These bits are cleared (to 00000) when the Auxiliary snapshot FIFO clear bit is set.

Bit 24 **ATSSTM**: Auxiliary Timestamp Snapshot Trigger Missed

This bit is set when the Auxiliary timestamp snapshot FIFO is full and external trigger was set. This indicates that the latest snapshot is not stored in the FIFO.

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **ATSSTN[3:0]**: Auxiliary Timestamp Snapshot Trigger Identifier

These bits identify the Auxiliary trigger inputs for which the timestamp available in the Auxiliary Snapshot Register is applicable. When more than one bit is set at the same time, it means that corresponding auxiliary triggers were sampled at the same clock. These bits are applicable only if the number of Auxiliary snapshots is more than one. One bit is assigned for each trigger as shown in the following list:

Bit 16: Auxiliary trigger 0

Bit 17: Auxiliary trigger 1

Bit 18: Auxiliary trigger 2

Bit 19: Auxiliary trigger 3

The software can read this register to find the triggers that are set when the timestamp is taken.

Bit 15 **TXTSSIS**: Tx Timestamp Status interrupt status

When drop transmit status is enabled in MTL, this bit is set when the captured transmit timestamp is updated in the *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* and *Tx timestamp status seconds register (ETH_MACTXTSSSR)*.

When PTP offload feature is enabled, this bit is set when the captured transmit timestamp is updated in the *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* and *Tx timestamp status seconds register (ETH_MACTXTSSSR)*, for PTO generated Delay Request and Pdelay request packets.

This bit is cleared when the *Tx timestamp status seconds register (ETH_MACTXTSSSR)* is read (or written to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bits 14:6 Reserved, must be kept at reset value.

Bit 5 **TSTRGTERR1**: Timestamp Target Time Error

This bit is set when the latest target time programmed in the ETH_MACPPSTTS1R and ETH_MACPPSTTSN1R registers elapses. It is cleared when the application reads this bit (or writes it to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 4 **TSTARGT1**: Timestamp Target Time Reached

When set, this bit indicates that the value of the system time is greater than or equal to the value specified in the ETH_MACPPSTTS1R and ETH_MACPPSTTSN1R registers.

This bit is cleared when the application reads this bit (or write it to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set)

Bit 3 **TSTRGTERR0**: Timestamp Target Time Error

This bit is set when the latest target time programmed in the ETH_MACPPSTTS0R and ETH_MACPPSTTSN0R elapses (see *PPS x target time seconds register (ETH_MACPPSTTSxR)* and *PPS x target time nanoseconds register (ETH_MACPPSTTNxR)*). This bit is cleared when the application reads this bit (or writes it to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 2 **AUXTSTRIG**: Auxiliary Timestamp Trigger Snapshot

This bit is set high when the auxiliary snapshot is written to the FIFO.

This bit is cleared when the application reads this bit (or writes it to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set).

Bit 1 **TSTARGT0**: Timestamp Target Time Reached

When set, this bit indicates that the value of system time is greater than or equal to the value specified in the ETH_MACPPSTTS0R and ETH_MACPPSTTSN0R registers (see *PPS x target time seconds register (ETH_MACPPSTTSxR)* and *PPS x target time nanoseconds register (ETH_MACPPSTTNxR)*).

This bit is cleared when the application reads this bit (or writes of 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set)

Bit 0 **TSSOVF**: Timestamp Seconds Overflow

When this bit is set, it indicates that the seconds value of the timestamp (when supporting version 2 format) has overflowed beyond 0xFFFF FFFF.

This bit is cleared when the application reads this bit (or writes it to 1 when RCWE bit in *CSR software control register (ETH_MACCSRSWCR)* is set)

Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)

Address offset: 0x0B30

Reset value: 0x0000 0000

This register contains the nanosecond part of timestamp captured for transmit packets when Tx status is disabled.The *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* register, along with *Tx timestamp status seconds register (ETH_MACTXTSSSR)*, gives the 64-bit timestamp captured for the PTP packet successfully transmitted by the MAC. This value is considered to be read by the application when the last byte of *Tx timestamp status nanoseconds register (ETH_MACTXTSSNR)* is read (bits[31:24]).
If the application does not read these registers and timestamp of another packet is captured, then either the current timestamp is lost (overwritten) or the new timestamp is lost (dropped), depending on the setting of the TXTSSTSM bit of the *Timestamp control register (ETH_MACTSCR)*. The status bit TXTSSIS bit [15] in *Timestamp status register (ETH_MACTSSR)* register is set when the MAC transmitter captures the timestamp.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXTSSMIS	TXTSSLO[30:16]														
r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXTSSLO[15:0]															
rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r	rc_r

Bit 31 **TXTSSMIS**: Transmit Timestamp Status Missed

When this bit is set, it indicates one of the following:

– The timestamp of the current packet is ignored if TXTSSTSM bit of the *Timestamp control register (ETH_MACTSCR)* is reset

– The timestamp of the previous packet is overwritten with timestamp of the current packet if TXTSSTSM bit of the *Timestamp control register (ETH_MACTSCR)* is set.

Bits 30:0 **TXTSSLO[30:0]**: Transmit Timestamp Status Low

This field contains the 31 bits of the Nanoseconds field of the transmit packet's captured timestamp.

Tx timestamp status seconds register (ETH_MACTXTSSSR)

Address offset: 0x0B34

Reset value: 0x0000 0000

The register contains the higher 32 bits of the timestamp (in seconds) captured when a PTP packet is transmitted.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TXTSSHI[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TXTSSHI[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **TXTSSHI[31:0]**: Transmit Timestamp Status High

This field contains the lower 32 bits of the Seconds field of transmit packet's captured timestamp.

Auxiliary control register (ETH_MACACR)

Address offset: 0x0B40

Reset value: 0x0000 0000

The auxiliary timestamp control register controls the auxiliary timestamp snapshot.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATSEN3	ATSEN2	ATSEN1	ATSEN0	Res.	Res.	Res.	ATSFC
								rw	rw	rw	rw				rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **ATSEN3**: Auxiliary Snapshot 3 Enable

– This bit controls the capturing of Auxiliary Snapshot Trigger 3. When this bit is set, the auxiliary snapshot of the event on eth_ptp_trg3 input is enabled. When this bit is reset, the events on this input are ignored.

Bit 6 **ATSEN2**: Auxiliary Snapshot 2 Enable

– This bit controls the capturing of Auxiliary Snapshot Trigger 2. When this bit is set, the auxiliary snapshot of the event on eth_ptp_trg2 input is enabled. When this bit is reset, the events on this input are ignored.

Bit 5 **ATSEN1**: Auxiliary Snapshot 1 Enable

– This bit controls the capturing of Auxiliary Snapshot Trigger 1. When this bit is set, the auxiliary snapshot of the event on eth_ptp_trg1 input is enabled. When this bit is reset, the events on this input are ignored.

Bit 4 **ATSEN0**: Auxiliary Snapshot 0 Enable

This bit controls the capturing of Auxiliary Snapshot Trigger 0. When this bit is set, the auxiliary snapshot of the event on eth_ptp_trg0 input is enabled. When this bit is reset, the events on this input are ignored.

Bits 3:1 Reserved, must be kept at reset value.

Bit 0 **ATSFC**: Auxiliary Snapshot FIFO Clear

When set, this bit resets the pointers of the Auxiliary Snapshot FIFO. This bit is cleared when the pointers are reset and the FIFO is empty. When this bit is high, the auxiliary snapshots are stored in the FIFO.

Auxiliary timestamp nanoseconds register (ETH_MACATSNR)

Address offset: 0x0B48

Reset value: 0x0000 0000

The *Auxiliary timestamp nanoseconds register (ETH_MACATSNR)*, along with *Auxiliary timestamp seconds register (ETH_MACATSSR)*, gives the 64-bit timestamp stored as auxiliary snapshot. These two registers form the read port of a 64-bit wide FIFO with a depth of 4 words.

You can store multiple snapshots in this FIFO. Bits[29:25] in *Timestamp status register (ETH_MACTSSR)* indicate the fill-level of the FIFO. The top of the FIFO is removed only when *Auxiliary timestamp seconds register (ETH_MACATSSR)* is read.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	AUXTSLO[30:16]														
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AUXTSLO[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bits 30:0 **AUXTSLO[30:0]**: Auxiliary Timestamp

Contains the lower 31 bits (nanoseconds field) of the auxiliary timestamp.

Auxiliary timestamp seconds register (ETH_MACATSSR)

Address offset: 0x0B4C

Reset value: 0x0000 0000

The Auxiliary Timestamp Seconds register contains the lower 32 bits of the Seconds field of the auxiliary timestamp register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AUXTSHI[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AUXTSHI[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **AUXTSHI[31:0]**: Auxiliary Timestamp

Contains the lower 32 bits of the Seconds field of the auxiliary timestamp.

Timestamp ingress asymmetric correction register (ETH_MACTSIACR)

Address offset: 0x0B50

Reset value: 0x0000 0000

The MAC Timestamp Ingress Asymmetry Correction register contains the Ingress Asymmetry Correction value to be used while updating correction field in PDelay_Resp PTP messages.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSTIAC[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSTIAC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **OSTIAC[31:0]**: One-Step Timestamp Ingress Asymmetry Correction

This field contains the ingress path asymmetry value to be added to correctionField of Pdelay_Resp PTP packet. The programmed value should be in units of nanoseconds and multiplied by 2^{16}. For example, 2.5 ns is represented as 0x00028000.

The value can also be negative, which is represented in 2's complement form with bit 31 representing the sign bit.

Timestamp egress asymmetric correction register (ETH_MACTSEACR)

Address offset: 0x0B54

Reset value: 0x0000 0000

The MAC Timestamp Egress Asymmetry Correction register contains the Egress Asymmetry Correction value to be used while updating the correction field in PDelay_Req PTP messages.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
OSTEAC[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSTEAC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **OSTEAC[31:0]**: One-Step Timestamp Egress Asymmetry Correction

This field contains the egress path asymmetry value to be subtracted from correctionField of Pdelay_Resp PTP packet. The programmed value must be the negated value in units of nanoseconds multiplied by 2^{16}.

For example, if the required correction is +2.5 ns, the programmed value must be 0xFFFD 8000, which is the 2's complement of 0x0002 8000(2.5×2^{16}). Similarly, if the required correction is -3.3 ns, the programmed value is 0x0003 4CCC (3.3×2^{16}).

Timestamp ingress correction nanosecond register (ETH_MACTSICNR)

Address offset: 0x0B58

Reset value: 0x0000 0000

This register contains the correction value in nanoseconds to be used with the captured timestamp value in the ingress path.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSIC[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSIC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TSIC[31:0]**: Timestamp Ingress Correction

This field contains the ingress path correction value as defined by the Ingress Correction expression.

Timestamp egress correction nanosecond register (ETH_MACTSECNR)

Address offset: 0x0B5C

Reset value: 0x0000 0000

This register contains the correction value in nanoseconds to be used with the captured timestamp value in the egress path.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSEC[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSEC[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TSEC[31:0]**: Timestamp Egress Correction

This field contains the nanoseconds part of the egress path correction value as defined by the Egress Correction expression.

Timestamp ingress latency register (ETH_MACTSILR)

Address offset: 0x0B68

Reset value: 0x0000 0000

This register holds the ingress MAC latency.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res	Res	Res	Res	ITLNS[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ITLSNS[7:0]								Res	Res	Res	Res	Res	Res	Res	Res
r	r	r	r	r	r	r	r								

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **ITLNS[11:0]**: Ingress Timestamp Latency, in nanoseconds

This register holds the average latency (in nanoseconds) between the MAC input ports and the actual point (GMII/MII) where the ingress timestamp is taken. Ingress correction value is computed as described in *Section : Ingress correction*.

Bits 15:8 **ITLSNS[7:0]**: Ingress Timestamp Latency, in subnanoseconds

This register holds the average latency (in subnanoseconds) between the MAC input ports and the actual point (GMII/MII) where the ingress timestamp is taken. ingress correction value is computed as described in *Section : Ingress correction*.

Bits 7:0 Reserved, must be kept at reset value.

Timestamp egress latency register (ETH_MACTSELR)

Address offset: 0x0B6C

Reset value: 0x0000 0000

This register holds the Egress MAC latency.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	ETLNS[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ETLSNS[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	r								

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:16 **ETLNS[11:0]**: Egress Timestamp Latency, in nanoseconds

This register holds the average latency in nanoseconds between the actual point (GMII/MII) where the egress timestamp is taken and the output ports of the MAC. Egress correction value is computed as described in *Section : Egress correction*.

Bits 15:8 **ETLSNS[7:0]**: Egress Timestamp Latency, in subnanoseconds

This register holds the average latency in subnanoseconds between the actual point (GMII/MII) where the egress timestamp is taken and the output ports of the MAC. Egress correction value is computed as described in *Section : Egress correction*.

Bits 7:0 Reserved, must be kept at reset value.

PPS control register (ETH_MACPPSCR)

Address offset: 0x0B70

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	TIMESEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			rw												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCGREN0	TRGTMODSEL0[1:0]		PPSEN0	PPSCTRL[3:0]			
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **TIMESEL**: Time Select

When this bit is set, 64 bit PTP time is used to capture time at MCGR trigger[0] input
When this bit is reset, presentation time is used to capture time at trigger input, maintaining backward compatibility

Bits 27:8 Reserved, must be kept at reset value.

Bit 7 **MCGREN0**: MCGR Mode Enable for PPS0 Output

This field enables the 0th PPS instance to operate in PPS or MCGR mode. When set it operates in MCGR mode and on reset it operates in PPS mode.

0: PPS mode

1: MCGR mode

Bits 6:5 **TRGTMODSEL0[1:0]**: Target Time Register Mode for PPS Output

This field indicates the Target Time registers (*PPS x target time seconds register (ETH_MACPPSTTSxR)* and *PPS x target time nanoseconds register (ETH_MACPPSTTNxR)*) mode for PPS output signal:

00: Target Time registers are programmed only for generating the interrupt event.

01: Enables MCGR Interrupt whose status bit is indicated by TSTARGT0 bit in *Timestamp status register (ETH_MACTSSR)* register

10: Target Time registers are programmed for generating the interrupt event and starting or stopping the PPS output signal generation.

11: Target Time registers are programmed only for starting or stopping the PPS output signal generation. No interrupt is asserted.

Bit 4 **PPSEN0**: Flexible PPS Output Mode Enable

When this bit is set, PPSCTRL[3:0] function as PPSCMD[3:0]. When this bit is reset, PPSCTRL[3:0] function as PPSCTRL (Fixed PPS mode).

Bits 3:0 **PPSCTRL[3:0]**: PPS Output Frequency Control

This field controls the frequency of the PPS output (eth_ptp_pps_out) signal. The default value of PPSCTRL is 0000, and the PPS output is 1 pulse (of width clk_ptp_i) every second. For other values of PPSCTRL, the PPS output becomes a generated clock of following frequencies:

0001: The binary rollover is 2 Hz, and the digital rollover is 1 Hz.

0010: The binary rollover is 4 Hz, and the digital rollover is 2 Hz.

0011: The binary rollover is 8 Hz, and the digital rollover is 4 Hz.

0100: The binary rollover is 16 Hz, and the digital rollover is 8 Hz.

..

1111: The binary rollover is 32.768 KHz and the digital rollover is 16.384 KHz.

Note: In the binary rollover mode, the PPS output (eth_ptp_pps_out) has a duty cycle of 50 percent with these frequencies. In the digital rollover mode, the PPS output frequency is an average number. The actual clock is of different frequency that gets synchronized every second. For example:

- *When PPSCTRL = 0001, the PPS (1 Hz) has a low period of 537 ms and a high period of 463 ms.*
- *When PPSCTRL = 0010, the PPS (2 Hz) is a sequence of*
  One clock of 50 percent duty cycle and 537 ms period
  Second clock of 463 ms period (268 ms low and 195 ms high).
- *When PPSCTRL = 0011, the PPS (4 Hz) is a sequence of*
  Three clocks of 50 percent duty cycle and 268 ms period
  Fourth clock of 195 ms period (134 ms low and 61 ms high)

This behavior is because of the non-linear toggling of bits in the digital rollover mode in the ETH_MACSTNR register.

PPS control register [alternate] (ETH_MACPPSCR)

Address offset: 0x0B70

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	TIMESEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			rw												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCGREN1	TRGTMODSEL1[1:0]		Res.	PPSCMD1[3:0]				MCGREN0	TRGTMODSEL0[1:0]		PPSEN0	PPSCMD[3:0]			
rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:29 Reserved, must be kept at reset value.

Bit 28 **TIMESEL**: Time Select

When this bit is set, 64 bit PTP time is used to capture time at MCGR trigger[0] input
When this bit is reset, presentation time is used to capture time at trigger input, maintaining backward compatibility

Bits 27:16 Reserved, must be kept at reset value.

Bit 15 **MCGREN1**: MCGR Mode Enable for PPS Output 1

This field enables the 0th PPS instance to operate in PPS or MCGR mode. When set, it operates in MCGR mode and when reset it operates in PPS mode.
0: PPS mode
1: MCGR mode

Bits 14:13 **TRGTMODSEL1[1:0]**: Target Time Register Mode for PPS Output 1

This field indicates the target time register (*PPS x target time seconds register (ETH_MACPPSTTSxR)* and *PPS x target time nanoseconds register (ETH_MACPPSTTNxR)*) mode for PPS output signal:
00: Target time registers are programmed only for generating the interrupt event.
01: Enabled MCGR Interrupt whose status bit is indicated by TSTARGT1 bit in *Timestamp status register (ETH_MACTSSR)* register
10: Target time registers are programmed for generating the interrupt event and starting or stopping the PPS output signal generation.
11: Target time registers are programmed only for starting or stopping the PPS output signal generation. No interrupt is asserted.

Bit 12 Reserved, must be kept at reset value.

Bits 11:8 **PPSCMD1[3:0]:** Flexible PPS Output 1 Control

This field controls the flexible PPS output 1 signal. This field is similar to the PPSCMD[3:0] bitfield.

– When MCGREN1 is set, PPSCMD1 indicated by the four bits [11:8] are taken as Presentation Time Control bits for media clock generation and recovery for comparator instance 1. This field is similar to the PPSCMD0 Presentation Time Control bits.
– When MCGREN1 is cleared, only the three bits [10:8] are used as PPSCMD1 and the 4th bit must be cleared.

Bit 7 **MCGREN0**: MCGR Mode Enable for PPS Output 0

This field enables the 0th PPS instance to operate in PPS or MCGR mode. When set it operates in MCGR mode and on reset it operates in PPS mode.

0: PPS mode

1: MCGR mode

Bits 6:5 **TRGTMODSEL0[1:0]**: Target Time Register Mode for PPS Output 0

This field indicates the Target Time registers (*PPS x target time seconds register (ETH_MACPPSTTSxR)* and *PPS x target time nanoseconds register (ETH_MACPPSTTNxR)*) mode for PPS output signal:

00: Target Time registers are programmed only for generating the interrupt event.

01: Enables MCGR Interrupt whose status bit is indicated by TSTARGT0 bit in *Timestamp status register (ETH_MACTSSR)* register

10: Target Time registers are programmed for generating the interrupt event and starting or stopping the PPS output 0 signal generation.

11: Target Time registers are programmed only for starting or stopping the PPS output 0 signal generation. No interrupt is asserted.

Bit 4 **PPSEN0**: Flexible PPS Output 0 Mode Enable

When this bit is set, Bits[3:0] function as PPSCMD[3:0]. When this bit is reset, Bits[3:0] function as PPSCTRL(Fixed PPS mode).

Bits 3:0 **PPSCMD[3:0]:** Flexible PPS Output 0 (eth_ptp_pps_out) Control

Programming these bits with a non-zero value instructs the MAC to initiate an event. When the command is transferred or synchronized to the PTP clock domain, these bits get cleared automatically. The software should ensure that these bits are programmed only when they are ‘all zero’. The following list describes the values of PPSCMD0:

0000: No Command

0001: START Single Pulse.

This command generates single pulse rising at the start point defined in Target Time Registers (register 455 and 456) and of a duration defined in the PPS Width Register.

0010: START Pulse Train.

This command generates the train of pulses rising at the start point defined in the Target Time Registers and of a duration defined in the PPS Width Register and repeated at interval defined in the PPS Interval Register. By default, the PPS pulse train is free-running unless stopped by the 'Stop Pulse train at time' or 'Stop Pulse Train immediately' commands.

0011: Cancel START.

This command cancels the START Single Pulse and START Pulse Train commands if the system time has not crossed the programmed start time.

0100: STOP Pulse Train at time.

This command stops the train of pulses initiated by the START Pulse Train command (PPSCMD[3:0] = 0010) after the time programmed in the Target Time registers elapses.

0101: STOP Pulse Train immediately.

This command immediately stops the train of pulses initiated by the START Pulse Train command (PPSCMD[3:0] = 0010).

0110: Cancel STOP Pulse train.

This command cancels the STOP pulse train at time command if the programmed stop time has not elapsed. The PPS pulse train becomes free-running on the successful execution of this command.

0111 to 1111: Reserved, must not be used

PPS x target time seconds register (ETH_MACPPSTTSxR)

Address offset: 0x0B80 + 0x10 *x, (x = 0 to 1)

Reset value: 0x0000 0000

The PPS output x Target Time Seconds register, along with *PPS x target time nanoseconds register (ETH_MACPPSTTNxR)*, is used to schedule an interrupt event (Bit TSSOVF of *Timestamp status register (ETH_MACTSSR)*) when the system time exceeds the value programmed in these registers.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSTRH0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSTRH0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TSTRH0[31:0]**: PPS Target Time Seconds Register

This field stores the time in seconds. When the timestamp value matches or exceeds both Target Timestamp registers, the MAC starts or stops the PPS signal output and generates an interrupt (if enabled) based on Target Time mode selected for the corresponding PPS output in the *PPS control register (ETH_MACPPSCR)*.

If the PTGE field of *Timestamp control register (ETH_MACTSCR)* is set with Presentation time control set in Recovery mode, then these bits indicate the TPT being programmed by the application and in Generation mode it indicates the CPT generated at the sampled trigger.

PPS x target time nanoseconds register (ETH_MACPPSTTNxR)

Address offset: 0x0B84 + 0x10 *x, (x = 0 to 1)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TRGTBUSY0	TTSL0[30:16]														
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TTSL0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **TRGTBUSY0**: PPS Target Time Register Busy

The MAC sets this bit when the PPSCMD0 field in the *PPS control register (ETH_MACPPSCR)* is programmed to 010 or 011. Programming the PPSCMD0 field to 010 or 011 instructs the MAC to synchronize the Target Time Registers to the PTP clock domain. The MAC clears this bit after synchronizing the Target Time Registers with the PTP clock domain The application must not update the Target Time Registers when this bit is read as 1. Otherwise, the synchronization of the previous programmed time gets corrupted.

Bits 30:0 **TTSL0[30:0]**: Target Time Low for PPS Register

This register stores the time in (signed) nanoseconds. When the value of the timestamp matches the value in both Target Timestamp registers, the MAC starts or stops the PPS signal output and generates an interrupt (if enabled) based on the TRGTMODSEL0 field (Bits [6:5]) in *PPS control register (ETH_MACPPSCR)*.

When the TSCTRLSSR bit is reset in the *Timestamp control register (ETH_MACTSCR)*, this value should be (time in ns / 0.465). The actual start or stop time of the PPS signal output might have an error margin up to one unit of subsecond increment value.

When the TSCTRLSSR bit is set in the *Timestamp control register (ETH_MACTSCR)*, this value should not exceed 0x3B9A C9FF. The actual start or stop time of the PPS signal output may have an error margin up to one unit of subsecond increment value.

PPS x interval register (ETH_MACPPSIxR)

Address offset: 0x0B88 + 0x10 *x, (x = 0 to 1)

Reset value: 0x0000 0000

The PPS Interval register contains the number of units of subsecond increment value between the rising edges of PPS output x.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PPSINT0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PPSINT0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PPSINT0[31:0]**: PPS Output Signal Interval

These bits store the interval between the rising edges of PPS signal output. The interval is stored in terms of number of units of subsecond increment value.

You need to program one value less than the required interval. For example, if the PTP reference clock is 50 MHz (period of 20 ns), and desired interval between the rising edges of PPS signal output is 100 ns (that is, 5 units of subsecond increment value), you should program value 4 (5-1) in this register.

PPS x width register (ETH_MACPPSWxR)

Address offset: 0x0B8C + 0x10 *x, (x = 0 to 1)

Reset value: 0x0000 0000

The PPS Width register contains the number of units of subsecond increment value between the rising and corresponding falling edges of PPS output x.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PPSWIDTH0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PPSWIDTH0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PPSWIDTH0[31:0]**: PPS Output Signal Width

These bits store the width between the rising edge and corresponding falling edge of PPS signal output. The width is stored in terms of number of units of subsecond increment value. You need to program one value less than the required interval. For example, if PTP reference clock is 50 MHz (period of 20 ns), and width between the rising and corresponding falling edges of PPS signal output is 80 ns (that is, four units of subsecond increment value), you should program value 3 (4-1) in this register.

Note: The value programmed in this register must be lesser than the value programmed in PPS x interval register (ETH_MACPPSIxR).

PTP offload control register (ETH_MACPOCR)

Address offset: 0x0BC0

Reset value: 0x0000 0000

This register controls the PTP Offload Engine operation.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DN[7:0]								PDRDIS	DRRDIS	APDREQTRIG	ASYNCTRIG	Res.	APDREQEN	ASYNCEN	PTOEN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **DN[7:0]**: Domain Number

This field indicates the domain Number in which the PTP node is operating.

Bit 7 **PDRDIS**: Disable Peer Delay Response response generation

When this bit is set, the Peer Delay Response (Pdelay_Resp) response is not generated for received Peer Delay Request (Pdelay_Req) request packet, as required by the programmed mode.

Setting this bit to 1 affects the normal PTP Offload operation and the time synchronization. So, this bit must be set only if there is problem with Pdelay_Resp generation in hardware and/or Pdelay_Resp generation is handled by software.

Bit 6 **DRRDIS**: Disable PTO Delay Request/Response response generation

When this bit is set, the Delay Request and Delay response are not generated for received SYNC and Delay request packet respectively, as required by the programmed mode.

Bit 5 **APDREQTRIG**: Automatic PTP Pdelay_Req message Trigger

When this bit is set, one PTP Pdelay_Req message is transmitted. This bit is automatically cleared after the PTP Pdelay_Req message is transmitted. The application should set the APDREQEN bit for this operation.

Bit 4 **ASYNCTRIG**: Automatic PTP SYNC message Trigger

When this bit is set, one PTP SYNC message is transmitted. This bit is automatically cleared after the PTP SYNC message is transmitted. The application should set the ASYNCEN bit for this operation.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **APDREQEN**: Automatic PTP Pdelay_Req message Enable
When this bit is set, PTP Pdelay_Req message is generated periodically based on interval programmed or trigger from application, when the MAC is programmed to be in Peer-to-Peer Transparent mode.

Bit 1 **ASYNCEN**: Automatic PTP SYNC message Enable
When this bit is set, PTP SYNC message is generated periodically based on interval programmed or trigger from application, when the MAC is programmed to be in Clock Master mode.

Bit 0 **PTOEN**: PTP Offload Enable
When this bit is set, the PTP Offload feature is enabled.

PTP source port identity 0 register (ETH_MACSPI0R)

Address offset: 0x0BC4

Reset value: 0x0000 0000

This register contains Bits[31:0] of the 80-bit Source Port Identity of the PTP node.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SPI0[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI0[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SPI0[31:0]**: Source Port Identity 0
This field indicates bits [31:0] of sourcePortIdentity of PTP node.

PTP Source port identity 1 register (ETH_MACSPI1R)

Address offset: 0x0BC8

Reset value: 0x0000 0000

This register contains Bits[63:32] of the 80-bit Source Port Identity of the PTP node.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SPI1[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI1[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SPI1[31:0]**: Source Port Identity 1
This field indicates bits [63:32] of sourcePortIdentity of PTP node.

PTP Source port identity 2 register (ETH_MACSPI2R)

Address offset: 0x0BCC

Reset value: 0x0000 0000

This register contains Bits[79:64] of the 80-bit Source Port Identity of the PTP node.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPI2[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **SPI2[15:0]**: Source Port Identity 2
This field indicates bits [79:64] of sourcePortIdentity of PTP node.

Log message interval register (ETH_MACLMIR)

Address offset: 0x0BD0

Reset value: 0x0000 0000

This register contains the periodic intervals for automatic PTP packet generation.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LMPDRI[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	DRSYNCR[2:0]			LSI[7:0]							
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **LMPDRI[7:0]**: Log Min Pdelay_Req Interval
This field indicates logMinPdelayReqInterval of PTP node. This is used to schedule the periodic Pdelay request packet transmission. Allowed values are -15 to 15.Negative value must be represented in 2's-complement form. For example, if the required value is -1, the value programmed must be 0xFF.

Bits 23:11 Reserved, must be kept at reset value.

Bits 10:8 **DRSYNCR[2:0]**:
Delay_Req to SYNC Ratio
In Slave mode, it is used for controlling frequency of Delay_Req messages transmitted.
0: DelayReq generated for every received SYNC
1: DelayReq generated every alternate reception of SYNC
2: for every 4 SYNC messages
3: for every 8 SYNC messages
4: for every 16 SYNC messages
5: for every 32 SYNC messages
Others: Reserved, must not be used
The master sends this information (logMinDelayReqInterval) in the DelayResp PTP messages to the slave. The reception processes this value from the received DelayResp messages and updates this field accordingly. In the Slave mode, the host must not write/update this register unless it has to override the received value. In Master mode, the sum of this field and logSyncInterval (LSI) field is provided in the logMinDelayReqInterval field of the generated multicast Delay_Resp PTP message.

Bits 7:0 **LSI[7:0]**:

Log Sync Interval

This field indicates the periodicity of the automatically generated SYNC message when the PTP node is Master. Allowed values are -15 to 15. Negative value must be represented in 2's-complement form. For example, if the required value is -1, the value programmed must be 0xFF.

Ethernet MAC register map and reset values

Table 852. Ethernet MAC register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000	ETH_MACCR	ARPEN	SARC[2:0]			IPC	IPG[2:0]			GPSLCE	S2KP	CST	ACS	WD	BE	JD	JE	PS	FES	DM	LM	ECRSFD	DO	DCRS	DR	Res.	BL[1:0]		DC	PRELEN[1:0]		TE	RE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0
0x0004	ETH_MACECR	Res.	APDIM	EIPG[4:0]					EIPGEN	Res.	Res.	Res.	Res.	Res.	USP	SPEN	DCRCC	Res.	Res.	GPSL[13:0]														
	Reset value		0	0	0	0	0	0	0						0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0008	ETH_MACPFR	RA	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DNTU	IPFE	Res.	Res.	Res.	VTFE	Res.	Res.	Res.	Res.	Res.	HPF	SAF	SAIF	PCF[1:0]		DBF	PM	DAIF	HMC	HUC	PR
	Reset value	0										0	0				0						0	0	0	0	0	0	0	0	0	0	0
0x000C	ETH_MACWTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PWE	Res.	Res.	Res.	Res.	WTO[3:0]			
	Reset value																								0					0	0	0	0
0x0010	ETH_MACHT0R	HT31T0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0014	ETH_MACHT1R	HT63T32[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0018 - 0x004C	Reserved																																
0x0050	ETH_MACVTCR	EIVLRXS	Res.	EIVLS[1:0]		ERIVLT	EDVLP	VTHM	EVLRXS	Res.	EVLS[1:0]		DOVLTC	ERSVLM	ESVL	VTIM	ETV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OFS[1:0]		CT	OB
	Reset value	0		0	0	0	0	0	0		0	0	0	0	0	0	0												0	0	0	0
0x0054	ETH_MACVTDR	Res.	Res.	Res.	Res.	Res.	Res.	DMACHN	DMACHEN	Res.	Res.	Res.	ERIVLT	ERSVLM	DOVLTC	ETV	VEN	VID[15:0]															
	Reset value							0	0				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0058	ETH_MACVHTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VLHT[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x005C	Reserved																																
0x0060	ETH_MACVIR	BUSY	RDWR	Res.	Res.	Res.	Res.	Res.	ADDR	Res.	Res.	CBTI	VLTI	CSVL	VLP	VLC[1:0]		VLT[15:0]															
	Reset value	0	0						0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0060	ETH_MACVIR [alternate]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CSV	Res.	Res.	Res.	VLT[15:0]															
	Reset value													0				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 852. Ethernet MAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0064	ETH_MACiVIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VLTI	CSVL	VLP	VLC[1:0]		VLT[15:0]															
	Reset value												0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0068 - 0x006C	Reserved																																
0x0070	ETH_ MACQ0TXFCR	PT[15:0]																Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DZPQ	PLT[2:0]			Res.	Res.	TFE	FCB_BP
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									0	0	0	0			0	0
0x0074 - 0x008C	Reserved																																
0x0090	ETH_MACRXFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UP	RFE
	Reset value																															0	0
0x0094	ETH_MACRXQCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VFFQ	VFFQE	Res.	Res.	Res.	Res.	Res.	Res.	MFFQ	MFFQE	Res.	Res.	Res.	Res.	Res.	Res.	UFFQ	UFFQE
	Reset value															0	0							0	0							0	0
0x0098- 0x009C	Reserved																																
0x00A0	ETH_ MACRXQC0R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXQ1EN[1:0]		RXQ0EN[1:0]	
	Reset value																													0	0	0	0
0x00A4	ETH_ MACRXQC1R	Res.	Res.	TBRQE	OMCBCQ	Res.	FPRQ[2:0]			TPQC[1]	TPQC[0]	TACPQE	MCBCQEN	Res.	MCBCQ[2:0]			Res.	UPQ[2:0]			Res.	Res.	Res.	Res.	Res.	PTPQ[2:0]			Res.	AVCPQ[2:0]		
	Reset value			0	0		0	0	0	0	0	0	0		0	0	0		0	0	0						0	0	0		0	0	0
0x00A8	ETH_ MACRXQC2R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSRQ1[7:0]								PSRQ0[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00AC	Reserved																																
0x00B0	ETH_MACISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MFRIS	MFTIS	MDIOIS	FPEIS	Res.	Res.	RXSTSIS	TXSTSIS	TSIS	Res.	MMCTXIS	MMCRXIS	MMCIS	Res.	Res.	LPIIS	PMTIS	PHYIS	Res.	Res.	RGSMIII
	Reset value												0	0	0	0			0	0	0		0	0	0			0	0	0			0
0x00B4	ETH_MACIER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MDIOIE	FPEIE	Res.	Res.	RXSTSIE	TXSTSIE	TSIE	Res.	Res.	Res.	Res.	Res.	Res.	LPIIE	PMTIE	PHYIE	Res.	Res.	RGSMIIIE
	Reset value														0	0			0	0	0							0	0	0			0

Table 852. Ethernet MAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00B8	ETH_ MACRXTXSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RWT	Res.	Res.	EXCOL	LCOL	EXDEF	LCARR	NCARR	TJT
	Reset value																								0			0	0	0	0	0	0
0x00BC	Reserved																																
0x00C0	ETH_MACPCSR	RWKFILTRS	Res.	Res.	RWKPTR[4:0]					Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RWKPFE	GLBLUCAST	Res.	Res.	RWKPRCVD	MGKPRCVD	Res.	Res.	RWKPKTEN	MGKPKTEN	PWRDWN
	Reset value	0			0	0	0	0	0														0	0			0	0			0	0	0
0x00C4	ETH_ MACRWKPFR	MACRWKPFR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C8 - 0x00CC	Reserved																																
0x00D0	ETH_MACLCSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPITCSE	LPITE	LPITXA	PLSEN	PLS	LPIEN	Res.	Res.	Res.	Res.	Res.	Res.	RLPIST	TLPIST	Res.	Res.	Res.	Res.	RLPIEX	RLPIEN	TLPIEX	TLPIEN
	Reset value											0	0	0	0	0	0							0	0					0	0	0	0
0x00D4	ETH_MACLTCR	Res.	Res.	Res.	Res.	Res.	Res.	LST[9:0]										TWT[15:0]															
	Reset value							1	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00D8	ETH_MACLETR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPIET[19:0]																			
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00DC	ETH_ MAC1USTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TIC_1US_CNTR[11:0]											
	Reset value																					0	0	0	0	0	1	1	0	0	0	1	1
0x00E0 - 0x00F4	Reserved																																
0x00F8	ETH_ MACPHYCSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LNKSTS	LNKSPEED[1:0]		LNKMOD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LUD	TC
	Reset value													0	0	0	0															0	0
0x00FC - 0x010C	Reserved																																
0x0110	ETH_MACVR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	USERVER[7:0]								SNPSVER[7:0]							
	Reset value	0x0000 2052																															
0x0114	ETH_MACDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TFCSTS[1:0]		TPESTS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RFCFCSTS[1:0]		RPESTS
	Reset value														0	0	0														0	0	0

Table 852. Ethernet MAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0118	Reserved																																
0x011C	ETH_MACHWF0R	ACTPHYSEL[3:0]				SAVLANINS	TSSTSSEL[1:0]			MACADR64SEL	MACADR32SEL	ADDMACADRSEL[4:0]					Res.	RXCOESEL	Res.	TXCOESEL	EEESEL	TSSEL	Res.	Res.	ARPOFFSEL	MMCSEL	MGKSEL	RWKSEL	SMASEL	VLHASH	PCSSEL	HDSEL	GMIISEL	MIISEL
	Reset value	0x0A0D 73F7																															
0x0120	ETH_MACHWF1R	Res.	L3L4FNUM[3:0]			Res.	HASHTBLSZ[1:0]			POUOST	Res.	RAVSEL	AVSEL	DBGMEMA	TSOEN	SPHEN	DCBEN	ADDR64[1:0]		ADVTHWORD	PTOEN	OSTEN	TXFIFOSIZE[4:0]				SPRAM	RXFIFOSIZE[4:0]				
	Reset value	0x1114 1965																															
0x0124	ETH_MACHWF2R	Res.	AUXSNAPNUM[2:0]			Res.	PPSOUTNUM[2:0]			TDCSZ[1:0]		TXCHCNT[3:0]				RDCSZ[1:0]		RXCHCNT[3:0]			Res.	Res.	TXQCNT[3:0]				Res.	Res.	RXQCNT[3:0]			
	Reset value	0x4204 1041																															
0x0128	ETH_MACHWF3R	Res.	Res.	ASP[1:0]		TBSSEL	FPESEL	Res.	Res.	Res.	Res.	ESTWID[1:0]		ESTDEP[2:0]		ESTSEL	Res.	FRPES[1:0]		FRPBS[1:0]		FRPSEL	PDUPSEL	Res.	Res.	Res.	DVLAN	CBTISEL	Res.	NRVF[2:0]		
	Reset value	0x0C33 0031																															
0x012C - 0x01FC	Reserved																																
0x0200	ETH_ MACMDIOAR	Res.	Res.	Res.	Res.	PSE	BTB	PA[4:0]					RDA[4:0]					Res.	NTC[2:0]			CR[3:0]				Res.	Res.	Res.	SKAP	GOC[1:0]		C45E	GB
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0		0	0	0	0	0	0	0				0	0	0	0	0
0x0204	ETH_ MACMDIODR	RA[15:0]																GD[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0208 - 0x020C	Reserved																																
0x0210	ETH_MACARPAR	ARPPA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0214 - 0x022C	Reserved																																
0x0230	ETH_ MACCSRSWCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SEEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RCWE
	Reset value																								0								0

Table 852. Ethernet MAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0234	ETH_MACFPECSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRSP	TVER	RRSP	RVER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SRSP	SVER	EFPE
	Reset value													0	0	0	0														0	0	0
0x0238-0x23C	Reserved																																
0x0240	ETH_MACPRSTIMR	MPTN[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0244	ETH_MACPRSTIMUR	MPTU[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0248 - 0x02FC	Reserved																																
0x0300	ETH_MACA0HR	AE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCS	ADDRHI[15:0]															
	Reset value	1															0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0304	ETH_MACA0LR	ADDRLO[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0308	ETH_MACA1HR	AE	SA	MBC[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCS	ADDRHI[15:0]															
	Reset value	0	0	0	0	0	0	0	0								0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x30C	ETH_MACA1LR	ADDRLO[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0310	ETH_MACA2HR	AE	SA	MBC[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCS	ADDRHI[15:0]															
	Reset value	0	0	0	0	0	0	0	0								0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0314	ETH_MACA2LR	ADDRLO[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0318	ETH_MACA3HR	AE	SA	MBC[5:0]						Res.	Res.	Res.	Res.	Res.	Res.	Res.	DCS	ADDRHI[15:0]															
	Reset value	0	0	0	0	0	0	0	0								0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x031C	ETH_MACA3LR	ADDRLO[31:0]																															
	Reset value	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0x0320 - 0x06FC	Reserved																																

Table 852. Ethernet MAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0700	ETH_MMC_ CONTROL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	UCDBC	Res.	Res.	CNTPRSTLVL	CNTPRST	CNTFREEZ	RSTONRD	CNTSTOPRO	CNTRST
	Reset value																								0			0	0	0	0	0	0
0x0704	ETH_MMC_RX_ INTERRUPT	Res.	Res.	Res.	Res.	RXLPITRCIS	RXLPIUSCIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXUCGPIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXALGNERPIS	RXCRCERPIS	Res.	Res.	Res.	Res.	Res.
	Reset value					0	0									0											0	0					
0x0708	ETH_MMC_TX_ INTERRUPT	Res.	Res.	Res.	Res.	TXLPITRCIS	TXLPIUSCIS	Res.	Res.	Res.	Res.	TXGPKTIS	Res.	Res.	Res.	Res.	Res.	TXMCOLGPIS	TXSCOLGPIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value					0	0					0						0	0														
0x070C	ETH_MMC_RX_ INTERRUPT_ MASK	Res.	Res.	Res.	Res.	RXLPITRCIM	RXLPIUSCIM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXUCGPIM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RXALGNERPIM	RXCRCERPIM	Res.	Res.	Res.	Res.	Res.
	Reset value					0	0									0											0	0					
0x0710	ETH_MMC_TX_ INTERRUPT_ MASK	Res.	Res.	Res.	Res.	TXLPITRCIM	TXLPIUSCIM	Res.	Res.	Res.	Res.	TXGPKTIM	Res.	Res.	Res.	Res.	Res.	TXMCOLGPIM	TXSCOLGPIM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value					0	0					0						0	0														
0x0714 - 0x0748	Reserved																																
0x074C	ETH_TX_SINGLE_ COLLISION_ GOOD_PACKETS	TXSNGLCOLG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0750	ETH_TX_MULTIPLE _COLLISION_ GOOD_PACKETS	TXMULTCOLG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0754 - 0x0764	Reserved																																
0x0768	ETH_TX_PACKET_ COUNT_GOOD	TXPKTG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x076C - 0x0790	Reserved																																

Table 852. Ethernet MAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0794	ETH_RX_CRC_ERROR_PACKETS	RXCRCERR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0798	ETH_RX_ALIGNMENT_ERROR_PACKETS	RXALGNERR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x079C - 0x07C0	Reserved																																
0x07C4	ETH_RX_UNICAST_PACKETS_GOOD	RXUCASTG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07C8 - 0x07E8	Reserved																																
0x07EC	ETH_TX_LPI_USEC_CNTR	TXLPIUSC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07F0	ETH_TX_LPI_TRAN_CNTR	TXLPITRC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07F4	ETH_RX_LPI_USEC_CNTR	RXLPIUSC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07F8	ETH_RX_LPI_TRAN_CNTR	RXLPITRC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x07FC- 0x89C	Reserved																																
0x08A0	ETH_MMC_FPE_TX_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HRCIS	FCIS
	Reset value																															0	0
0x08A4	ETH_MMC_FPE_TX_IMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HRCIM	FCIM
	Reset value																															0	0
0x08A8	ETH_MMC_FPE_TX_FCR	TXFFC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08AC	ETH_MMC_TX_HRCR	TXHRC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08B0- 0x08BC	Reserved																																
0x08C0	ETH_MMC_FPE_RX_ISR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCIS	PAOCIS	PSECIS	PAECIS	
	Reset value																												0	0	0	0	

Table 852. Ethernet MAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x08C4	ETH_MMC_FPE_RX_IMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FCIM	PAOCIM	PSECIM	PAECIM
	Reset value																													0	0	0	0
0x08C8	ETH_RX_PACKET_ASM_ERR	PAEC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08CC	ETH_RX_PACKET_SMD_ERR	PSEC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08D0	ETH_RX_PACKET_ASM_OKR	PAOC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08D4	ETH_RX_FPE_FRAG_CR	FFC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08D8-0x08FC	Reserved																																
0x0900	ETH_MACL3L4C0R	Res.	Res.	Res.	DMCHEN0	Res.	Res.	Res.	DMCHN0	Res.	Res.	L4DPIM0	L4DPM0	L4SPIM0	L4SPM0	Res.	L4PEN0	L3HDBM0[4:0]					L3HSBM0[4:0]					L3DAIM0	L3DAM0	L3SAIM0	L3SAM0	Res.	L3PEN0
	Reset value				0				0			0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0
0x0904	ETH_MACL4A0R	L4DP0[15:0]																L4SP0[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0908 - 0x090C	Reserved																																
0x0910	ETH_MACL3A00R	L3A00[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0914	ETH_MACL3A10R	L3A10[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0918	ETH_MACL3A20R	L3A20[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x091C	ETH_MACL3A30R	L3A30[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0920 - 0x092C	Reserved																																
0x0930	ETH_MACL3L4C1R	Res.	Res.	Res.	DMCHEN1	Res.	Res.	Res.	DMCHN1	Res.	Res.	L4DPIM1	L4DPM1	L4SPIM1	L4SPM1	Res.	L4PEN1	L3HDBM1[4:0]					L3HSBM1[4:0]					L3DAIM1	L3DAM1	L3SAIM1	L3SAM1	Res.	L3PEN1
	Reset value				0				0			0	0	0	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		0

Table 852. Ethernet MAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0934	ETH_MACL4A1R	L4DP1[15:0]																L4SP1[15:0]															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0938 - 0x093C	Reserved																																
0x0940	ETH_MACL3A01R	L3A01[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0944	ETH_MACL3A11R	L3A11[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0948	ETH_MACL3A21R	L3A21[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x094C	ETH_MACL3A31R	L3A31[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0950 - 0x0A6C	Reserved																																
0x0A70	ETH_MAC_IACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MSEL[3:0]				AOFF[7:0]								Res.	Res.	AUTO	Res.	Res.	Res.	COM.	OB
	Reset value													0	0	0	0	0	0	0	0	0	0	0	0			0				0	0
0x0A74	ETH_MAC_ TMRQR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PFEX	Res.	TMRW[2:0]			TYP[15:0]																	
	Reset value										0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0A78 - 0x0AFC	Reserved																																
0x0B00	ETH_MACTSCR	Res.	Res.	Res.	AV8021ASMEN	Res.	Res.	Res.	TXTSSTSM	Res.	Res.	Res.	Res.	Res.	TSENMACADDR	SNAPTYPSEL[1:0]		TSMSTRENA	TSEVNTENA	TSIPV4ENA	TSIPV6ENA	TSIPENA	TSVER2ENA	TSCTRLSSR	TSENALL	Res.	PTGE	TSADDREG	Res.	TSUPDT	TSINIT	TSCFUPDT	TSENA
	Reset value				0				0						0	0	0	0	0	1	0	0	0	0	0		0	0		0	0	0	0
0x0B04	ETH_MACSSIR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SSINC[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value									0	0	0	0	0	0	0	0																
0x0B08	ETH_MACSTSR	TSS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B0C	ETH_MACSTNR	Res.	TSSS[30:0]																														
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B10	ETH_MACSTSUR	TSS[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 852. Ethernet MAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0B14	ETH_MACSTNUR	ADDSUB	TSSS[30:0]																														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B18	ETH_MACTSAR	TSAR[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B1C	Reserved																																
0x0B20	ETH_MACTSSR	Res.	Res.	ATSNS[4:0]					ATSSTM	Res.	Res.	Res.	Res.	ATSSTN[3:0]				TXTSSIS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TSTRGTERR1	TSTARGT1	TSTRGTERR0	AUXTSTRIG	TSTARGT0	TSSOVF
	Reset value			0	0	0	0	0	0					0	0	0	0	0										0	0	0	0	0	0
0x0B24-0xB28	Reserved																																
0x0B2C	Reserved																																
0x0B30	ETH_MACTXTSSNR	TXTSSMIS	TXTSSLO[30:0]																														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B34	ETH_MACTXTSSR	TXTSSHI[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B38 - 0x0B3C	Reserved																																
0x0B40	ETH_MACACR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATSEN3	ATSEN2	ATSEN1	ATSEN0	Res.	Res.	Res.	ATSFC
	Reset value																									0	0	0	0				0
0x0B44	Reserved																																
0x0B48	ETH_MACATSNR	Res.	AUXTSLO[30:0]																														
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B4C	ETH_MACATSSR	AUXTSHI[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B50	ETH_MACTSIACR	OSTIAC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B54	ETH_MACTSEACR	OSTEAC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B58	ETH_MACTSICNR	TSIC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 852. Ethernet MAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0B5C	ETH_ MACTSECNR	TSEC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B60 - 0x0B64	Reserved																																
0x0B68	ETH_ MACTSILR	Res.	Res.	Res.	Res.	ITLNS[11:0]												ITLSNS[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								
0x0B6C	ETH_ MACTSELR	Res.	Res.	Res.	Res.	ETLNS[11:0]												ETLSNS[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0								
0x0B70	ETH_MACPPSCR	Res.	Res.	Res.	TIMESEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCGREN0	TRGTMODSEL0[1:0]		PPSEN0	PPSCTRL[3:0]			
	Reset value				0																					0	0	0	0	0	0	0	0
0x0B70	ETH_MACPPSCR (alternate)	Res.	Res.	Res.	TIMESEL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MCGREN1	TRGTMODSEL1[1:0]		Res.	PPSCMD1[3:0]				MCGREN0	TRGTMODSEL0[1:0]		PPSEN0	PPSCMD[3:0]			
	Reset value				0													0	0	0		0	0	0	0	0	0	0	0	0	0	0	0
0x0B74 - 0x0B7C	Reserved																																
0x0B80	ETH_ MACPPSTTS0R	TSTRH0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B84	ETH_ MACPPSTTN0R	TRGTBUSY0	TTSL0[30:0]																														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B88	ETH_MACPPSI0R	PPSINT0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B8C	ETH_MACPPSW0R	PPSWIDTH0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B90	ETH_ MACPPSTTS1R	TSTRH0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 852. Ethernet MAC register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0B94	ETH_ MACPPSTTN1R	TRGTBUSY0	TTSL0[30:0]																														
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B98	ETH_MACPPSI1R	PPSINT0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0B9C	ETH_MACPPSW1R	PPSWIDTH0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0BA0 - 0x0BBC	Reserved																																
0x0BC0	ETH_MACPOCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DN[7:0]								PRDIS	DRRDIS	APDREQTRIG	ASYNCTRIG	Res.	APDREQEN	ASYNCEN	PTOEN
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0		0	0	0
0x0BC4	ETH_MACSPI0R	SPI0[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0BC8	ETH_MACSPI1R	SPI1[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0BCC	ETH_MACSPI2R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPI2[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0BD0	ETH_MACLMIR	LMPDRI[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DRSYNCR[2:0]			LSI[7:0]							
	Reset value	0	0	0	0	0	0	0	0														0	0	0	0	0	0	0	0	0	0	0

Refer to *Section 2.3: Memory organization* for the register boundary addresses.

77 Hardware debug port (HDP)

77.1 HDP introduction

The hardware debug port (HDP) is used to observe internal signals, thanks to some multiplexers: up to 16 signals for each of 8-bit output.

77.2 HDP functional description

HDP main features:

- 8 output signals
- 15 input signals for each output signal
- 8 software programmable signals for pinout agnostic code debugging (one for each output signal)
- output disabling by security signal

The HDP is mainly composed of eight multiplexers under software control through HDP_CTRL. Each multiplexer can output an observed signal on an output pin. The observed signal is selected among a pool of 15 interesting signals for each multiplexer. For each multiplexer, a software programmable signal can be observed.

Table 853 presents for each of the muxes which signal can be selected.

The HDP value at the main multiplexer output can be read by software thanks to HDP_VAL. During the debug, it is often useful to output some signals that are controlled by the software: they can be used to precisely observe a code sequence, or to indicate the current executed task. The debug set #15 is reserved for this purpose: it provides on HDP outputs, the HDP_GPOVAL value. This register is associated with HDP_GPOSET and HDP_GPOCLR, used to toggle signals without a read-modify-write sequence.

Some debug sets expose internal data that require to be protected: HDPEN signal, when low, forces low all HDP block outputs.

Note: Internal signals may toggle at a rate higher than pins can support: their observability may then be compromised.

Table 853. Muxing of HDP signals

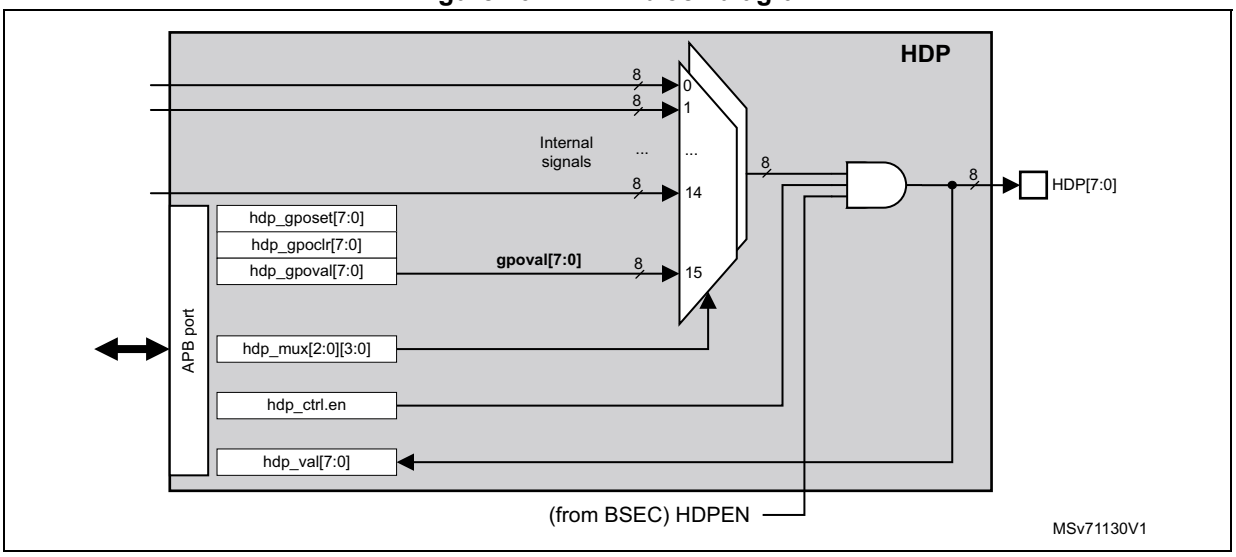
HDP_MUX	HDP7 MUX7[3:0]	HDP6 MUX6[3:0]	HDP5 MUX5[3:0]	HDP4 MUX4[3:0]	HDP3 MUX3[3:0]	HDP2 MUX2[3:0]	HDP1 MUX1[3:0]	HDP0 MUX0[3:0]
0	-	-	-	PWR SEL_VTH_ VDDCORE	-	-	-	PWR pwrwake _sys
1	-	-	-	CPU SLEEPING	-	-	CPU HALTED	CPU SLEEP DEEP

Table 853. Muxing of HDP signals (continued)

HDP_MUX	HDP7 MUX7[3:0]	HDP6 MUX6[3:0]	HDP5 MUX5[3:0]	HDP4 MUX4[3:0]	HDP3 MUX3[3:0]	HDP2 MUX2[3:0]	HDP1 MUX1[3:0]	HDP0 MUX0[3:0]
2	BSEC dbg_ unlocked[7]	BSEC dbg_ unlocked[6]	BSEC dbg_ unlocked[5]	BSEC dbg_ unlocked[4]	BSEC dbg_ unlocked[3]	BSEC dbg_ unlocked[2]	BSEC dbg_ unlocked[1]	BSEC dbg_ unlocked[0]
3	BSEC dbg_ unlocked_ sec[7]	BSEC dbg_ unlocked_ sec[6]	BSEC dbg_ unlocked_ sec[5]	BSEC dbg_ unlocked_ sec[4]	BSEC dbg_ unlocked_ sec[3]	BSEC dbg_ unlocked_ sec[2]	BSEC dbg_ unlocked_ sec[1]	BSEC dbg_ unlocked_ sec[0]
4	BSEC ap_ unlocked[7]	BSEC ap_ unlocked[6]	BSEC ap_ unlocked[5]	BSEC ap_ unlocked[4]	BSEC ap_ unlocked[3]	BSEC ap_ unlocked[2]	BSEC ap_ unlocked[1]	BSEC ap_ unlocked[0]
5	-	-	-	-	-	-	EXTI1 sys_ wakeup	-
6	-	-	ETH_ASYN C_IT	ETH_SYNC _IT	-	-	-	RCC pwrds_sys
7	SMPS_HDP [7]	SMPS_HDP [6]	SMPS_HDP [5]	SMPS_HD P[4]	SMPS_HDP [3]	SMPS_HD P[2]	SMPS_HD P[1]	SMPS_HDP [0]
8	LSE_ CKDV08SW	HSE_CKDV 18m/V08m	SDMMCDLL 2_Lock	SDMMCDL L1_Lock	PLL4_lock	PLL3_lock	PLL2_lock	PLL1_lock
9	GFXTIM_ EVT[3]	GFXTIM_ EVT[2]	GFXTIM_ EVT[1]	GFXTIM_ EVT[0]	GPU2D_GP _FLAG[3]	GPU2D_GP _FLAG[2]	GFXTIM_ EVT[3]	GFXTIM_ EVT[2]
10	MSS_HDP [7]	MSS_HDP [6]	MSS_HDP [5]	MSS_HDP [4]	MSS_HDP [3]	MSS_HDP [2]	MSS_HDP [7]	MSS_HDP [6]
11	-	-	MSS_HDP [13]	MSS_HDP [12]	MSS_HDP [11]	MSS_HDP [10]	-	-
12	USB_HDP [7]	USB_HDP [6]	USB_HDP [5]	USB_HDP [4]	USB_HDP [3]	USB_HDP [2]	USB_HDP [7]	USB_HDP [6]
13	-	-	USB_HDP [13]	USB_HDP [12]	USB_HDP [11]	USB_HDP [10]	-	-
14	-	-	CAMSS dcmipp1_ evt_ frameend[2]	CAMSS dcmipp1_ evt_ frameend[1]	CAMSS dcmipp1_evt _frameend [0]	CAMSS dcmipp1_ evt_ lineend[2]	-	-
15	gpoval[7]	gpoval[6]	gpoval[5]	gpoval[4]	gpoval[3]	gpoval[2]	gpoval[7]	gpoval[6]

77.3 HDP block diagram

Figure 1074. HDP block diagram



77.4 HDP register description

77.4.1 HDP control register (HDP_CTRL)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **EN**: Enable HDP, valid if enabled in BSEC

77.4.2 HDP multiplexer control register (HDP_MUX)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MUX7[3:0]				MUX6[3:0]				MUX5[3:0]				MUX4[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MUX3[3:0]				MUX2[3:0]				MUX1[3:0]				MUX0[3:0]			
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MUXy[3:0]**: Select the HDPy output among the 16 available signals (y = 7 to 0)

77.4.3 HDP read back value register (HDP_VAL)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPVAL[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **HDPVAL[7:0]**: Value of the HDP signals

77.4.4 HDP general-purpose output set register (HDP_GPOSET)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPGPOSET[7:0]							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **HDPGPOSET[7:0]**: When a bit is written to 1, the corresponding HDP GPO is set
Writing a bit to 0 has no effect.

77.4.5 HDP general purpose output clear register (HDP_GPOCLR)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPGPOCLR[7:0]							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **HDPGPOCLR[7:0]**: When a bit is written to 1, the corresponding HDP GPO is cleared. Writing a bit to 0 has no effect.

77.4.6 HDP general purpose output value register (HDP_GPOVAL)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPGPOVAL[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **HDPGPOVAL[7:0]**: When written, define the value of the HDP GPO. When read, provide the current GPO value.

77.4.7 HDP register map

Table 854. HDP register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	HDP_CTRL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN
	Reset value																																0
0x004	HDP_MUX	MUX7[3:0]				MUX6[3:0]				MUX5[3:0]				MUX4[3:0]				MUX3[3:0]				MUX2[3:0]				MUX1[3:0]				MUX0[3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008 - 0x00C	Reserved	Res.erved																															
0x010	HDP_VAL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPVAL[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

Table 854. HDP register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x014	HDP_GPOSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPGPOSET[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x018	HDP_GPOCLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPGPOCLR[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x01C	HDP_GPOVAL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDPGPOVAL[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

Refer to *Section 2.3* for the register boundary addresses.

78 Debug support (DBG)

78.1 DBG introduction

A comprehensive set of debug features is provided to support software development and system integration:

- Independent breakpoint debugging of CPU core in the system
- Code execution tracing
- Software instrumentation
- Cross-triggering

The debug features can be controlled via a JTAG/serial-wire debug access port (SWJ-DP), using industry standard debugging tools.

A trace port allows data to be captured for logging and analysis.

The debug features are based on the following Arm® CoreSight™ components:

- SWJ-DP: JTAG/serial-wire debug port
- AXI-AP: AXI access port
- AHB-AP: AHB access port
- APB-AP: APB access port
- ITM: instrumentation trace macrocell
- DWT: data watchpoint and trace
- ETM: Embedded Trace Macrocell
- ETF: embedded trace FIFO
- ETR: embedded trace router
- TPIU: trace port interface unit
- SWO: serial-wire output
- CTI: cross-trigger interface
- CTM: cross-trigger matrix
- Timestamp generator
- STM: system trace macrocell

These components are described in *Section 78.11* and *Section 78.12*. More information can be found in the Arm documents referenced in *Section 78.13*.

Debug use cases

The trace and debug system are designed to support a variety of typical use cases:

- Low-cost software trace

  A limited trace capability is available over the single-wire debug output. This supports code instrumentation using “printf”, tracing of data and address watchpoints, interrupt detection and program counter sampling.
- External debugging

  The software can be debugged using equipment connected to the JTAG/SWD debug port. This allows breakpoint and watchpoint settings, code stepping, and memory access.

- Self-hosted debugging

  Debug resources can be accessed by software running on the processor. This allows autonomous fault analysis, and communication with external tools via a functional interface (such as USB or UART). Halting the CPU is not possible.
- Tracing code execution from via the trace port

  The trace information is sent to a trace port analyzer in real time. An ID embedded in the trace allows the analyzer to identify the source of each information packet.
- Code and hardware instrumentation via the trace port

  The software data from the processor, memory content via DMA, and hardware event trace information can be generated using the STM. The information is combined with the processor trace stream, and output to a trace port analyzer in real time.
- Capturing trace continuously into the ETF programmed as a circular buffer (ETB)

  The combined trace information can be stored on-chip in a private circular buffer. The trace storage can be started and stopped by a debugger command, a software command, an external trigger signal, or an internal event.
- Capturing trace continuously into a system memory

  Thanks to the ETR, the combined trace information can be sent to a system memory. The trace storage can be started and stopped by a debugger command, a software command, an external trigger signal, or an internal event.
- Reading the stored trace with the debugger

  The debugger can read the contents of the recorded trace data via the debug port. This is slower than the trace port, but allows basic trace functionality on the debugger without the cost of a trace port analyzer.
- Self-hosted trace analysis

  The recorded trace data can be read by the Cortex-M55. This powerful feature allows built-in test software to monitor code execution in real time, analyze and identify faults, and autonomously handle exceptions.
- Uploading stored trace

  The stored trace can also be uploaded to a host machine using one of the communication interfaces (such as USB, USART, SPI, I2C, Ethernet, or CAN). This is especially useful if the trace port is not accessible, for example remote monitoring and failure analysis of a deployed product.

78.2 DBG functional description

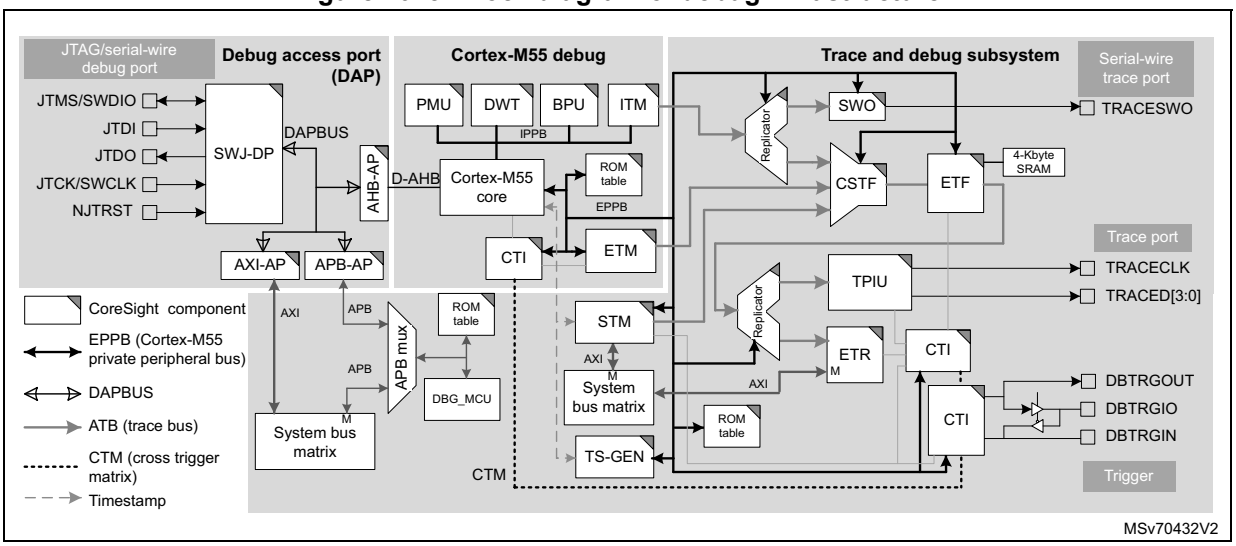
The debug infrastructure is composed of three subsystems:

- The Cortex-M55 incorporates the following CoreSight components:
  - SCS (system control space): contains registers used to identify and halt the processor. It accesses core data registers, and other debug features included in the processor core.
  - DWT: generates trace or debug events on access to specific addresses
  - BPU (breakpoint unit): allows the processor to be halted at specific addresses
  - ITM: generates software instrumentation trace, and adds timestamp, synchronization, and DWT event trace packets
  - ETM: generates non-intrusive program flow trace
  - PMU (performance monitoring unit): counts events for performance analysis

  - CTI: allows cross-triggering between different trace and debug components
  - processor ROM table: allows a debugger to discover CoreSight components in the Cortex-M55
- The trace and debug subsystem includes the following CoreSight components:
  - ITM replicator: routes the ITM trace to both SWO and trace funnel
  - SWO: provides a single-wire asynchronous trace port for low-bandwidth trace coming from the ITM
  - CSTF (CoreSight trace funnel): merges trace from STM, ITM, and ETM
  - ETF: smooths the trace stream by buffering in a local memory
  - ETR replicator: routes trace from the CSTF to both ETR and TPIU
  - ETR: routes trace to a buffer in the system RAM
  - TPIU: provides a parallel trace port up to 16-bit wide for high-bandwidth trace coming from both ITM and ETM
  - CTI: allows cross triggering between different trace and debug components
  - system ROM table: allows a debugger to discover CoreSight components in the trace and debug subsystem
  - DBGMCU (MCU debug unit): controls the debug authentication, the low-power mode emulation, and freezes certain peripherals in debug mode and other features
  - STM: allows AXI bus masters to write trace data
- The DAP (debug access port) allows an external debugger to access the debug and trace functionality. It contains the following CoreSight components:
  - SWJ-DP: implements a JTAG interface as well as a serial-wire debug (SWD) interface together with debug port (DP) registers
  - AHB-AP: AHB bus master that provides access to the trace and debug components in the Cortex-M55
  - APB-AP: APB bus master that provides access to the trace and debug subsystem

78.3 DBG block diagram

Figure 1075. Block diagram of debug infrastructure



78.4 DBG pins

Table 855. SWJ-DP pins

Pin name	JTAG debug port		SWD debug port		Pin assignment
	Type	Description	Type	Description	
JTMS/SWDIO	I	JTAG test mode select	I/O	Serial-wire data in/out	Refer to the product datasheet
JTCK/SWCLK	I	JTAG test clock	I	Serial-wire clock	
JTDI	I	JTAG test data input	-	-	
JTDO	O	JTAG test data output	-	-	
NJTRST	I	JTAG test reset	-	-	

Table 856. Trace port pins

Pin name	Type	Description	Pin assignment
TRACED[3:0]	O	Trace synchronous data out (can be 1, 2, or 4 pins)	Refer to the product datasheet
TRACECLK	O	Trace clock	

Table 857. Serial-wire trace port pins

Pin name	Type	Description	Pin assignment
TRACESWO	O	Trace asynchronous data out	Refer to the product datasheet(1)

1. TRACESWO is multiplexed with JTDO. This means that single-wire trace is only available when using the JTAG port in serial-wire debug interface mode, not when using JTAG.

Table 858. Trigger pins

Pin name	Type	Description	Pin assignment
DBTRGIN	I	External trigger input to CTI	Refer to the product datasheet
DBTRGOUT	O	External trigger output from CTI	
DBTRGIO	I/O	External bi-directional trigger(1)	

1. DBTRGIO can be configured as an input or an output by DBTRGOEN in DBGMCU_CR. If configured as an input, it is connected to DBTRGIN. If configured as an output, it is connected to DBTRGOUT. This is because DBTRGIN and DBTRGOUT may not be available on some packages.

78.5 DBG power, clock, and reset

78.5.1 DBG power domains

Debug components are located in the core power domain. This means that a debug session is disconnected if the core is powered down (for example in Standby mode).

Debug and low-power modes

The device includes power saving features that allow the core power domain to be switched off, or stopped when not required. If the power is switched off, or the core is not clocked, all debug components are inaccessible to the debugger. To avoid this, a power-saving mode emulation is implemented.

If emulation is enabled for a domain, the domain still enters power-saving mode, but its clock and power are maintained: the domain behaves as in power-saving mode, but the debugger does not lose the connection.

This emulation mode is programmed in the DBGMCU (see *Section 78.12.20*).

78.5.2 DBG clocks

The debugger supplies the clock for the debug port (swtclk) via the debug interface pin JTCK/SWCLK. This clock is used to register the serial input data in both serial-wire and JTAG modes, as well as to operate state machines and internal logic of the debug port. This clock must therefore continue to toggle for at least five cycles after the end of an access, to ensure that the debug port returns to the idle state.

The SWJ-DP contains an asynchronous interface to the system debug clock domain (ck_sys_dbg). All debug and trace components are in this same clock domain.

The ck_sys_dbg clock is enabled by the debugger using CDBGPWRUPREQ in DP_CTRLSTAT. It can also be enabled by software setting DBGCLKEN in DBGMCU_CR. This clock must be enabled before the debugger or software can access any of the debug features on the device. Disable this clock when debug and trace are not used, to reduce power consumption.

The TPIU output clock (ck_trace, also called traceclkin) is generated by the RCC. Its frequency can be programmed to match the I/O port capability. This clock is also used to drive the SWO. It can only be enabled by setting TRACECLKEN in DBGMCU_CR.

Note: ck_trace must be activated whenever tracing is required, even if neither the TPIU nor the SWO are used (for example, when using the ETR as a software buffer). This avoids

filling up the SWO FIFO, which may then block the trace output from the ITM in the Cortex-M55.

The debug and trace components included in the processor are clocked with the processor clock.

78.5.3 Debug reset

The SWJ-DP is reset by a power-on reset (POR), and when waking up from Standby mode.

The debug and trace components, including the Cortex-M55 debug part, can be reset in three different ways:

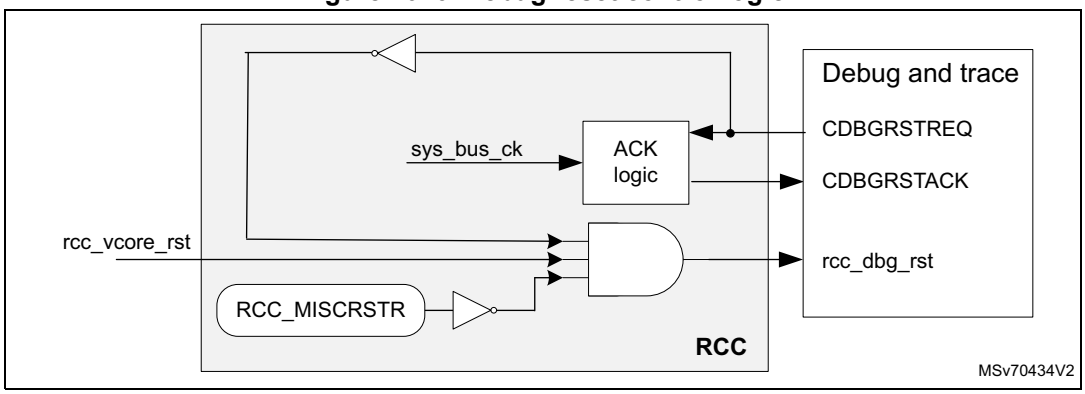
- Over the DAP by setting CDBGRSTREQ in DP_CTRLSTAT

  Setting this bit asserts the debug reset request signal (cdbgrstreq) connected to the RCC. The RCC then activates rcc_dbg_rst, and returns a handshake signal (cdbgrstack) to acknowledge the DAP request. This acknowledge can be read by the debugger on CDBGRSTACK in DP_CTRLSTAT. The rcc_dbg_rst is active as long as CDBGRSTREQ is set.
- By the application setting DBGRST in RCC_MISCRSTR

  When this bit is set, the rcc_dbg_rst is activated. It is released when DBGRST is cleared.
- When a POR occurs

  The rcc_dbg_rst is activated after a power-on reset, or when the product exits Standby mode.

Figure 1076. Debug reset control logic



The rcc_dbg_rst can be applied only when the processor is in a quiescent state (when all system transactions have been completed, or when the processor is in debug mode).

78.6 Security

The trace and debug components allow a high degree of access to the processor, and system during product development. To protect user code, and to ensure that debug features cannot be used to alter or compromise the normal operation of the finished product, these features can be disabled, or limited in scope. For example, a secure software debug and trace can be disabled without preventing the debug of non-secure code.

78.6.1 BSEC control over debug

The BSEC features the following registers dedicated to debug:

- BSEC_AP_UNLOCK register

  It can be used to open the debug access port to the Cortex-M55 (AP1) when a specific value is written in it. This register drives ap_unlocked[7:0] signals toward the debug logic that decodes these signals to open the access port if the expected value is met.

- BSEC_DBGCR register

  It can be used to enable the non-secure/secure debug when a specific value is written in it. This register drives dbg_unlocked_sec[7:0] and dbg_unlocked[7:0] signals toward the debug logic that decodes these signals to determine the values of the authentication signals (see the next section), enabling or not the debug for full secure or only non-secure world.

The BSEC also defines the HDPL (hide protection level) from which the debug is enabled. Its hardware ensures that dbg_unlocked and dbg_unlocked_sec automatically adopt the specific values as soon as the appropriate HDPL is reached.

On an open device, these registers are automatically set to the values that open the debug (refer to *Section 4: Boot and security control (BSEC)* for details on these registers.)

78.6.2 Authentication signals

The following authentication signals are used by a CoreSight system to determine which features are enabled or disabled. These signals are derived from the BSEC_DBGCR register value. This applies not only when the debug authentication sequence is executed, but also during the regular debug in open, closed, or closed secure states (refer to *Section 3: System security* for details on these states):

- **dbgen**: global enable for all debug and trace features

  0: All debug features are disabled.

  1: Debug features in non-secure mode are enabled. Debug features in secure mode depend on the spiden state.

- **spiden**: enables debug in secure privileged mode when dbgen = 1.

  0: Debug features are disabled in secure privileged mode. External access to secure system interconnect is disabled.

  1: Debug features are enabled in secure privileged mode. External access to secure system interconnect is enabled.

- **niden**: enables trace and performance monitoring (non invasive debug).

  0: Trace generation is disabled.

  1: Trace generation in non-secure mode is enabled. Trace generation in non-secure mode depends on the spiden state.

- **spniden**: enables trace and performance monitoring in secure privileged mode when niden = 1.

  0: Trace generation is disabled in secure privileged mode

  1: Trace generation is enabled in secure privileged mode

For detailed information on the Arm Coresight debug infrastructure, refer to the relevant component section, or to the relevant Arm technical documentation, which describes the dbgen, niden, spiden and spniden signals. When the BSEC_DBGCR register is configured

to enable debug (either secure or non-secure), the dbgen and niden signals are set. The spiden and spniden signals are only set if secure debug is enabled.

The initial states of the signals are set at POR, according to *Table 859*. Signal states can be modified through an authentication process, by a trusted software with secure and privileged access rights, in collaboration with the debugger (see *Section 78.7* for more details on the authentication process).

Table 859. Authentication signal initial states

Device state	Authentication signal default state	Description
Open or Closed_Unlocked DEV BOOT	All 1	All debug and trace enabled at boot-up.
Closed	All 0	All debug and trace disabled at boot-up. Secure software must enable debug as required after booting.

78.7 Debug authentication

78.7.1 Main characteristics

The debug authentication relies on the fact that, even when debug is disabled, there is always an access port available (the APB-AP, AP0) to support communication between the host (debugger) and the device (STM32 product).

Debug rights (controlled by BSEC) are reset by a POR.

The debug authentication starts only at boot (HDPL1). This implies that a debugger hot plug is not possible on a closed part.

In BSEC-open state, the debugger can configure debug peripherals, and has full access to the system without the need for an authentication (but not to the fuse secrets, and unmapped ROM code).

During bootROM execution (HDPL0), the debug is automatically disabled via hardware.

When the debug is opened via a certificate, the debugger can proceed with debug infrastructure configuration. The debug certificate contains debugger access rights (secure, non-secure, HDPL).

When the debug is disabled, the debugger can still proceed with debug infrastructure ‘discovery’, but can only see the authentication mailbox. Other debug and trace features are disabled.

78.7.2 Debug authentication protocol

Principle

The protocol implements a challenge response mechanism based on asymmetric cryptography to authenticate the host. It relies on a key pair, with a public key stored in the device, and a private key from the host library, used to sign a random value (the challenge) generated by the device.

The protocol implements a bidirectional communication between the debugger (host) and the SoC (device) through a mailbox interface located in the DBGMCU.

The host can write to the mailbox via the JTAG/SWD interface. It expects to get responses and messages from the device via the same mailbox.

The debug authentication protocol is launched on a POR of the device, when an “open request” message is posted by the host. The protocol is based on the following steps:

1. Initial message: posted by the host, combined with a reset to launch the debug authentication process on the device
2. Challenge message: The device generates a random value, to be signed by the host, when sending back the response.
3. Response: The host sends a message to the device proving its authenticity. This is done using a tool to generate a token.

After a first sequence of mutual authentication to align (on protocol version, or device type for example), the device generates a random value that must be signed by the host with a private key when building the response.

The device verifies the signed response including the random value using a public key embedded in the device.

Status mailbox

A mailbox status information is provided to inform host or device that a message is available for reading. This is especially useful when one side sends a message larger than 32 bits. For instance, when the device sends such a long message, it checks that the host has read each 32-bit element before writing a new one by polling the clear of DEVICE_ACK.

The DBGMCU_DBG_AUTH_ACK register is used for that purpose. This register is read only for the host and the device:

- HOST_ACK: this bit is set when the host writes a message in DBGMCU_DBG_AUTH_HOST. It indicates that a new message is present in the mailbox. This bit is reset when the device reads DBGMCU_DBG_AUTH_HOST. The device can poll this bit to know when the host has written a message.
- DEVICE_ACK: this bit is set when the device writes a message in DBGMCU_DBG_AUTH_DEV, and reset when the host reads it. The host can poll this bit to know when the device has written a new message.

78.8 Chip-level TAP controller (CLTAPC)

The JTAG interface is connected to the internal scan chain of the device. It is used for testing as well as for debug. The first component in the scan chain is the chip-level TAP controller, which implements a TAP state machine based on IEEE Std 1149.1-1990. It allows access to the built-in manufacturing test features.

The CLTAPC must be bypassed in order to access the debug support functions provided by the SWJ-DP. This is done by prefixing SWJ-DP instructions with the CLTAPC bypass instruction sequence (0b11111).

78.9 Serial-wire and JTAG debug port (SWJ-DP)

The SWJ-DP is a CoreSight component that implements an external access port for connecting debugging equipment.

Two types of interface can be configured:

- 5-pin standard JTAG interface (JTAG-DP)
- 2-pin (clock + data) serial-wire debug port (SW-DP)

The two modes are mutually exclusive, since they share the same I/O pins.

By default, the JTAG-DP is selected after a system or POR. The five I/O pins are configured by hardware in debug alternative function mode. The SWJ-DP incorporates pull-up resistors on JTDI, JTMS/SWDIO, and NJTRST, as well as a pull-down resistor on JTCK/SWCLK.

A debugger can select the SW-DP by transmitting the following serial data sequence on JTMS/SWDIO:

...(50 or more ones)...,0,1,1,1,1,0,0,1,1,1,1,0,0,1,1,1,...(50 or more ones)...

JTCK/SWCLK must be cycled for each data bit.

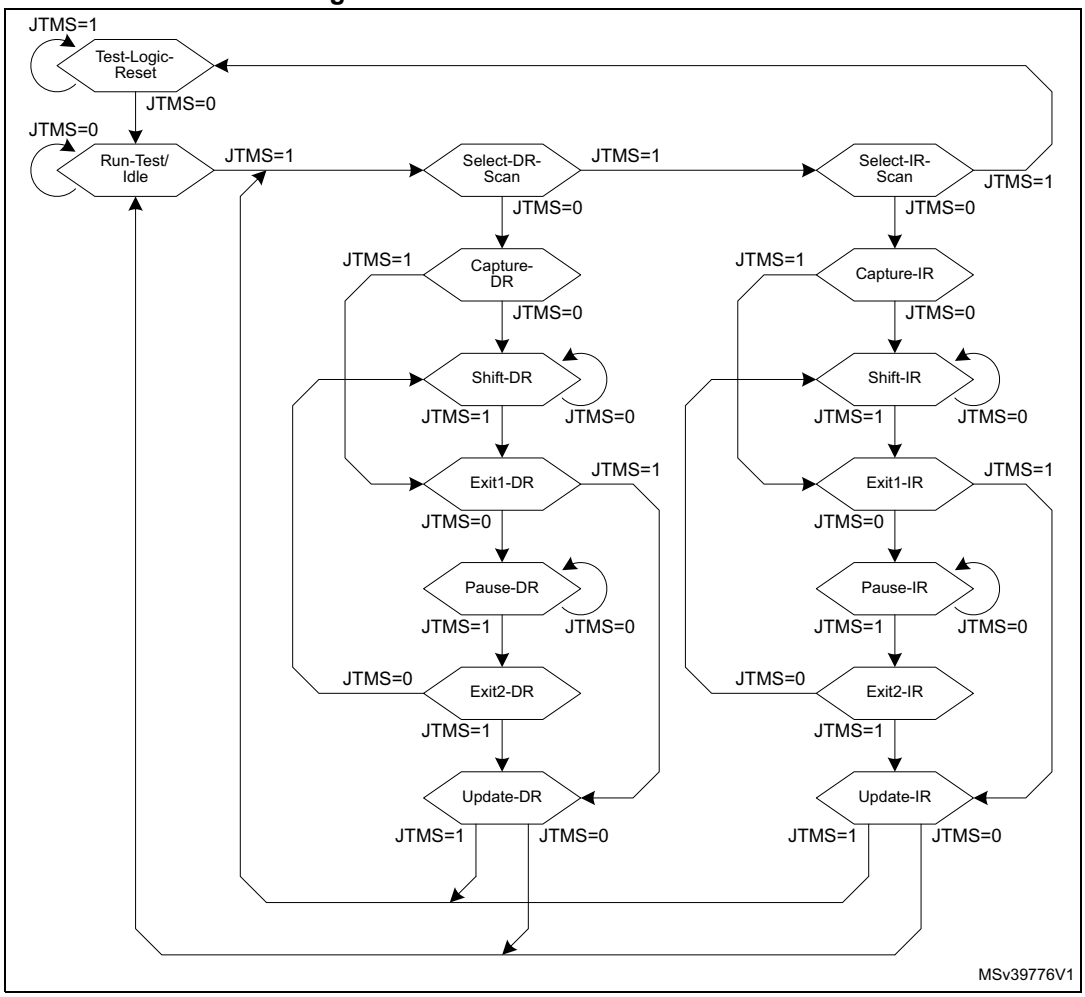
In SW-DP mode, the unused JTAG pins (JTDI, JTDO, and NJTRST) can be used for other functions.

Note: All SWJ port I/Os can be reconfigured to other functions by software, but debugging is no longer possible.

78.9.1 JTAG debug port

The JTAG-DP implements a TAP state machine (TAPSM) based on IEEE Std 1149.1-1990, and shown in *Figure 1077*. It controls two scan chains: one associated with an instruction register (IR), and one with a number of data registers (DR).

Figure 1077. JTAG TAP state machine



When the TAPSM goes through the Capture-IR state, 0b0001 is transferred into the IR scan chain connected between JTDI and JTDO.

While the TAPSM is in the Shift-IR state, the IR scan chain shifts one bit for each JTCK rising edge. This means that on the first tick:

- The LSB of the IR scan chain is output on JTDO.
- Bit [n] of the IR scan chain is transferred to bit [n-1].
- The value on JTDI is transferred to the MSB of the IR scan chain.

When the TAPSM goes through the Update-IR state, the value scanned in the IR scan chain is transferred to the instruction register.

When the TAPSM goes through the Capture-DR state, a value is transferred from one of DR to one of the DR scan chains, connected between JTDI and JTDO.

The value held in the instruction register determines which DR and associated DR scan chain are selected.

Data are then shifted while the TAPSM is in the Shift-DR state, in the same way as the IR shift in the Shift-IR state.

When the TAPSM goes through the Update-DR state, the value scanned in the DR scan chain is transferred to the selected DR.

When the TAPSM is in the Run-Test/Idle state, no special actions occur. The IDCODE instruction is loaded in IR.

When active, the NJTRST signal resets the state machine asynchronously to the test-logic-reset state.

The table below lists DRs corresponding to the 4-bit IR instructions.

Table 860. JTAG-DP data registers

Instruction register	Data register	Scan chain length	Description
0000 to 0111	(BYPASS)	1	Not implemented: BYPASS selected
1000	ABORT	35	**Abort register** – Bits 34:1 = reserved – Bit 0 = APABORT: write 1 to generate an AP abort
1001	(BYPASS)	1	Reserved: BYPASS selected
1010	DPACC	35	**Debug port access register** Initiates the debug port and allows access to a debug port register. – When transferring data IN: Bits 34:3 = DATA[31:0] = 32-bit data to transfer for a write request Bits 2:1 = A[3:2] = 2-bit address of a debug port register. Bit 0 = RnW = read request (1) or write request (0). – When transferring data OUT: Bits 34:3 = DATA[31:0] = 32-bit data which is read following a read request Bits 2:0 = ACK[2:0] = 3-bit acknowledge: 0b010= OK/fault 0b00 = Wait Other = reserved
1011	APACC	35	**Access port access register** Initiates an access port and allows access to an access port register. – When transferring data IN: Bits 34:3 = DATA[31:0] = 32-bit data to shift in for a write request Bits 2:1 = A[3:2] = 2-bit sub-address of an access port register. Bit 0 = RnW= read request (1) or write request (0). – When transferring data OUT: Bits 34:3 = DATA[31:0] = 32-bit data which is read following a read request Bits 2:0 = ACK[2:0] = 3-bit acknowledge: 010b = OK/Fault 001b = Wait Other = reserved
1100	(BYPASS)	1	Reserved: BYPASS selected
1101	(BYPASS)	1	Reserved: BYPASS selected

Table 860. JTAG-DP data registers (continued)

Instruction register	Data register	Scan chain length	Description
1110	IDCODE	32	**ID code** 0x6BA00477: Arm JTAG debug port ID code
1111	BYPASS	1	**Bypass** A single JTCK cycle delay is inserted between JTDI and JTDO

For more details on DR registers and the JTAG debug protocol, refer to the Arm Debug Interface Architecture Specification *[1]*.

78.9.2 SWD debug port

The serial-wire debug protocol uses the two following pins:

- SWCLK: clock from the host (debugger) to the device (STM32 product)
- SWDIO: bidirectional serial data (with internal pull-up)

Serial data are transferred LSB first, synchronously with the clock. A transfer comprises the following phases:

1. packet request (8 bits) transmitted by the host
2. acknowledge response (3 bits) transmitted by the target
3. data transfer (33 bits) transmitted by the host (for a write) or the device (for a read)

The data transfer only occurs if the acknowledge response is OK.

If the direction of the data is reversed between each phase, a single clock cycle turn-around time is inserted.

Table 861. Packet request

Bit(s)	Name	Description
0	Start	Must be 1
1	APnDP	0: DP register access (see *Table 864* for a list of DP registers 1: AP register access (see *Section 78.10*)
2	RnW	0: Write request 1: Read request
4:3	A(3:2)	Address field of DP or AP register (refer to *Table 864* and *Section 78.10*)
5	Parity	Single bit parity of preceding bits
6	Stop	0
7	Park	Not driven by host. Must be read as 1 by the device.

Table 862. ACK response

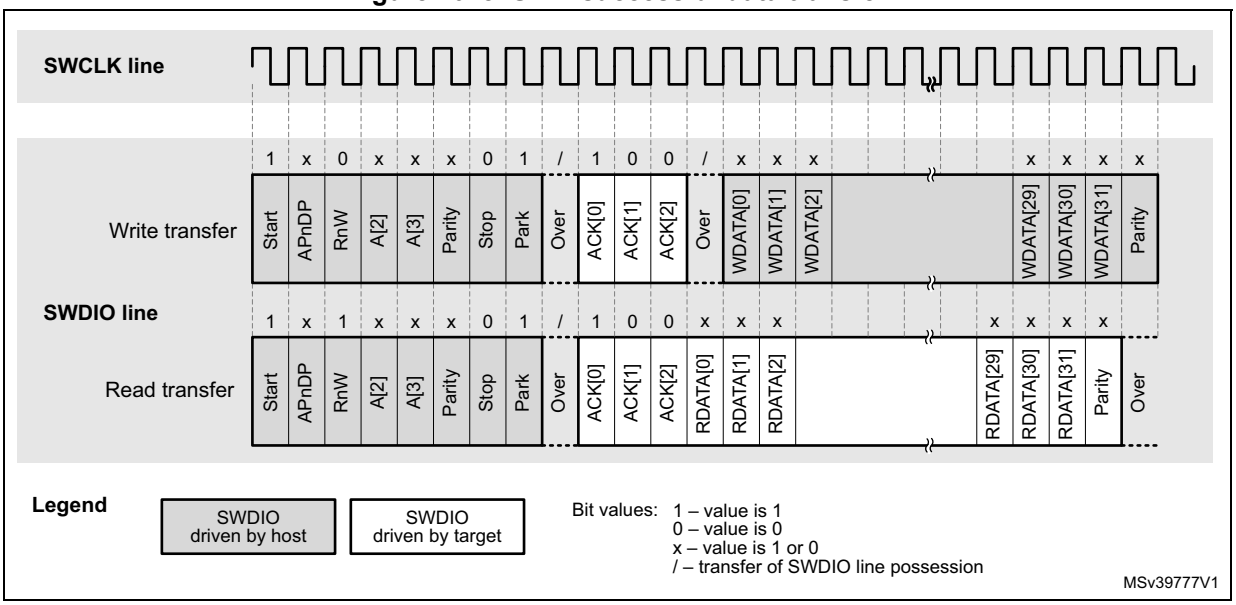
Bitfield	Name	Description
2:0	ACK	0b000: Fault 0b010: Wait 0b100: OK

Table 863. Data transfer

Bitfield	Name	Description
31:0	WDATA or RDATA	Write or read data
32	Parity	Single bit parity of 32 data bits

Figure 1078 shows successful write and read transfers.

Figure 1078. SWD successful data transfer



For any FAULT or WAIT ACK response from the device, the data transfer phase is canceled, unless overrun detection is enabled. If and overrun detection is enabled, data are ignored by the device (for a write), or not driven (for a read).

A line reset must be generated by the host when it is first connected, or following a protocol error. The line reset consists of 50 or more SWCLK cycles with SWDIO high, followed by two SWCLK cycles with SWDIO low.

For more details on the serial-wire debug protocol, refer to the Arm Debug Interface Architecture Specification *[1]*.

Note: The SWJ-DP implements SWD protocol version 2.

78.9.3 Debug port registers

The SW-DP and JTAG-DP access the debug port (DP) registers, listed in *Table 864*.

The debugger can access DP registers as follows:

1. Program DPBANKSEL in DP_SELECT to select the register bank to be accessed (see *Table 864*).
2. If using JTAG, program A[3:2] in the DPACC register with the register address within the bank. Program the R/W bit to select a read or a write. For a write, program the DATA field with the write data.

   If using SWD, the A[3:2] and R/W fields are part of the packet request word sent to the SW-DP with the APnDP bit reset (see *Table 861*). The write data are sent in the data phase.

Table 864. Debug port registers

Address	A[3:2] value	R/W	Description
0x0	00	R	DP_DPIDR register(2). Contains the IDCODE for the debug port.
		W	DP_ABORT register(1). Aborts the current AP transaction. This register is also used to clear the error flags in DP_CTRL/STAT.
0x4	01	R/W	If DPBANKSEL[3:0] = 0x0 in DP_SELECT: DP_CTRLSTAT register. Controls the DP and provides status information.
			If DPBANKSEL[3:0] = 0x1 in DP_SELECT: DP_DLCR register(2). Controls the operating mode of the SWD data link.
			If DPBANKSEL[3:0] = 0x2 in DP_SELECT: DP_TARGETID register. Provides target identification information.
			If DPBANKSEL[3:0] = 0x3 in DP_SELECT: DLPIDR register(2). Provides the SWD protocol version.
0x8	10	R	RESEND register(2). Returns the value that was returned by the last AP read or DP_RDBUFF read, used in the event of a corrupted read transfer.
		W	DP_SELECT register. Selects the access port, access port register bank, and DP register at address 0x4.
0xC	11	R	DP_RDBUFF register – Via JTAG-DP, enables the debugger to get the final result after a sequence of operations (without requesting new JTAG-DP operation). – Via SW-DP, contains the result of the preceding AP read access, allowing a new AP access to be avoided.
		W	DP_TARGETSEL register(2). On a write to DP_TARGETSEL immediately following a line reset sequence, the target is selected if the following conditions are both met: – Bits [31:28] match bits [31:28] in DP_DLPIDR. – Bits [27:0] match bits [27:0] in DP_TARGETID. Writing any other value deselects the target. Debug tools must write 0xFFFFFFFF to deselect all targets. This is an invalid DP_TARGETID value. All other invalid DP_TARGETID values are reserved.

1. Access to the AP ABORT register from the JTAG-DP is done using the ABORT instruction.
2. Only accessible via SW-DP. Register is “reserved” via JTAG-DP.

Debug port identification register (DP_DPIDR)

Address offset: 0x0

Reset value: 0x6BA0 2477

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REVISION[3:0]				PARTNO[7:0]								Res.	Res.	Res.	MIN
r	r	r	r	r	r	r	r	r	r	r	r				r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VERSION[3:0]				DESIGNER[10:0]											Res.
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	

Bits 31:28 **REVISION[3:0]**: Revision code
0x6

Bits 27:20 **PARTNO[7:0]**: Debug port part number
0xBA

Bits 19:17 Reserved, must be kept at reset value.

Bit 16 **MIN**: Minimal debug port (MINDP) implementation
0: MINDP not implemented (transaction counter and pushed operations are supported)

Bits 15:12 **VERSION[3:0]**: DP architecture version
0x2: DPv2

Bits 11:1 **DESIGNER[10:0]**: JEDEC designer identity code
0x23B: Arm

Bit 0 Reserved, must be kept at reset value.

Debug port abort register (DP_ABORT)

Address offset: 0x0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ORUN ERRCLR	WDER RCLR	STKER RCLR	STKCM PCLR	DAP ABORT
											w	w	w	w	w

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **ORUNERRCLR**: Overrun error clear bit
0: No effect
1: Clear STICKYORUN in DP_CTRLSTAT.

Bit 3 **WDERRCLR**: Write data error clear bit
0: No effect
1: Clear WDATAERR in DP_CTRLSTAT.

Bit 2 **STKERRCLR**: Sticky error clear bit
0: No effect
1: Clear STICKYERR in DP_CTRLSTAT.

Bit 1 **STKCMPCLR**: Sticky compare clear bit
0: No effect
1: Clear STICKYCMP in DP_CTRLSTAT.

Bit 0 **DAPABORT**: Abort current AP transaction
The transaction is aborted if an excessive number of WAIT responses are returned. This indicates that the transaction has stalled.
0: No effect
1: Abort the transaction.

Debug port control/status register (DP_CTRLSTAT)

Address offset: 0x4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CSYSPWRUPACK	CSYSPWRUPREQ	CDBGPWRUPACK	CDBGPWRUPREQ	CDBGRSTACK	CDBGRSTREQ	Res.	Res.	TRNCNT[11:4]							
r	rw	r	rw	r	rw			rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRNCNT[3:0]				MASKLANE[3:0]				WDATAERR	READOK	STICKYERR	STICKYCMP	TRNMODE[1:0]		STICKYORUN	ORUNDETECT
rw	rw	rw	rw	rw	rw	rw	rw	r	r	rw	w	w	w	w	w

Bit 31 **CSYSPWRUPACK**: System domain power-up status bit - not used in this device

Bit 30 **CSYSPWRUPREQ**: System domain power-up control bit - not used in this device

Bit 29 **CDBGPWRUPACK**: Debug domain power-up status bit
This bit is read-only. It returns the status of the debug domain power-up acknowledge signal from the power controller.
0: domain powered down
1: domain powered up

Bit 28 **CDBGPWRUPREQ:** Debug domain power-up/down control bit
This bit controls the debug domain power-up/down request signal to the power controller.
0: power-down requested
1: power-up requested

Bit 27 **CDBGRSTACK**: Debug domain reset status bit - not used in this device

Bit 26 **CDBGRSTREQ**: Debug domain reset control bit - not used in this device

Bits 25:24 Reserved, must be kept at reset value.

Bits 23:12 **TRNCNT[11:0]**: Transaction counter
To program a sequence of transactions to incremental addresses via an AP, this field is loaded with the number of transactions to perform. It is decremented on successful completion of each transaction.

Bits 11:8 **MASKLANE[3:0]**: Pushed-compare and pushed-verify masking bits

The field indicates the bytes to be masked in pushed-compare and pushed-verify operations (TRNMODE = 1 or 2 in this register). In the pushed operations, the word supplied in an AP write transaction is compared with the current value at the target AP address.

1xxx: Include byte lane 3 in comparisons.

x1xx: Include byte lane 2 in comparisons.

xx1x: Include byte lane 1 in comparisons.

xxx1: Include byte lane 0 in comparisons.

Bit 7 **WDATAERR**: Write data error in SW-DP

The bit indicates one of the following:

- a parity or a framing error on the data phase of a write operation
- a write operation that had been accepted by the DP, and then discarded without being submitted to the AP

This bit is read-only. It is reset by writing 1 to WDERRCLR in DP_ABORT.

0: No error

1: Error has occurred.

Note: This bit is reserved in JTAG-DP.

Bit 6 **READOK**: AP read response in SW-DP

This bit indicates the response to the last AP read access. It is read-only.

0: Read not ok

1: Read ok

Note: This bit is reserved in JTAG-DP.

Bit 5 **STICKYERR**: Transaction error (read-only in SW-DP, R/W in JTAG-DP)

This bit indicates that an error occurred during an AP transaction. In SW-DP, this bit is reset by writing 1 to STKERRCLR in DP_ABORT. In JTAG-DP, this bit is reset by programming it to 1.

0: No error

1: Error has occurred.

Bit 4 **STICKYCMP**: Compare match (read-only in SW-DP, R/W in JTAG-DP)

This bit indicates that a match occurred in a pushed operation. In SW-DP, this bit is reset by writing 1 to STKCMPCLR in DP_ABORT. In JTAG-DP, this bit is reset by programming it to 1.

0: Match if TRNMODE = 0x1; no match if TRNMODE = 0x2

1: No match if TRNMODE = 0x1; match if TRNMODE = 0x2

Bits 3:2 **TRNMODE[1:0]**: Transfer mode for AP write operations

For read operations, this field must be set to 0x0.

0x0: Normal operation - AP transactions are passed directly to the AP.

0x1: Pushed-verify operation. The DP stores the write data and performs a read transaction at the target AP address. The result of the read operation is compared with the stored data. If they do not match, the STICKYCMP bit is set in this register.

0x2: Pushed-compare operation. The DP stores the write data and performs a read transaction at the target AP address. The result of the read is compared with the stored data. If they match, the STICKYCMP bit is set.

0x3: Reserved

Note: In pushed operations, only the data bytes indicated by the MASKLANE field are included in the comparison.

Bit 1 **STICKYORUN:** Overrun (read-only in SW-DP, R/W in JTAG-DP)

This bit indicates that an overrun occurred (a new transaction received before previous transaction completed). This bit is only set if ORUNDETECT is set in this register. In SW-DP, this bit is reset by writing 1 to ORUNERRCLR in DP_ABORT. In JTAG-DP, this bit is reset by writing a 1 to it.

0: No overrun

1: Overrun occurred.

Bit 0 **ORUNDETECT:** Overrun detection mode enable

0: Overrun detection disabled

1: Overrun detection enabled. In the event of an overrun, the STICKYORUN bit is set and subsequent transactions are blocked until STICKYORUN is cleared.

Debug port data link control register (DP_DLCR)

Address offset: 0x4

Reset value: 0x0000 0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	TURNROUND[1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
						rw	rw								

Bits 31:10 Reserved, must be kept at reset value.

Bits 9:8 **TURNROUND[1:0]**: Tristate period for SWDIO

0x0: 1 data bit period

0x1: 2 data bit periods

0x2: 3 data bit periods

0x3: 4 data bit periods

Bits 7:0 Reserved, must be kept at reset value.

Debug port target identification register (DP_TARGETID)

Address offset: 0x4

Reset value: 0x0486 0041

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TREVISION[3:0]				TPARTNO[15:4]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TPARTNO[3:0]				TDESIGNER[10:0]											Res.
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	

Bits 31:28 **TREVISION[3:0]**: Device revision number

0x0: revision 1

Bits 27:12 **TPARTNO[15:0]**: Target part number
0x4860: STM32N6x5/x7xx

Bits 11:1 **TDESIGNER[10:0]**: Target designer JEDEC code
0x020: STMicroelectronics

Bit 0 Reserved, must be kept at reset value.

Debug port data link protocol identification register (DP_DLPIDR)

Address offset: 0x4

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TINSTANCE[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r												
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PROTSVN[3:0]			
												r	r	r	r

Bits 31:28 **TINSTANCE[3:0]**: Target instance number
These bits define the instance number for this device in a multi-drop system: 0x0.

Bits 27:4 Reserved, must be kept at reset value.

Bits 3:0 **PROTSVN[3:0]**: Serial--wire debug protocol version
0x1: Version 2

Debug port resend register (DP_RESEND)

Address offset: 0x8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESEND[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESEND[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RESEND[31:0]**: Last AP read or DP RDBUFF read value
These bits contain the value that was returned by the last AP read or DP RDBUFF read. Used in the event of a corrupted read transfer.

Debug port access port select register (DP_SELECT)

Address offset: 0x8

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
APSEL[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
w	w	w	w	w	w	w	w								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APBANKSEL[3:0]				DPBANKSEL[3:0]			
								w	w	w	w	w	w	w	w

Bits 31:24 **APSEL[7:0]**: Access port select bits

These bits select the access port for the next transaction.
0x00: AP0 - System debug access port (APB-AP)
0x01: AP1 - Cortex-M55 debug access port (AHB-AP)
Others: reserved

Bits 23:8 Reserved, must be kept at reset value.

Bits 7:4 **APBANKSEL[3:0]**: AP register bank select bits

These bits select the 4-word register bank on the active AP for the next transaction.

Bits 3:0 **DPBANKSEL[3:0]**: DP register bank select bits

These bits select the register at address 0x4 of the debug port.
0x0: DP_CTRLSTAT register
0x1: DP_DLCR register
0x2: DP_TARGETID register
0x3: DP_DLPIDR register
Others: reserved

Debug port read buffer register (DP_RDBUFF)

Address offset: 0xC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RDBUFF[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RDBUFF[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RDBUFF[31:0]**: Last AP read value

The field contains the value returned by the last AP read access. There are two ways to retrieve the value returned by an AP read access:

- Perform a second read access to the same address, which initiates a new transaction on the corresponding bus./
- Read the value returned by the last AP read access from the DP_RDBUFF register, in which case no new AP transaction occurs.

Debug port target identification register (DP_TARGETSEL)

Address offset: 0xC

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TINSTANCE[3:0]				TPARTNO[15:4]											
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TPARTNO[3:0]				TDESIGNER[10:0]											Res.
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	

Bits 31:28 **TINSTANCE[3:0]**: Target instance number

This field defines the instance number for the target device in a multi-drop system. It must be programmed with the same value as TINSTANCE in DP_DLPIDR in order to select this device.

Bits 27:12 **TPARTNO[15:0]**: Target part number

This field defines the part number for the target device. It must be programmed with the same value as TPARTNO in DP_TARGETID in order to select this device.

Bits 11:1 **TDESIGNER[10:0]**: Target designer JEDEC code

This field defines the JEDEC code for the target device. It must be programmed with the same value as TDESIGNER in DP_TARGETID in order to select this device.

Bit 0 Reserved, must be kept at reset value.

Debug port register map

These registers are not on the CPU memory bus. They are only accessed through SW-DP and JTAG-DP debug interfaces.

The debug port address offset is 4-bit wide, where the two most significant bits are defined in the JTAG-DP register DPACC or SW-DP packet request A[3:2] field. The two least significant bits are 00.

Table 865. Debug port register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0	DP_DPIDR	REVISION [3:0]				PARTNO[7:0]								Res.	Res.	Res.	MIN	VERSION [3:0]				DESIGNER[10:0]											Res.
	Reset value	0	1	1	0	1	0	1	1	1	0	1	0				0	0	0	1	0	0	1	0	0	0	1	1	1	0	1	1	
0x0	DP_ABORT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ORUNERRCLR	WDERRCLR	STKERRCLR	STKCMPCLR	DAPABORT
	Reset value																												0	0	0	0	0
0x4(1)	DP_CTRLSTAT	CSYSPWRUPACK	CSYSPWRUPREQ	CDBGPWRUPACK	CDBGPWRUPREQ	CDBGRSTACK	CDBGRSTREQ	Res.	Res.	TRNCNT[11:0]												MASKLANE[3:0]				WDATAERR	READOK	STICKYERR	STICKYCMP	TRNMODE[1:0]		STICKYORUN	ORUNDETECT
	Reset value	0	0	0	0	0	0			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x4(2)	DP_DLCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TURNROUND [1:0]		Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value	0																						0	0								

Table 865. Debug port register map and reset values (continued)

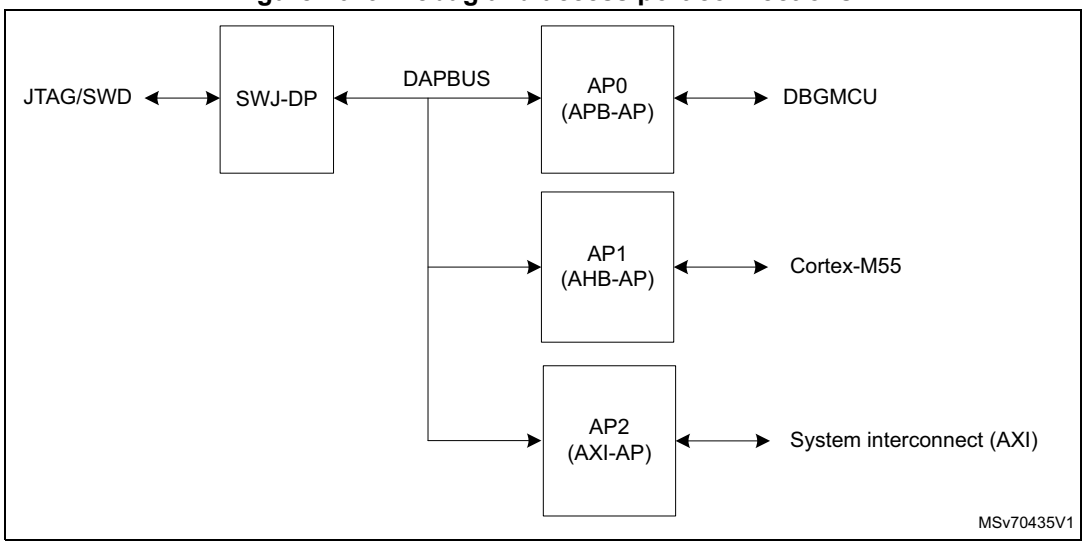
Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x4[3]	DP_TARGETID	TREVISION [3:0]				TPARTNO[15:0]																TDESIGNER[10:0]											Res.
	Reset value	0	0	0	0	0	1	0	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
0x4[4]	DP_DLPIDR	TINSTANCE [3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PROTSVN [3:0]			
	Reset value	0	0	0	0																									0	0	0	1
0x8	DP_RESEND	RESEND[31:0]]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x8	DP_SELECT	APSEL[7:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APBANKSEL [3:0]				DPBANKSEL [3:0]			
	Reset value	X	X	X	X	X	X	X	X																	X	X	X	X	X	X	X	X
0xC	DP_RDBUFF	RDBUFF[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xC	DP_TARGETSEL	TINSTANCE [3:0]				TPARTNO[15:4]												TPARTNO [3:0]				TDESIGNER[10:0]											Res.
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	

1. DPBANKSEL[3:0] = 0x0
2. DPBANKSEL[3:0] = 0x1
3. DPBANKSEL[3:0] = 0x2
4. DPBANKSEL[3:0] = 0x3

78.10 Access ports

There are three access ports (AP) attached to the DP.

Figure 1079. Debug and access port connections



- AP0 debug access port to DBGMCU (APB-AP): is always accessible during reset so that a debugger can initiate an authentication process even if it is a closed device.
- AP1 Cortex-M55 access port (AHB-AP): allows access to the debug and trace features integrated in the Cortex-M55 processor via its PPB bus, and all other debug

components located on the EPPB interface (external private peripheral bus). In a closed device, AP1 is locked until the debug authentication process is complete.

- AP2 system bus access port (AXI-AP): Allows access to the system bus matrix, which gives visibility of all the memory and peripherals accessible via the AXI interconnect. In a closed device, AP2 is locked until the debug authentication process is complete.

All access ports are of MEM-AP type: debug and trace component registers are mapped in the address space of the associated debug bus. The AP is seen by the debugger as a set of 32-bit registers organized in banks of four registers each. Some of these registers are used to configure or monitor the AP itself, while others are used to perform a transfer on the bus.

The address of the AP registers is composed of:

- bits [7:4]: content of the DP_SELECT register's APBANKSEL field
- bits [3:2]: content of the A(3:2) field of the APACC data register in the JTAG-DP (see *Table 860*) or of the SW-DP packet request (see *Table 861*), depending on the debug interface used
- bits [1:0]: Always set to 0

The content of APSEL field in DP_SELECT defines which MEM-AP is being accessed.

The debugger can access the AP registers as follows:

- Program APSEL field in DP_SELECT to choose one of the APs, and APBANKSEL field to select the register bank to be accessed.
- If using JTAG, program A[3:2] field in the APACC register with the register address within the bank. Program the RnW bit to select a read or a write. For a write, program the DATA field with the write data.

  If using SWD, the A(3:2) and RnW fields are part of the packet request word sent to the SW-DP with the APnDP bit set (see *Table 861*). The write data are sent in the data phase.

The debugger can access the memory-mapped debug component registers through the MEM-AP registers (using the AP register access procedure described above) as follows:

1. Program the transaction target address in APx_TAR.
2. Program APx_CSW, if necessary, with the transfer parameters (AddrInc for example).
3. Write to or read from APx_DRW to initiate a bus transaction at the address held in the APx_TAR register. Alternatively, a read or write to banked data register BDn triggers an access to address TAR[31:4] + n (this allows an access to the next four consecutive addresses without changing the address in APx_TAR).

For more detailed information on the MEM-AP, refer to the Arm Debug Interface Architecture Specification *[1]*.

Authentication

The effect of the authentication signals for AP1 and AP2 is shown in *Table 866*.

Table 866. AP1-2 authentication behavior

spiden	Others	Behavior
0	X	Secure transfers blocked. Any attempt to perform a secure transfer results in an error. Nonsecure transfers are enabled.
1		Both non-secure and secure transfers are enabled.

78.10.1 AP0 registers

Access port 0 control/status word register (AP0_CSW)

Address offset: 0x0

Reset value: 0x8000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DBGSWENABLE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	MODE[3:0]				TRINPROG	DEVICEEN	ADDRINC[1:0]		Res.	SIZE[2:0]		
				rw	rw	rw	rw	r	r	rw	rw		r	r	r

Bit 31 **DBGSWENABLE**: Software debug enable
This bit enables or disables software access to the APB bus via the APB interconnect. It must always be set to 1 to enable software access to the DBGMCU.
0 Disable software access.
1 Enable software access.

Bits 30:12 Reserved, must be kept at reset value.

Bits 11:8 **MODE[3:0]**: Mode of operation
0x0: Normal download or upload model.
Others: reserved

Bit 7 **TRINPROG**: Transfer in progress
This bit indicates that an AP bus transfer is in progress.
0: No transfer in progress
1: Bus transfer in progress

Bit 6 **DEVICEEN**: Device enable
This bit defines whether the AP can be accessed or not.
1: AP access enabled

Bits 5:4 **ADDRINC[1:0]**: Auto-increment mode bits
These bits define whether the TAR address is automatically incremented after a transaction.
0x0: No auto-increment
0x1: Address is incremented by the size in bytes of the transaction (SIZE in this register).
Others: reserved

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **SIZE[2:0]**: Size of next memory access transaction
0x2: Word (32-bit)

AP0 transfer address register (AP0_TAR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDRESS[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRESS[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ADDRESS[31:0]**: address of the current transfer

AP0 data read/write register (AP0_DRW)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATA[31:0]**: read/write data for current transfer
In write mode, this is the write data value. In read mode, this is the read data value.

AP0 banked data register x (AP0_BDx)

Address offset: 0x10 + 0x4 * x (x = 0 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATA[31:0]**: Read/write data for current transfer
The transaction address is TAR[31:4] << 4 + 0x4 * x.

AP0 base address register (AP0_BASE)

Address offset: 0xF8

Reset value: 0x8000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BASEADDR[19:4]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BASEADDR[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FORMAT	ENTRYPRESENT
r	r	r	r											r	r

Bits 31:12 **BASEADDR[19:0]**: Base address (bits 31 to 12) of the AP ROM table
The twelve LSBs are 0 since the ROM table must be aligned on a 4-Kbyte boundary.
0x80000: ROM table base address is 0x8000 0000

Bits 11:2 Reserved, must be kept at reset value.

Bit 1 **FORMAT**: Base-address register format
1: Arm debug interface v5

Bit 0 **ENTRYPRESENT**: Debug component present status
This bit indicates that debug components are present on the access port bus:
1: Debug components are present.

AP0 identification register (AP0_IDR)

Address offset: 0xFC

Reset value: 0x5477 0002 (AP0)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REVISION[3:0]				JEDECBANK[3:0]				JEDECCODE[6:0]							MEMAP
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDENTITY[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:28 **REVISION[3:0]**: Arm core revision
0x5: r1p0

Bits 27:24 **JEDECBANK[3:0]**: JEDEC bank
0x4: Arm

Bits 23:17 **JEDECCODE[6:0]**: JEDEC code
0x3B: Arm

Bit 16 **MEMAP**: Memory access port
1: Standard register map

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **IDENTITY[7:0]**: AP type identification
0x02: APB-AP

AP0 register map

These registers are not on the CPU memory bus. They are only accessed through SW-DP and JTAG-DP debug interfaces.

The access port address is 8-bit wide, defined by APBANKSEL[3:0] in DP_SELECT and by JTAG-DP register DPACC or SW-DP packet request A[3:2] field.

Table 867. AP0 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	AP0_CSW	DBGSWENABLE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE[3:0]				TRINPROG	DEVICEEN	ADDRINC[1:0]		Res.	SIZE[2:0]		
	Reset value	1																				0	0	0	0	0	0	0	0		0	1	0
0x04	AP0_TAR	ADDRESS[31:0]																															
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	Reserved	Reserved																															
0x0C	AP0_DRW	DATA[31:0]																															
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10 + 0x4 * x (x = 0 to 3)	AP0_BDx	DATA[31:0]																															
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20-0xF4	Reserved	Reserved																															
0xF8	AP0_BASE	BASEADDR[19:0]																				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FORMAT	ENTRYPRESENT
	Reset value	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0										1	1
0xFC	AP0_IDR	REVISION [3:0]				JEDECBANK [3:0]				JEDECCODE[6:0]							MEMAP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDENTITY[7:0]							
	Reset value	0	1	0	1	0	1	0	0	0	1	1	1	0	1	1	1									0	0	0	0	0	0	1	0

78.10.2 AP1 registers

Access port 1 control/status word register (AP1_CSW)

Address offset: 0x0

Reset value: 0x4000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	SPROT	Res.	PROT[4:0]					SPISTATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw		rw	rw	rw	rw	rw	r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	MODE[3:0]				TRINPROG	DBGSTATUS	ADDRINC[1:0]		Res.	SIZE[2:0]		
				rw	rw	rw	rw	r	r	rw	rw		rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bit 30 **SPROT**: Secure transfer request
This bit sets the protection attribute HPROT[6] of the bus transfer.
0: If spiden is high, secure transfer. If spiden is low, no transfer.
1: Nonsecure transfer

Bit 29 Reserved, must be kept at reset value.

Bits 28:24 **PROT[4:0]**: Bus transfer protection
This field sets the protection attributes HPROT[4:0] of the bus transfer.
xxxx0: Instruction fetch
xxxx1: Data access
xxx0x: User mode
xxx1x: Privileged mode
xx0xx: Non-bufferable
xx1xx: Bufferable
x0xxx: Non-cacheable
x1xxx: Cacheable
0xxxx: Non-exclusive
1xxxx: Exclusive

Bit 23 **SPISTATUS**: Status of spiden authentication signal
This bit determines whether the debugger can access secure memory.
0: Secure AHB transfers not supported
1: Secure AHB transfers allowed

Bits 22:12 Reserved, must be kept at reset value.

Bits 11:8 **MODE[3:0]**: Specifies the mode of operation
0x0: Normal download or upload model
Others: reserved

Bit 7 **TRINPROG**: Transfer in progress
This bit indicates that an AP bus transfer is in progress.
0: No transfer in progress
1: Bus transfer in progress

Bit 6 **DBGSTATUS**: Debug status
Indicates the state of the dbgen authentication signal
0: No AHB transfers permitted
1: AHB transfers permitted

Bits 5:4 **ADDRINC[1:0]**: Auto-increment mode
These bits define whether the TAR address is automatically incremented after a transaction.
0x0: No auto-increment
0x1: Address incremented by the size in bytes of the transaction (SIZE field)
0x2: Packed transfers enabled. A 32-bit AP access generates a 1 x 32-bit, 2 x 16-bit, or 4 x 8-bit bus transaction corresponding to the programmed transaction size. Data are packed or unpacked accordingly.
0x3: Reserved

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **SIZE[2:0]**: Size of next memory access transaction
0x0: Byte (8-bit)
0x1: Half-word (16-bit)
0x2: Word (32-bit)
Others: Reserved

AP1 transfer address register (AP1_TAR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDRESS[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRESS[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ADDRESS[31:0]**: Address of the current transfer

AP1 data read/write register (AP1_DRW)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATA[31:0]**: Read/write data for current transfer
In write mode, this is the write data value. In read mode, this is the read data value.

AP1 banked data register x (AP1_BDx)

Address offset: 0x10 + 0x4 * x (x = 0 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATA[31:0]**: read/write data for current transfer
The transaction address is TAR[31:4] << 4 + 0x4 * x.

AP1 base address register (AP1_BASE)

Address offset: 0xF8

Reset value: 0xE00F E003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BASEADDR[19:4]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BASEADDR[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FORMAT	ENTRYPRESENT
r	r	r	r											r	r

Bits 31:12 **BASEADDR[19:0]**: Base address (bits 31 to 12) of the AP ROM table
The twelve LSBs are 0 since the ROM table must be aligned on a 4-Kbyte boundary.
0xE00FE: ROM table base address is at 0xE00F E000.

Bits 11:2 Reserved, must be kept at reset value.

Bit 1 **FORMAT**: Base address register format
1: Arm debug interface v5.

Bit 0 **ENTRYPRESENT**: Debug component present status
This bit indicates that debug components are present on the access port bus:
1: Debug components are present.

AP1 identification register (AP1_IDR)

Address offset: 0xFC

Reset value: 0x8477 0001 (AP1)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REVISION[3:0]				JEDECBANK[3:0]				JEDECCODE[6:0]							MEMAP
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDENTITY[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:28 **REVISION[3:0]**: Arm core revision
0x8: r0p9

Bits 27:24 **JEDECBANK[3:0]**: JEDEC bank
0x4: Arm

Bits 23:17 **JEDECCODE[6:0]**: JEDEC code
0x3B: Arm

Bit 16 **MEMAP**: Memory access port
1: Standard register map

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **IDENTITY[7:0]**: AP type identification
0x01: AHB-AP

AP1 register map

These registers are not on the CPU memory bus. They are accessed only through SW-DP and JTAG-DP debug interfaces.

The access port address is 8-bit wide, defined by APBANKSEL[3:0] in DP_SELECTR, and by JTAG-DP register DPACC or SW-DP packet request A[3:2] field.

Table 868. AP1 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	AP1_CSW	Res.	SPROT	Res.	PROT[4:0]					SPISTATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE[3:0]				TRINPROG	DEVICEEN	ADDRINC[1:0]		Res.	SIZE[2:0]		
	Reset value		1		0	0	0	0	0	0												0	0	0	0	0	0	0	0		0	1	0
0x04	AP1_TAR	ADDRESS[31:0]																															
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	Reserved	Reserved																															
0x0C	AP1_DRW	DATA[31:0]																															
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10 + 0x4 * x (x = 0 to 3)	AP1_BDx	DATA[31:0]																															
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20-0xF4	Reserved	Reserved																															
0xF8	AP1_BASE	BASEADDR[19:0]																				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FORMAT	ENTRYPRESENT
	Reset value	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0										1	1
0xFC	AP1_IDR	REVISION[3:0]				JEDECBANK[3:0]				JEDECCODE[6:0]							MEMAP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDENTITY[7:0]							
	Reset value	1	0	0	0	0	1	0	0	0	1	1	1	0	1	1	1									0	0	0	0	0	0	0	1

78.10.3 AP2 registers

AP2 control/status word register (AP2_CSW)

Address offset: 0x0

Reset value: 0x30X0 60X2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	PROT[2:0]			CACHE[3:0]				SPISTATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	rw	rw	rw	rw	rw	rw	rw	r							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DOMAIN[1:0]		ACEENABLE	MODE[3:0]				TRINPROG	DBGSTATUS	ADDRINC[1:0]		Res.	SIZE[2:0]		
	rw	rw	rw	rw	rw	rw	rw	r	r	rw	rw		rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **PROT[2:0]**: Access permissions

This field sets ARPROT[2:0] or AWPROT[2:0] attributes of the bus transfer.

xx0: Unprivileged access

xx1: Privileged access

x0x: Secure access

x1x: Nonsecure access

0xx: Data access

1xx: Instruction access

Bits 27:24 **CACHE[3:0]**: Memory type

This field sets ARCACHE[3:0] or AWCACHE[3:0] attributes of the bus transfer (see AMBA AXI and ACE Protocol Specification *[7]*).

Bit 23 **SPISTATUS**: Status of spiden authentication signal

This bit determines whether the debugger can access secure memory.

0: Secure AXI transfers not supported

1: Secure AXI transfers are allowed.

Bits 22:15 Reserved, must be kept at reset value.

Bits 14:13 **DOMAIN[1:0]**: Shareable transaction encoding for ACE

0: Non-shareable

1: Shareable, inner domain, includes additional masters.

2: Shareable, outer domain, also includes inner or additional masters.

3: Shareable, system domain, all masters included

Bit 12 **ACEENABLE**: Enable ACE transactions, including barriers

0: Disable

1: Enable

Bits 11:8 **MODE[3:0]**: Specify mode of operation

0x0: Normal download or upload

0x1: Barrier transaction

Others: Reserved

Bit 7 **TRINPROG**: Transfer in progress

This bit indicates that an AP bus transfer is in progress.

0: No transfer in progress

1: Bus transfer in progress

Bit 6 **DBGSTATUS**: Debug status

This bit indicates the state of the dbgen authentication signal

0: No AXI transfers permitted

1: AXI transfers are permitted.

Bits 5:4 **ADDRINC[1:0]**: Auto-increment mode

These bits define whether the TAR address is automatically incremented after a transaction.
0x0: No auto-increment
0x1: Address is incremented by the size in bytes of the transaction (SIZE field).
0x2: Packed transfers enabled. A 32-bit AP access generates a 1 x 32-bit, 2 x 16-bit or 4 x 8-bit bus transaction corresponding to the programmed transaction size. The data is packed or unpacked accordingly.
0x3: Reserved

Bit 3 Reserved, must be kept at reset value.

Bits 2:0 **SIZE[2:0]**: Size of next memory access transaction
0x0: Byte (8-bit)
0x1: Half-word (16-bit)
0x2: Word (32-bit)
0x3: Double word (64-bit)
Others: Reserved

AP2 transfer address register (AP2_TAR)

Address offset: 0x04

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDRESS[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDRESS[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **ADDRESS[31:0]**: Address of the current transfer

AP2 data read/write register (AP2_DRW)

Address offset: 0x0C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATA[31:0]**: Read/write data for current transfer

In write mode, this is the write data value. In read mode, this is the read data value.
For 64-bit transfers, two accesses must be made to DRW for a single 64-bit AXI access. The first access in read mode initiates the read on the AXI bus and returns the lower 32-bit word of data.The second access returns the upper 32-bit word. In write mode, the first access contains the lower 32-bit word of data, the second access contains the upper 32-bit word and initiates the write on the AXI bus.

AP2 banked data register x (AP2_BDx)

Address offset: 0x10 + 0x4 * x (x = 0 to 3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DATA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DATA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **DATA[31:0]**: Read/write data for current transfer
The transaction address is TAR[31:4] << 4 + 0x4 * x.
64-bit banked data transactions are not supported

AP2 configuration register (AP2_CONFIG)

Address offset: 0xF4

Reset value: 0x0000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LD	LA	BE
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **LD**: Large data support
1: 64-bit data transfers supported

Bit 1 **LA**: Long address support
0: 32-bit address only

Bit 0 **BE**: Big-endian
0: Little-endian

AP2 base address register (AP2_BASE)

Address offset: 0xF8

Reset value: 0x0000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BASEADDR[19:4]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BASEADDR[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FORMAT	ENTRY PRESENT
r	r	r	r											r	r

Bits 31:12 **BASEADDR[19:0]**: Base address (bits 31 to 12) of the AP ROM table
Not applicable, no ROM table present

Bits 11:2 Reserved, must be kept at reset value.

Bit 1 **FORMAT**: Base address register format
1: Arm debug interface v5

Bit 0 **ENTRYPRESENT**: Debug component present status
This bit indicates that debug components are present on the access port bus:
0: No debug components are present.

AP2 identification register (AP2_IDR)

Address offset: 0xFC

Reset value: 0x4477 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REVISION[3:0]				JEDECBANK[3:0]				JEDECCODE[6:0]							MEMAP
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDENTITY[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:28 **REVISION[3:0]**: Arm core revision
0x4: r1p1

Bits 27:24 **JEDECBANK[3:0]**: JEDEC bank
0x4: Arm

Bits 23:17 **JEDECCODE[6:0]**: JEDEC code
0x3B: Arm

Bit 16 **MEMAP**: Memory access port
1: Standard register map

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **IDENTITY[7:0]**: AP type identification
0x04: AXB-AP

AP2 register map

These registers are not on the CPU memory bus. They are accessed only through SW-DP and JTAG-DP debug interfaces.

The access port address is 8-bit wide, defined by APBANKSEL[3:0] in DP_SELECTR, and by JTAG-DP register DPACC or SW-DP packet request A[3:2] field.

Table 869. AP2 register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x00	AP2_CSW	Res.	PROT[2:0]			CACHE[3:0]				SPISTATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.		DOMAIN[2:0]		ACEENABLE	MODE[3:0]				TRINPROG	DBGSTATUS	ADDRINC[1:0]		Res.	SIZE[2:0]		
	Reset value		0	1	1	0	0	0	0	X									1	1	0	0	0	0	0	X	X	X	X		0	1	0
0x04	AP2_TAR	ADDRESS[31:0]																															
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x08	Reserved	Reserved																															
0x0C	AP2_DRW	DATA[31:0]																															
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x10 + 0x4 * x (x = 0 to 3)	AP2_BDx	DATA[31:0]																															
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x20-0xF0	Reserved	Reserved																															
0xF4	AP2_CONFIG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LD	LA	BE
	Reset value																														0	1	0
0xF8	AP_BASE	BASEADDR[19:0]																				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FORMAT	ENTRYPRESENT
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0											1	0
0xFC	AP_IDR	REVISION [3:0]				JEDECBANK [3:0]				JEDECCODE[6:0]							MEMAP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	IDENTITY[7:0]							
	Reset value	0	1	0	0	0	1	0	0	0	1	1	1	0	1	1	1									0	0	0	0	0	1	0	0

78.10.4 ROM tables

The ROM table is a CoreSight component that contains the base addresses of all the CoreSight debug components accessible via the access port to which they are attached. These tables allow a debugger to discover the topology of the CoreSight system automatically.

The system ROM table is pointed to by the AP0 base register. It contains the base address pointer for the DBGMCU.

The system ROM table occupies a 4-Kbyte, 32-bit wide chunk of address space, from 0x8000 0000 to 0x8000 0FFC when accessed by the debugger.

Alternatively, it can be accessed by the CPU at address range 0x4400 0000 to 0x4400 0FFF.

Table 870. System ROM table

Address offset in ROM table	Component name	Component base address (debugger)	Component base address (system bus)	Component address offset	Size (Kbytes)	Entry
0x000	DBGMCU	0x8000 1000	0x4400 1000	0x1000	4	0x0000 1003
0x004	DFT register(1)	0x8000 2000	0x4400 2000	0x2000	4	0x0000 2002
0x008	Top of table	-	-	-	-	0x0000 0000
0x00C to 0xFC8	Reserved	-	-	-	-	0x0000 0000
0xFCC to 0xFFC	ROM table registers	-	-	-	-	-

1. Not CoreSight compatible. Bit 0 is reset to indicate that there are no CoreSight registers in this component.

The MCU ROM table is pointed to by the AP1 base register. It contains the base address pointer for the Cortex-M55 ROM table and for the trace subsystem ROM table.

It occupies a 4-Kbyte, 32-bit wide chunk of address space, from 0xE00F E000 to 0xE00F EFFF.

Table 871. MCU ROM table

Address offset in ROM table	Component name	Component base address	Component address offset	Size (Kbytes)	Entry
0x000	Cortex-M55 processor ROM table (*Table 873*)	0xE00F F000	0x0000 1000	4	0x0000 1003
0x004	Trace subsystem ROM table (*Table 880*)	0xE008 0000	0xFFF8 2000	4	0xFFF8 2003
0x008	Top of table	-	-	-	0x0000 0000
0x00C to 0xFC8	Reserved	-	-	-	0x0000 0000
0xFCC to 0xFFC	ROM table registers	-	-	-	-

The top of ROM tables contains a number of read-only registers, including the standard CoreSight component and peripheral identity registers (see *Section 78.10.5*).

Each debug component occupies one or more 4-Kbyte blocks of address space. This block of address space is referred to as the debug register file for the component.

The component address offset field of a ROM table entry points to the start of the last 4-Kbyte block of the address space of the component. This block always contains the component and peripheral ID registers for the component, starting at offset 0xFD0 from the start of the block. The 4-Kbyte count field PIDR4 [7:4], specifies the number of 4-Kbyte blocks for the component.

The process for finding the start of the address space for a component is:

1. Read the ROM-table entry for the component and extract its Address_Offset[18:0] from bits [31:12] of the ROM-table entry.
2. Use the address offset, together with the base address of the ROM table, ROM_Base_Address, to calculate the base address of the component:

   Component_Base_Address = ROM_Base_Address + Address_Offset

   The Component_Base_Address is the start address of the 4-Kbyte address space containing the CoreSight identity registers for the component.
3. Read the peripheral ID4 (PIDR4) register for the component. The address of this register is:

   Peripheral_ID4_address = Component_Base_Address + 0xFD0
4. Extract the SIZE[7:4] field from the value of the peripheral ID4 register. This gives the number of 4-Kbyte sections of address space occupied by the component.
5. Use the SIZE field value to calculate the start address of the address space for the component:
   Component_Start_Address = Component_Base_Address - SIZE × 0x1000. If the field value is 0b0000, which corresponds to one 4-Kbyte section, the address space for the component starts at Component_Base_Address obtained at stage 2.

For more information on the use of the ROM tables, refer to the Arm Debug Interface Architecture Specification *[1]*.

78.10.5 System/MCU ROM registers

SYSROM memory type register (SYSROM_MEMTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SYSMEM**: System memory

0: No system memory is present on this bus.

SYSROM CoreSight peripheral identity register 4 (SYSROM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: Register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x0: STMicroelectronics JEDEC continuation code

SYSROM CoreSight peripheral identity register 0 (SYSROM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0099

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Device part number field, bits [7:0]
0x86: STM32N6 device

SYSROM CoreSight peripheral identity register 1 (SYSROM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0x0: STMicroelectronics JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Device part number field, bits [11:8]
0x4: STM32N6x5/x7xx device

SYSROM CoreSight peripheral identity register 2 (SYSROM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 001A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Device revision number
0x0: Not used

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x2: STMicroelectronics JEDEC code

SYSROM CoreSight peripheral identity register 3 (SYSROM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

SYSROM CoreSight component identity register 0 (SYSROM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value

SYSROM CoreSight component identity register 1 (SYSROM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0x1: ROM table component

Bits 3:0 **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

SYSROM CoreSight component identity register 2 (SYSROM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

SYSROM CoreSight component identity register 3 (SYSROM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value

System ROM register map

Table 872. System ROM table register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFCC	SYSROM_ MEMTYPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
	Reset value																																0
0xFD0	SYSROM_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON [3;0			
	Reset value																									0	0	0	0	0	0	0	0
0xFD4-0xFDC	Reserved	Reserved																															
0xFE0	SYSROM_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									1	0	0	0	0	1	1	0
0xFE4	SYSROM_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									0	0	0	0	0	1	0	0
0xFE8	SYSROM_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	0
0xFEC	SYSROM_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	SYSROM_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	SYSROM_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									0	0	0	1	0	0	0	0
0xFF8	SYSROM_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1

Table 872. System ROM table register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFFC	SYSROM_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

78.11 Cortex-M55 debug features

The Cortex-M55 integrates the following CoreSight components:

- Cortex-M55 ROM table
- Cortex-M55 SCS (system control space)
- BPU (breakpoint unit)
- DWT (data watchpoint and trace unit)
- ITM (instrumentation trace macrocell)
- ETM (Embedded Trace Macrocell)
- CTI (cross trigger interface)

These components are accessible by the debugger via the Cortex-M55 AHB-AP.

78.11.1 Processor ROM table

The MCU ROM table in the Cortex-M55 sub-system pointed to by the BASE register in the Cortex-M55 AHB-AP links to the Cortex-M55 processor ROM table.

Table 873 contains the base address pointer for the SCS registers, which allows the debugger to identify the CPU core, as well as for the FPB, DWT, ITM, ETM, and CTI.

The Cortex-M55 processor ROM table occupies a 4-Kbyte, 32-bit wide chunk of address space, from 0xE00FF000 to 0xE00FFFFC.

Table 873. Processor ROM table

Address in ROM table	Component name	Component base address	Component address offset	Size (Kbytes)	Entry
0xE00F F000	SCS	0xE000 E000	0xFFF0 F000	4	0xFFF0 F003
0xE00F F004	DWT	0xE000 1000	0xFFF0 2000	4	0xFFF0 2003
0xE00F F008	FPB	0xE000 2000	0xFFF0 3000	4	0xFFF0 3003
0xE00F F00C	ITM	0xE000 0000	0xFFF0 1000	4	0xFFF0 1003
0xE00F F010	TPIU(1)	0xE004 0000	0xFFF4 1000	4	0xFFF4 1002
0xE00F F014	ETM	0xE004 1000	0xFFF4 2000	4	0xFFF4 2003
0xE00F E018	PMU	0xE000 3000	0xFFF0 4000	4	0xFFF0 4003
0xE00F E01C	CTI	0xE004 2000	0xFFF4 3000	4	0xFFF4 3003
0xE00F F020	PMC-100(2)	0xE004 6000	0xFFF4 7000	4	0xFFF4 7002
0xE00F F024	Top of table	-	-	-	0x0000 0000

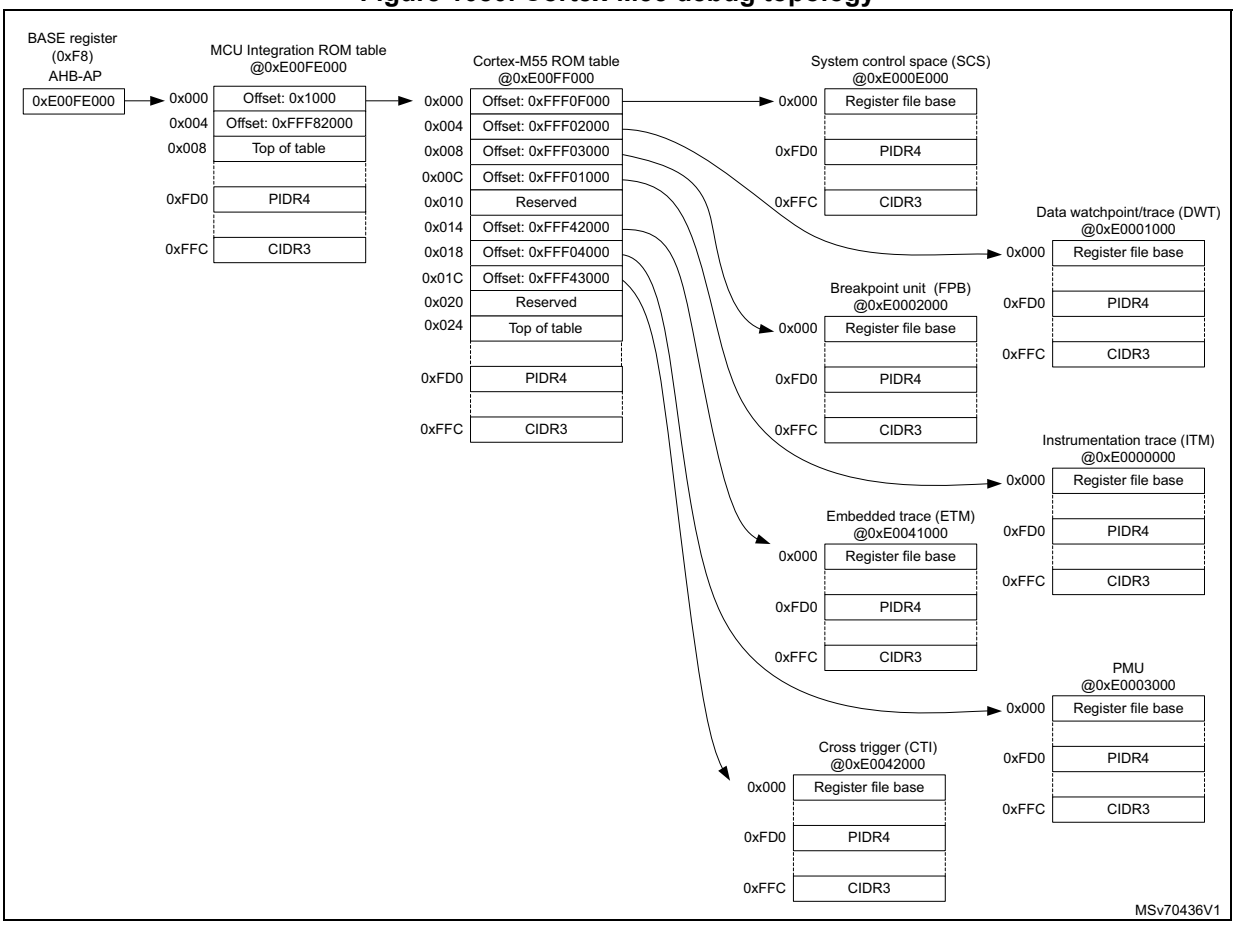
Table 873. Processor ROM table (continued)

Address in ROM table	Component name	Component base address	Component address offset	Size (Kbytes)	Entry
0xE00F F024 to 0xE00F FFC8	Reserved	-	-	-	0x0000 0000
0xE00FF FCC to 0xE00F FFFC	ROM table registers	-	-	-	-

1. The TPIU is included in this table by default, but bit 0 is reset to indicate that it is not present.
2. The PMC-100 is included in this table by default, but bit 0 is reset to indicate that it is not present.

The topology for the debug components in the Cortex-M55 subsystem is shown in *Figure 1080*.

Figure 1080. Cortex-M55 debug topology



78.11.2 Processor ROM registers

Processor ROM memory type register (CPUROM_MEMTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SYSMEM**: System memory presence
1: System memory is present on this bus.

Processor ROM CoreSight peripheral identity register 4 (CPUROM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: Register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm JEDEC continuation code

Processor ROM CoreSight peripheral identity register 0 (CPUROM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 00D2

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number field, bits [7:0]
0xD2: Cortex-M55 processor ROM table

Processor ROM CoreSight peripheral identity register 1 (CPUROM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0xB: Arm JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number field, bits [11:8]
0x4: Cortex-M55 processor ROM table

Processor ROM CoreSight peripheral identity register 2 (CPUROM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x0: rev r0p0

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x3: Arm JEDEC code

Processor ROM CoreSight peripheral identity register 3 (CPUROM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

Processor ROM CoreSight component identity register 0 (CPUROM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value

Processor ROM CoreSight component identity register 1 (CPUROM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0x1: ROM table component

Bits 3:0 **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

Processor ROM CoreSight component identity register 2 (CPUROM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

Processor ROM CoreSight component identity register 3 (CPUROM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value

Processor ROM register map

Table 874. Processor ROM register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFCC	CPUROM_MEMTYPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
	Reset value																																1
0xFD0	CPUROM_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4-0xFDC	Reserved	Reserved																															
0xFE0	CPUROM_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									1	1	0	1	0	0	1	0

Table 874. Processor ROM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFE4	CPUROM_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	0	1	0	0
0xFE8	CPUROM_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	1
0xFEC	CPUROM_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	CPUROM_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	CPUROM_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									0	0	0	1	0	0	0	0
0xFF8	CPUROM_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	CPUROM_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

78.11.3 Data watchpoint and trace unit (DWT)

The DWT provides eight comparators that can be used as:

- watchpoint
- ETM trigger
- PC sampling trigger
- data address sampling trigger
- data comparator (available on comparator 1 only)
- clock cycle counter comparator (available on comparator 0 only)

It also contains counters for:

- clock cycles
- folded instructions
- LSU (load store unit) operations
- sleep cycles
- number of cycles per instruction
- interrupt overhead

A DWT comparator compares one of the following with the value held in its DWT_COMP:

- a data address
- an instruction address
- a data value
- the cycle count value (for comparator 0 only)

For address matching, the comparator can use a mask, so it matches a range of addresses.

On a successful match, the comparator generates one of the following:

- one or more DWT data trace packets, containing one or more of:
  - the address of the instruction that caused a data access
  - an address offset, bits[15:0] of the data access address
  - the matched data value.
- a watchpoint debug event, on either the PC value, or the accessed data address
- a CMPMATCH[N] event that signals the match outside the DWT unit

A watchpoint debug event either generates a DebugMonitor exception, or causes the processor to halt execution and enter debug state.

For more details on how to use the DWT, refer to the Arm Cortex-M55 Processor Technical Reference Manual *[4]*.

78.11.4 Cortex-M55 DWT registers

The register file base address for the DWT is defined in *Table 873: Processor ROM table*.

DWT control register (DWT_CTRL)

Address offset: 0x000

Reset value: 0x8000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMCOMP[3:0]				NOTRCPKT	NOEXTTRIG	NOCYCCNT	NOPRFCNT	CYCDISS	CYCEVTENA	FOLDEVTENA	LSUEVTENA	SLEEPEVTENA	EXCEVTENA	CPIEVTENA	EXCTRCENA
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	PCSAMPLENA	SYNCTAP[1:0]		CYCTAP	POSTINIT[3:0]				POSTRESET[3:0]				CYCCNTENA
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:28 **NUMCOMP[3:0]**: Number of implemented comparators (read-only)
0x8: Eight comparators

Bit 27 **NOTRCPKT**: Trace sampling and exception tracing support (read-only)
0: Supported

Bit 26 **NOEXTTRIG**: External match signal, CMPMATCH support (read-only)
0: Supported

Bit 25 **NOCYCCNT**: Cycle counter support (read-only)
0: Supported

Bit 24 **NOPRFCNT**: Profiling counter support (read-only)
0: Supported

Bit 23 **CYCDISS**: Cycle counter disabled secure
0: No effect
1: Cycle counter is not incremented when the processor is in secure state

Bit 22 **CYCEVTENA**: Enable POSTCNT underflow event counter packet generation
0: Disabled
1: Enabled

Bit 21 **FOLDEVTENA**: Enable folded instruction counter overflow event generation
0: Disabled
1: Enabled

Bit 20 **LSUEVTENA**: Enable LSU counter overflow event generation
0: Disabled
1: Enabled

Bit 19 **SLEEPEVTENA**: Enable sleep counter overflow event generation
0: Disabled
1: Enabled

Bit 18 **EXCEVTENA**: Enable exception overhead counter overflow event generation
0: Disabled
1: Enabled

Bit 17 **CPIEVTENA**: Enable CPI counter overflow event generation
0: Disabled
1: Enabled

Bit 16 **EXCTRCENA**: Enable exception trace generation
0: Disabled
1: Enabled

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **PCSAMPLENA**: POSTCNT counter use enable
Enables the use of POSTCNT counter as a timer for periodic PC sample packet generation.
0: Disabled
1: Enabled

Bits 11:10 **SYNCTAP[1:0]**: Position of synchronization packet counter tap on CYCCNT counter
This selection determines the synchronization packet rate.
0x0: Disabled - no synchronization packets
0x1: Tap at CYCCNT[24]
0x2: Tap at CYCCNT[26]
0x3: Tap at CYCCNT[28]

Bit 9 **CYCTAP**: Position of the POSTCNT tap on the CYCCNT counter
0: Tap at CYCCNT[6]
1: Tap at CYCCNT[10]

Bits 8:5 **POSTINIT[3:0]**: Initial value of the POSTCNT counter
Writes to this field are ignored if POSTCNT counter is enabled (CYCEVTENA or PCSAMPLENA must be reset prior to writing POSTINIT).

Bits 4:1 **POSTRESET[3:0]**: Reload value of the POSTCNT counter.

Bit 0 **CYCCNTENA**: CYCCNT counter enable
0: Disabled
1: Enabled

DWT cycle count register (DWT_CYCCNT)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CYCCNT[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CYCCNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CYCCNT[31:0]**: Processor clock-cycle counter

DWT CPI count register (DWT_CPICNT)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPICNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **CPICNT[7:0]**: CPI counter

Counts additional cycles required to execute multi-cycle instructions, except those recorded by DWT_LSUCNT, and counts any instruction fetch stalls.

DWT exception count register (DWT_EXCCNT)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXCCNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **EXCCNT[7:0]**: Exception overhead cycle counter

Counts the number of cycles spent in exception processing.

DWT sleep count register (DWT_SLEEPCNT)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLEEPCNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **SLEEPCNT[7:0]**: Sleep cycle counter

Counts the number of cycles spent in sleep mode (WFI, WFE, sleep-on-exit).

DWT LSU count register (DWT_LSUCNT)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSUCNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **LSUCNT[7:0]**: Load store counter

Counts additional cycles required to execute load and store instructions.

DWT fold count register (DWT_FOLDCNT)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FOLDCNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **FOLDCNT[7:0]**: Folded instruction counter

Increments on each instruction that takes 0 cycles (in parallel with another instruction).

DWT program counter sample register (DWT_PCSR)

Address offset: 0x01C

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EIASAMPLE[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EIASAMPLE[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **EIASAMPLE[31:0]**: Executed instruction address sample value
Samples the current value of the program counter.

DWT comparator register x (DWT_COMPx)

Address offset: 0x020 + 0x10 * x (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
COMP[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COMP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **COMP[31:0]**: Reference value for comparison

DWT function register x (DWT_FUNCTx)

Address offset: 0x028 + 0x10 * x (x = 0 to 7)

Reset value: 0x5800 0000, 0xF000 0000, 0x5000 0000, 0xF000 0000, 0x5000 0000, 0xD000 0000, 0x5000 0000, 0xD000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r			r								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DATAVSIZE[1:0]		Res.	Res.	Res.	Res.	ACTION[1:0]		MATCH[3:0]			
				rw	rw					rw	rw	rw	rw	rw	rw

Bits 31:27 **ID[4:0]**: Identify capability (read-only)
Identifies the MATCH[3:0] capabilities for the comparator.
0x08: Data address, and data address with value
0x09: Cycle counter, data address, and data address with value
0x0A: Instruction address, data address, and data address with value
0x0B: Cycle counter, instruction address, data address and data address with value
0x18: Data address, data address limit, and data address with value
0x1A: Instruction address, instruction address limit, data address, data address limit, and data address with value
0x1C: Data address, data address limit, data value, linked data value, and data address with value
0x1E: Instruction address, instruction address limit, data address, data address limit, data value, linked data value, and data address with value
Others: Reserved

Bits 26:25 Reserved, must be kept at reset value.

Bit 24 **MATCHED**: Comparator match (read-only)
Indicates if a comparator match has occurred since the register was last read.
0: No match
1: Match occurred

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:10 **DATAVSIZE[1:0]**: Size of required data comparison
For data value matching, specifies the size of the required data comparison.
0: Byte
1: Half word
2: Word
3: Reserved

Bits 9:6 Reserved, must be kept at reset value.

Bits 5:4 **ACTION[1:0]**: Action on match
Defines the action when a match occurs
0: Trigger only
1: Generate debug event
2: For a cycle counter, instruction address, data address, data value or linked data value comparator, generate a data trace match packet. For a data address with value comparator, generate a data trace data value packet
3: For a data address limit comparator, generate a data trace data address packet. For a cycle counter, instruction address limit, or data address comparator, generate a data trace PC value packet. For a data address with value comparator, generate both a data trace PC value packet and a data trace data value packet.

Bits 3:0 **MATCH[3:0]**: Match type

Controls the type of match generated by this comparator (see ID[4:0] for valid types).

0: Disabled. Never generates a match

1: Cycle counter. Matches if DWT_CYCCNT equals the comparator value. The comparator is checked each time DWT_CYCCNT is written to, directly or indirectly.

2: Instruction address. If not linked to, an instruction matches if the address of the first byte of the instruction matches the comparator address.

3: Instruction address limit. When comparator <n-1> is configured as instruction address (0b0010), an access generates a match if the first byte of the instruction lies between the lower address (specified by the comparator <n-1>), and the limit address (specified by this comparator, <n>). Both the lower and the limit address are inclusive to the address range

4: Data address. If not linked to by a data address limit comparator, an access matches if any accessed byte lies between the comparator address, and a limit defined by the DATAVSIZE field.

5: Data address, writes. As 4, except that only write accesses generate a match.

6: Data address, reads. As 0b0100, except that only read accesses generate a match.

7: Data address limit. An access generates a match when all of the following are true:

- Comparator <n-1> is configured as one of data address (4-6), or data address with value (12-14).
- Any accessed byte lies between the lower address (specified by comparator <n-1>), and the limit address (specified by this comparator <n>).
- Both the lower address, and limit address are inclusive to the range.

8: Data value. An access matches if the value accessed matches the comparator value.

9: Data value, writes. As 8, except that only write accesses generate a match.

10: Data value, reads. As 0b1000, except that only read accesses generate a match.

11: Linked data value. An access generates a match when all of the following are true:

- Comparator <n-1> is configured as data address (4-6), or data address with value (12-14).
- DATAVSIZE for the two comparators <n> and <n-1> are equal.
- The value accessed matches the comparator value specified by comparator <n>.
- Any accessed byte lies between the lower address (specified by comparator <n-1>), and limit defined by the DATAVSIZE field.
- Both the lower address, and the limit address are inclusive to the range.

12: Data address with value. As 4, except that the data value is traced

13: Data address with value, writes. As 12, except that only write accesses generate a match

14: Data address with value, reads. As 12, except that only read accesses generate a match

15: Reserved

DWT comparator value mask register x (DWT_VMASKx)

Address offset: 0x02C + 0x10 * x (x = 1,3)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
VMASK[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VMASK[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **VMASK[31:0]**: Data value mask.

Mask value for use in the comparison with load or store data. This only applies to comparators which support data value matching.

For every bit m, if VMASK[m] = 0, the comparison matches only if DWT_COMPx[m] matches bit [m] of the candidate data value. If VMASK[m] = 1, the comparison ignores bit [m] of the candidate data value.

DWT device type architecture register (DWT_DEVARCH)

Address offset: 0xFBC

Reset value: 0x4771 1A02

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARCHITECT[10:0]											PRESENT	REVISION[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARCHVER[3:0]				ARCHPART[11:0]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 **ARCHITECT[10:0]**: Architect JEDEC code

0x23B: Arm JEDEC code

Bit 20 **PRESENT**: DEVARCH register present

1: Present

Bits 19:16 **REVISION[3:0]**: Architecture revision

0x1: DWT architecture v2.1

Bits 15:12 **ARCHVER[3:0]**: Architecture version

0x1: DWT architecture v2

Bits 11:0 **ARCHPART[11:0]**: Architecture

0xA02: DWT architecture

DWT device type register (DWT_DEVTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUB[3:0]				MAJOR[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUB[3:0]**: Component subtype

0x0: Other

Bits 3:0 **MAJOR[3:0]**: Component major type
0x0: Miscellaneous

DWT CoreSight peripheral identity register 4 (DWT_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: Register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm JEDEC code

DWT CoreSight peripheral identity register 0 (DWT_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number field, bits [7:0]
0x02: DWT part number

DWT CoreSight peripheral identity register 1 (DWT_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0xB: Arm JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number field, bits [11:8]
0x0: DWT part number

DWT CoreSight peripheral identity register 2 (DWT_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x0: r0p0

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x3: Arm JEDEC code

DWT CoreSight peripheral identity register 3 (DWT_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

DWT CoreSight component identity register 0 (DWT_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value

DWT CoreSight component identity register 1 (DWT_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00E0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0xE: Trace generator component

Bits 3:0 **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

DWT CoreSight component identity register 2 (DWT_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

DWT CoreSight component identity register 3 (DWT_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value

DWT register map

Table 875. DWT register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	DWT_CTRL	NUMCOMP [3:0]				NOTRCPKT	NOEXTTRIG	NOCYCCNT	NOPRFCNT	CYCDISS	CYCEVTENA	FOLDEVTENA	LSUEVTENA	SLEEPEVTENA	EXCEVTENA	CPIEVTENA	EXCTRCENA	Res.	Res.	Res.	PCSAMPLENA	SYNCTAP[1:0]		CYCTAP	POSINIT[3:0]				POSTPRESET [3:0]				CYCCNTENA
	Reset value	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				0	0	0	0	0	0	0	0	0	0	0	0	0
0x004	DWT_CYCCNT	CYCCNT[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	DWT_CPICNT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CPICNT[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x00C	DWT_EXCCNT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXCCNT[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x010	DWT_SLEEPCNT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SLEEPCNT[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

Table 875. DWT register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x014	DWT_LSUCNT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LSUCNT[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x018	DWT_FOLDCNT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FOLDCNT[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x01C	DWT_PCSR	EIASAMPLE[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x020	DWT_COMP0	COMP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	Reserved	Reserved																															
0x028	DWT_FUNCT0	ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATAVSIZE [1:0]		Res.	Res.	Res.	Res.	ACTION [1:0]		MATCH[3:0]			
	Reset value	0	1	0	1	1			0													0	0					0	0	0	0	0	0
0x02C	Reserved	Reserved																															
0x030	DWT_COMP1	COMP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	Reserved	Reserved																															
0x038	DWT_FUNCT1	ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATAVSIZE [1:0]		Res.	Res.	Res.	Res.	ACTION [1:0]		MATCH[3:0]			
	Reset value	1	1	1	1	0			0													0	0					0	0	0	0	0	0
0x03C	DWT_VMASK1	VMASK[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x040	DWT_COMP2	COMP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x044	Reserved	Reserved																															
0x048	DWT_FUNCT2	ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATAVSIZE [1:0]		Res.	Res.	Res.	Res.	ACTION [1:0]		MATCH[3:0]			
	Reset value	0	1	0	1	0			0													0	0					0	0	0	0	0	0
0x04C	Reserved	Reserved																															
0x050	DWT_COMP3	COMP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x054	Reserved	Reserved																															
0x058	DWT_FUNCT3	ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATAVSIZE [1:0]		Res.	Res.	Res.	Res.	ACTION [1:0]		MATCH[3:0]			
	Reset value	1	1	0	1	0			0													0	0					0	0	0	0	0	0
0x05C	DWT_VMASK3	VMASK[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x060	DWT_COMP4	COMP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x064	Reserved	Reserved																															

Table 875. DWT register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x068	DWT_FUNCT4	ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATAVSIZE [1:0]		Res.	Res.	Res.	Res.	ACTION [1:0]		MATCH[3:0]			
	Reset value	0	1	0	1	0			0													0	0					0	0	0	0	0	0
0x06C	Reserved	Reserved																															
0x070	DWT_COMP5	COMP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x074	Reserved	Reserved																															
0x078	DWT_FUNCT5	ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATAVSIZE [1:0]		Res.	Res.	Res.	Res.	ACTION [1:0]		MATCH[3:0]			
	Reset value	1	1	0	1	0			0													0	0					0	0	0	0	0	0
0x07C	Reserved	Reserved																															
0x080	DWT_COMP6	COMP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x084	Reserved	Reserved																															
0x088	DWT_FUNCT6	ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATAVSIZE [1:0]		Res.	Res.	Res.	Res.	ACTION [1:0]		MATCH[3:0]			
	Reset value	0	1	0	1	0			0													0	0					0	0	0	0	0	0
0x08C	Reserved	Reserved																															
0x090	DWT_COMP7	COMP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x094	Reserved	Reserved																															
0x098	DWT_FUNCT7	ID[4:0]					Res.	Res.	MATCHED	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DATAVSIZE [1:0]		Res.	Res.	Res.	Res.	ACTION [1:0]		MATCH[3:0]			
	Reset value	1	1	0	1	0			0													0	0					0	0	0	0	0	0
0x09C-0xFB8	Reserved	Reserved																															
0xFBC	DWT_DEVARCH	ARCHITECT[10:0]											PRESENT	REVISION [3:0]				ARCHVER [3:0]				ARCHPART[11:0]											
	Reset value	0	1	0	0	0	1	1	1	0	1	1	1	0	0	0	1	0	0	0	1	1	0	1	0	0	0	0	0	0	0	1	0
0xFC0-0xFC8	Reserved	Reserved																															
0xFCC	DWT_DEVTYPE	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	SUB[3:0]				MAJOR[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFD0	DWT_PIDR4	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	SIZE[3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4-0xFDC	Reserved	Reserved																															
0xFE0	DWT_PIDR0	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	PARTNUM[7:0]							
	Reset value																									0	0	1	0	0	0	1	1

Table 875. DWT register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFE4	DWT_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	1	0	0	1
0xFE8	DWT_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	1
0xFEC	DWT_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	DWT_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	DWT_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									0	1	1	1	0	0	0	0
0xFF8	DWT_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	DWT_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

78.11.5 Cortex-M55 performance monitoring unit (PMU)

The PMU enables software to get information about events that are taking place in the processor, and can be used for performance analysis and system debug.

The PMU supports eight 16-bit event counters, and one 32-bit cycle counter. Each event counter can count one event from the list of PMU events in the Arm Cortex-M55 Processor Technical Reference Manual *[4]*. The PMU also supports a chain function, which allows the PMU to cascade two of the 16-bit counters into one 32-bit counter. Only odd event counters support the chain feature. PMU counters increment if the corresponding enable bit is set in PMU_CNTENSET.

The PMU cycle counter PMU_CCNTR is an alias of the DWT_CYCCNT register. All derived functions of the counter are available whenever either the DWT or the PMU enables the cycle counter.

Generating interrupts

If a counter is configured to generate an interrupt when it overflows, a debug monitor exception becomes pending: MON_PEND bit is set to 1 in the Debug Exception and Monitor Control Register (DEMCR) of the Cortex-M55 core debug registers. At the same time, the PMU bit in the Cortex-M55 Debug Fault Status Register (DFSR) is set to 1.
The associated overflow bit programmed by PMU_OVSSET and PMU_OVSCLR indicates which counter triggered the exception.

The interrupts are enabled if their corresponding bit programmed by PMU_INTENSET, and PMU_INTENCLR is set and DEMCR.MON_EN is 1.

Exporting trace

The PMU can export trace whenever the lower 8 bits of the counters overflow. The PMU issues an event counter packet with the appropriate counter flag set to 1. This occurs on counter increment only, not on software or debugger write. For each counter n, if the lower 8 bits of that counter overflows, the associated OVn bit of the event counter packet is set.

If multiple counters overflow during the same period, multiple bits may be set.

The PMU can serve as an event source for the CTI.

78.11.6 Cortex-M55 PMU registers

The register file base address for the PMU is defined in *Table 873: Processor ROM table*.

PMU event counter register x (PMU_EVCNTRx)

Address offset: 0x000 + 0x4 * x (x = 0 to 7)

Reset value: 0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
COUNTER[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **COUNTER[15:0]**: Event counter

The counter increments whenever the selected event occurs.

PMU cycle counter register (PMU_CCNTR)

Address offset: 0x07C

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CCNT[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CCNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CCNT[31:0]**: Cycle count

The counter increments every processor clock cycle.

PMU event type and filter register (PMU_EVTYPERx)

Address offset: 0x400 + 0x4 * x (x = 0 to 7)

Reset value: 0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EVTCOUNT[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **EVTCOUNT[15:0]**: Event to count

The number of the event that is counted by event counter PMU_EVCNTRx.

PMU count enable set register (PMU_CNTENSET)

Address offset: 0xC00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
C	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PN[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **C:** Cycle-count enable

Condition: write operation
0: No effect
1: Enable cycle counter.
Condition: read operation
0: Cycle counter disabled
1: Cycle counter enabled

Bits 30:8 Reserved, must be kept at reset value.

Bits 7:0 **PN[7:0]**: Event counter enable

Condition: write operation
Writing a 1 to bit n enables event counter PMU_EVCNTRn. Writing a 0 has no effect.
Condition: read operation
When bit n is 0, event counter PMU_EVCNTRn is disabled.
When bit n is 1, event counter PMU_EVCNTRn is enabled.

PMU count enable clear register (PMU_CNTENCLR)

Address offset: 0xC20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
C	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PN[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **C:** Cycle-count disable
Condition: write operation
0: No effect
1: Disable cycle counter.
Condition: read operation
0: Cycle counter disabled
1: Cycle counter enabled

Bits 30:8 Reserved, must be kept at reset value.

Bits 7:0 **PN[7:0]**: Event counter disable
Condition: write operation
Writing a 1 to bit n disables event counter PMU_EVCNTRn. Writing a 0 has no effect.
Condition: read operation
When bit n is 0, event counter PMU_EVCNTRn is disabled.
When bit n is 1, event counter PMU_EVCNTRn is enabled.

PMU count interrupt set register (PMU_INTENSET)

Address offset: 0xC40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
C	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PN[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **C:** Cycle-counter-overflow interrupt request enable
Condition: write operation
0: No effect
1: Enable cycle-counter-overflow interrupt request.
Condition: read operation
0: Cycle-counter-overflow interrupt request disabled
1: Cycle-counter-overflow interrupt request enabled

Bits 30:8 Reserved, must be kept at reset value.

Bits 7:0 **PN[7:0]**: Event-counter interrupt request enable
Condition: write operation
Writing a one to bit n enables the interrupt request for event counter PMU_EVCNTRn.
Writing a 0 has no effect.
Condition: read operation
When bit n is 0, the event-counter interrupt request for PMU_EVCNTRn is disabled.
When bit n is 1, the event-counter interrupt request for PMU_EVCNTRn is enabled.

PMU count interrupt clear register (PMU_INTENCLR)

Address offset: 0xC60

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
C	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PN[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **C:** Cycle-counter-overflow interrupt request disable
Condition: write operation
0: No effect
1: Disable cycle-counter-overflow interrupt request.
Condition: read operation
0: Cycle-counter-overflow interrupt request disabled
1: Cycle-counter-overflow interrupt request enabled

Bits 30:8 Reserved, must be kept at reset value.

Bits 7:0 **PN[7:0]**: Event-counter interrupt request disable
Condition: write operation
Writing a one to bit n disables the interrupt request for event counter PMU_EVCNTRn.
Writing a 0 has no effect.
Condition: read operation
When bit n is 0, the event-counter interrupt request for PMU_EVCNTRn is disabled.
When bit n is 1, the event-counter interrupt request for PMU_EVCNTRn is enabled.

PMU overflow flag status clear register (PMU_OVSCLR)

Address offset: 0xC80

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
C	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PN[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **C:** Cycle-counter-overflow flag clear
Condition: write operation
0: No effect
1: Clear cycle-counter-overflow flag.
Condition: read operation
0: Cycle counter has not overflowed.
1: Cycle counter has overflowed.

Bits 30:8 Reserved, must be kept at reset value.

Bits 7:0 **PN[7:0]**: Even- counter-overflow flag clear
Condition: write operation
Writing a one to bit n clears the overflow flag for event counter PMU_EVCNTRn. Writing a 0 has no effect
Condition: read operation
When bit n is 0, the event counter for PMU_EVCNTRn has not overflowed.
When bit n is 1, the event counter for PMU_EVCNTRn has overflowed.

PMU software increment register (PMU_SWINC)

Address offset: 0xCA0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PN[7:0]							
								w	w	w	w	w	w	w	w

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PN[7:0]**: Event counter software increment
Writing a one to bit n increments event counter PMU_EVCNTRn, if it is configured for software increments. Writing a 0 has no effect.

PMU overflow flag status set register (PMU_OVSSET)

Address offset: 0xCC0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
C	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PN[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 **C:** Cycle-counter-overflow flag set
Condition: write operation
0: No effect
1: Set cycle-counter-overflow flag.
Condition: read operation
0: Cycle counter has not overflowed.
1: Cycle counter has overflowed.

Bits 30:8 Reserved, must be kept at reset value.

Bits 7:0 **PN[7:0]**: Event-counter-overflow flag set
Condition: write operation
Writing a one to bit n sets the overflow flag for event counter PMU_EVCNTRn. Writing a 0 has no effect
Condition: read operation
When bit n is 0, the event counter for PMU_EVCNTRn has not overflowed.
When bit n is 1, the event counter for PMU_EVCNTRn has overflowed.

PMU type register (PMU_TYPE)

Address offset: 0xE00

Reset value: 0x00A0 5F08

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRO	Res.	FZO	Res.	Res.	Res.	Res.	Res.
								r		r					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	CC	SIZE[5:0]						N[7:0]							
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **TRO:** Trace on overflow support
1: Trace on overflow supported

Bit 22 Reserved, must be kept at reset value.

Bit 21 **FZO:** Freeze on overflow support
1: Freeze on overflow mechanism supported

Bits 20:15 Reserved, must be kept at reset value.

Bit 14 **CC:** Cycle counter present
1: Dedicated cycle counter is present.

Bits 13:8 **SIZE[5:0]**: Counter size
This field indicates the spacing of counters in the memory-map.
0x1F: 32 bits

Bits 7:0 **N[7:0]**: Number of event counters
0x8

PMU type register (PMU_CTRL)

Address offset: 0xE04

Reset value: 0x0000 XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TRO	Res.	FZO	Res.	Res.	Res.	DP	Res.	Res.	C	P	E
				rw		rw				rw			w	w	rw

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **TRO:** Trace on overflow enable
0: Trace on overflow disabled
1: Trace when any of the event counters overflows an 8-bit value

Bit 10 Reserved, must be kept at reset value.

Bit 9 **FZO:** Freeze on overflow enable
0: Event counters continue counting while the overflow flag is set.
1: Event counters stop while the overflow flag is set.

Bits 8:6 Reserved, must be kept at reset value.

Bit 5 **DP**: Controls whether the cycle counter is disabled in secure state
. This bit is an alias of CYCDISS in DWT_CTRL.
0: No effect
1: Cycle counter is not incremented when the processor is in secure state.

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **C**: Cycle counter reset
0: No effect
1: Reset PMU_CCNTR to 0.

Bit 1 **P**: Event counter reset
0: No effect
1: Reset all event counters, not including PMU_CCNTR, to 0.

Bit 0 **E**: Event counter enable
0: All counters, including PMU_CCNTR, are disabled.
1: All counters are enabled by PMU_CNTENSET.

PMU overflow flag status set register (PMU_AUTHSTAT)

Address offset: 0xFB8

Reset value: 0x0000 00XX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **SNID[1:0]**: Secure non-invasive debug
10: Secure non-invasive debug prohibited
11: Secure non-invasive debug allowed

Bits 5:4 **SID[1:0]**: Secure invasive debug
10: Secure invasive debug prohibited
11: Secure invasive debug allowed

Bits 3:2 **NSNID[1:0]**: Nonsecure non-invasive debug
10: Nonsecure non-invasive debug prohibited
11: Nonsecure non-invasive debug allowed

Bits 1:0 **NSID[1:0]**: Nonsecure invasive debug
10: Nonsecure invasive debug prohibited
11: Nonsecure invasive debug allowed

PMU device type architecture register (PMU_DEVARCH)

Address offset: 0xFBC

Reset value: 0x4770 0A06

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARCHITECT[10:0]											PRESENT	REVISION[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARCHVER[3:0]				ARCHPART[11:0]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 **ARCHITECT[10:0]**: Architect JEDEC code
0x23B: Arm JEDEC code

Bit 20 **PRESENT**: DEVARCH register present
1: Present

Bits 19:16 **REVISION[3:0]**: Architecture revision
0x0: PMU architecture minor revision

Bits 15:12 **ARCHVER[3:0]**: Architecture version
0x0: PMU architecture version

Bits 11:0 **ARCHPART[11:0]**: Architecture
0xA06: PMU architecture part number

PMU device type register (PMU_DEVTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0016

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUB[3:0]				MAJOR[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUB[3:0]**: Component subtype
0x1

Bits 3:0 **MAJOR[3:0]**: Component major type
0x6

PMU CoreSight peripheral identity register 4 (PMU_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: Register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm JEDEC code

PMU CoreSight peripheral identity register 0 (PMU_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0023

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number field, bits [7:0]
0x23: PMU part number

PMU CoreSight peripheral identity register 1 (PMU_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00BD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0xB: Arm JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number field, bits [11:8]
0xD: PMU part number

PMU CoreSight peripheral identity register 2 (PMU_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x0: r0p0

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x3: Arm JEDEC code

PMU CoreSight peripheral identity register 3 (PMU_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

PMU CoreSight component identity register 0 (PMU_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value

PMU CoreSight component identity register 1 (PMU_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0x9: PMU

Bits 3:0 **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

PMU CoreSight component identity register 2 (PMU_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

PMU CoreSight component identity register 3 (PMU_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value

PMU register map

Table 876. PMU register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000 + 0x4 * x (x=0 to 7) Last address: 0x01C	PMU_EVCNTRx	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COUNTER[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020-0x078	Reserved	Reserved																															
0x07C	PMU_CCNTR	CCNT[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x080-0x3FC	Reserved	Reserved																															

Table 876. PMU register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x400 + 0x4 * x (x=0 to 7) Last address: 0x41C	PMU_EVTYPERx	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EVTCNT[15:0]															
	Reset value																	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x420-0xBFC	Reserved	Reserved																															
0xC00	PMU_CNTENSET	C	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PN[7:0]							
	Reset value	0																								0	0	0	0	0	0	0	0
0xC04-0xC1C	Reserved	Reserved																															
0xC20	PMU_CNTENCLR	C	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PN[7:0]							
	Reset value	0																								0	0	0	0	0	0	0	0
0xC24-0xC3C	Reserved	Reserved																															
0xC40	PMU_INTENSET	C	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PN[7:0]							
	Reset value	0																								0	0	0	0	0	0	0	0
0xC44-0xC5C	Reserved	Reserved																															
0xC60	PMU_INTENCLR	C	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PN[7:0]							
	Reset value	0																								0	0	0	0	0	0	0	0
0xC64-0xC7C	Reserved	Reserved																															
0xC80	PMU_OVSCLR	C	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PN[7:0]							
	Reset value	0																								0	0	0	0	0	0	0	0
0xC84-0xC9C	Reserved	Reserved																															
0xCA0	PMU_SWINC	C	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PN[7:0]							
	Reset value	0																								0	0	0	0	0	0	0	0
0xCA4 - 0xCBC	Reserved	Reserved																															
0xCC0	PMU_OVSSET	C	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PN[7:0]							
	Reset value	0																								0	0	0	0	0	0	0	0
0xCC4-0xDFC	Reserved	Reserved																															
0xE00	PMU_TYPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRO	Res.	FZO	Res.	Res.	Res.	Res.	Res.	Res.	CC	SIZE[5:0]							N[7:0]							
	Reset value								1		1							1	0	1	1	1	1	1	1	0	0	0	0	1	0	0	0
0xE04	PMU_CTRL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRO	Res.	FZO	Res.	Res.	Res.	DP	Res.	Res.	C	P	E
	Reset value																					X		X				X			X	X	X
0xE04-0xFB4	Reserved	Reserved																															
0xFB8	PMU_AUTHSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID [1:0]		SID [1:0]		NSNID [1:0]		NSID [1:0]	
	Reset value																									X	X	X	X	X	X	X	X

Table 876. PMU register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFBC	PMU_DEVARCH	ARCHITECT[10:0]											PRESENT	REVISION [3:0]				ARCHVER [3:0]				ARCHPART[11:0]											
	Reset value	0	1	0	0	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	1	1	0
0xFC0-0xFC8	Reserved	Reserved																															
0xFCC	PMU_DEVTYPE	Res.																								SUB[3:0]				MAJOR[3:0]			
	Reset value																									0	0	0	1	0	1	1	0
0xFD0	PMU_PIDR4	Res.																								SIZE[3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4-0xFDC	Reserved	Reserved																															
0xFE0	PMU_PIDR0	Res.																								PARTNUM[7:0]							
	Reset value																									0	0	1	0	0	0	1	1
0xFE4	PMU_PIDR1	Res.																								JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	1	1	0	1
0xFE8	PMU_PIDR2	Res.																								REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	1
0xFEC	PMU_PIDR3	Res.																								REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	PMU_CIDR0	Res.																								PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	PMU_CIDR1	Res.																								CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	PMU_CIDR2	Res.																								PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	PMU_CIDR3	Res.																								PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

78.11.7 Instrumentation trace macrocell (ITM)

The ITM generates trace information as packets. There are four sources that can generate packets. If multiple sources generate packets at the same time, the ITM arbitrates the order in which packets are output. The four sources in decreasing order of priority are:

- Software trace (highest priority)

  The software can write directly to any of 32 x 32-bit ITM stimulus registers to generate packets. The permission level for each port can be programmed. When software writes to an enabled stimulus port, the ITM combines the port identity, the write access size, and data written, into a packet that it writes to a FIFO. The ITM outputs packets from the FIFO onto the trace bus. Reading a stimulus port register returns the status of the stimulus register (empty or pending) in bit 0.

- Hardware trace

  The DWT generates trace packets in response to a data trace event, a PC sample or a performance profiling counter wraparound. The ITM outputs these packets on the trace bus.
- Local timestamping

  The ITM contains a 21-bit counter clocked by the (pre-divided) processor clock. The counter value is output in a timestamp packet on the trace bus. The counter is reset every time a timestamp packet is generated. The timestamps thus indicate the time elapsed since the previous timestamp packet.
- Global system timestamping (lower priority)

  Timestamps can also be generated using the system-wide 64-bit count value coming from the timestamp generator component.

For more details on how to use the ITM, refer to the Arm Cortex-M55 Processor Technical Reference Manual *[4]*.

78.11.8 Cortex-M55 ITM registers

The register file base address for the ITM is defined in *Table 873: Processor ROM table*.

ITM stimulus register x (ITM_STIMx)

Address offset: 0x000 + 0x4 * x (x = 0 to 31)

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STIMULUS[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STIMULUS[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **STIMULUS[31:0]**: Software event packet/disabled/fifoready
Write data are output on the trace bus as a software event packet.
When reading, bit 0 is a fifoready indicator:
0: Stimulus port buffer is full (or port is disabled).
1: Stimulus port can accept new write data.
When reading, bit 1 is a disabled indicator:
0: Stimulus port and ITM are enabled.
1: Stimulus port or ITM are disabled.

ITM trace enable register (ITM_TER)

Address offset: 0xE00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STIMENA[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STIMENA[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **STIMENA[31:0]**: Stimulus port enable

Each bit n (31:0) enables the stimulus port associated with ITM_STIMn register.

0: Port disabled

1: Port enabled

ITM trace privilege registers (ITM_TPR)

Address offset: 0xE40

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIVMASK[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PRIVMASK[3:0]**: Enable unprivileged access to ITM stimulus ports

Each bit controls eight stimulus ports:

xxx0: Unprivileged access permitted on ports 0 to 7

xxx1: Only privileged access permitted on ports 0 to 7

xx0x: Unprivileged access permitted on ports 8 to 15

xx1x: Only privileged access permitted on ports 8 to 15

x0xx: Unprivileged access permitted on ports 16 to 23

x1xx: Only privileged access permitted on ports 16 to 23

0xxx: Unprivileged access permitted on ports 24 to 31

1xxx: Only privileged access permitted on ports 24 to 31

ITM trace control register (ITM_TCR)

Address offset: 0xE80

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSY	TRACEBUSID[6:0]						
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	GTSFREQ[1:0]		TSPRESCALE [1:0]		Res.	Res.	STALL ENA	SWO ENA	TX ENA	SYNC ENA	TS ENA	ITM ENA
				rw	rw	rw	rw			rw	r	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **BUSY**: ITM busy

Indicates whether the ITM is currently processing events (read-only):
0: Not busy
1: Busy

Bits 22:16 **TRACEBUSID[6:0]**: Identifier for multi-source trace stream formatting

If multi-source trace is in use, the debugger must write a non-0 value to this field.

Note: Different IDs must be used for each trace source in the system.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:10 **GTSFREQ[1:0]**: Global timestamp frequency

Defines how often the ITM generates a global timestamp, based on the global timestamp clock frequency, or disables generation of global timestamps.
0x0: Disable generation of global timestamps.
0x1: Generate timestamp request whenever the ITM detects a change in global timestamp counter bits [63:7]; this is approximately every 128 cycles.
0x2: Generate timestamp request whenever the ITM detects a change in global timestamp counter bits [63:13]; this is approximately every 8192 cycles.
0x3: Generate a timestamp after every packet, if the output FIFO is empty.

Bits 9:8 **TSPRESCALE[1:0]**: Local timestamp prescale

Prescaler used with the trace packet reference clock.
0x0: No prescaling
0x1: Divide by 4.
0x2: Divide by 16.
0x3: Divide by 64.

Bits 7:6 Reserved, must be kept at reset value.

Bit 5 **STALLENA**: Stall the processor to guarantee delivery of trace packets

0: Drop hardware source packets, and generate an overflow packet if the ITM output is stalled.
1: Stall the processor to guarantee delivery of trace packets.

Bit 4 **SWOENA**: Asynchronous clocking enable for the timestamp counter (read-only)

0: Timestamp counter uses the processor clock.

Bit 3 **TXENA**: Hardware event packet forwarding enable

This bit enables forwarding of hardware event packets from the DWT unit to the trace port.
0: Disabled
1: Enabled

Bit 2 **SYNCENA**: Synchronization packet transmission enable

If a debugger sets this bit, it must also configure SYNCTAP in DWT_CTRL for the correct synchronization speed.
0: Disabled
1: Enabled

Bit 1 **TSENA**: Local timestamp generation enable

0: Disabled
1: Enabled

Bit 0 **ITMENA**: ITM enable

0: Disabled
1: Enabled

ITM device type architecture register (ITM_DEVARCH)

Address offset: 0xFBC

Reset value: 0x4770 1A01

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARCHITECT[10:0]											PRESENT	REVISION[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARCHVER[3:0]				ARCHPART[11:0]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 **ARCHITECT[10:0]**: Architect JEDEC code
0x23B: Arm JEDEC code

Bit 20 **PRESENT**: DEVARCH register present
1: Present

Bits 19:16 **REVISION[3:0]**: Architecture revision
0x0: ITM architecture v2.0

Bits 15:12 **ARCHVER[3:0]**: Architecture version
0x1: ITM architecture v2

Bits 11:0 **ARCHPART[11:0]**: Architecture
0xA01: ITM architecture

ITM device type register (ITM_DEVTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0043

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUB[3:0]				MAJOR[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUB[3:0]**: Component subtype
0x4: Associated with a bus, stimulus derived from bus activity.

Bits 3:0 **MAJOR[3:0]**: Component major type
0x3: Trace source

ITM CoreSight peripheral identity register 4 (ITM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: Register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm JEDEC code

ITM CoreSight peripheral identity register 0 (ITM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number field, bits [7:0]
0x01: ITM part number

ITM CoreSight peripheral identity register 1 (ITM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0xB: Arm JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number field, bits [11:8]
0x1: ITM part number

ITM CoreSight peripheral identity register 2 (ITM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x0: r0p0

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x3: Arm JEDEC code

ITM CoreSight peripheral identity register 3 (ITM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

ITM CoreSight component identity register 0 (ITM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value

ITM CoreSight component identity register 1 (ITM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00E0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0xE: Trace generator component

Bits 3:0 **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

ITM CoreSight component identity register 2 (ITM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

ITM CoreSight component identity register 3 (ITM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value

ITM register map

Table 877. ITM register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000 + 0x4 * x (x=0 to 31) Last address: 0x07C	ITM_STIMx	STIMULUS[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x080-0xDFC	Reserved	Reserved																															
0xE00	ITM_TER	STIMENA[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xE04-0xE3C	Reserved	Reserved																															
0x0E40	ITM_TPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIVMASK [3:0]			
	Reset value																													0	0	0	0
0xE44-0xE7C	Reserved	Reserved																															
0xE80	ITM_TCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSY	TRACEBUSID[6:0]							Res.	Res.	Res.	Res.	GTSFREQ[1:0]		TSPRESCALE [1:0]		Res.	Res.	STALLENA	SWOENA	TXENA	SYNCENA	TSENA	ITMENA
	Reset value									0	0	0	0	0	0	0	0					0	0	0	0			0	0	0	0	0	0
0xE84-0xFB8	Reserved	Reserved																															
0xFBC	ITM_DEVARCH	ARCHITECT[10:0]											PRESENT	REVISION [3:0]				ARCHVER [3:0]				ARCHPART[11:0]											
	Reset value	0	1	0	0	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0	1
0xFC0-0xFC8	Reserved	Reserved																															
0xFCC	ITM_DEVTYPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUB[3:0]				MAJOR[3:0]			
	Reset value																									0	1	0	0	0	0	1	1

Table 877. ITM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFD0	ITM_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4-0xFDC	Reserved	Reserved																															
0xFE0	ITM_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	0	0	0	0	0	1
0xFE4	ITM_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value																									1	0	1	1	0	0	0	0
0xFE8	ITM_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value																									0	0	0	0	1	0	1	1
0xFEC	ITM_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	ITM_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	ITM_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value																									1	1	1	0	0	0	0	0
0xFF8	ITM_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	ITM_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

78.11.9 Breakpoint Unit (BPU)

The BPU allows hardware breakpoints to be set. It contains eight comparators which monitor the instruction fetch address. If a match occurs, the instruction comparators can be configured to generate a breakpoint instruction.

78.11.10 Cortex-M55 BPU registers

The register file base address for the BPU is defined in *Table 873: Processor ROM table*.

BPU control register (BPU_CTRL)

Address offset: 0x000

Reset value: 0x1000 0080

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REV[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r												

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	NUM_CODE[6:4]			NUM_LIT[3:0]				NUM_CODE[3:0]				Res.	Res.	KEY	ENABLE
	r	r	r	r	r	r	r	r	r	r	r			rw	rw

Bits 31:28 **REV[3:0]**: BPU architecture revision
1: Revision 2

Bits 27:15 Reserved, must be kept at reset value.

Bits 14:12 **NUM_CODE[6:4]**: Instruction address comparator number (three MSBs)
This read-only field holds the three MSBs of the number of instruction address comparators supported.
0x0: the MSBs of the number are all 0.

Bits 11:8 **NUM_LIT[3:0]**: Number of literal address comparators supported (read-only)
0x0: No literal comparators supported

Bits 7:4 **NUM_CODE[3:0]**: Instruction address comparator number (four LSBs)
This read-only field holds the four LSBs of the number of instruction address comparators supported.
0x8: Eight instruction comparators supported

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **KEY**: Write protect key
A write to BPU_CTRL register is ignored if this bit is not set to 1.

Bit 0 **ENABLE**: BPU enable
0: Disabled
1: Enabled

BPU remap register (BPU_REMAP)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	RMPSPT	REMAP[23:11]												
		r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REMAP[10:0]											Res.	Res.	Res.	Res.	Res.
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw					

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **RMPSPT**: Flash patch remap support (read-only)
0: Remapping not supported

Bits 28:5 **REMAP[23:0]**: Not supported

Bits 4:0 Reserved, must be kept at reset value.

BPU comparator registers (BPU_COMPx)

Address offset: 0x008 + 0x4 * x (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BPADDR[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BPADDR[15:1]															ENABLE
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:1 **BPADDR[31:1]**: Breakpoint address
Specifies bits[31:1] of the breakpoint instruction address.

Bit 0 **ENABLE**: Comparator enable
The comparator is only enabled if both this bit and BPU ENABLE in BPU_CTRL are set.
0: Disabled
1: Enabled

BPU device type architecture register (BPU_DEVARCH)

Address offset: 0xFBC

Reset value: 0x4770 1A03

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARCHITECT[10:0]											PRESENT	REVISION[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARCHVER[3:0]				ARCHPART[11:0]											
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 **ARCHITECT[10:0]**: Architect JEDEC code
0x23B: Arm JEDEC code

Bit 20 **PRESENT**: DEVARCH register present
1: Present

Bits 19:16 **REVISION[3:0]**: Architecture revision
0x0: BPU architecture v2.0

Bits 15:12 **ARCHVER[3:0]**: Architecture version
0x1: BPU architecture v2

Bits 11:0 **ARCHPART[11:0]**: Architecture
0xA03: BPU architecture

BPU CoreSight peripheral identity register 4 (BPU_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: Register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm JEDEC code

BPU CoreSight peripheral identity register 0 (BPU_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0023

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number field, bits [7:0]
0x23: BPU part number

BPU CoreSight peripheral identity register 1 (BPU_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00BD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0xB: Arm JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number field, bits [11:8]
0xD: BPU part number

BPU CoreSight peripheral identity register 2 (BPU_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x0: r0p0

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x3: Arm JEDEC code

BPU CoreSight peripheral identity register 3 (BPU_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

BPU CoreSight component identity register 0 (BPU_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value

BPU CoreSight component identity register 1 (BPU_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0x9: BPU

Bits 3:0 **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

BPU CoreSight component identity register 2 (BPU_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

BPU CoreSight component identity register 3 (BPU_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value

BPU register map

Table 878. BPU register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	BPU_CTRL	REV[3:0]				Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NUM_CODE [6:4]			NUM_LIT [3:0]				NUM_CODE [3:0]				Res.	Res.	KEY	ENABLE
	Reset value	0	0	0	1														0	0	0	0	0	0	0	1	0	0	0			0	0
0x004	BPU_REMAP	Res.	Res.	RMPSP	REMAP[23:0]																								Res.	Res.	Res.	Res.	Res.
	Reset value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					
0x008 + 0x4 * x (x=0 to 7) Last address: 0x020	BPU_COMPx	BPADDR[31:1]																														Res.	ENABLE
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024-0xFB8	Reserved	Reserved																															
0xFBC	BPU_DEVARCH	ARCHITECT[10:0]											PRESENT	REVISION [3:0]				ARCHVER [3:0]				ARCHPART[11:0]											
	Reset value	0	1	0	0	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0	1	1
0xFC0-0xFC8	Reserved	Reserved																															
0xFD0	BPU_PIDR4	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	SIZE[3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4-0xFDC	Reserved	Reserved																															
0xFE0	BPU_PIDR0	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	PARTNUM[7:0]							
	Reset value																									0	0	1	0	0	0	1	1

Table 878. BPU register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFE4	BPU_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	1	1	0	1
0xFE8	BPU_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	1
0xFEC	BPU_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	BPU_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	BPU_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	BPU_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	BPU_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

78.11.11 Embedded Trace Macrocell (ETM)

The Cortex-M55 ETM is a CoreSight component closely coupled to the CPU. The ETM generates trace packets that allow the execution of the Cortex-M55 core to be traced.

Note: Data accesses are not included in the trace information.

The ETM receives information from the CPU over the processor trace interface, including:

- the number of instructions executed in the same cycle
- changes in program flow
- the current processor instruction state
- addresses of memory locations accessed by load and store instructions
- type, direction and size of a transfer
- condition code information
- exception information
- wait for interrupt state information

For more information, refer to the Arm CoreSight ETM-M55 Technical Reference Manual *[5]*.

78.11.12 Cortex-M55 ETM registers

The register file base address for the ETM is defined in *Table 873: Processor ROM table*.

ETM programming control register (ETM_PRGCTL)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **EN**: Trace program enable
0: Trace unit disabled
1: Trace unit enabled

ETM status register (ETM_STAT)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PMSTABLE	IDLE
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **PMSTABLE**: Programmers model stable
This bit indicates whether ETM registers are stable and can be read.
0: Registers not stable
1: Registers stable

Bit 0 **IDLE**: Trace unit inactive
0: ETM not idle
1: ETM idle

ETM trace configuration register (ETM_CONFIG)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	RS	TS	COND[2:0]			Res.	Res.	Res.	CCI	BB	INSTP0[1:0]		Res.
			rw	rw	rw	rw	rw				rw	rw	r	r	

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **RS**: Return stack enable
0: Disabled
1: Enabled

Bit 11 **TS**: Global timestamp tracing
0: Disabled
1: Enabled

Bits 10:8 **COND[2:0]**: Conditional instruction tracing
0x0: Conditional instruction tracing disabled
0x1: Conditional load instructions are traced.
0x2: Conditional store instructions are traced.
0x3: Conditional load and store instructions are traced.
0x7: All conditional instructions are traced.
Others: Reserved

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **CCI**: Cycle counting in instruction trace
0: Disabled
1: Enabled

Bit 3 **BB**: Branch broadcast mode
0: Disabled
1: Enabled

Bits 2:1 **INSTP0[1:0]**:
This bitfield determines which instructions are P0 instructions (read-only)
0x0: Only branches are P0 instructions.

Bit 0 Reserved, must be kept at reset value.

ETM event control 0 register (ETM_EVENTCTL0)

Address offset: 0x020

Reset value: 0x0000 0000

Writes to this register are accepted on when the trace unit is disabled.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EVENT1[7:0]								EVENT0[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:8 **EVENT1[7:0]**: Event selector 1

Bits 7:0 **EVENT0[7:0]**: Event selector 0

ETM event control 1 register (ETM_EVENTCTL1)

Address offset: 0x024

Reset value: 0x0000 0000

Writes to this register are accepted on when the trace unit is disabled.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	LPOVERRIDE	ATB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INSTEN[1:0]	
			rw	rw										rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **LPOVERRIDE**: Low-power state behavior override
0: Low-power state normal behavior
1: Entry to low-power state does not affect resources and event trace generation.

Bit 11 **ATB**: ATB trigger enable
0: Disabled
1: Enabled

Bits 10:2 Reserved, must be kept at reset value.

Bits 1:0 **INSTEN[1:0]**: Instruction trace event element enable
Each bit corresponds to an event:
x0: Event 0 does not cause an event element.
x1: Event 0 causes an event element.
0x: Event 1 does not cause an event element.
1x: Event 1 causes an event element.

ETM stall control register (ETM_STALLCTL)

Address offset: 0x02C

Reset value: 0x0000 0000

Writes to this register are accepted on when the trace unit is disabled.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	ISTALL	Res.	Res.	Res.	Res.	LEVEL[1:0]		Res.	Res.
							rw					rw	rw		

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **ISTALL**: Stall processor based on instruction trace buffer space
0: Do not stall processor.
1: Stall processor.

Bits 7:4 Reserved, must be kept at reset value.

Bits 3:2 **LEVEL[1:0]**: Stalling threshold level
A low level minimizes the amount of processor stalling, with a higher risk of FIFO overflow. A high level minimizes the risk of FIFO overflow but increases the amount of processor stalling.

Bits 1:0 Reserved, must be kept at reset value.

ETM global timestamp control register (ETM_TSCTL)

Address offset: 0x030

Reset value: 0x0000 0000

Writes to this register are accepted on when the trace unit is disabled.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EVENT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **EVENT[7:0]**: Event selector

When the selected event is triggered, the ETM inserts a global timestamp into the trace streams.

ETM synchronization period register (ETM_SYNCP)

Address offset: 0x034

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERIOD[4:0]				
											r	r	r	r	r

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **PERIOD[4:0]**: Trace bytes between synchronization requests

This bitfield defines the number of bytes of trace information between trace synchronization requests.

0xA: 1024 bytes

ETM cycle count control register (ETM_CCCTL)

Address offset: 0x038

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	THRESHOLD[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **THRESHOLD[11:0]**: Threshold value for instruction trace cycle counting
The threshold represents the minimum interval between cycle count trace packets.
0x0 - 0x03: Reserved
Other: Threshold

ETM trace ID register (ETM_TRACEID)

Address offset: 0x040

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRACEID[6:0]						
									rw	rw	rw	rw	rw	rw	rw

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:0 **TRACEID[6:0]**: Trace ID
0x01 to 0x6F: Valid ID
Others: Reserved

ETM ViewInst main control register (ETM_VICTL)

Address offset: 0x080

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXLEVEL_S3	Res.	Res.	EXLEVEL_S0
												rw			rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TRCERR	TRCRESET	SSSTATUS	Res.	EVENT[7:0]							
				rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **EXLEVEL_S3**: Trace disable, exception level 3 (handler mode)
This bit disables tracing in the specified exception level in secure state for exception level 3.
0: Enable ViewInst in this exception level.
1: Disable ViewInst in this exception level.

Bits 18:17 Reserved, must be kept at reset value.

Bit 16 **EXLEVEL_S0**: Trace disable, exception level 0 (thread mode)
This bit disables tracing in the specified exception level in secure state for exception level 0.
0: Enable ViewInst in this exception level.
1: Disable ViewInst in this exception level.

Bits 15:12 Reserved, must be kept at reset value.

Bit 11 **TRCERR**: Tracing of system error exception
This bit selects whether a system error exception must always be traced.
0: System error exception is traced only if the instruction or exception immediately before the system error exception is traced.
1: System error exception is always traced regardless of the value of ViewInst.

Bit 10 **TRCRESET**: Tracing of reset exception
This bit selects whether a reset exception must always be traced.
0: Reset exception is traced only if the instruction or exception immediately before the reset exception is traced.
1: Reset exception is always traced regardless of the value of ViewInst.

Bit 9 **SSSTATUS**: Current status of the start/stop logic
0: Stop state
1: Started state

Bit 8 Reserved, must be kept at reset value.

Bits 7:0 **EVENT[7:0]**: Event selector
Whenever the resource selected by this bitfield becomes active, the ViewInst function becomes active.

ETM ViewInst start/stop processor comparator control register (ETM_VIPCSSCTL)

Address offset: 0x08C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STOP[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	START[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:16 **STOP[7:0]**: Processor comparator input selector to stop trace
This bitfield selects which processor comparator inputs are in use with ViewInst start-stop control, to stop the trace. One bit is provided for each processor comparator input.

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **START[7:0]**: Processor comparator input selector to start trace
This bitfield selects which processor comparator inputs are in use with ViewInst start-stop control, to start the trace. One bit is provided for each processor comparator input.

ETM external input selection register (ETM_EXTINSEL)

Address offset: 0x120

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEL3[7:0]								SEL2[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEL1[7:0]								SEL0[7:0]							
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:24 **SEL3[7:0]**: This bitfield selects an external input to use as a resource.

Bits 23:16 **SEL2[7:0]**: This bitfield selects an external input to use as a resource.

Bits 15:8 **SEL1[7:0]**: This bitfield selects an external input to use as a resource.

Bits 7:0 **SEL0[7:0]**: This bitfield selects an external input to use as a resource.

ETM counter reload value register (ETM_CNTRLDV)

Address offset: 0x140

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VALUE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **VALUE[15:0]**: Counter reload value
This value is loaded into the counter each time the reload event occurs.

ETM ID register 8 (ETM_IDR8)

Address offset: 0x180

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MAXSPEC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MAXSPEC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **MAXSPEC[31:0]**: Maximum speculation depth

This bitfield indicates the maximum speculation depth of the instruction trace stream. This is the maximum number of P0 elements that have not been committed in the trace stream at any one time.

0x0: Maximum trace speculation depth is 0.

ETM ID register 9 (ETM_IDR9)

Address offset: 0x184

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMP0KEY[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMP0KEY[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **NUMP0KEY[31:0]**: Number of P0 right-hand keys used

0x0: No P0 keys used in instruction trace only configuration

ETM ID register 10 (ETM_IDR10)

Address offset: 0x188

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMP1KEY[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMP1KEY[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **NUMP1KEY[31:0]**: Total number of P1 right-hand keys

This bitfield indicates the total number of P1 right-hand keys, including normal and special keys.

0x0: No P1 keys used in instruction trace only configuration

ETM ID register 11 (ETM_IDR11)

Address offset: 0x18C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMP1SPC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMP1SPC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **NUMP1SPC[31:0]**: Total number of special P1 right-hand keys used
0x0: No special P1 keys used

ETM ID register 12 (ETM_IDR12)

Address offset: 0x190

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMCONDKEY[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMCONDKEY[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **NUMCONDKEY[31:0]**:
This bitfield indicates the total number of conditional instruction right-hand keys, including normal and special keys.
0x1: One conditional instruction right hand-key implemented

ETM ID register 13 (ETM_IDR13)

Address offset: 0x194

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMCONDSPC[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMCONDSPC[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **NUMCONDSPC[31:0]**: Number of special conditional instruction right-hand keys
0x0: No special conditional instruction right hand-keys implemented

ETM ID register 0 (ETM_IDR0)

Address offset: 0x1E0

Reset value: 0x2800 06E1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	COMMOPT	TSSIZE[4:0]					Res.	Res.	Res.	Res.	BF[1:0]		TRCEXDATA	QSUPP[1]
		r	r	r	r	r	r					r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
QSUPP[0]	Res.	CONDTYPE[1:0]		NUMEVENT[1:0]		RETSTACK	Res.	TRCCCI	TRCCOND	TRCBB	TRCDATA[1:0]		INSTP0[1:0]		Res.
r		r	r	r	r	r		r	r	r	r	r	r	r	

Bits 31:30 Reserved, must be kept at reset value.

Bit 29 **COMMOPT**: Meaning of the commit field in some packets
1: Commit mode 1

Bits 28:24 **TSSIZE[4:0]**: Global timestamp size
0x08: Maximum of 64-bit global timestamp implemented

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:18 **BF[1:0]**: Branch future support
0x0: Not supported

Bit 17 **TRCEXDATA**:
This bit indicates support for tracing of data transfers for exceptions and exception returns.
0: Not supported

Bits 16:15 **QSUPP[1:0]**: Q element support
0x0: Not supported

Bit 14 Reserved, must be kept at reset value.

Bits 13:12 **CONDTYPE[1:0]**: Conditional result tracing type
0x0: ETM indicates only if a conditional instruction passes or fails its condition code check.

Bits 11:10 **NUMEVENT[1:0]**: Number of events supported in the trace
0x1: Two events supported

Bit 9 **RETSTACK**: Return stack support
1: Two entry return stack supported

Bit 8 Reserved, must be kept at reset value.

Bit 7 **TRCCCI**: Support for cycle counting in the instruction trace
1: Cycle counting in the instruction trace is implemented.

Bit 6 **TRCCOND**: Support for conditional instruction tracing
1: Conditional instruction trace is implemented.

Bit 5 **TRCBB**: Support for branch broadcast tracing
1: Branch broadcast trace is implemented.

Bits 4:3 **TRCDATA[1:0]**: Support for data tracing
0x0: Not supported

Bits 2:1 **INSTP0[1:0]**: Support for tracing of load and store instructions as P0 elements
0x0: Not supported

Bit 0 Reserved, must be kept at reset value.

ETM ID register 1 (ETM_IDR1)

Address offset: 0x1E4

Reset value: 0x4100 F450

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DESIGNER[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
r	r	r	r	r	r	r	r								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	TRCARCHMAJ[3:0]				TRCARCHMIN[3:0]				REVISION[3:0]			
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 **DESIGNER[7:0]**: Trace unit designer entity
0x41: Arm

Bits 23:12 Reserved, must be kept at reset value.

Bits 11:8 **TRCARCHMAJ[3:0]**: Major trace unit architecture version number
0x4: ETM v4

Bits 7:4 **TRCARCHMIN[3:0]**: Minor trace unit architecture version number
0x5: Minor version 5

Bits 3:0 **REVISION[3:0]**: Implementation revision number
0x0: Rev 0

ETM ID register 2 (ETM_IDR2)

Address offset: 0x1E8

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	CCSIZE[3:0]				DVSIZE[4:0]					DASIZE[4:1]			
			r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DASIZE[0]	VMIDSIZE[4:0]					CIDSIZE[4:0]					IASIZE[4:0]				
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:29 Reserved, must be kept at reset value.

Bits 28:25 **CCSIZE[3:0]**: Cycle counter size
This bitfield indicates the size of the cycle counter in bits minus 12.
0x0: Cycle counter is 12 bits

Bits 24:20 **DVSIZE[4:0]**: Data value size in bytes
0x0: Data value size is not supported in instruction only configuration.

Bits 19:15 **DASIZE[4:0]**: Data address size in bytes
0x0: Data address size is not supported in instruction only configuration.

Bits 14:10 **VMIDSIZE[4:0]**: Virtual machine ID size
0x0: Virtual machine ID tracing not implemented

Bits 9:5 **CIDSIZE[4:0]**: Context ID size
0x0: Context ID tracing not implemented

Bits 4:0 **IASIZE[4:0]**: Instruction address size
0x4: 32-bit maximum address size

ETM ID register 3 (ETM_IDR3)

Address offset: 0x1EC

Reset value: 0x0509 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NOOVERFLOW	NUMPROC[2:0]			SYSSTALL	STALLCTL	SYNCPR	TRCERR	Res.	Res.	Res.	Res.	EXLEVEL_S[3:0]			
r	r	r	r	r	r	r	r					r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	CCITMIN[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **NOOVERFLOW**: Support of NOOVERFLOW
This bitfield indicates whether the NOOVERFLOW of trace stall control is implemented.
0: Not implemented

Bits 30:28 **NUMPROC[2:0]**: Number of processors available for tracing
0x00: Only one processor can be traced.

Bit 27 **SYSSTALL**: System support for stall control of the processor
0: Stall control of the processor is supported.

Bit 26 **STALLCTL**: Stall control support
1: Trace stall control (STALLCTL register) is implemented.

Bit 25 **SYNCPR**: Trace synchronization period support
1: ETM_SYNCPR is read-only for instruction trace only configuration. The trace synchronization period is fixed.

Bit 24 **TRCERR**: Support of ETM_VICTL.TRCERR
Indicates whether ETM_VICTL.TRCERR is implemented.
1: TRCERR is implemented.

Bits 23:20 Reserved, must be kept at reset value.

Bits 19:16 **EXLEVEL_S[3:0]**: Support of privilege levels
Privilege levels are implemented; one bit for each level.
0x9: Privilege levels thread and handler are implemented.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **CCITMIN[11:0]**: Instruction trace cycle counting minimum threshold
Minimum value that can be programmed in ETM_CCCTL.THRESHOLD
0x4: Minimum threshold is 4 instruction trace cycle

ETM ID register 4 (ETM_IDR4)

Address offset: 0x1F0

Reset value: 0x0011 8000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
NUMVMIDC[3:0]				NUMCIDC[3:0]				NUMSSCC[3:0]				NUMRSPAIR[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMPC[3:0]				Res.	Res.	Res.	SUPPADC	NUMDVC[3:0]				NUMACPAIRS[3:0]			
r	r	r	r				r	r	r	r	r	r	r	r	r

Bits 31:28 **NUMVMIDC[3:0]**: Number of virtual machine ID comparators implemented
0x0: None

Bits 27:24 **NUMCIDC[3:0]**: Number of context ID comparators implemented
0x0: None

Bits 23:20 **NUMSSCC[3:0]**: Number of single-shot comparator controls implemented
0x1: One

Bits 19:16 **NUMRSPAIR[3:0]**: Number of resource selection pairs implemented
0x1: Two

Bits 15:12 **NUMPC[3:0]**: Number of processor comparator inputs implemented in DWT
0x8: Eight

Bits 11:9 Reserved, must be kept at reset value.

Bit 8 **SUPPADC**: Support of data address comparisons
0: Not implemented

Bits 7:4 **NUMDVC[3:0]**: Number of data value comparators implemented
0x0: None

Bits 3:0 **NUMACPAIRS[3:0]**: Number of address comparator pairs implemented.
0x0: None

ETM ID register 5 (ETM_IDR5)

Address offset: 0x1F4

Reset value: 0x90C7 0084

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REDFUNCNTR	NUMCNTR[2:0]			NUMSEQSTATE[2:0]			Res.	LPOVERRIDE	ATBTRIG	TRACEIDSIZE[5:0]					
r	r	r	r	r	r	r		r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	NUMEXTINSEL[2:0]			NUMEXTIN[8:0]								
				r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 **REDFUNCNTR**: Support of reduced function counter
1: Implemented

Bits 30:28 **NUMCNTR[2:0]**: Number of counters implemented
0x1: One counter implemented

Bits 27:25 **NUMSEQSTATE[2:0]**: Number of sequencer states implemented
0x0: None

Bit 24 Reserved, must be kept at reset value.

Bit 23 **LPOVERRIDE**: Support of low-power state override
1: Implemented

Bit 22 **ATBTRIG**: Support of ATB trigger
1: Implemented

Bits 21:16 **TRACEIDSIZE[5:0]**: Number of trace ID bits
0x07: Seven-bit trace ID implemented.

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:9 **NUMEXTINSEL[2:0]**: Number of implemented external input selectors
0x4: Four external input selectors implemented

Bits 8:0 **NUMEXTIN[8:0]**: Number of implemented external inputs
0x4: Four external inputs implemented

ETM resource selection register 2 (ETM_RSCTL2)

Address offset: 0x208

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PAIRINV	INV	Res.	Res.	GROUP[1:0]	
										rw	rw			rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELECT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:22 Reserved, must be kept at reset value.

Bit 21 **PAIRINV**: Inversion of result of a combined pair of resources
0: Not inverted
1: Inverted

Bit 20 **INV**: Inversion of the selected resources
0: Not inverted
1: Inverted

Bits 19:18 Reserved, must be kept at reset value.

Bits 17:16 **GROUP[1:0]**: Selects a group of resources for SELECT
0x0: External inputs
0x1: Processor comparator inputs
0x2: Counter at 0
0x3: Single-shot comparator control 0

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **SELECT[7:0]**: Resource selector from group
This bitfield selects one or more resources from the group selected in GROUP. One bit is provided per resource from the group.
If GROUP = 0x0, SELECT[3:0] select from external input 3 to 0. SELECT[7:4] are not used.
If GROUP = 0x1, SELECT[7:0] select from comparator input 7 to 0.
If GROUP = 0x2, SELECT[0] selects counter at 0. SELECT[7:1] are not used.
If GROUP = 0x3, SELECT[0] selects single-shot comparator 0. SELECT[7:1] are not used.

ETM resource selection register 3 (ETM_RSCTL3)

Address offset: 0x20C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INV	Res.	GROUP[2:0]		
											rw		rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELECT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:21 Reserved, must be kept at reset value.

Bit 20 **INV**: Inversion of the selected resources
0: Not inverted
1: Inverted

Bit 19 Reserved, must be kept at reset value.

Bits 18:16 **GROUP[2:0]**: Selects a group of resources for SELECT
0x0: External inputs
0x1: Processor comparator inputs
0x2: Counter at 0
0x3: Single-shot comparator control 0

Bits 15:8 Reserved, must be kept at reset value.

Bits 7:0 **SELECT[7:0]**: Resource selector from group
Selects one or more resources from the group selected in GROUP. One bit is provided per resource from the group.
If GROUP = 0x0, SELECT[3:0] select from external input 3 to 0. SELECT[7:4] are not used.
If GROUP = 0x1, SELECT[7:0] select from comparator input 7 to 0.
If GROUP = 0x2, SELECT[0] selects counter at 0. SELECT[7:1] are not used.
If GROUP = 0x3, SELECT[0] selects single-shot comparator 0. SELECT[7:1] are not used.

ETM single-shot comparator control register 0 (ETM_SSCC0)

Address offset: 0x280

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							rw								
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

Bits 31:25 Reserved, must be kept at reset value.

Bit 24 **RST**: Single-shot comparator resource reset enable

Enables the single-shot comparator resource to be reset when it occurs, which enables another comparator match to be detected.

0: Disabled

1: Reset enabled; multiple matches can occur

Bits 23:0 Reserved, must be kept at reset value.

ETM single-shot comparator status register 0 (ETM_SSCS0)

Address offset: 0x2A0

Reset value: 0x0000 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
rw															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PC	DV	DA	INST
												r	r	r	r

Bit 31 **STATUS**: Single-shot status

This bit indicates whether any of the selected comparators have matched. If RST = 0 in ETM_SSCC0, this bit must be written with 0 to enable single-shot comparator control.

0: No match occurred

1: Match occurred at least once.

Bits 30:4 Reserved, must be kept at reset value.

Bit 3 **PC**: Processor comparator

Indicates that the single-shot comparator is sensitive to processor comparator inputs.

1: Single-shot comparator is sensitive to processor comparator inputs.

Bit 2 **DV**: Data value comparator support

0: Single-shot data value comparisons not supported

Bit 1 **DA**: Data address comparator support

0: Single-shot data address comparisons not supported

Bit 0 **INST**: Instruction address comparator support

0: Single-shot instruction address comparisons not supported

ETM single-shot processor comparator input control register (ETM_SSPCIC0)

Address offset: 0x2C0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PC[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PC[7:0]**: Comparator input selector for single-shot control

This bitfield selects one or more processor comparator inputs for single-shot control. One bit is provided for each processor comparator input.

ETM power-down control register (ETM_PDC)

Address offset: 0x310

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PU	Res.	Res.	Res.
												r			

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **PU**: Power-up request

This bitfield requests to maintain power to the ETM, and to access to the trace registers.

0: Power not requested

1: Power requested

Bits 2:0 Reserved, must be kept at reset value.

ETM power-down status register (ETM_PDS)

Address offset: 0x314

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STICKYPD	POWER
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **STICKYPD**: Sticky power-down state
This bit is set to 1 when power to ETM registers is removed, to indicate that programming state has been lost. It is cleared after a read of this register.
0: Trace register power uninterrupted since the last read of this register
1: Trace register power interrupted since the last read of this register

Bit 0 **POWER**: ETM powered up
1: ETM is powered up. All registers are accessible.

ETM claim tag set register (ETM_CLAIMSET)

Address offset: 0xFA0

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMSET[3:0]**: Set claim tag bits
Write:
0000: No effect
xxx1: Set bit 0
xx1x: Set bit 1
x1xx: Set bit 2
1xxx: Set bit 3
Read:
0xF: Indicate there are four bits in claim tag.

ETM claim tag clear register (ETM_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMCLR[3:0]**: Reset claim tag bits
Write:
0000: No effect
xxx1: Clear bit 0
xx1x: Clear bit 1
x1xx: Clear bit 2
1xxx: Clear bit 3
Read: Returns current value of claim tag.

ETM authentication status register (ETM_AUTHSTAT)

Address offset: 0xFB8

Reset value: 0x0000 00XX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **SNID[1:0]**: Security level for secure non-invasive debug
0x2: Disabled
0x3: Enabled

Bits 5:4 **SID[1:0]**: Security level for secure invasive debug
0x0: Not implemented

Bits 3:2 **NSNID[1:0]**: Security level for nonsecure non-invasive debug
0x2: Disabled
0x3: Enabled

Bits 1:0 **NSID[1:0]**: Security level for nonsecure invasive debug
0x0: Not implemented

ETM CoreSight device architecture register (ETM_DEVARCH)

Address offset: 0xFBC

Reset value: 0x4775 4A13

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARCHITECT[10:0]											PRESENT	REVISION[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARCHID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 **ARCHITECT[10:0]**: Component architect
0x23B: Arm

Bit 20 **PRESENT**: Indicates the presence of this register
1: Present

Bits 19:16 **REVISION[3:0]**: Architecture revision
0x5: Rev 4.5

Bits 15:0 **ARCHID[15:0]**: Architecture ID
0x4A13: ETMv4.5 component

ETM CoreSight device type identity register (ETM_DEVTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0013

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUBTYPE[3:0]**: Device subtype identifier
0x1: Processor trace

Bits 3:0 **MAJORTYPE[3:0]**: Device main type identifier
0x3: Trace source

ETM CoreSight peripheral identity register 4 (ETM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: Register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm JEDEC code

ETM CoreSight peripheral identity register 0 (ETM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0023

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number field, field, bits [7:0]
0x23: ETM part number

ETM CoreSight peripheral identity register 1 (ETM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00BD

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0xB: Arm JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number field, bits [11:8]
0xD: ETM part number

ETM CoreSight peripheral identity register 2 (ETM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 001B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x1: rev 1

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x3: Arm JEDEC code

ETM CoreSight peripheral identity register 3 (ETM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

ETM CoreSight component identity register 0 (ETM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value

ETM CoreSight component identity register 1 (ETM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0x9: Debug component with CoreSight-compatible registers

Bits 3:0 **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

ETM CoreSight component identity register 2 (ETM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

ETM CoreSight component identity register 3 (ETM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value

ETM register map

Table 879. ETM register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x004	ETM_PRGCTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EN
	Reset value																																0
0x008	Reserved	Reserved																															
0x00C	ETM_STAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PMSTABLE	IDLE
	Reset value																														0	0
0x010	ETM_CONFIG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RS	TS	COND [2:0]			Res.	Res.	Res.	CCI	BB	INSTP0 [1:0]		Res.
	Reset value																				0	0	0	0	0				0	0	0	0	
0x014-0x01C	Reserved	Reserved																															
0x020	ETM _EVENTCTL0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EVENT1[7:0]								EVENT0[7:0]							
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024	ETM _EVENTCTL1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LPOVERRIDE	ATB	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INSTEN[3:0]			
	Reset value																				0	0								0	0	0	0
0x02C	ETM _STALLCTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ISTALL	Res.	Res.	Res.	Res.	LEVEL [1:0]		Res.	Res.
	Reset value																								0					0	0		
0x030	ETM_TSCTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EVENT[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x034	ETM_SYNCP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PERIOD[4:0]				
	Reset value																												0	1	0	1	0
0x038	ETM_CCCTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	THRESHOLD[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x03C	Reserved	Reserved																															
0x040	ETM_TRACEID	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRACEID[6:0]						
	Reset value																										0	0	0	0	0	0	0
0x044 - 0x07C	Reserved	Reserved																															
0x080	ETM_VICTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXLEVEL_S3	Res.	Res.	EXLEVEL_S0	Res.	Res.	Res.	Res.	TRCERR	TRCRESET	SSSTATUS	Res.	EVENT[7:0]							
	Reset value													0			0					0	0	0		0	0	0	0	0	0	0	0
0x084-0x088	Reserved	Reserved																															
0x08C	ETM_VIPCSSCTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STOP[3:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	START[3:0]							
	Reset value									0	0	0	0	0	0	0	0									0	0	0	0	0	0	0	0

Table 879. ETM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x090-0x12C	Reserved	Reserved																															
0x120	ETM_EXTINSEL	SEL3[7:0]								SEL2[7:0]								SEL1[7:0]								SEL0[7:0]							
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x124-0x13C	Reserved	Reserved																															
0x140	ETM_CNTRLDV	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VALUE[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x144-0x17C	Reserved	Reserved																															
0x180	ETM_IDR8	MAXSPEC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0x184	ETM_IDR9	NUMP0KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x188	ETM_IDR10	NUMP1KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x18C	ETM_IDR11	NUMP1SPC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x190	ETM_IDR12	NUMCONDKEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x194	ETM_IDR13	NUMCONDSPC[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x198-0x1DC	Reserved	Reserved																															
0x1E0	ETM_IDR0	Res.	Res.	COMMOPT	TSSIZE[4:0]					Res.	Res.	Res.	Res.	BF[1:0]		TRCEXDATA	QSUPP[1:0]		Res.	CONDTYPE [1:0]		NUMEVENT [1:0]		RETSTACK	Res.	TRCCCI	TRCCOND	TRCBB	TRCDATA[1:0]		INSTP0[1:0]		Res.
	Reset value			1	0	1	0	0	0					0	0	0	0	0		0	0	0	1	1		1	1	1	0	0	0	0	
0x1E4	ETM_IDR1	DESIGNER[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRCARCHMAJ [3:0]				TRCARCHMIN [3:0]				REVISION [3:0]			
	Reset value	0	1	0	0	0	0	0	1													0	1	0	0	0	1	0	1	0	0	0	0
0x1E8	ETM_IDR2	Res.	Res.	Res.	CCSIZE[3:0]				DVSIZE[4:0]					DASIZE[4:0]				VMIDSIZE[4:0]				CIDSIZE[4:0]				IASIZE[4:0]							
	Reset value				0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0x1EC	ETM_IDR3	NOOVERFLOW	NUMPROC[2:0]		SYSSTALL	STALLCTL	SYNCPR	TRCERR		Res.	Res.	Res.	Res.	EXLEVEL_S [3:0]				Res.	Res.	NUMPROC [4:3]		CCITMIN[11:0]											
	Reset value	0	0	0	0	0	1	0	1					1	0	0	1			0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0x1F0	ETM_IDR4	NUMVMIDC [3:0]			NUMCIDC [3:0]				NUMSSCC [3:0]					NUMRSPAIR [3:0]				NUMPC[3:0]				Res.	Res.	Res.	SUPPDAC	NUMDVC [3:0]				NUMACPAIRS [3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0	0				0	0	0	0	0	0	0	0	0

Table 879. ETM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1F4	ETM_IDR5	REDFUNCNTR	NUMCNTR[2:0]			NUMSEQSTATE [2:0]			Res.	LPOVERRIDE	ATBTRIG	TRACEIDSIZE[5:0]						Res.	Res.	Res.	Res.	NUMEXTINSEL [2:0]			NUMEXTIN[8:0]								
	Reset value	1	0	0	1	0	0	0		1	1	0	0	0	1	1	1					1	0	0	0	0	0	0	0	0	1	0	0
0x1F8-0x204	Reserved	Reserved																															
0x208	ETM_RSCTL2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PAIRINV	INV	Res.	GROUP [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELECT[7:0]							
	Reset value											0	0		0	0	0									0	0	0	0	0	0	0	0
0x20C	ETM_RSCTL3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INV	Res.	GROUP [2:0]			Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SELECT[7:0]							
	Reset value												0		0	0	0									0	0	0	0	0	0	0	0
0x210-0x27C	Reserved	Reserved																															
0x280	ETM_SSCC0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RST	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value								0																								
0x284-0x29C	Reserved	Reserved																															
0x2A0	ETM_SSCS0	STATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PC	DV	DA	INST
	Reset value	0																												1	0	0	0
0x2A4-0x2BC	Reserved	Reserved																															
0x2C0	ETM_SSPCIC0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PC[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x2C4-0x30C	Reserved	Reserved																															
0x310	ETM_PDC	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PU	Res.	Res.	Res.
	Reset value																													0			
0x314	ETM_PDS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STICKYPD	POWER
	Reset value																															1	1
0x318-0xF9C	Reserved	Reserved																															
0xFA0	ETM_CLAIMSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET [3:0]			
	Reset value																													1	1	1	1
0xFA4	ETM_CLAIMCLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR [3:0]			
	Reset value																													0	0	0	0
0xFA8-0xFB4	Reserved	Reserved																															
0xFB8	ETM_AUTHSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID [1:0]		SID [1:0]		NSNID [1:0]		NSID [1:0]	
	Reset value																									X	X	X	X	X	X	X	X

Table 879. ETM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFBC	ETM_DEVARCH	ARCHITECT[10:0]											PRESENT	REVISION [3:0]				ARCHID[15:0]															
	Reset value	0	1	0	0	0	1	1	1	0	1	1	1	0	1	0	1	0	0	1	0	1	0	1	0	0	0	0	1	0	0	1	1
0xFC0-0xFC8	Reserved	Reserved																															
0xFCC	ETM_DEVTYPE	Res.																								SUBTYPE [3:0]				MAJORTYPE [3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1
0xFD0	ETM_PIDR4	Res.																								SIZE[3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4 - 0xFDC	Reserved	Reserved																															
0xFE0	ETM_PIDR0	Res.																								PARTNUM[7:0]							
	Reset value																									0	0	1	0	0	0	1	1
0xFE4	ETM_PIDR1	Res.																								JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	1	1	0	1
0xFE8	ETM_PIDR2	Res.																								REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	0	1	0	1	1
0xFEC	ETM_PIDR3	Res.																								REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	ETM_CIDR0	Res.																								PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	ETM_CIDR1	Res.																								CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	ETM_CIDR2	Res.																								PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	ETM_CIDR3	Res.																								PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

78.11.13 Cross trigger interface (CTI)

See *Section 78.12.4*.

78.12 Trace and debug subsystem

The trace and debug subsystem includes the following CoreSight components:

- trace subsystem ROM table
- global trace timestamp generator (TSGEN)
- system cross-trigger interface (CTI)

- cross-trigger matrix (CTM)
- trace port interface unit (TPIU)
- trace bus funnel (CSTF)
- embedded trace FIFO (ETF)
- embedded trace router (ETR)
- replicator
- system trace macrocell (STM)
- serial-wire output (SWO)

The MCU ROM table in the Cortex-M55 subsystem pointed to by the BASE register in the Cortex-M55 AHB-AP links to the trace subsystem ROM table. This secondary ROM table contains the base address pointer for the CoreSight trace components.

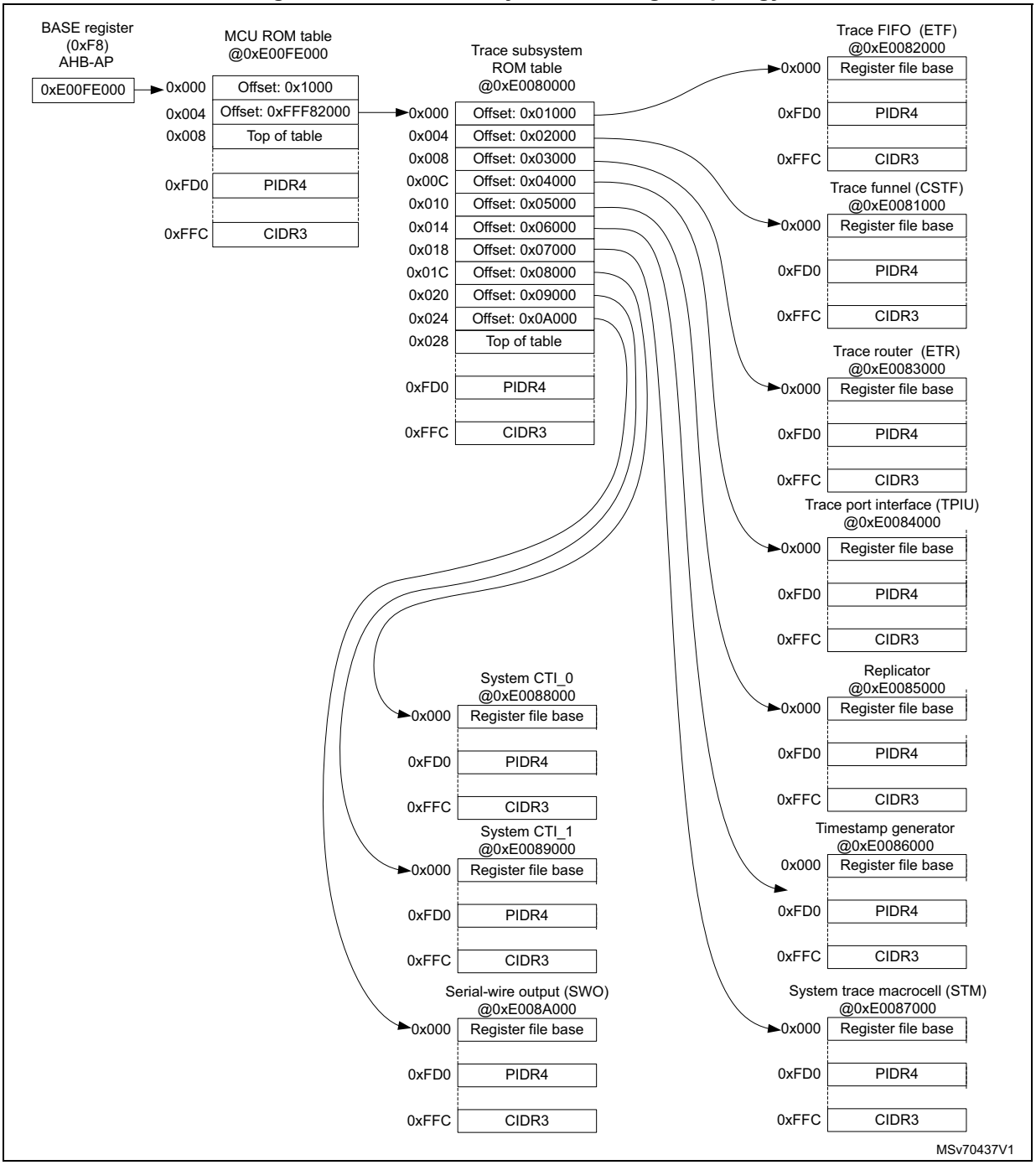
These components are accessible by the debugger and the Cortex-M55 processor via the system AHB-AP and its associated EPPB debug bus.

Table 880. Trace subsystem ROM table

Address in ROM table	Component name	Component base address	Component address offset	Size (Kbytes)	Entry
0xE008 0000	Funnel	0xE008 1000	0x0000 1000	4	0x0000 1003
0xE008 0004	ETF	0xE008 2000	0x0000 2000	4	0x0000 2003
0xE008 0008	ETR	0xE008 3000	0x0000 3000	4	0x0000 3003
0xE008 000C	TPIU	0xE008 4000	0x0000 4000	4	0x0000 4003
0xE008 0010	Replicator	0xE008 5000	0x0000 5000	4	0x0000 5003
0xE008 0014	Timestamp generator	0xE008 6000	0x0000 6000	4	0x0000 6003
0xE008 0018	STM	0xE008 7000	0x0000 7000	4	0x0000 7003
0xE008 001C	CTI0	0xE008 8000	0x0000 8000	4	0x0000 8003
0xE008 0020	CTI1	0xE008 9000	0x0000 9000	4	0x0000 9003
0xE008 0024	SWO	0xE008 A000	0x0000 A000	4	0x0000 A003
0xE008 0028	Top of table	-	-	-	0x0000 0000
0xE008 002C to 0xE008 00C8	Reserved	-	-	-	0x0000 0000
0xE008 0FCC to 0xE008 0FFC	ROM table registers	-	-	-	-

The topology for the CoreSight components in the trace an/d debug subsystem is shown in *Figure 1081*.

Figure 1081. Trace subsystem CoreSight topology



78.12.1 Trace subsystem ROM table registers

Trace subsystem ROM memory type register (TSSROM_MEMTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
															r

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **SYSMEM**: System memory presence
1: System memory is present on this bus.

Trace subsystem ROM CoreSight peripheral identity register 4 (TSSROM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: Register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x0: Arm JEDEC continuation code

Trace subsystem ROM CoreSight peripheral identity register 0 (TSSROM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0086

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number field, bits [7:0]
0x86: STM32N6

Trace subsystem ROM CoreSight peripheral identity register 1 (TSSROM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0x0: Arm JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number field, bits [11:8]
0x4: STM32N6x5/x7xx

Trace subsystem ROM CoreSight peripheral identity register 2 (TSSROM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x0: rev 1

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x2: STMicroelectronics JEDEC code

Trace subsystem ROM CoreSight peripheral identity register 3 (TSSROM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

Trace subsystem ROM CoreSight component identity register 0 (TSSROM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value

Trace subsystem ROM CoreSight component identity register 1 (TSSROM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0010

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0x1: ROM table component

Bits 3:0 **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

Trace subsystem ROM CoreSight component identity register 2 (TSSROM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

Trace subsystem ROM CoreSight component identity register 3 (TSSROM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value

Trace subsystem ROM table register map

Table 881. Trace subsystem ROM table register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFCC	TSSROM_MEMTYPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SYSMEM
	Reset value																																1
0xFD0	TSSROM_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFD4-0xFDC	Reserved	Reserved																															
0xFE0	TSSROM_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									1	0	0	0	0	1	1	0
0xFE4	TSSROM_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value																									0	0	0	0	0	1	0	0
0xFE8	TSSROM_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value																									0	0	0	0	1	0	1	0
0xFEC	TSSROM_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	TSSROM_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	TSSROM_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value																									0	0	0	1	0	0	0	0
0xFF8	TSSROM_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	TSSROM_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

78.12.2 Global timestamp generator (TSGEN)

The CoreSight TSGEN contains a 64-bit counter that provides a common timing reference for all the trace sources in the system, namely the ETM, ITM, and STM. These components insert timestamps in the trace streams that allow the trace analyzer to recover the chronological order of trace packets, which can be lost when multiple trace sources are multiplexed into one stream at the funnel.

The TSGEN registers are accessible over the EPPB, and allow the debugger or debug software to perform the following:

- to start and stop the timestamp incrementing.

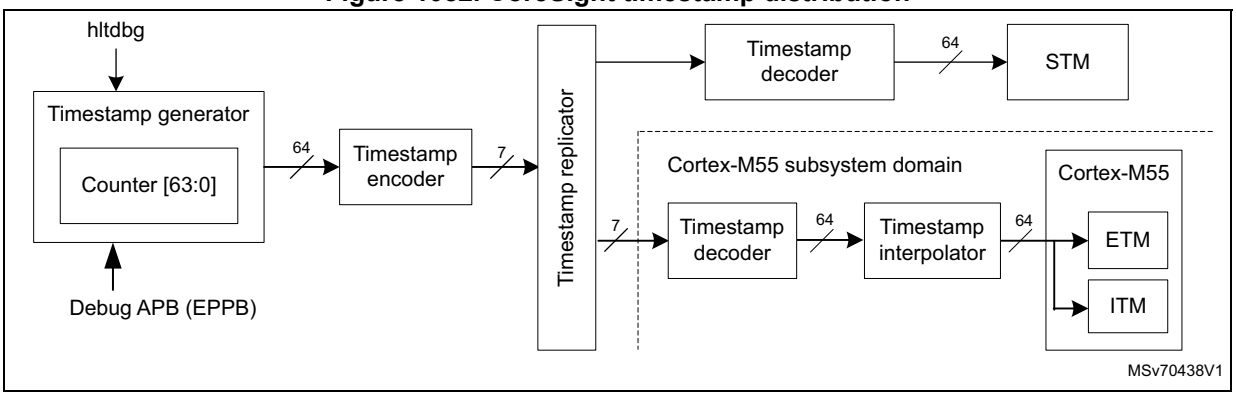
- to cause the timestamp counter to stop when the Cortex-M55 has entered in debug state (when hltdbg input goes active).
- to read the current timestamp value.
- to change the current timestamp value. The timestamp counter must be halted while it is changed. When the timestamp value is changed, the timestamp generator resynchronizes all the trace sources.
- to change the reported timestamp increment.

The TSGEN has its own clock domain ck_ker_tsgen. The timestamp is distributed to the Cortex-M55 to provide time information for the trace macrocell, and to the STM to provide time information for stamping its messages. To simplify the timestamp distribution over clock domain boundaries, the 64-bit timestamp is encoded in 7 bits, then decoded in the destination domain. The timestamp distribution is shown in *Figure 1082*.

Whenever the frequency of the TSGEN or Cortex-M55 changes, the ETM of the Cortex-M55 processor receives a pulse on its tsclkchange input, to insert a marker in the trace stream as a hint that there is discontinuity in time represented by timestamps.

For more information on the CoreSight timestamp generator component, refer to the Arm CoreSight SoC-400 Technical Reference Manual *[2]*.

Figure 1082. CoreSight timestamp distribution



78.12.3 TSGEN registers

The register file base address for the TSGEN is defined in *Table 880: Trace subsystem ROM table*.

TSGEN counter control register (TSG_CNTCR)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDBG	EN
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **HDBG**: Halt on debug
0: Normal operation
1: Halts counter when system-wide debug state is detected (hltdbg goes active).

Bit 0 **EN**: Enable
0: Counter disabled
1: Counter enabled and incrementing

TSGEN counter status register (TSG_CNTSR)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBGH	Res.
														r	

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **DBGH**: Debug halted
0: Normal operation
1: Counter halted due to system-wide debug state

Bit 0 Reserved, must be kept at reset value.

TSGEN current counter value lower register (TSG_CNTCVL)

Address offset: 0x008

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNTCVU_L_32[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNTCVU_L_32[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CNTCVU_L_32[31:0]**: Timestamp counter current value (least significant 32 bits)
To change the current timestamp value, write the least significant 32 bits of the new value to this register before writing the most significant 32 bits to TSG_CNTCVU. The timestamp value is not changed until the TSG_CNTCVU register is written to.
Note: EN must be cleared in TSG_CNTCR before writing to this register TSG_CNTCVL.

TSGEN current counter value upper register (TSG_CNTCVU)

Address offset: 0x00C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNTCVU_U_32[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNTCVU_U_32[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **CNTCVU_U_32[31:0]**: Timestamp counter current value (most significant 32 bits)
To change the current timestamp value, write the least significant 32 bits of the new value to TSG_CNTCVL before writing the most significant 32 bits to this register. The 64-bit timestamp value is updated with the value from both writes when this register is written to.
Note: EN must be cleared in TSG_CNTCR before writing to this register TSG_CNTCVU.

TSGEN base frequency ID register (TSG_CNTFID0)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FREQ[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FREQ[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **FREQ[31:0]**: Increment frequency of TSG counter (in Hz)
This field must be programmed with the trace generator clock frequency whenever it changes.

TSGEN CoreSight peripheral identity register 4 (TSG_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm JEDEC code

TSGEN CoreSight peripheral identity register 0 (TSG_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]
0x01: TSGEN part number

TSGEN CoreSight peripheral identity register 1 (TSG_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm JEDEC code

Bits 3:0 **PARTNUM[11:8]**: part number bits [11:8]
0x1: TSGEN part number

TSGEN CoreSight peripheral identity register 2 (TSG_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 001B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x1: r0p1

Bit 3 **JEDEC**: JEDEC assigned value
0x1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm JEDEC code

TSGEN CoreSight peripheral identity register 3 (TSG_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: no customer modifications

TSGEN CoreSight component identity register 0 (TSG_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: component ID bits [7:0]
0x0D: common ID value

TSGEN CoreSight peripheral identity register 1 (TSG_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 00F0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID bits [15:12] - component class
0xF: CoreSight Soc-400 component

Bits 3:0 **PREAMBLE[11:8]**: Component ID bits [11:8]
0x0: Common ID value

TSGEN CoreSight component identity register 2 (TSG_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: component ID bits [23:16]
0x05: common ID value

TSGEN CoreSight component identity register 3 (TSG_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID bits [31:24]
0xB1: Common ID value

TSGEN register map

Table 882. TSGEN register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	TSG_CNTCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HDBG	EN
	Reset value																															0	0
0x004	TSG_CNTSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBGH	Res.
	Reset value																															0	
0x008	TSG_CNTCVL	CNTCVU_L_32[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x00C	TSG_CNTCVU	CNTCVU_U_32[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	TSG_CNTFID0	FREQ[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x024-0xFCC	Reserved	Reserved																															
0xFD0	TSG_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4-0xFDC	Reserved	Reserved																															
0xFE0	TSG_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	0	0	0	0	0	1
0xFE4	TSG_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	0	0	0	1
0xFE8	TSG_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	1	1	0	1	1
0xFEC	TSG_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND [3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	TSG_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	TSG_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	1	1	1	0	0	0	0

Table 882. TSGEN register map and reset values (continued)

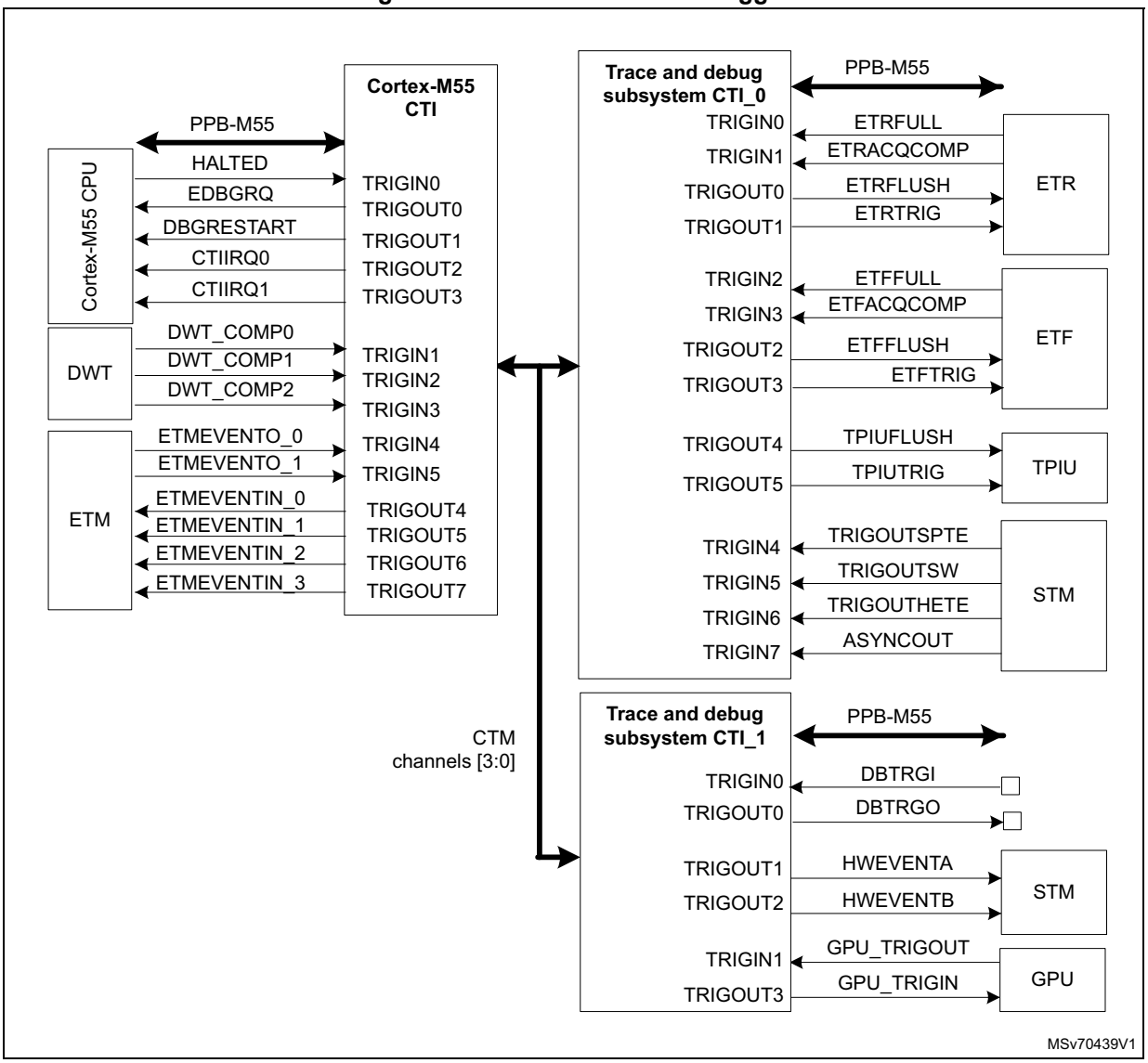
Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFF8	TSG_CIDR2	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	TSG_CIDR3	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

78.12.4 Cross trigger interface (CTI) and matrix (CTM)

The CTI and CTM together form the CoreSight embedded cross trigger feature. There are three CTI components: two in the trace and debug subsystem, and one dedicated to the Cortex-M55.

The CTIs are connected to each other via the CTM. The trace and debug subsystem CTIs and the Cortex-M55 CTI are accessible on the PPB (private peripheral bus) via the Cortex-M55.

Figure 1083. Embedded cross trigger



The CTIs allow events from various sources to trigger debug and/or trace activity. For example, a transition detected on an external trigger input can start code trace.

Each CTI has up to eight trigger inputs and eight trigger outputs. Any input can be connected to any output, on the same CTI, or on another CTI via the CTM.

The trigger input and output signals for each CTI are listed in the following tables.

Table 883. Trace and debug subsystem CTI_0 inputs

-	Source signal	Source component	Comments
0	ETRFULL	ETR	ETR full flag: allows a debug event to be generated when the trace buffer is full.
1	ETRACQCOMP	ETR	ETR capture finished: allows a debug event to be generated when trace acquisition is complete.
2	ETFFULL	ETF	ETF full flag: allows a debug event to be generated when the trace FIFO is full.
3	ETFACQCOMP	ETF	ETF capture finished: allows a debug event to be generated when the trace FIFO is empty.
4	TRIGOUTSPTE	STM	The STM asserts this signal for one clock cycle when a trigger event is detected on a match using the STMSPTER.
5	TRIGOUTSW	STM	The STM asserts this signal for one clock cycle when a trigger event is generated on writes to a TRIG location in the extended stimulus port registers.
6	TRIGOUTHETE	STM	The STM asserts this signal for one clock cycle when a trigger event is detected on a match using the STMHETER.
7	ASYNCOUT	STM	Alignment synchronization signal. The STM asserts this signal for one clock cycle when an ASYNC-VERSION-FREQ sequence is output on the ATB interface, and the ASYNCOUT signal can be used for cross-triggering.

Table 884. Trace and debug subsystem CTI_0 outputs

-	Output signal	Destination component	Comments
0	ETRFLUSH	ETR	ETR flush trigger: causes the trace sources to be flushed into destination memory.
1	ETRTRIG	ETR	ETR trigger: causes various kinds of flush, stops the TMC, and causes additional trigger insertion.
2	ETFFLUSH	ETF	ETF flush trigger: causes the trace FIFO to be flushed.
3	ETFTRIG	ETF	ETF enable trigger: starts filling the trace FIFO.
4	TPIUFLUSH	TPIU	Trace port flush trigger: causes the TPIU FIFO to be flushed.
5	TPIUTRIG	TPIU	Trace Port enable trigger: starts trace output on the external trace port.
6	-	-	Not used
7	-	-	Not used

Table 885. Trace and debug subsystem CTI_1 inputs

-	Source signal	Source component	Comments
0	DBTRIGI	GPIO	External trigger input: allows an external signal to generate a debug event.
1	GPU_TRIGOUT	GPU	Trigger out from GPU
2	-	-	Not used
3	-	-	Not used
4	-	-	Not used

Table 885. Trace and debug subsystem CTI_1 inputs (continued)

-	Source signal	Source component	Comments
5	-	-	Not used
6	-	-	Not used
7	-	-	Not used

Table 886. Trace and debug subsystem CTI_1 outputs

-	Output signal	Destination component	Comments
0	DBTRIGO	GPIO	External I/O trigger output: allows monitoring events on the external DBTRIGO pin.
1	HWEVENTA	STM	STM hardware event
2	HWEVENTB	STM	STM hardware event
3	GPU_TRIGIN	GPU	Trigger in to GPU
4	-	-	Not used
5	-	-	Not used
6	-	-	Not used
7	-	-	Not used

Table 887. Cortex-M55 CTI inputs

-	Source signal	Source component	Comments
0	HALTED	M55 CPU	Pulsed high when the core enters in debug state
1	DWTCOMP_0	M55 DWT	DWT comparator output
2	DWTCOMP_1	M55 DWT	DWT comparator output
3	DWTCOMP_2	M55 DWT	DWT comparator output
4	ETMEVENTO_0	M55 ETM	ETM event output
5	ETMEVENTO_1	M55 ETM	ETM event output
6	-	-	Not used
7	-	-	Not used

Table 888. Cortex-M55 CTI outputs

-	Source signal	Destination component	Comments
0	EDBGRQ	M55 CPU	CPU halt request - puts CPU in debug mode
1	DBGRESTART	M55 CPU	CPU restart request - CPU exits debug mode
2	IRQ0	M55 NVIC	CTI interrupt
3	IRQ1	M55 NVIC	CTI interrupt

Table 888. Cortex-M55 CTI outputs (continued)

-	Source signal	Destination component	Comments
4	ETMEVENTIN_0	M55 ETM	ETM event input
5	ETMEVENTIN_1	M55 ETM	ETM event input
6	ETMEVENTIN_2	M55 ETM	ETM event input
7	ETMEVENTIN_3	M55 ETM	ETM event input

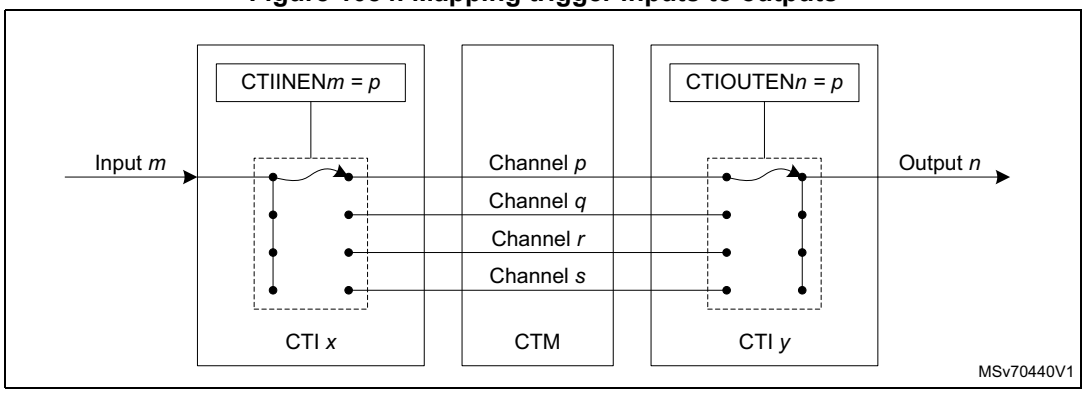
There are four event channels in a cross trigger matrix, which allows up to four, parallel, bidirectional connections between trigger inputs and outputs on different CTIs.

To connect input number *m* on CTI*x* to output number *n* on CTI*y*, the input must be connected to an event channel *p* using the CTIINEN*m* register of CTI*x*. The same channel *p* must be connected to the output using the CTIOUTEN*n* register of CTI*y*.

Note: This applies even if the input and output belong to the same CTI.

An input can be connected to more than one channel (up to four). An input can then be routed to several outputs. Similarly, an output can be connected to several inputs. It is also possible to connect several inputs/outputs to the same channel.

Figure 1084. Mapping trigger inputs to outputs



For more information on the cross-trigger interface CoreSight component, refer to the Arm CoreSight SoC-400 Technical Reference Manual *[2]*.

78.12.5 CTI registers

The register file base addresses for CTI_0 and CTI_1 in the trace and debug subsystem are defined in *Table 880*. The register file base address for the Cortex-M55 CTI is defined in *Table 873*. Registers are the same for each CTI.

CTI control register (CTI_CONTROL)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GLBEN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **GLBEN**: Global enable

0: Cross-triggering disabled

1: Cross-triggering enabled

CTI trigger acknowledge register (CTI_INTACK)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTACK[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **INTACK[7:0]**: Trigger acknowledge

There is one bit of this register for each CTITRIGOUT output. When a one is written to a bit, the corresponding CTITRIGOUT output is acknowledged, causing it to be cleared.

CTI application trigger set register (CTI_APPSET)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPSET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **APPSET[3:0]**: Channel event set
Read:
xxx0: Channel 0 event inactive
xxx1: Channel 0 event active
xx0x: Channel 1 event inactive
xx1x: Channel 1 event active
x0xx: Channel 2 event inactive
x1xx: Channel 2 event active
0xxx: Channel 3 event inactive
1xxx: Channel 3 event active
Write:
xxx0: No effect
xxx1: Set event on channel 0.
xx0x: No effect
xx1x: Set event on channel 1.
x0xx: No effect
x1xx: Set event on channel 2.
0xxx: No effect
1xxx: Set event on channel 3.

CTI application trigger clear register (CTI_APPCLEAR)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPCLEAR[3:0]			
												w	w	w	w

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **APPCLEAR[3:0]**: Channel event clear
xxx0: No effect
xxx1: Clear event on channel 0.
xx0x: No effect
xx1x: Clear event on channel 1.
x0xx: No effect
x1xx: Clear event on channel 2.
0xxx: No effect
1xxx: Clear event on channel 3.

CTI application pulse register (CTI_APPPULSE)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPPULSE[3:0]			
												w	w	w	w

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **APPPULSE[3:0]**: Pulse channel event

This bitfield clears itself immediately.
xxx0: No effect
xxx1: Generate pulse on channel 0.
xx0x: No effect
xx1x: Generate pulse on channel 1.
x0xx: No effect
x1xx: Generate pulse on channel 2.
0xxx: No effect
1xxx: Generate pulse on channel 3.

CTI trigger IN x enable register (CTI_INENx)

Address offset: 0x020 + 0x4 * x (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGINEN[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **TRIGINEN[3:0]**: Cross-trigger event enable

This bitfield enables or disables a cross-trigger event on each of the four channels when CTITRIGINx is activated (x = 0 to 7).
xxx0: Trigger n does not generate events on channel 0.
xxx1: Trigger n generates events on channel 0.
xx0x: Trigger n does not generate events on channel 1.
xx1x: Trigger n generates events on channel 1.
x0xx: Trigger n does not generate events on channel 2.
x1xx: Trigger n generates events on channel 2.
0xxx: Trigger n does not generate events on channel 3.
1xxx: Trigger n generates events on channel 3.

CTI trigger OUT x enable register (CTI_OUTENx)

Address offset: 0x0A0 + 0x4 * x (x = 0 to 7)

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGOUTEN[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **TRIGOUTEN[3:0]**: Enable trigger upon event

For each channel, this bitfield defines whether an event on that channel generates a trigger on CTITRIGOUTx (x = 0 to 7).

xxx0: Channel 0 events do not generate triggers on trigger output n.

xxx1: Channel 0 events generate triggers on trigger output n.

xx0x: Channel 1 events do not generate triggers on trigger output n.

xx1x: Channel 1 events generate triggers on trigger output n.

x0xx: Channel 2 events do not generate triggers on trigger output n.

x1xx: Channel 2 events generate triggers on trigger output n.

0xxx: Channel 3 events generate triggers on trigger output n.

CTI trigger IN status register (CTI_TRGISTS)

Address offset: 0x130

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGINSTATUS[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TRIGINSTATUS[7:0]**: Trigger input status

There is one bit of this bitfield for each CTITRIGIN input. When a bit is set to 1, it indicates that the corresponding trigger input is active. When it is set to 0, the corresponding trigger input is inactive.

CTI trigger OUT status register (CTI_TRGOSTS)

Address offset: 0x134

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGOUTSTATUS[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TRIGOUTSTATUS[7:0]**: Trigger output status

There is one bit of this bitfield for each CTITRIGOUT output. When a bit is set to 1, it indicates that the corresponding trigger output is active. When it is set to 0, the corresponding trigger output is inactive.

CTI channel IN status register (CTI_CHINSTS)

Address offset: 0x138

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHINSTATUS[3:0]			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CHINSTATUS[3:0]**: Channel input status

There is one bit of this bitfield for each channel input. When a bit is set to 1, it indicates that the corresponding channel input is active. When it is set to 0, the corresponding channel input is inactive.

CTI channel OUT status register (CTI_CHOUTSTS)

Address offset: 0x13C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHOUTSTATUS[3:0]			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CHOUTSTATUS[3:0]**: Channel output status

There is one bit of this bitfield for each channel output. When a bit is set to 1, it indicates that the corresponding channel output is active. When it is set to 0, the corresponding channel output is inactive.

CTI channel gate register (CTI_GATE)

Address offset: 0x140

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GATEEN[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **GATEEN[3:0]**: Channel output enable

For each channel, a bit defines whether an event on that channel can propagate over the CTM to other CTIs.

xxx0: Channel 0 events do not propagate.
xxx1: Channel 0 events propagate.
xx0x: Channel 1 events do not propagate.
xx1x: Channel 1 events propagate.
x0xx: Channel 2 events do not propagate.
x1xx: Channel 2 events propagate.
0xxx: Channel 3 events do not propagate.
1xxx: Channel 3 events propagate.

CTI claim tag set register (CTI_CLAIMSET)

Address offset: 0xFA0

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMSET[3:0]**: Set claim tag bits

Write:
0000: No effect
xxx1: Set bit 0.
xx1x: Set bit 1.
x1xx: Set bit 2.
1xxx: Set bit 3.
Read:
0xF: Indicate there are four bits in claim tag.

CTI claim tag clear register (CTI_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMCLR[3:0]**: Reset claim tag bits

Write:
0000: No effect
xxx1: Clear bit 0.
xx1x: Clear bit 1.
x1xx: Clear bit 2.
1xxx: Clear bit 3.
Read: Return current value of claim tag.

CTI lock access register (CTI_LAR)

Address offset: 0xFB0

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACCESS_W[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACCESS_W[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **ACCESS_W[31:0]**: CTI register write access enable

This bitfield enables write access to some CTI registers by the processor core (debuggers do not need to unlock the component).
0xC5ACCE55: Enable write access.
Other values: Disable write access.

CTI lock status register (CTI_LSR)

Address offset: 0xFB4

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKTYPE	LOCKGRANT	LOCKEXIST
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **LOCKTYPE**: Size of the CTI_LAR register
0: 32 bits

Bit 1 **LOCKGRANT**: Current status of lock
This bit always returns 0 when read by an external debugger.
0: Write access is permitted.
1: Write access is blocked. Only read access is permitted.

Bit 0 **LOCKEXIST**: Existence of lock control mechanism
The bit indicates whether a lock control mechanism exists. It always returns 0 when read by an external debugger.
0: No lock control mechanism exists.
1: Lock control mechanism is implemented.

CTI authentication status register (CTI_AUTHSTAT)

Address offset: 0xFB8

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **SNID[1:0]**: Security level for secure non-invasive debug
0x0: Not implemented

Bits 5:4 **SID[1:0]**: Security level for secure invasive debug
0x0: Not implemented

Bits 3:2 **NSNID[1:0]**: Security level for nonsecure non-invasive debug
0x2: Disabled
0x3: Enabled

Bits 1:0 **NSID[1:0]**: Security level for nonsecure invasive debug
0x2: Disabled
0x3: Enabled

CTI device configuration register (CTI_DEVID)

Address offset: 0xFC8

Reset value: 0x0004 0800

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NUMCH[3:0]			
												r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMTRIG[7:0]								Res.	Res.	Res.	EXTMUXNUM[4:0]				
r	r	r	r	r	r	r	r				r	r	r	r	r

Bits 31:20 Reserved, must be kept at reset value.

Bits 19:16 **NUMCH[3:0]**: Number of ECT channels available
0x4: Four channels

Bits 15:8 **NUMTRIG[7:0]**: Number of ECT triggers available
0x8: Eight trigger inputs and eight trigger outputs

Bits 7:5 Reserved, must be kept at reset value.

Bits 4:0 **EXTMUXNUM[4:0]**: Number of trigger input/output multiplexers
0x0: None

CTI device type identifier register (CTI_DEVTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0014

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUBTYPE[3:0]**: Sub-classification
0x1: This bitfield indicates that this component is a cross-triggering component.

Bits 3:0 **MAJORTYPE[3:0]**: Major classification
0x4: This bitfield indicates that this component allows a debugger to control other components in a CoreSight SoC-400 system.

CTI CoreSight peripheral identity register 4 (CTI_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: Register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm JEDEC code

CTI CoreSight peripheral identity register 0 (CTI_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0006

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number field, bits [7:0]
0x06: CTI part number

CTI CoreSight peripheral identity register 1 (CTI_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0xB: Arm JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number field, bits [11:8]
0x9: CTI part number

CTI CoreSight peripheral identity register 2 (CTI_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 005B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x5: r1p0

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x3: Arm JEDEC code

CTI CoreSight peripheral identity register 3 (CTI_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

CTI CoreSight component identity register 0 (CTI_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value

CTI CoreSight component identity register 1 (CTI_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0x9: CoreSight component

Bits 3:0 **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

CTI CoreSight component identity register 2 (CTI_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

CTI CoreSight component identity register 3 (CTI_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value

CTI register map

Table 889. CTI register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	CTI_CONTROL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GLBEN
	Reset value																																0
0x010	CTI_INTACK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	INTACK[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x014	CTI_APPSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPSET[3:0]			
	Reset value																													0	0	0	0
0x018	CTI_APPCLEAR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPCLEAR [3:0]			
	Reset value																													0	0	0	0
0x01C	CTI_APPPULSE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	APPPULSE [3:0]			
	Reset value																													0	0	0	0
0x020 + 0x4 * x (x=0 to 7) Last address: 0x03C	CTI_INENx	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGINEN [3:0]			
	Reset value																													0	0	0	0
0x040-0x09C	Reserved	Reserved																															
0x0A0 + 0x4 * x (x=0 to 7) Last address: 0x0BC	CTI_OUTENx	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGOUTEN [3:0]			
	Reset value																													0	0	0	0
0x0C0-0x12C	Reserved	Reserved																															
0x130	CTI_TRIGISTS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGINSTATUS[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x134	CTI_TRIGOSTS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGOUTSTATUS[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x138	CTI_CHINSTS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHINSTATUS [3:0]			
	Reset value																													0	0	0	0
0x13C	CTI_CHOUTSTS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CHOUSTATUS [3:0]			
	Reset value																													0	0	0	0
0x140	CTI_GATE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	GATEEN[3:0]			
	Reset value																													1	1	1	1
0x144-0xF9C	Reserved	Reserved																															

Table 889. CTI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFA0	CTI_CLAIMSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET [3:0]			
	Reset value																													1	1	1	1
0xFA4	CTI_CLAIMCLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR [3:0]			
	Reset value																													0	0	0	0
0xFA8-0xFAC	Reserved	Reserved																															
0xFB0	CTI_LAR	ACCESS_W[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0xFB4	CTI_LSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKTYPE	LOCKGRANT	LOCKEXIST
	Reset value																														0	1	1
0xFB8	CTI_AUTHSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
	Reset value																									0	0	0	0	1	0	1	0
0xFC8	CTI_DEVID	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NUMCH[3:0]				NUMTRIG[7:0]								Res.	Res.	Res.	EXMUXNUM [4:0]				
	Reset value													0	1	0	0	0	0	0	0	1	0	0	0				0	0	0	0	0
0xFCC	CTI_DEVTYPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE [3:0]				MAJORTYPE [3:0]			
	Reset value																									0	0	0	1	0	1	0	0
0xFD0	CTI_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4-0xFDC	Reserved	Reserved																															
0xFE0	CTI_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	0	0	0	1	1	0
0xFE4	CTI_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	1	0	0	1
0xFE8	CTI_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	1	0	1	1	0	1	1
0xFEC	CTI_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	CTI_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	CTI_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	CTI_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1

Table 889. CTI register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFFC	CTI_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

78.12.6 Trace funnel (CSTF)

The trace funnel is a CoreSight component that combines ATB buses from the three trace sources into one single ATB. The CSTF has three ATB slave ports and one ATB master port. An arbiter selects the slave ports according to a programmable priority.

The slave ports are connected as follows:

- S0: Cortex-M55 ETM
- S1: Cortex-M55 ITM
- S2: STM

CSTF registers allow the slave ports to be individually enabled, and their priority settings to be configured. The priorities can be modified only when trace is disabled. The arbitration works as follows:

1. The arbiter selects the slave port with the highest assigned priority that has data valid.
2. Up to MIN_HOLD_TIME transfers are passed from the selected slave to the master port, where MIN_HOLD_TIME is programmable in CSTF_CTRL.
3. A new arbitration is then performed.

High priority must be assigned to slave ports connected to sources with a small amount of buffering, or where data loss cannot be tolerated. Low priority must be assigned to less critical sources or those with large buffers.

For more information on the ATB funnel CoreSight component, refer to the Arm CoreSight SoC-400 Technical Reference Manual *[2]*.

78.12.7 Trace funnel registers

The register file base address for the CSTF is defined in *Table 880: Trace subsystem ROM table*.

CSTF control register (CSTF_CTRL)

Address offset: 0x000

Reset value: 0x0000 0300

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	MIN_HOLD_TIME[3:0]				Res.	Res.	Res.	Res.	Res.	ENS2	ENS1	ENS0
				rw	rw	rw	rw						rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **MIN_HOLD_TIME[3:0]**: Number of transactions between arbitrations
0x0: 1 transaction
:
0xE: 15 transactions
0xF: Reserved

Bits 7:3 Reserved, must be kept at reset value.

Bit 2 **ENS2**: S2 slave port enable
0: Disable port.
1: Enable port.

Bit 1 **ENS1**: S1 slave port enable
0: Disable port.
1: Enable port.

Bit 0 **ENS0**: S0 slave port enable
0: Disable port.
1: Enable port.

CSTF priority register (CSTF_PRIORITY)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIPORT2[2:0]			PRIPORT1[2:0]			PRIPORT0[2:0]		
							rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:9 Reserved, must be kept at reset value.

Bits 8:6 **PRIPORT2[2:0]**: S1 slave port priority
0: Highest priority
:
7: Lowest priority

Bits 5:3 **PRIPORT1[2:0]**: S1 slave port priority
0: Highest priority
:
7: Lowest priority

Bits 2:0 **PRIPORT0[2:0]**: S0 slave port priority
0: Highest priority
:
7: Lowest priority

CSTF claim tag set register (CSTF_CLAIMSET)

Address offset: 0xFA0

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMSET[3:0]**: Set claim tag

Write:
0000: No effect
xxx1: Set bit 0.
xx1x: Set bit 1.
x1xx: Set bit 2.
1xxx: Set bit 3.
Read:
0xF: Indicate there are four bits in claim tag.

CSTF claim tag clear register (CSTF_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMCLR[3:0]**: Reset claim tag

Write:
0000: No effect
xxx1: Clear bit 0.
xx1x: Clear bit 1.
x1xx: Clear bit 2.
1xxx: Clear bit 3.
Read: Return current value of claim tag.

CSTF lock access register (CSTF_LAR)

Address offset: 0xFB0

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACCESS_W[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACCESS_W[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **ACCESS_W[31:0]**: CSTF register write access enable

The field enables write access to some CSTF registers by the processor cores (debuggers do not need to unlock the component).

0xC5ACCE55: Enable write access.

Other values: Disable write access.

CSTF lock status register (CSTF_LSR)

Address offset: 0xFB4

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKTYPE	LOCKGRANT	LOCKEXIST
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **LOCKTYPE**: Size of the CSTF_LAR register

0: 32 bits

Bit 1 **LOCKGRANT**: Current status of lock

This bit always returns 0 when read by an external debugger.

0: Write access is permitted.

1: Write access is blocked. Only read access is permitted.

Bit 0 **LOCKEXIST**: Existence of lock control mechanism

The bit indicates whether a lock control mechanism exists. It always returns 0 when read by an external debugger.

0: No lock control mechanism exists.

1: Lock control mechanism is implemented.

CSTF authentication status register (CSTF_AUTHSTAT)

Address offset: 0xFB8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **SNID[1:0]**: Security level for secure non-invasive debug
0x0: Not implemented

Bits 5:4 **SID[1:0]**: Security level for secure invasive debug
0x0: Not implemented

Bits 3:2 **NSNID[1:0]**: Security level for nonsecure non-invasive debug
0x0: Not implemented

Bits 1:0 **NSID[1:0]**: Security level for nonsecure invasive debug
0x0: Not implemented

CSTF CoreSight device identity register (CSTF_DEVID)

Address offset: 0xFC8

Reset value: 0x0000 0033

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SCHEME[3:0]				PORTCNT[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SCHEME[3:0]**: Priority scheme
0x3: Program the slave ports to have higher or lower priority with respect to each other.

Bits 3:0 **PORTCNT[3:0]**: Number of input ports connected
0x3: Two input ports

CSTF CoreSight device type identity register (CSTF_DEVTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0012

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEVTYPEID[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **DEVTYPEID[7:0]**: Device type identifier
0x12: Trace funnel

CSTF CoreSight peripheral identity register 4 (CSTF_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: Register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm JEDEC code

CSTF CoreSight peripheral identity register 0 (CSTF_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number field, bits [7:0]
0x08: CSTF part number

CSTF CoreSight peripheral identity register 1 (CSTF_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0xB: Arm JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number field, bits [11:8]
0x9: CSTF part number

CSTF CoreSight peripheral identity register 2 (CSTF_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 003B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x3: r1p1

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x3: Arm JEDEC code

CSTF CoreSight peripheral identity register 3 (CSTF_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

CSTF CoreSight component identity register 0 (CSTF_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value

CSTF CoreSight component identity register 1 (CSTF_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0x9: CoreSight component

Bits 3:0 **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

CSTF CoreSight component identity register 2 (CSTF_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

CSTF CoreSight component identity register 3 (CSTF_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value

CSTF register map

Table 890. CSTF register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	CSTF_CTRL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MIN_HOLD_TIME [3:0]				Res.	Res.	Res.	Res.	Res.	ENS2	ENS1	ENS0
	Reset value																					0	0	1	1						0	0	0
0x004	CSTF_PRIORITY	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRIPORT2 [2:0]			PRIPORT1 [2:0]			PRIPORT0 [2:0]		
	Reset value																								0	0	0	0	0	0	0	0	0
0x008-0xF9C	Reserved	Reserved																															
0xFA0	CSTF_CLAIMSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET [3:0]			
	Reset value																													1	1	1	1

Table 890. CSTF register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFA4	CSTF_CLAIMCLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR [3:0]			
	Reset value																													0	0	0	0
0xFA8-0xFAC	Reserved	Reserved																															
0xFB0	CSTF_LAR	ACCESS_W[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0xFB4	CSTF_LSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCTYPE	LOCKGRANT	LOCKEXIST
	Reset value																														0	1	1
0xFB8	CSTF_AUTHSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID [1:0]		SID [1:0]		NSNID [1:0]		NSID [1:0]	
	Reset value																									0	0	0	0	0	0	0	0
0xFC8	CSTF_DEVID	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SCHEME[3:0]				PORTCNT [3:0]			
	Reset value																									0	0	1	1	0	0	1	1
0xFCC	CSTF_DEVTYPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEVTYPEID[7:0]							
	Reset value																									0	0	0	1	0	0	1	0
0xFD0	CSTF_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4-0xFDC	Reserved	Reserved																															
0xFE0	CSTF_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	0	0	1	0	0	0
0xFE4	CSTF_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	1	0	0	1
0xFE8	CSTF_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	1	1	1	0	1	1
0xFEC	CSTF_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	CSTF_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	CSTF_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	CSTF_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	CSTF_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

78.12.8 Embedded trace FIFO (ETF)

The ETF is a 4-Kbyte memory, which captures trace data from three trace sources: the ETM/ITM of the Cortex-M55, and the STM. The ETF is a configuration of the CoreSight trace memory controller component.

The ETF can be used in three modes (selected in ETF_MODE register):

- hardware-FIFO mode

  The trace memory is used as a FIFO that is drained through the ATB master interface. Trace data are captured into the trace RAM, and when full, the incoming trace stream is stalled. When the trace buffer is not empty, trace data are drained out through the ATB master interface to the TPIU.

  In this mode, the FIFO role is to smooth the flow of trace information arriving at the trace port. Since trace data can be very bursty in nature, the peak data rate can easily exceed the port capability, resulting in an overflow. The ETF allows a steady data rate at the trace port, which can then be sized according to the average rate rather than the peak. The trace is stored off-chip in real time by the trace port analyzer tool, and so the trace log can be very large.

- software-FIFO mode

  The trace memory is used as a FIFO that can be read through ETF_RRD while the trace is being captured. Trace data are captured into the trace RAM, and when full, the incoming trace stream is stalled.

  This mode allows the trace to be transferred by DMA into a system memory, or to a high-speed interface (such as SPI or USB), or monitored by software running on the core.

Note: Unlike the hardware-FIFO mode, this mode is invasive, since it uses system resources shared by the processors.

- circular-buffer mode (embedded trace buffer)

  The trace memory is used as a circular buffer. Trace data are captured into the trace memory starting from the location pointed to by ETF_RWP. Even when the trace memory is full, incoming trace data continue to be overwritten in the trace memory until a stop condition occurs.

  In this mode, the ETF stores trace data on-chip: the trace log size is limited to the ETF SRAM size (64 kbytes in this case). Being a circular buffer, if the FIFO becomes full, incoming trace data overwrite oldest stored data (which are lost). The trace buffer contents represent then the most recent activity of the processor: up to the point when the buffer was stopped, rather than all the activity since the trace was started.

  There are four possible methods to read out the buffer contents once the trace has stopped:

  – via the trace port: with the TPIU enabled, the contents of the buffer are output over the trace port. This can be done by setting DRAINBUF in ETF_FFCR.
  – via the ETR: the buffer can be sent out to memory such as SRAM. With the ETR enabled, this can be done by setting DRAINBUF in ETF_FFCR.
  – via the debug port: the debugger can read the buffer via ETF_RRD register, which is accessible over the debug APB.
  – by software: the processor can read the buffer via ETF_RRD, as the debug APB is accessible from the system bus. This can also be performed by software triggered DMA transfers. No hardware DMA request is available so FULL in ETF_STS can

be used to trigger an interrupt to the Cortex-M55 via the cross trigger. The software can then read ETF_CBUFLVL to find out the number of data to be read.

For more information on the trace memory controller CoreSight component, refer to the Arm CoreSight Trace Memory Controller Technical Reference Manual *[3]*.

78.12.9 Embedded trace FIFO registers

ETF RAM size register (ETF_RSZ)

Address offset: 0x004

Reset value: 0x0000 0400

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	RSZ[30:16]														
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSZ[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bit 31 Reserved, must be kept at reset value.

Bits 30:0 **RSZ[30:0]**: size of the RAM in 32-bit words

0x400: 4 Kbytes

ETF status register (ETF_STS)

Address offset: 0x00C

Reset value: 0x0000 001C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EMPTY	FTEMPTY	READY	TRIGD	FULL
											r	r	r	r	r

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **EMPTY:** Trace FIFO empty

This bit is valid only when TCEN is set in ETF_CTL. This EMPTY bit is read as 0 when TCEN = 0.

0: Trace FIFO contains data.

1: Trace FIFO is empty.

Note: This bit does not inform about ETF pipeline emptiness which is indicated by FTEMPTY.

Bit 3 **FTEMPTY**: Formatter empty

This bit is set when trace capture has stopped, and all internal pipelines and buffers have drained. Unlike READY, it is not affected by buffer drains. The ACQCOMP output reflects the value of this bit.

Bit 2 **READY**: ETF ready

This bit is set when the trace capture has stopped, and all internal pipelines and buffers have drained (stopped or disabled mode).

Bit 1 **TRIGD**: Triggered

This bit is set when the trace capture is in progress and the TMC has detected a trigger event. This bit is cleared when leaving disabled state.

This bit is operational only in the circular-buffer mode. In all other modes, this bit is always low.

This bit does not indicate that a trigger is embedded in formatted output trace data from the TMC. The trigger indication on the output trace stream is determined by ETF_FFCR programming.

Bit 0 **FULL**: Trace buffer full

In circular-buffer mode, this flag is set when the RAM write pointer wraps around the top of the buffer, and remains set until TCEN in ETF_CTL is cleared and set.

In software- and hardware-FIFO modes, this flag indicates that the current space in the trace memory is less than or equal to the value programmed in ETF_BUFWM, that is, Fill level ≥ MEM_SIZE - BUFWM.

This bit is cleared when leaving the disabled state. The FULL output reflects the value of this bit.

ETF RAM read data register (ETF_RRD)

Address offset: 0x010

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RRD[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RRD[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RRD[31:0]**: RAM read data

- Circular-buffer mode: When in stopped state and the buffer is not empty, reading this register returns the next word of data from the trace buffer. When all of the trace buffer has been read, EMPTY is set in ETF_STS, and subsequent reads return 0xFFF FFFF. Reading this register when not in stopped state returns 0xFFFF FFFF.
- Software-FIFO mode: Reading this register returns data from the FIFO. If this register is read when the FIFO is empty, 0xFFFF FFFF is returned.
- Hardware-FIFO mode: Reading this register returns 0xFFFF FFFF.

ETF RAM read pointer register (ETF_RRP)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	RRP[12:0]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:0 **RRP[12:0]**: RAM read pointer

This bitfield contains the value of the read pointer that is used to read entries from the trace memory over the APB interface via ETF_RRD. The pointer can be programmed with a byte address, 64-bit aligned (bits 0 to 3 must be 0). The pointer is incremented by eight each time a full 64-bit FIFO entry is written. When the pointer reaches its maximum value, it wraps around.

This register can be written only in disabled state. It can be read in disabled state, in stopped state in circular-buffer mode and software-FIFO mode, and also in running and stopping states in software-FIFO mode.

ETF RAM write pointer register (ETF_RWP)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	RWP[12:0]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:0 **RWP[12:0]**: RAM write pointer

This bitfield contains the value of the write pointer that is used to write entries into the trace memory over the APB interface via the ETF_RWD register. The pointer can be programmed with a byte address, 64-bit aligned (meaning that bits 0 to 3 must be 0). The pointer is incremented by eight each time a full 64-bit FIFO entry is read. When the pointer reaches its maximum value, it wraps around.

This register can be written only in disabled state. It can be read in disabled state, in stopped state in circular-buffer mode and software-FIFO mode, and also in running and stopping states in software-FIFO mode.

ETF trigger counter register (ETF_TRG)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	TRG[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:0 **TRG[10:0]**: trigger counter

In circular-buffer mode, this bitfield specifies the number of 32-bit words to capture in the trace RAM following the detection of either a rising edge on the TRIGIN input, or a trigger packet in the incoming trace stream (ATID = 0x7D). On capturing the specified number of data words, a trigger event occurs. The effect of a trigger event on the ETF behavior is controlled by ETF_FFCR register.

The number of 32-bit words written into the trace RAM following the trigger is the value stored in this register, plus one. This register is ignored when the ETF is in software- or hardware-FIFO mode. When the trigger counter starts counting, any additional triggers, either on TRIGIN or in the incoming trace stream, are ignored until the counter reaches 0. When the trigger counter has reached 0, it remains at 0 until it is reprogrammed with a write to this register.

This register is cleared when READY is set to 1, so that the state of the counter when trace capture has stopped does not affect a subsequent trace capture session. Writing to this register when not in disabled state results in unpredictable behavior.

A read access to this register is permitted at any time when in disabled state, or in circular-buffer mode. A read access returns the current value of the trigger counter.

ETF control register (ETF_CTL)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCEN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **TCEN:** Trace capture enable

When writing:

0: This bit disables the trace capture (moves from running, stopping, or stopped state into disabling or disabled state).

1: This bit enables the trace capture (moves from disabled state into running state).

Note: When reading, this bit is low in disabling or disabled state. It is high otherwise.

ETF RAM write data register (ETF_RWD)

Address offset: 0x024

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RWD[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RWD[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **RWD[31:0]**: RAM write data

When in disabled state, a write to this register stores data at the location pointed to by ETF_RWP. Writes to this register when not in disabled state are ignored. When a full memory width (64 bits) of data is written, data are written to the memory, and the RAM write pointer is incremented to the next memory word.

Note: This register is used for test purposes.

ETF mode register (ETF_MODE)

Address offset: 0x028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **MODE[1:0]**: operation mode

0: Circular-buffer mode. The trace memory is used as a circular buffer. Trace data are captured into the trace memory starting from the location pointed to by the write pointer register. Even when the trace memory is full, incoming trace data continue to be overwritten into the trace memory until a stop condition occurs.

1: Software-FIFO mode.The trace memory is used as a FIFO that can be read through ETF_RRD while trace is being captured. Trace data are captured into the trace RAM, and when full, the incoming trace stream is stalled.

2: Hardware-FIFO mode. The trace memory is used as a FIFO that is drained through the ATB master interface. Trace data are captured into the trace RAM, and when full, the incoming trace stream is stalled. When the trace buffer is not empty, trace data are drained out through the ATB master interface to the TPIU.

3: Reserved

ETF latched buffer fill level register (ETF_LBUFLVL)

Address offset: 0x02C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	LBUFLEVEL[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **LBUFLEVEL[11:0]**: latched buffer fill level

Reading this register returns the maximum fill level of the trace memory in 32-bit words since this register was last read. Reading this register also results in its contents being updated to the current fill level.

When entering disabled state, this register retains its last value. While in disabled state, reads from this register do not affect its value. When exiting disabled state, this register is cleared.

This register is used for performance analysis of the trace system.

ETF current buffer fill level register (ETF_CBUFLVL)

Address offset: 0x030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	CBUFLEVEL[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **CBUFLEVEL[11:0]**: current buffer fill level

Reading this register returns the current fill level of the trace memory in 32-bit words.

This register is cleared when TCEN = 0 in ETF_CTL.

ETF buffer level watermark register (ETF_BUFWM)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	BUFWM[10:0]										
					rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:0 **BUFWM[10:0]**: buffer level watermark

The value programmed in this register indicates the required threshold vacancy level in 32-bit words in the trace memory. When the space in the FIFO is less than or equal to this value (Fill level ≥ MEM_SIZE - BUFWM), the FULL output is pulled high, and the FULL bit is set in ETF_STS.

This register is used only in software- and hardware-FIFO modes. In circular-buffer mode, this functionality can be obtained by programming ETF_RWP to the required vacancy trigger level, so that when the pointer wraps around, the FULL bit is set indicating that the vacancy level has fallen below the required level.

The maximum value that can be written into this register is MEM_SIZE - 1; in this case, the FULL output is asserted after the first 32-bit word is written to trace memory.

Writing to this register other than when in disabled state results in unpredictable behavior.

ETF formatter and flush status register (ETF_FFSR)

Address offset: 0x300

Reset value: 0x0000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FTSTOPPED	FLINPROG
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **FTSTOPPED:** formatter stopped

This bit behaves in the same way as FTEMPTY in the ETF_STS register.

Bit 0 **FLINPROG:** flush in progress

This bit indicates whether a flush on the ATB slave port is in progress. It reflects the status of the AFVALIDS output. A flush can be initiated by the flush control bits in ETF_FFCR, or requested by the ATB master port.

0: No flush in progress

1: Flush in progress

ETF formatter and flush control register (ETF_FFCR)

Address offset: 0x304

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	DRAINBUF	STPONTRGEV	STOPONFL	Res.	TRIGONFL	TRGONTRGEV	TRGONTRGIN	Res.	FLUSHMAN	FONTRGEV	FONFLIN	Res.	Res.	ENTI	ENFT
	rw	rw	rw		rw	rw	rw		rw	rw	rw			rw	rw

Bits 31:15 Reserved, must be kept at reset value.

Bit 14 **DRAINBUF:** Drain buffer

This bit is used to enable draining of the trace data through the ATB master interface after the formatter has stopped. This is useful in circular-buffer mode to capture trace data in the trace memory, and then to drain the captured trace through the ATB master interface.
Writing a one to this bit when in stopped state starts the drain of the contents of the trace buffer through the ATB master interface. This bit always reads as 0. READY in ETF_STS goes low while the drain is in progress.
This bit is functional only when the ETF is in circular-buffer mode and formatting is enabled (if ENFT = 1 in ETF_FFCR). Setting this bit when the ETF is in any other mode, or when not in stopped state, results in unpredictable behavior.
When trace capture is complete in circular-buffer mode, all of the captured trace must be retrieved from the trace memory through the same mechanism, either read trace data out through ETF_RRD reads, or drain trace data by setting the DRAINBUF bit.
Setting DRAINBUF after some of the captured trace has been read out through ETF_RRD results in an unpredictable behavior.

Bit 13 **STPONTRGEV:** Stop on trigger event

0: No effect
1: Stop trace capture when a trigger event occurs.

Note: Enabling the ETF in software- or hardware-FIFO mode with this bit set, results in an unpredictable behavior.

Bit 12 **STOPONFL:** Stop on flush

0: No effect
1: Stop trace capture when the flush is complete.
If a flush is initiated by the ATB master interface, its completion does not lead to a formatter stop, regardless of the value programmed in this bit.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **TRIGONFL:** Trigger on flush

0: No effect
1: Indicate a trigger in the trace stream when the flush is complete.
If ENFT and ENTI are both cleared in ETF_FFCR, this bit is ignored and no trigger is inserted into the trace stream.
If a flush is initiated by the ATB master interface, its completion does not lead to a trigger indication regardless of the value programmed in this bit.

Bit 9 **TRGONTRGEV:** Trigger on trigger event

0: No effect
1: Indicate a trigger in the trace stream when trigger event occurs.
If ENFT and ENTI are both cleared in ETF_FFCR, this bit is ignored and no trigger is inserted into the trace stream.

Note: This bit is not supported in software- or hardware-FIFO mode.

Bit 8 **TRGONTRGIN:** Trigger on trigger in

0: No effect
1: Indicate a trigger in the trace stream when a rising edge is detected on the TRIGIN input.
If ENFT and ENTI are both cleared i n ETF_FFCR, this bit is ignored and no trigger is inserted into the trace stream.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **FLUSHMAN:** Manual flush

0: No effect

1:Flush the trace FIFO and pipeline.

This bit is cleared automatically when the flush completes. If TCEN = 0 in ETF_CTL, writes to this bit are ignored.

Bit 5 **FONTRGEV:** flush on trigger event

0: no effect

1: Flushes the trace FIFO and pipeline if a trigger event occurs.

Note: This bit is not supported in software- or hardware-FIFO mode. If STPONTRGEV is set in this register, this bit is ignored.

Bit 4 **FONFLIN:** Flush on flush in

0: No effect

1: Flush the trace FIFO and pipeline when a rising edge is detected on FLUSHIN input.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **ENTI**: Enable trigger insertion

Setting this bit enables the insertion of triggers in the formatted trace stream. A trigger is indicated by inserting one byte of data 0x00 with ATID = 0x7D in the trace stream. Trigger indication on the trace stream is additionally controlled by TRIGONFL, TRGONTRGEV, and TRGONTRGIN in ETF_FFCR. This bit can only be changed when READY = 1, and TCEN = 0. This bit takes effect only when ENFT = 1 in this register. If ENTI bit is set to high when ENFT = 0, it results in formatting being enabled.

Bit 0 **ENFT**: Enable formatting

0: Formatting is disabled. Incoming trace data are assumed to be from a single trace source.

1: Formatting is enabled.

If multiple ATIDs are received by the ETF when the trace capture is enabled and the formatter is disabled, trace data are interleaved. Disabling of formatting is supported only in circular-buffer mode. If the ETF is enabled in a mode other than circular-buffer mode with ENFT = 0, the formatting is enabled. If the ENTI bit is set to high when ENFT = 0, the formatting is enabled.

Note: This bit is ignored when in disabled state.

ETF periodic synchronization counter register (ETF_PSCR)

Address offset: 0x308

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSCOUNT[4:0]				
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **PSCOUNT[4:0]**: Synchronization counter reload value

This bitfield determines the reload value of the synchronization counter. The reload value takes effect the next time the counter reaches 0. Reads from this register return the reload value programmed into this register. This register is set to 0xA on reset, corresponding to a synchronization period of 1024 bytes.

0x0: Synchronization disabled

0x7-0x1B: Synchronization period is $2^{PSCOUNT}$ bytes.

Others: Reserved

ETF claim tag set register (ETF_CLAIMSET)

Address offset: 0xFA0

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMSET[3:0]**: Claim tag bit set

Write:

0000: No effect

xxx1: Set bit 0.

xx1x: Set bit 1.

x1xx: Set bit 2.

1xxx: Set bit 3.

Read:

0xF: Indicate there are four bits in claim tag.

ETF claim tag clear register (ETF_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMCLR[3:0]**: Claim tag bit reset
Write:
0000: No effect
xxx1: Clear bit 0.
xx1x: Clear bit 1.
x1xx: Clear bit 2.
1xxx: Clear bit 3.
Read: Return current value of claim tag.

ETF lock access register (ETF_LAR)

Address offset: 0xFB0

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACCESS_W[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACCESS_W[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **ACCESS_W[31:0]**: ETF register write access enable
This bitfield enables write access to some ETF registers by the processor core (debuggers do not need to unlock the component).
0xC5ACCE55: Enable write access.
Other values: Disable write access.

ETF lock status register (ETF_LSR)

Address offset: 0xFB4

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKTYPE	LOCKGRANT	LOCKEXIST
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **LOCKTYPE:** ETF_LAR register size
This bit indicates the ETF_LAR size.
0: 32-bit

Bit 1 **LOCKGRANT:** Current lock status
This bit is always read as 0 by an external debugger.
0: Write access is permitted.
1: Write access is blocked. Only reads are permitted.

Bit 0 **LOCKEXIST:** Lock control existence

This bit indicates whether a lock control mechanism exists. This bit is always read as 0 by an external debugger.

0: No lock-control mechanism exists.

1: A lock-control mechanism is implemented.

ETF authentication status register (ETF_AUTHSTAT)

Address offset: 0xFB8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **SNID[1:0]**: Security level for secure non-invasive debug

0x0: Not implemented

Bits 5:4 **SID[1:0]**: Security level for secure invasive debug

0x0: Not implemented

Bits 3:2 **NSNID[1:0]**: Security level for nonsecure non-invasive debug

0x0: Not implemented

Bits 1:0 **NSID[1:0]**: Security level for nonsecure invasive debug

0x0: Not implemented

ETF device configuration register (ETF_DEVID)

Address offset: 0xFC8

Reset value: 0x0000 0380

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	MEMWIDTH[2:0]			CONFIGTYP[1:0]		CLKSCHEM	ATBINPORTCNT[4:0]				
					r	r	r	r	r	r	r	r	r	r	r

Bits 31:11 Reserved, must be kept at reset value.

Bits 10:8 **MEMWIDTH[2:0]**: Memory interface data bus width

0x3: 64 bits (corresponds to 32-bit ATB data)

Bits 7:6 **CONFIGTYP[1:0]**: Configuration type of component (ETB, ETR, or ETF)

0x2: ETF

Bit 5 **CLKSCHEM:** RAM clocking scheme (synchronous or asynchronous)
0x0: Synchronous

Bits 4:0 **ATBINPORTCNT[4:0]**: Number/type of ATB input port multiplexing
0x0: None

ETF device type identifier register (ETF_DEVTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0032

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUBTYPE[3:0]**: Sub-classification
0x3: Capture trace data from the ATB slave interface into the RAM that can be drained through the ATB master interface.

Bits 3:0 **MAJORTYPE[3:0]**: Major classification
0x2: Component is a trace link because it has an ATB master interface through which trace data can be drained out in hardware-FIFO mode.

ETF CoreSight peripheral identity register 4 (ETF_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: Register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm JEDEC code

ETF CoreSight peripheral identity register 0 (ETF_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0061

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number bits [7:0]
0x61: ETF part number

ETF CoreSight peripheral identity register 1 (ETF_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code bits [3:0]
0xB: Arm JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number bits [11:8]
0x9: ETF part number

ETF CoreSight peripheral identity register 2 (ETF_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 001F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x1: r0p1

Bit 3 **JEDEC**: JEDEC assigned value
0x1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code bits [6:4]
0x3: Arm JEDEC code

ETF CoreSight peripheral identity register 3 (ETF_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

ETF CoreSight component identity register 0 (ETF_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID bits [7:0]
0x0D: Common ID value

ETF CoreSight peripheral identity register 1 (ETF_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID bits [15:12] - component class
0x9: CoreSight component

Bits 3:0 **PREAMBLE[11:8]**: Component ID bits [11:8]
0x0: Common ID value

ETF CoreSight component identity register 2 (ETF_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID bits [23:16]
0x05: Common ID value

ETF CoreSight component identity register 3 (ETF_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID bits [31:24]
0xB1: Common ID value

ETF register map

Table 891. ETF register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x004	ETF_RSZ	Res.	RSZ[30:0]																														
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0x00C	ETF_STS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EMPTY	FTEMPTY	READY	TRIGD	FULL
	Reset value																												1	1	1	0	0
0x010	ETF_RRD	RRD[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x014	ETF_RRP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RRP[12:0]												
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	ETF_RWP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	RWP[12:0]												
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	ETF_TRG	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRG[10:0]										
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x020	ETF_CTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCEN
	Reset value																																0
0x024	ETF_RWD	RWD[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x028	ETF_MODE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE [1:0]	
	Reset value																															0	0
0x02C	ETF_LBUFLVL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LBUFLEVEL[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x030	ETF_CBUFLVL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CBUFLEVEL[11:0]											
	Reset value																					0	0	0	0	0	0	0	0	0	0	0	0
0x034	ETF_BUFWM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUFWM[10:0]										
	Reset value																						0	0	0	0	0	0	0	0	0	0	0
0x300	ETF_FFSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FTSTOPPED	FLINPROG
	Reset value																															1	0

Table 891. ETF register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x304	ETF_FFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DRAINBUF	STPONTRGEV	STOPONFL	Res.	TRGONFL	TRGONTRGEV	TRGONTRGIN	Res.	FLUSHMAN	FONTRGEV	FONFLIN	Res.	Res.	ENTI	ENFT
	Reset value																		0	0	0		0	0	0		0	0	0			0	0
0x308	ETF_PSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSCOUNT[3:0]				
	Reset value																												0	1	0	1	0
0x30C-0xF9C	Reserved	Reserved																															
0xFA0	ETF_CLAIMSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET [3:0]			
	Reset value																													1	1	1	1
0xFA4	ETF_CLAIMCLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR [3:0]			
	Reset value																													0	0	0	0
0xFB0	ETF_LAR	ACCESS_W[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0xFB4	ETF_LSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKTYPE	LOCKGRANT	LOCKEXIST
	Reset value																														0	1	1
0xFB8	ETF_AUTHSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID [1:0]		SID [1:0]		NSNID [1:0]		NSID [1:0]	
	Reset value																									0	0	0	0	0	0	0	0
0xFC8	ETF_DEVID	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEMWDTH [2:0]			CONFIGTYP [1:0]		CLKSCHEM	ATBINPORTCNT [4:0]				
	Reset value																						0	1	1	1	0	0	0	0	0	0	0
0xFD0	ETF_DEVTYPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE [3:0]				MAJORTYPE [3:0]			
	Reset value																									0	0	1	1	0	0	1	0
0xFD0	ETF_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CN [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4-0xFDC	Reserved	Reserved																															

Table 891. ETF register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFE0	ETF_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	1	1	0	0	0	0	1
0xFE4	ETF_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
	Reset value																									1	0	1	1	1	0	0	1
0xFE8	ETF_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
	Reset value																									0	0	0	1	1	0	1	1
0xFEC	ETF_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	ETF_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	ETF_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	ETF_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	ETF_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

78.12.10 Embedded trace router (ETR)

The ETR allows trace data from the Arm trace bus to be sent to external memory, using a DMA scatter/gather interface over an AXI bus.

The ETR also supports programmable burst options for an optimum efficiency of the system interconnect and memory controller. Trace capture can be intrusive to system performance, because less memory is available to the rest of the system, and because the bandwidth must be shared between the ETR and regular system functions. However, the depth of trace capture available is likely to be much larger than available when storing in the ETF and a trace port is not required. Reading of the AXI based trace buffer can either be done directly over the AXI from a normal bus master or through the ETR as if it is an ETB.

The ETR is configured at design time, but some registers accessible through its programming interface allow a few parameters to be modified. In particular, the following:

- The AXI control register contains a field indicating the maximum number of data transfers that can occur within each burst initiated by the TMC on the AXI master interface.
- Data buffer address low and high registers contain the 32-bit address used to locate the trace buffer in the system memory.

- The formatter and flush control register that enables trace data to be TWP formatted with only frames (no padding or syncs). When disabled, data are raw data from the ATB input, without the ID.

From a security perspective, the ETR is associated with a RIMU that controls its master access to the interconnect. Refer to *Section 6: Resource isolation framework security controller (RIFSC)* for more details on the RIMU.

The ETR is a configuration of the CoreSight trace memory controller (TMC) component. It allows trace generated by the Cortex-Mxx core to be stored in a buffer in the on-chip system RAM. The ETR implements a 64-bit AXI bus master together with read and write pointers to manage the buffer.

The ETR supports programmable burst options for an optimum efficiency of the system interconnect and memory controller. Trace capture can be intrusive to system performance, because less memory is available to the rest of the system, and because the bandwidth must be shared between the ETR and regular system functions. However, the depth of trace capture available is likely to be much larger than available when storing in the ETF and a trace port is not required. Reading of the AXI based trace buffer can either be done directly over the AXI from a normal bus master or through the ETR as if it is an ETB.

The ETR is configured at design time, but some registers accessible through its programming interface allow a few parameters to be modified. In particular, the following:

- The AXI control register contains a field indicating the maximum number of data transfers that can occur within each burst initiated by the TMC on the AXI master interface.
- Data buffer address low and high registers contain the 32-bit address used to locate the trace buffer in the system memory.
- The formatter and flush control register enables trace data to be TWP formatted with only frames (no padding or syncs). When disabled, data are raw data from the ATB input, without the ID.

From a security perspective, the ETR is associated with a RIMU that controls its master access to the interconnect. Refer to the Resource isolation framework chapter for more details on the RIMU.

The following subsections give an overview of the TMC configured as an ETR. For more information on the TMC, refer to the Arm CoreSight Trace Memory Controller Technical Reference Manual *[3]*.

ETR mode

Two types of buffer are supported (selected in ETR_MODE):

- software FIFO

  The trace is stored in the buffer and can be read in the same order as it is written. When the buffer is full, the trace is stalled until a read is performed to free up space.
- circular buffer

  The trace is stored in a circular buffer. When the buffer is full, incoming trace data overwrite the oldest stored data, and the oldest stored data is lost. The trace buffer contents represent the most recent activity of the processor, up to the point when the buffer was stopped, rather than all the activity since the trace was started.

The debugger or the processor can read the buffer via ETF_RRD that is accessible over the system APB-D.

Reading via ETR_RRD maintains the read and write pointers, allowing the ETR to keep track of the unread data remaining in the buffer, and manage the FULL/EMPTY flags correctly. The write bandwidth is greater than the read bandwidth, so filtering may be applied to the trace generation to avoid losing trace packets.

The buffer can be read directly from the SRAM. This does not affect the read and write pointers, so must be done when trace has stopped. After reading the contents of the buffer, it must be disabled to reset the pointers to the initial (empty) state.

ETR state machine

The ETR can be moved to any one of these states:

- Disabled

  This state is entered after a reset, or when trace capture is disabled by setting TCEN = 0 in ETR_CTL. The ETR must only be programmed in this state.
- Running

  Trace capture is performed in this state. It is entered by enabling trace capture (TCEN = 1) while in disabled state.
- Stopped

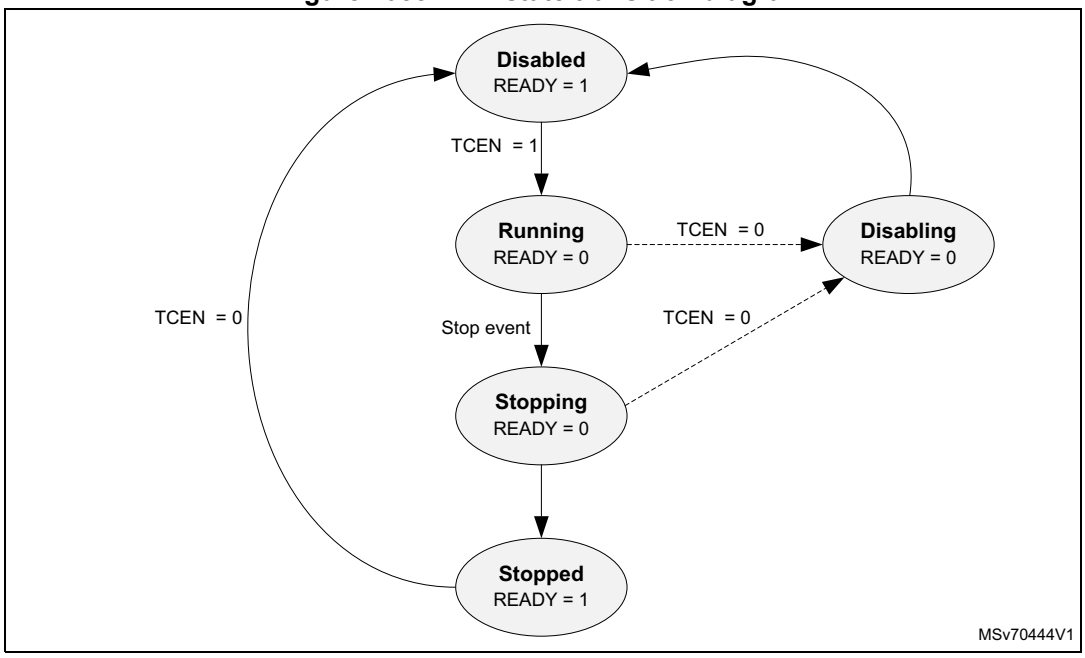
  Trace capture is stopped in this state, but the contents of the buffer can be read out. This state is entered after a stop event (trigger or flush).
- Disabling

  This is a transition state while disabling trace capture.
- Stopping

  This is a transition state while stopping trace capture.

The state transition diagram is shown in the figure below.

Figure 1085. ETR state transition diagram



Scatter-gather operation

The trace buffer can be implemented as one continuous memory with a single base address (ETR_AXICTL.SGMODE = 0), or as multiple 4-Kbyte sections of RAM, in scatter-gather mode (ETR_AXICTL.SGMODE = 1). In the latter case, a linked list of page tables defines the base address of each section. The ETR_DBALO register is programmed with the base address of the buffer, in normal mode. In scatter-gather mode it contains the address of the first entry in the first page table.

The size of page tables is up to 4 Kbytes. Each 32-bit entry has the following format:

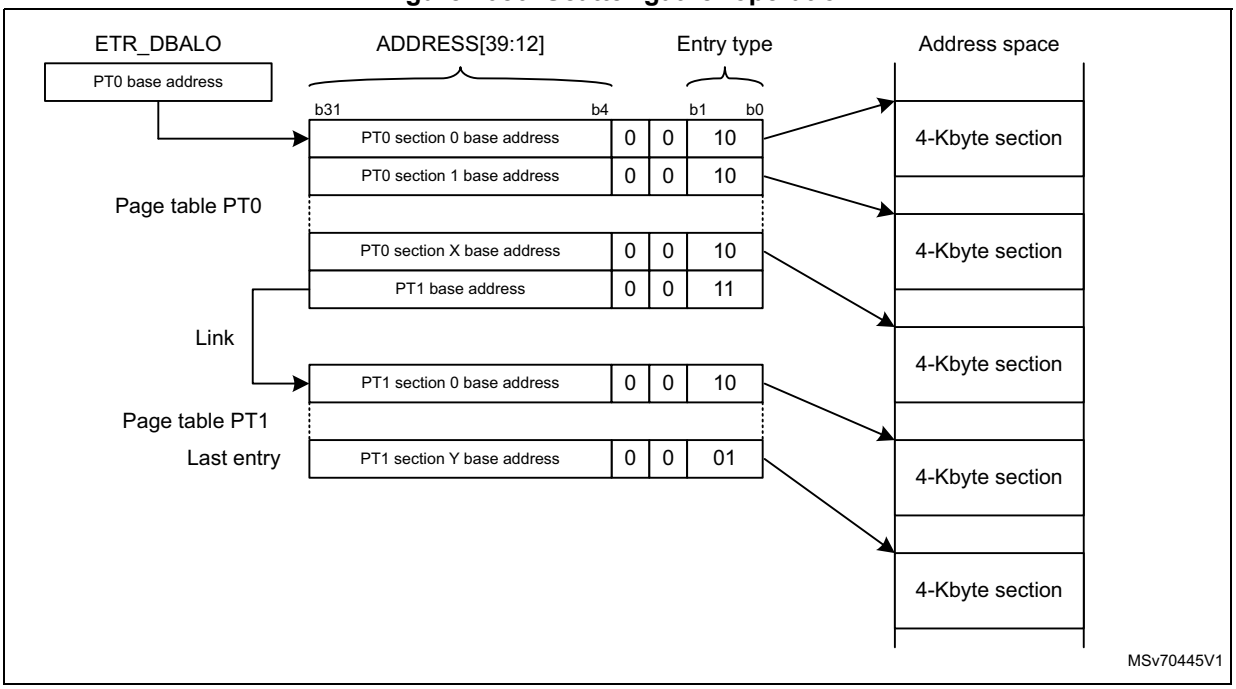
Table 892. ETR scatter-gather page format

Bit range	Content	Description
Bits 31:4	Address[39:12]	Base address of a 4-Kbyte section of RAM
Bits 3:2	00	Not used
Bits 1:0	00	Reserved
	01	Entry points to the last 4-Kbyte section of the trace buffer
	10	Entry points to a 4-Kbyte section of the trace buffer
	11	Entry points to another page table

Note: Trace-buffer addresses are 40-bit wide. However, only 32-bit addressing is used, and the 8 most significant bits are ignored.

The figure below illustrates a scatter-gather operation.

Figure 1086. Scatter-gather operation



Trigger and flush events

Two input signals (trigin and flushin) come from the CTI and can be asserted by software or by a cross trigger coming from another CoreSight component via the cross trigger matrix.

When trigin is asserted, or when a trigger packet is detected on the ATB interface (ATID = 0x7D), a counter is loaded with the value in ETR_TRG, and starts decrementing by one each time a trace word is written into the trace buffer. When the counter reaches 0, a trigger event is generated (in circular-buffer mode only). The trigger event can be programmed in ETR_FFCR:

- to insert a trigger in the trace stream
- to flush the ETR internal buffers, and to issue a flush request to empty any upstream buffers on the ATB
- to stop the trace capture (stop event)

A trigger packet can also be inserted in the trace stream at the moment a rising edge is detected on trigin.

When flushin is asserted, or when the FLUSHMAN = 1 in ETR_FFCR, the same actions can be programmed to occur as for the trigger event.

Authentication

The ETR behaviour is affected by the authentication signals dbgen and spiden.

If dbgen is low, no AXI accesses are possible. Any attempted AXI access behaves as if an immediate error response had been returned, and sets MEMERR in ETR_STS. No transactions are issued on the AXI master port.

If spiden is low, no secure AXI accesses are possible. Any attempted AXI access while PROTCTRLBIT[1] is cleared in ETR_AXICTL behaves as if an immediate error response has been returned, and sets MEMERR in ETR_STS. No transactions are issued on the AXI master port.

78.12.11 ETR registers

The register file base address for the ETR is defined in *Table 880*.

ETR RAM size register (ETR_RSZ)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	RSZ[30:16]														
	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSZ[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bit 31 Reserved, must be kept at reset value.

Bits 30:0 **RSZ[30:0]**: RAM size.

The number of 32-bit words of RAM allocated to the buffer. If scatter gather operation is programmed, this register is ignored.

The maximum size is 4 Gbytes, the minimum size is 512 bytes in software-FIFO mode or 4 bytes in circular-buffer mode.

ETR status register (ETR_STS)

Address offset: 0x00C

Reset value: 0x0000 001C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MEMERR	EMPTY	FTEMPTY	READY	TRIGD	FULL
										r	r	r	r	r	r

Bits 31:6 Reserved, must be kept at reset value.

Bit 5 **MEMERR**: AXI bus error

This bit indicates that an error has occurred on the AXI master interface. The error may be because of an error response from an AXI slave or because of the status of the authentication interface inputs prevents a transaction from taking place.

0: No error

1: Error

Bit 4 **EMPTY**: Trace buffer empty

This bit is valid only when TCEN in ETR_CTL is high. This bit reads as 0 when TCEN is low.

0: Trace buffer contains data.

1: Trace buffer is empty.

Note: Empty trace buffer does not mean that the ETR pipeline is empty. The latter is indicated by FTEMPTY.

Bit 3 **FTEMPTY**: Formatter empty

This bit is set when a trace capture has stopped, and all internal pipelines and buffers have been drained. The acqcomp output reflects the value of this bit.

Bit 2 **READY**: ETR ready

This bit is set when a trace capture has stopped, and all internal pipelines and buffers have been drained (stopped or disabled state), and the AXI interface is not busy.

Bit 1 **TRIGD**: Triggered

This bit is set when a trace capture is in progress, and the TMC has detected a trigger event. This bit is cleared when leaving disabled state. This bit is operational only in circular-buffer mode. In all other modes, this bit is always low.

Bit 0 **FULL**: Trace buffer full

In circular-buffer mode, this flag is set when the RAM write pointer wraps around the top of the buffer, and remains set until TCEN of ETR_CTL is cleared and set.

In software-FIFO mode, this flag indicates that the current space in the trace memory is less than or equal to the value programmed in ETR_BUFWM (CBUFLVL ≥ RSZ - BUFWM).

If the TMC is programmed for scatter-gather operation, this bit indicates whether the trace memory is currently full regardless of the value programmed in ETR_BUFWM.

This bit is cleared when leaving disabled state. The FULL output reflects the value of this bit.

ETR RAM read data register (ETR_RRD)

Address offset: 0x010

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RRD[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RRD[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RRD[31:0]**: RAM read data

Circular-buffer mode:

When in stopped state and the buffer is not empty, reading this register returns the next word of data from the trace buffer (two reads are required to retrieve one 64-bit word of data). When the whole trace buffer has been read, EMPTY is set in ETR_STS, and subsequent reads return 0xFFFFFFF. Reading this register when not in stopped state returns 0xFFFFFFFF.

Software-FIFO mode:

Reading this register returns data from the FIFO. If this register is read when the FIFO is empty, 0xFFFFFFFF is returned.

If MEMERR is set in ETR_STS, reading this register returns an error response on the APB bus.

ETR RAM read pointer register (ETR_RRP)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RRP[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RRP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **RRP[31:0]**: RAM read pointer

This register contains the value of the read pointer that is used to read entries from the trace memory over the APB interface via the ETR_RRD register. The pointer can be programmed with a byte address, 64-bit aligned (bits 0 to 2 must be 0). The pointer is incremented by eight each time a 64-bit buffer entry has been written. When the pointer reaches its maximum value, it wraps around.

This register can be written to only while in disabled state, to set the value of the trace memory address from which data are fetched on a subsequent RRD read. It can be read in disabled state, in stopped state in circular-buffer and software-FIFO modes, and also in running and stopping states in software-FIFO mode.

ETR RAM write pointer register (ETR_RWP)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RWP[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RWP[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **RWP[31:0]**: RAM write pointer

This register contains the value of the write pointer that is used to write entries into the trace memory. The pointer can be programmed with a byte address, 64-bit aligned (that is, bits 0 to 2 should be 0). The pointer is incremented by 8 each time a new entry is written. When the pointer reaches its maximum value, it wraps around.

This register can be written to only while in Disabled state. It can be read in Disabled state, in Stopped state in circular-buffer and software-FIFO modes, and also in Running and Stopping states in software-FIFO mode.

ETR trigger counter register (ETR_TRG)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TRG[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRG[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **TRG[31:0]**: Trigger counter

In circular-buffer mode, this field specifies the number of 32-bit words to capture in the trace buffer following the detection of either a rising edge on the TRIGIN input or a trigger packet in the incoming trace stream, ATID = 0x7D. On capturing the specified number of data words, a trigger event occurs. The effect of a trigger event on the ETR behavior is controlled by the ETR_FFCR register.

The number of 32-bit words written into the trace buffer following the trigger is the value stored in this register, plus one. This register is ignored when the ETR is in software-FIFO mode. When the trigger counter starts counting, any additional triggers, either on TRIGIN or in the incoming trace stream, are ignored until the counter reaches 0. When the trigger counter has reached 0, it remains at 0 until it is reprogrammed with a write to this register. This register is cleared when READY goes high, so that the state of the counter when trace capture has stopped does not affect a subsequent trace capture session. Writing to this register when not in disabled state results in unpredictable behavior.

A read access to this register is permitted at any time when in disabled state, or in circular-buffer mode. A read access returns the current value of the trigger counter.

ETR control register (ETR_CTL)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCEN
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **TCEN**: Trace capture enable

When writing:

0: Disables trace capture (moves from running, stopping or stopped state into disabling or disabled state).

1: Enables trace capture (moves from disabled state to running state).

When reading, this bit is low when in disabling or disabled states, and high otherwise.

ETR RAM write data register (ETR_RWD)

Address offset: 0x024

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RWD[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RWD[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **RWD[31:0]**: RAM write data

When in disabled state, a write to this register stores data at the location pointed to by the ETR_RWP. Writes to this register when not in disabled state are ignored. When a full memory width (64 bits) of data is written, data are written to memory and the RAM write pointer is incremented to the next memory word.
This register is used for test purposes.

ETR mode register (ETR_MODE)

Address offset: 0x028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **MODE[1:0]**: Operation mode

0x0: Circular-buffer mode
In this mode, trace data are captured in the trace buffer memory starting from the location pointed to by the write pointer register. Even when the trace buffer is full, incoming trace data continue to be overwritten into the trace memory until a stop condition has occurred.
0x1: Software-FIFO mode
In this mode, the trace buffer is a FIFO that can be read through ETR_RRD while a trace is being captured. Trace data are captured in the trace buffer memory and when full, the incoming trace stream is stalled.
Others: Reserved

ETR latched buffer fill level register (ETR_LBUFLVL)

Address offset: 0x02C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LBUFLEVEL[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LBUFLEVEL[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **LBUFLEVEL[31:0]**: Latched buffer fill level

Reading this register returns the maximum fill level of the trace buffer in 32-bit words since this register was last read. Reading this register also results in its contents being updated to the current fill level.

When entering disabled state, this register retains its last value. While in disabled state, reads from this register do not affect its value. When exiting disabled state, this register is cleared.

This register does not apply when the ETR is programmed for scatter/gather operation. In this case, reading this register returns 0x00000000.

This register is used for performance analysis of the trace system.

ETR current buffer fill level register (ETR_CBUFLVL)

Address offset: 0x030

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CBUFLEVEL[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CBUFLEVEL[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CBUFLEVEL[31:0]**: Current buffer fill level

Reading this register returns the current fill level of the trace buffer in 32-bit words.

This register does not apply when the ETR is programmed for scatter/gather operation. In this case, reading this register returns 0x00000000.

This register is cleared when TCEN is low.

ETR buffer level watermark register (ETR_BUFWM)

Address offset: 0x034

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BUFWM[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BUFWM[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **BUFWM[31:0]**: Buffer level watermark

The value programmed in this register indicates the required threshold vacancy level of the trace memory in 32-bit words. When the space in the FIFO is less than or equal to this value (CBUFLEVEL ≥ RSZ - BUFWM), the full output is pulled high and FULL is set in ETR_STS.
This register is used only in software-FIFO mode. In circular-buffer mode, this functionality can be obtained by programming ETR_RWP to the required vacancy trigger level, so that when the pointer wraps around, the FULL bit is set indicating that the vacancy level has fallen below the required level.
The maximum value that can be written into this register is RAM size (RSZ) - 1. In this case, the full output is asserted after the first 32-bit word is written to the buffer.
Writing to this register other than when in disabled state results in unpredictable behavior.

ETR AXI control register (ETR_AXICTL)

Address offset: 0x110

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	WRBURSTLEN[3:0]				SGMODE	Res.	CACHECTRLBIT[3:0]				PROTCTRLBIT[1:0]	
				rw	rw	rw	rw	rw		rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **WRBURSTLEN[3:0]**: Write burst length

This field indicates the maximum number of data transfers that can occur within each burst initiated by the ETR on the AXI master interface. The write burst initiated on the AXI can be of a smaller length that the programmed value in the case when the formatter has stopped because of a stop condition occurred. Programming this field to a burst length value greater than the write buffer depth, results in a burst length that is equal to the write buffer depth.
The burst length programmed must be compatible with the trace buffer size and the AXI data width: the total number of bytes of data transferred in a burst is not greater than the trace buffer size or, if scatter-gather operation is enabled, is not greater than 4 Kbytes. Programming an incompatible burst length results in an unpredictable behavior.
It is recommended that this value is set to no more than half the write buffer depth. It is also recommended that this value is set to enable an AXI burst of at least one frame of trace data.
0x0: one data transfer per burst (default)
0x1-0xF: maximum of (WRBURSTLEN + 1) data transfers per burst

Bit 7 **SGMODE**: Scatter-gather mode

This bit indicates whether trace memory is accessed as a single buffer in system memory or as a linked-list based scatter-gather memory. This bit is ignored when in disabled state.
0: Trace memory is a single contiguous block of system memory.
1: Trace memory is spread over multiple blocks of system memory based on a linked-list mechanism.

Bit 6 Reserved, must be kept at reset value.

Bits 5:2 **CACHECTRLBIT[3:0]**: Cache control

These bits control the value driven on ARCACHEM[3:0] or AWCACHEM[3:0] signals on the AXI master interface when performing AXI transfers.

xxx0: Non-bufferable

xxx1: Bufferable

000x: Non-cacheable

001x: Cacheable, no allocate on read, no allocate on write

011x: Cacheable, allocate on read, no allocate on write

101x: Cacheable, no allocate on read, allocate on write

111X: Cacheable, allocate on read, allocate on write

Others: Reserved

Bits 1:0 **PROTCTRLBIT[1:0]**: Protection control bits.

These bits control the value driven on ARPROTM[3:0] or AWPROTM[3:0] signals on the AXI master interface when performing AXI transfers.

x0: Normal access

x1: Privileged access

0x: Secure access

1x: Nonsecure access

ETR data buffer address low register (ETR_DBALO)

Address offset: 0x118

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
BUFADDRLO[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BUFADDRLO[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **BUFADDRLO[31:0]**: Buffer address low

If SGMODE = 0 in ETR_AXICTL, the content of this register is the base address of the trace buffer in system memory.

If SGMODE = 1, the content of this register is the address of first page table entry in the linked list.

This register can only be programmed in disabled state.

ETR formatter and flush status register (ETR_FFSR)

Address offset: 0x300

Reset value: 0x0000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FTSTOPPED	FLINPROG
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **FTSTOPPED**: Formatter stopped

This bit behaves in the same way as FTEMPTY in ETR_STS.

Bit 0 **FLINPROG**: Flush in progress

This bit indicates whether a flush on the ATB slave port is in progress. It reflects the status of the AFVALIDS output. A flush can be initiated by the flush control bits in ETR_FFCR.

0: No flush in progress

1: Flush in progress

ETR formatter and flush control register (ETR_FFCR)

Address offset: 0x304

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	STPONTRGEV	STOPONFL	Res.	TRIGONFL	TRGONTRGEV	TRGONTRGIN	Res.	FLUSHMAN	FONTRGEV	FONFLIN	Res.	Res.	ENTI	ENFT
		rw	rw		rw	rw	rw		rw	rw	rw			rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **STPONTRGEV**: Stop on trigger event

0: No effect

1: Stops trace capture when a trigger event occurs.

Enabling the ETR in software-FIFO mode with this bit set results in unpredictable behavior.

Bit 12 **STOPONFL**: Stop on flush

0: No effect

1: Stops trace capture when flush is completed.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **TRIGONFL**: Trigger on flush

0: No effect

1: Indicates a trigger in the trace stream when flush is completed.

If ENFT and ENTI are clear, this bit is ignored and no trigger is inserted into the trace stream.

Bit 9 **TRGONTRGEV**: Trigger on trigger event

0: No effect

1: Indicates a trigger in the trace stream when a trigger event occurs.

If ENFT and ENTI are both clear, this bit is ignored and no trigger is inserted into the trace stream. This bit is not supported in software-FIFO mode.

Bit 8 **TRGONTRGIN**: Trigger on trigger in

0: No effect

1: Indicates a trigger in the trace stream when a rising edge is detected on the TRIGIN input.

If ENFT and ENTI are clear, this bit is ignored and no trigger is inserted into the trace stream.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **FLUSHMAN**: Manual flush

0: No effect

1: Flushes the trace FIFO and pipeline.

This bit is cleared automatically when the flush is complete. If TCEN = 0 in ETR_CTL, writes to this bit are ignored.

Bit 5 **FONTRGEV**: Flush on trigger event

0: No effect

1: Flushes the trace FIFO and pipeline if a trigger event occurs.

This bit is not supported in software-FIFO mode. If STPONTRGEV is set, this bit is ignored.

Bit 4 **FONFLIN**: Flush on flush in

0: No effect

1: Flushes the trace FIFO and pipeline when a rising edge is detected on the FLUSHIN input.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **ENTI**: Enable trigger insertion

Setting this bit enables the insertion of triggers in the formatted trace stream. A trigger is indicated by inserting one byte of data 0x00 with ATID = 0x7D in the trace stream. Trigger indication on the trace stream is further controlled by TRIGONFL, TRGONTRGEV, and TRGONTRGIN in FFCR. This bit can be changed only when READY is high and TCEN is low. It takes effect only when ENFT in this register is set. If ENTI is set high when ENFT is low, it results in formatting being enabled.

Bit 0 **ENFT**: Enable formatting

0: Formatting is disabled. Incoming trace data are assumed to be from a single trace source.

1: Formatting is enabled.

If multiple ATIDs are received by the ETR when trace capture is enabled and the formatter is disabled, an interleaving of trace data occurs. Disabling of formatting is supported only in circular-buffer mode. If the ETR is enabled in a mode other than circular-buffer mode with ENFT low, then formatting is enabled. If ENTI bit is set to high when ENFT is low, formatting is enabled.

This bit is ignored when in disabled state.

ETR periodic synchronization counter register (ETR_PSCR)

Address offset: 0x308

Reset value: 0x0000 000A

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSCOUNT[4:0]				
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bits 4:0 **PSCOUNT[4:0]**: Synchronization counter reload value

Determines the reload value of the synchronization counter. The reload value takes effect the next time the counter reaches 0. Reads from this register return the reload value programmed in this register. This register is set to 0xA on reset, corresponding to a synchronization period of 1024 bytes.

0x0: Synchronization disabled

0x7-0x1B: Synchronization period is $2^{PSCOUNT}$ bytes

Others: Reserved

ETR claim tag set register (ETR_CLAIMSET)

Address offset: 0xFA0

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMSET[3:0]**: Set claim tag bits

Write:

0000: No effect

xxx1: Set bit 0.

xx1x: Set bit 1.

x1xx: Set bit 2.

1xxx: Set bit 3.

Read:

0xF: Indicates there are four bits in claim tag.

ETR claim tag clear register (ETR_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMCLR[3:0]**: Reset claim tag bits
Write:
0000: No effect
xxx1: Clear bit 0.
xx1x: Clear bit 1.
x1xx: Clear bit 2.
1xxx: Clear bit 3.
Read: Returns current value of claim tag.

ETR lock access register (ETR_LAR)

Address offset: 0xFB0

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACCESS_W[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACCESS_W[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **ACCESS_W[31:0]**: ETR register access enable
Enables write access to all ETR registers by the processor core (debuggers do not need to unlock the component)
0xC5ACCE55: Enable write access.
Other values: Disable write access.

ETR lock status register (ETR_LSR)

Address offset: 0xFB4

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKTYPE	LOCKGRANT	LOCKEXIST
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **LOCKTYPE**: Size of ETR_LAR register
0: 32 bits

Bit 1 **LOCKGRANT**: Current status of lock
This bit always returns 0 when read by an external debugger.
0: Write access is permitted.
1: Write access is blocked to all registers except ETR_LAR. Only read access is permitted.

Bit 0 **LOCKEXIST**: Existence of lock control mechanism
The bit indicates whether a lock control mechanism exists. It always returns 0 when read by an external debugger.
0: No lock control mechanism exists.
1: Lock control mechanism is implemented.

ETR authentication status register (ETR_AUTHSTAT)

Address offset: 0xFB8

Reset value: 0x0000 00XX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **SNID[1:0]**: Security level for secure non-invasive debug
0x0: Not implemented

Bits 5:4 **SID[1:0]**: Security level for secure invasive debug
0x2: Functionality disabled. This return value occurs when spiden = 0 or dbgen = 0.
0x3: Functionality enabled. This return value occurs when spiden = 1 and dbgen = 1.

Bits 3:2 **NSNID[1:0]**: Security level for nonsecure non-invasive debug
0x0: Not implemented

Bits 1:0 **NSID[1:0]**: Security level for nonsecure invasive debug
0x2: Functionality disabled. This return value occurs when dbgen = 0.
0x3: Functionality enabled. This return value occurs when dbgen = 1.

ETR device configuration register (ETR_DEVID)

Address offset: 0xFC8

Reset value: 0x0000 2B40

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	WBUF_DEPTH[2:0]			MEMWIDTH[2:0]			CONFIGTYP[1:0]		CLKSCHEM	ATBINPORTCNT[4:0]				
		r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:14 Reserved, must be kept at reset value.

Bits 13:11 **WBUF_DEPTH[2:0]**: Write buffer depth
Indicates, in powers of two, the number of entries in the ETR internal write buffer. Each entry is 32 bits.
0x5: 32 entries

Bits 10:8 **MEMWIDTH[2:0]**: Memory interface data bus width
0x3: 64 bits

Bits 7:6 **CONFIGTYP[1:0]**: Configuration type of component (ETB, ETR or ETF)
0x1: ETR

Bit 5 **CLKSCHEM**: RAM clocking scheme (synchronous or asynchronous)
0: Synchronous

Bits 4:0 **ATBINPORTCNT**[4:0]: Number/type of ATB input port multiplexing
0x0: None

ETR device type identifier register (ETR_DEVTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0021

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUBTYPE[3:0]**: Sub-classification
0x2: Captures the ATB slave interface trace data in RAM which can then be read through an APB interface.

Bits 3:0 **MAJORTYPE[3:0]**: Major classification
0x1: The component is a trace sink.

ETR CoreSight peripheral identity register 4 (ETR_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: Register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm JEDEC code

ETR CoreSight peripheral identity register 0 (ETR_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0061

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number field, bits [7:0]
0x61: ETR part number

ETR CoreSight peripheral identity register 1 (ETR_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0xB: Arm JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number field, bits [11:8]
0x9: ETR part number

ETR CoreSight peripheral identity register 2 (ETR_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 001B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x1: r0p1

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x3: Arm JEDEC code

ETR CoreSight peripheral identity register 3 (ETR_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

ETR CoreSight component identity register 0 (ETR_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value

ETR CoreSight component identity register 1 (ETR_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0x9: CoreSight component

Bits 3:0 **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

ETR CoreSight component identity register 2 (ETR_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

ETR CoreSight component identity register 3 (ETR_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value

ETR register map

Table 893. ETR register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x004	ETR_RSZ	Res.	RSZ[30:0]																														
	Reset value		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x008	Reserved	Reserved																															
0x00C	ETR_STS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EMPTY	FTEMPTY	READY	TRIGD	FULL
	Reset value																												0	1	1	0	0
0x010	ETR_RRD	RRD[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x014	ETR_RRP	RRP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x018	ETR_RWP	RWP[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x01C	ETR_TRG	TRG[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x020	ETR_CTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCEN
	Reset value																																0
0x024	ETR_RWD	RWD[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0x028	ETR_MODE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE [1:0]	
	Reset value																															0	0
0x02C	ETR_LBUFLVL	LBUFLEVEL[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x030	ETR_CBUFLVL	CBUFLEVEL[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x034	ETR_BUFWM	BUFWM[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x038-0x10C	Reserved	Reserved																															

Table 893. ETR register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x110	ETR_AXICTL	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WRBURSTLEN[3:0]				SGMODE	Res.	CACHECTRLBIT[3:0]				PROTCTLBIT[1:0]	
	Reset value																					0	0	0	0	0		0	0	0	0	0	0
0x114	Reserved	Reserved																															
0x118	ETR_DBALO	BUFADDRLO[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x11C-0x2FC	Reserved	Reserved																															
0x300	ETR_FFSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FTSTOPPED	FLINPROG
	Reset value																															1	0
0x304	ETR_FFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STPONTRGEV	STOPONFL	Res.	TRGONFL	TRGONTRGEV	TRGONTRGIN	Res.	FLUSHMAN	FONTRGEV	FONFLIN	Res.	Res.	ENTI	ENFT
	Reset value																			0	0		0	0	0		0	0	0			0	0
0x308	ETR_PSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PSCOUNT[4:0]				
	Reset value																												0	1	0	1	0
0x30C-0xF9C	Reserved	Reserved																															
0xFA0	ETR_CLAIMSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
	Reset value																													1	1	1	1
0xFA4	ETR_CLAIMCLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
	Reset value																													0	0	0	0
0xFA8-0xFAC	Reserved	Reserved																															
0xFB0	ETR_LAR	ACCESS_W[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0xFB4	ETR_LSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKTYPE	LOCKGRANT	LOCKEXIST
	Reset value																														0	1	1
0xFB8	ETR_AUTHSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
	Reset value																									X	X	X	X	X	X	X	X
0xFBC-0xFC4	Reserved	Reserved																															

Table 893. ETR register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFC8	ETR_DEVID	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	WBUF_DEPTH [2:0]			MEMWDTH [2:0]			CONFIGTYP[1:0]		CLKSCHEM	ATBINPORTCNT [4:0]				
	Reset value																			1	0	1	0	1	1	0	1	0	0	0	0	0	0
0xFCC	ETR_DEVTYPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE [3:0]				MAJORTYPE [3:0]			
	Reset value																									0	0	1	0	0	0	0	1
0xFD0	ETR_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4-0xFDC	Reserved	Reserved																															
0xFE0	ETR_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	1	1	0	0	0	0	1
0xFE4	ETR_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	1	0	0	1
0xFE8	ETR_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	0	1	1	0	1	1
0xFEC	ETR_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	ETR_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	ETR_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	ETR_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	ETR_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

78.12.12 Trace port interface unit (TPIU)

The TPIU CoreSight component formats the trace stream and outputs it on the external trace port signals. The TPIU has a single ATB slave port for incoming trace data. The trace port is a synchronous parallel port, comprising a clock output, TRACECLK, and four data outputs, TRACED[3:0]. The trace port width is programmable in the 1 to 4 range. Using a smaller port width reduces the number of test points/connector pins needed, and frees up I/Os for other purposes. This restricts the bandwidth of the trace port, and hence quantity of trace information that can be output in real time. The trace clock must be enabled by setting TRACECLKEN in DBGMCU_CR before a trace is sent to the TPIU.

Trace data are output on TRACED[15:0] synchronously with the rising and falling edges of TRACECLK. TRACECLK is synchronous with traceclkin, at half the frequency.

This means that TRACECLK is asynchronous with respect to the ATB clock.
The asynchronism is handled by a FIFO in the TPIU, which also regulates the flow of data from the ETF. If the trace port bandwidth is not sufficient to handle the amount of trace data, the ETF buffer can overflow, and trace data can be missed. In this case, the traceclkin frequency can be increased, or the port size increased, up to a maximum of four pins, depending on the device package. Alternatively, the ETM can be programmed to filter unnecessary information.

For more information on the trace port interface CoreSight component, refer to the Arm CoreSight SoC-400 Technical Reference Manual *[2]*.

78.12.13 TPIU registers

The register file base address for the TPIU is defined in *Table 880: Trace subsystem ROM table*.

TPIU supported port size register (TPIU_SUPPSIZE)

Address offset: 0x000

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PORTSIZE[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PORTSIZE[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **PORTSIZE[31:0]**: Trace port size

This bitfield indicates supported trace port sizes, from 1 to 32 pins. Bit n-1 when set indicates that port size n is supported.
0x0000 0003: Port sizes 1 to 4 supported.

TPIU current port size register (TPIU_CURPSIZE)

Address offset: 0x004

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PORTSIZE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PORTSIZE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **PORTSIZE[31:0]**: Current trace port size

Setting bit n-1 indicates that the current port size is n-pin wide. The value of n must be within the range of supported port sizes (1-16). Only one bit can be set, or unpredictable behavior may result. This register must only be modified when the formatter is stopped.

TPIU supported trigger modes register (TPIU_SUPTRGM)

Address offset: 0x100

Reset value: 0x0000 011F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRGRUN	TRIGD
														r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCOUNT8	Res.	Res.	Res.	MULT64K	MULT256	MULT16	MULT4	MULT2
							r				r	r	r	r	r

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **TRGRUN**: Trigger running

0: Trigger has not occurred, or counter is at 0.

1: Trigger has occurred, and counter is not at 0.

Bit 16 **TRIGD**: Triggered

0: Trigger has not occurred.

1: Trigger has occurred, and counter has reached 0.

Bits 15:9 Reserved, must be kept at reset value.

Bit 8 **TCOUNT8**: 8-bit counter register

1: Implemented

Bits 7:5 Reserved, must be kept at reset value.

Bit 4 **MULT64K**: Multiplying the trigger counter by 65536

1: Supported

Bit 3 **MULT256**: Multiplying the trigger counter by 256

1: Supported

Bit 2 **MULT16**: Multiplying the trigger counter by 16

1: Supported

Bit 1 **MULT4**: Multiplying the trigger counter by 4

1: Supported

Bit 0 **MULT2**: Multiplying the trigger counter by 2

1: Supported

TPIU trigger counter value register (TPIU_TRGCNT)

Address offset: 0x104

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGCOUNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **TRIGCOUNT[7:0]**: Enable trigger delay indication

This bitfield is used to delay the indication of triggers to any externally connected trace capture or storage devices. This counter is only 8-bit wide, and is intended to be used only with the counter multipliers in the trigger multiplier register (0x108). When a trigger is started, this value, in conjunction with the multiplier, specifies the number of words before the trigger is indicated. When the trigger counter reaches 0, the value written here is reloaded. Writing to this register causes the trigger counter value to reset but does not reset any value on the multiplier. Reading this register returns the preset value, not the current count.

TPIU trigger multiplier register (TPIU_TRGMULT)

Address offset: 0x108

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MULT64K	MULT256	MULT16	MULT4	MULT2
											rw	rw	rw	rw	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **MULT64K**: Multiply the trigger counter by 65536
0: Disabled
1: Enabled

Bit 3 **MULT256**: Multiply the trigger counter by 256
0: Disabled
1: Enabled

Bit 2 **MULT16**: Multiply the trigger counter by 16
0: Disabled
1: Enabled

Bit 1 **MULT4**: Multiply the trigger counter by 4
0: Disabled
1: Enabled

Bit 0 **MULT2**: Multiply the trigger counter by 2
0: Disabled
1: Enabled

TPIU supported test patterns/modes register (TPIU_SUPTPM)

Address offset: 0x200

Reset value: 0x0003 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCONTEN	PTIMEEN
														r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PATF0	PATA5	PATW0	PATW1
												r	r	r	r

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **PCONTEN**: Continuous mode
1: Supported

Bit 16 **PTIMEEN**: Timed mode
1: Supported

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **PATF0**: FF/00 pattern
This bit indicates whether the FF/00 pattern is supported as output over the trace port.
1: Supported

Bit 2 **PATA5**: AA/55 pattern
This bit indicates whether the AA/55 pattern is supported as output over the trace port.
1: Supported

Bit 1 **PATW0**: Walking 0 pattern
This bit indicates whether the walking 0 pattern is supported as output over the trace port.
1: Supported

Bit 0 **PATW1**: Walking one pattern
This bit indicates whether the walking one pattern is supported as output over the trace port.
1: Supported

TPIU current test pattern/mode register (TPIU_CURTPM)

Address offset: 0x204

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCONTEN	PTIMEEN
														rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PATF0	PATA5	PATW0	PATW1
												rw	rw	rw	rw

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **PCONTEN**: Continuous mode enable
0: Disabled
1: Enabled

Bit 16 **PTIMEEN**: Timed mode enable
0: Disabled
1: Enabled

Bits 15:4 Reserved, must be kept at reset value.

Bit 3 **PATF0**: FF/00 pattern enable
This bit indicates whether the FF/00 pattern is enabled as output over the trace port.
0: Disabled
1: Enabled

Bit 2 **PATA5**: AA/55 pattern is enable
This bit indicates whether the AA/55 pattern is enabled as output over the trace port.
0: Disabled
1: Enabled

Bit 1 **PATW0**: Walking 0 pattern enable
This bit indicates whether the walking 0 pattern is enabled as output over the trace port.
0: Disabled
1: Enabled

Bit 0 **PATW1**: Walking one pattern enable
Indicates whether the walking one pattern is enabled as output over the trace port.
0: Disabled
1: Enabled

TPIU test pattern repeat counter register (TPIU_TPRCR)

Address offset: 0x208

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PATTCOUNT[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PATTCOUNT[7:0]**: Number of traceclkin cycles
The field provides an 8-bit counter value to indicate the number of traceclkin cycles for which a pattern runs before it switches to the next pattern.

TPIU formatter and flush status register (TPIU_FFSR)

Address offset: 0x300

Reset value: 0x0000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCPRESENT	FTSTOPPED	FLINPROG
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **TCPRESENT**: TRACECTL output pin availability

This bit indicates whether the optional TRACECTL output pin is available.

0: TRACECTL pin is not present in this device.

Bit 1 **FTSTOPPED**: Formatter stopped

The formatter has received a stop request signal. Trace data and post-amble are sent. Any additional trace data on the ATB interface are ignored.

0: Formatter has not stopped.

1: Formatter has stopped.

Bit 0 **FLINPROG**: Flush in progress

This bit indicates whether a flush on the ATB slave port is in progress. This bit reflects the status of the AFVALIDS output. A flush can be initiated by the flush control bits in TPIU_FFCR.

0: No flush in progress

1: Flush in progress

TPIU formatter and flush control register (TPIU_FFCR)

Address offset: 0x304

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	STOPTRIG	STOPFL	Res.	TRIGFL	TRIGEVT	TRIGIN	Res.	FONMAN	FONTRIG	FONFLIN	Res.	Res.	ENFCONT	ENFTC
		rw	rw		rw	rw	rw		rw	rw	rw			rw	rw

Bits 31:14 Reserved, must be kept at reset value.

Bit 13 **STOPTRIG**: Stop on trigger event

0: No effect

1: Stop formatter when a trigger event occurs.

Bit 12 **STOPFL**: Stop on flush

0: No effect

1: Stop formatter when flush is completed.

Bit 11 Reserved, must be kept at reset value.

Bit 10 **TRIGFL**: Trigger on flush
0: No effect
1: Indicate a trigger in the trace stream when flush is completed.

Bit 9 **TRIGEVT**: Trigger on trigger event
0: No effect
1: Indicate a trigger in the trace stream when trigger event occurs.

Bit 8 **TRIGIN**: Trigger on trigger in
0: No effect
1: Indicate a trigger in the trace stream when the TRIGIN input from the system CTI is asserted.

Bit 7 Reserved, must be kept at reset value.

Bit 6 **FONMAN**: Manual flush
0: No effect
1: Flush the trace.
Note: This bit is automatically cleared when the flush completes.

Bit 5 **FONTRIG**: Flush on trigger event
A trigger event occurs when the trigger counter reaches 0, or, if the trigger counter is 0, when the TRIGIN input from the system CTI is high.
0: No effect
1: Flush the trace if a trigger event occurs.

Bit 4 **FONFLIN**: Flush on flush in
0: No effect
1: Flush the trace if the FLUSHIN input from the system CTI is asserted.

Bits 3:2 Reserved, must be kept at reset value.

Bit 1 **ENFCONT:** Continuous formatting enable
0: Disabled
1: Enabled

Bit 0 **ENFTC**: Embedding of triggers in formatted trace
0: Formatting is disabled.
1: Formatting is enabled.

TPIU formatter synchronization counter register (TPIU_FSCR)

Address offset: 0x400

Reset value: 0x0000 0040

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	CYCCOUNT[11:0]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:0 **CYCCOUNT[11:0]**: Effective use of TPAs

This bitfield enables the effective use of different-sized TPAs without wasting large amounts of storage capacity of the capture device. This counter contains the number of formatter frames since the last synchronization packet of 128 bits. It is a 12-bit counter with a maximum count value of 4096. This equates to a synchronization every 65536 bytes (4096 packets x 16 bytes per packet). The default is set up for a synchronization packet every 1024 bytes (every 64 formatter frames). If the formatter is configured for continuous mode, full, and half-word sync frames are inserted during normal operation. The count value is then the maximum number of complete frames between full synchronization packets.

TPIU claim tag set register (TPIU_CLAIMSET)

Address offset: 0xFA0

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMSET[3:0]**: Set claim tag bits

Write:
0000: No effect
xxx1: Set bit 0.
xx1x: Set bit 1.
x1xx: Set bit 2.
1xxx: Set bit 3.
Read:
0xF: Indicate there are four bits in claim tag.

TPIU claim tag clear register (TPIU_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMCLR[3:0]**: Reset claim tag bits
Write:
0000: No effect
xxx1: Clear bit 0.
xx1x: Clear bit 1.
x1xx: Clear bit 2.
1xxx: Clear bit 3.
Read: Return current value of claim tag.

TPIU lock access register (TPIU_LAR)

Address offset: 0xFB0

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACCESS_W[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACCESS_W[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **ACCESS_W[31:0]**: TPIU register access enable
This bitfield enables write access to some TPIU registers by the processor cores (debuggers do not need to unlock the component).
0xC5ACCE55: Enable write access.
Other values: Disable write access.

TPIU lock status register (TPIU_LSR)

Address offset: 0xFB4

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKTYPE	LOCKGRANT	LOCKEXIST
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **LOCKTYPE**: Size of the TPIU_LAR register
0: 32-bit

Bit 1 **LOCKGRANT**: Current status of lock
This bit always returns 0 when read by an external debugger.
0: Write access is permitted.
1: Write access is blocked. Only read access is permitted.

Bit 0 **LOCKEXIST**: Implementation of a lock control mechanism
The bit indicates whether a lock control mechanism is implemented. It always returns 0 when read by an external debugger.
0: No lock control mechanism is available.
1: Lock control mechanism is implemented.

TPIU authentication status register (TPIU_AUTHSTAT)

Address offset: 0xFB8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **SNID[1:0]**: Security level for secure non-invasive debug
0x0: Not implemented

Bits 5:4 **SID[1:0]**: Security level for secure invasive debug
0x0: Not implemented

Bits 3:2 **NSNID[1:0]**: Security level for nonsecure non-invasive debug
0x0: Not implemented

Bits 1:0 **NSID[1:0]**: Security level for nonsecure invasive debug
0x0: Not implemented

TPIU device configuration register (TPIU_DEVID)

Address offset: 0xFC8

Reset value: 0x0000 00A0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	SWOUARTNRZ	SWOMAN	TCLKDATA	FIFOSIZE[2:0]			CLKRELAT	MAXNUM[4:0]				
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **SWOUARTNRZ**: SWO UART or NRZ support
0: Not supported

Bit 10 **SWOMAN**: SWO Manchester format support
0: Not supported

Bit 9 **TCLKDATA**: Trace clock plus data support
0: Not supported

Bits 8:6 **FIFOSIZE[2:0]**: FIFO size in powers of two
0x2: FIFO size = 4 (16 bytes)

Bit 5 **CLKRELAT**: ATB clock and traceclkin relation
This bit indicates the relationship between the ATB clock and traceclkin (synchronous or asynchronous)
1: Asynchronous

Bits 4:0 **MAXNUM[4:0]**: Number/type of ATB input port multiplexing
0x0: None

TPIU device type identifier register (TPIU_DEVTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0011

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUBTYPE[3:0]**: Sub-classification
0x1: Trace port component

Bits 3:0 **MAJORTYPE[3:0]**: Major classification
0x1: Trace sink component

TPIU CoreSight peripheral identity register 4 (TPIU_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: Register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm JEDEC code

TPIU CoreSight peripheral identity register 0 (TPIU_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0012

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number field, bits [7:0]
0x12: TPIU part number

TPIU CoreSight peripheral identity register 1 (TPIU_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0xB: Arm JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number field, bits [11:8]
0x9: TPIU part number

TPIU CoreSight peripheral identity register 2 (TPIU_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 005B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x5: r1p0

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x3: Arm JEDEC code

TPIU CoreSight peripheral identity register 3 (TPIU_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

TPIU CoreSight component identity register 0 (TPIU_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value

TPIU CoreSight component identity register 1 (TPIU_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0x9: CoreSight component

Bits 3:0 **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

TPIU CoreSight component identity register 2 (TPIU_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

TPIU CoreSight component identity register 3 (TPIU_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value

TPIU register map

Table 894. TPIU register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	TPIU_SUPPSIZE	PORTSIZE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
0x004	TPIU_CURPSIZE	PORTSIZE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x008-0x0FC	Reserved	Reserved																															
0x100	TPIU_SUPTRGM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRGRUN	TRIGD	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCOUNT8	Res.	Res.	Res.	MULT64K	MULT256	MULT16	MULT4	MULT2
	Reset value															0	0								1				1	1	1	1	1
0x104	TPIU_TRGCNT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TRIGCOUNT[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x108	TPIU_TRGMULT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MULT64K	MULT256	MULT16	MULT4	MULT2
	Reset value																												0	0	0	0	0
0x10C-0x1FC	Reserved	Reserved																															
0x200	TPIU_SUPTPM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCONTEN	PTIMEEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PATF0	PATA5	PATW0	PATW1
	Reset value															1	1													1	1	1	1
0x204	TPIU_CURTPM	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PCONTEN	PTIMEEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PATF0	PATA5	PATW0	PATW1
	Reset value															0	0													0	0	0	0
0x208	TPIU_TPRCR	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	PATTCOUNT[7:0]							
	Reset value																									0	0	0	0	0	0	0	0
0x20C-0x2FC	Reserved	Reserved																															
0x300	TPIU_FFSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	TCPRESENT	FTSTOPPED	FLINPROG
	Reset value																														0	1	0
0x304	TPIU_FFCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	STOPTRIG	STOPFL	Res.	TRIGFL	TRIGEVT	TRIGIN	Res.	FONMAN	FONTRIG	FONFLIN	Res.	Res.	ENFCONT	ENFTC
	Reset value																			0	0		0	0	0		0	0	0			0	0
0x308-0x30C	Reserved	Reserved																															
0x400	TPIU_FSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CYCCOUNT[11:0]											
	Reset value																					0	0	0	0	0	1	0	0	0	0	0	0
0x404-0xF9C	Reserved	Reserved																															

Table 894. TPIU register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFA0	TPIU_CLAIMSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET [3:0]			
	Reset value																													1	1	1	1
0xFA4	TPIU_CLAIMCLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR [3:0]			
	Reset value																													0	0	0	0
0xFA8-0xFAC	Reserved	Reserved																															
0xFB0	TPIU_LAR	ACCESS_W[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0xFB4	TPIU_LSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKTYPE	LOCKGRANT	LOCKEXIST
	Reset value																														0	1	1
0xFB8	TPIU_AUTHSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
	Reset value																									0	0	0	0	0	0	0	0
0xFBC-0xFC4	Reserved	Reserved																															
0xFC8	TPIU_DEVID	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWOUARTNRZ	SWOMAN	TCLKDATA	FIFOSIZE [2:0]			CLKRELAT	MAXNUM[4:0]				
	Reset value																					0	0	0	0	1	0	1	0	0	0	0	0
0xFCC	TPIU_DEVTYPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE [3:0]				MAJORTYP [3:0]			
	Reset value																									0	0	0	1	0	0	0	1
0xFD0	TPIU_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4-0xFDC	Reserved	Reserved																															
0xFE0	TPIU_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	0	1	0	0	1	0
0xFE4	TPIU_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	1	0	0	1
0xFE8	TPIU_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	1	0	1	1	0	1	1
0xFEC	TPIU_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	TPIU_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1

Table 894. TPIU register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFF4	TPIU_CIDR1	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	TPIU_CIDR2	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	TPIU_CIDR3	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

78.12.14 Trace replicator (REP)

The trace replicator CoreSight component splits the ATB bus into two branches: trace data can be routed to two different sinks. The replicator has two ATB master ports, and one ATB slave port. The trace packets received on the slave port are duplicated and output on the master ports.

There are two replicators in the trace and debug subsystem:

- ITM replicator

  Routes the trace coming from the ITM to the SWO and to the trace funnel (CSTF).
- ETR replicator

  Routes the combined ETM and ITM trace to the embedded trace router (ETR), and to the trace port interface (TPIU), as shown in the table below. The slave port is connected to the CSTF.

Table 895. ETR replicator allocation

ATB port	Trace sink
#0	ETR
#1	TPIU

As data are received from the trace source, it is passed on to all trace sinks at the same time. By default, the replicator does not accept more data from the trace source until all trace sinks have accepted data. This has the impact of reducing the throughput of the replicator to match that of the slowest trace sink. The ETR replicator has an APB bus interface and a register set that allows filtering of the trace forwarded to each branch.

The trace packets output on each master port can be filtered according to the source ID, by programming REP_IDFILTER[1:0] registers.

For more information on the ATB replicator CoreSight component, refer to the Arm CoreSight SoC-400 Technical Reference Manual *[2]*.

78.12.15 Trace replicator registers

The register file base address for the replicator is defined in *Table 880*.

Replicator ID filter register 0 (REP_IDFILTER0)

Address offset: 0x000

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ID0_70_7F	ID0_60_6F	ID0_50_5F	ID0_40_4F	ID0_30_3F	ID0_20_2F	ID0_10_1F	ID0_0_F
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **ID0_70_7F**: Enable or disable ID filtering for IDs 0x70-0x7F
0: Transactions with these IDs are passed on to ATB master port 0.
1: Transactions with these IDs are discarded on master port 0.

Bit 6 **ID0_60_6F**: Enable or disable ID filtering for IDs 0x60-0x6F
0: Transactions with these IDs are passed on to ATB master port 0.
1: Transactions with these IDs are discarded on master port 0.

Bit 5 **ID0_50_5F**: Enable or disable ID filtering for IDs 0x50-0x5F
0: Transactions with these IDs are passed on to ATB master port 0.
1: Transactions with these IDs are discarded on master port 0.

Bit 4 **ID0_40_4F**: Enable or disable ID filtering for IDs 0x40-0x4F
0: Transactions with these IDs are passed on to ATB master port 0.
1: Transactions with these IDs are discarded on master port 0.

Bit 3 **ID0_30_3F**: Enable or disable ID filtering for IDs 0x30-0x3F
0: Transactions with these IDs are passed on to ATB master port 0.
1: Transactions with these IDs are discarded on master port 0.

Bit 2 **ID0_20_2F**: Enable or disable ID filtering for IDs 0x20-0x2F
0: Transactions with these IDs are passed on to ATB master port 0.
1: Transactions with these IDs are discarded on master port 0.

Bit 1 **ID0_10_1F**: Enable or disable ID filtering for IDs 0x10-0x1F
0: Transactions with these IDs are passed on to ATB master port 0.
1: Transactions with these IDs are discarded on master port 0.

Bit 0 **ID0_0_F**: Enable or disable ID filtering for IDs 0x0-0xF
0: Transactions with these IDs are passed on to ATB master port 0.
1: Transactions with these IDs are discarded on master port 0.

Replicator ID filter register 1 (REP_IDFILTER1)

Address offset: 0x004

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ID1_70_7F	ID1_60_6F	ID1_50_5F	ID1_40_4F	ID1_30_3F	ID1_20_2F	ID1_10_1F	ID1_0_F
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **ID1_70_7F**: Enable or disable ID filtering for IDs 0x70-0x7F
0: Transactions with these IDs are passed on to ATB master port 1.
1: Transactions with these IDs are discarded on master port 1.

Bit 6 **ID1_60_6F**: Enable or disable ID filtering for IDs 0x60-0x6F
0: Transactions with these IDs are passed on to ATB master port 1.
1: Transactions with these IDs are discarded on master port 1.

Bit 5 **ID1_50_5F**: Enable or disable ID filtering for IDs 0x50-0x5F
0: Transactions with these IDs are passed on to ATB master port 1.
1: Transactions with these IDs are discarded on master port 1.

Bit 4 **ID1_40_4F**: Enable or disable ID filtering for IDs 0x40-0x4F
0: Transactions with these IDs are passed on to ATB master port 1.
1: Transactions with these IDs are discarded on master port 1.

Bit 3 **ID1_30_3F**: Enable or disable ID filtering for IDs 0x30-0x3F
0: Transactions with these IDs are passed on to ATB master port 1.
1: Transactions with these IDs are discarded on master port 1.

Bit 2 **ID1_20_2F**: Enable or disable ID filtering for IDs 0x20-0x2F
0: Transactions with these IDs are passed on to ATB master port 1.
1: Transactions with these IDs are discarded on master port 1.

Bit 1 **ID1_10_1F**: Enable or disable ID filtering for IDs 0x10-0x1F
0: Transactions with these IDs are passed on to ATB master port 1.
1: Transactions with these IDs are discarded on master port 1.

Bit 0 **ID1_0_F**: Enable or disable ID filtering for IDs 0x0-0xF
0: Transactions with these IDs are passed on to ATB master port 1.
1: Transactions with these IDs are discarded on master port 1.

Replicator claim tag set register (REP_CLAIMSET)

Address offset: 0xFA0

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMSET[3:0]**: Set claim tag bits
Write:
0000: No effect
xxx1: Set bit 0.
xx1x: Set bit 1.
x1xx: Set bit 2.
1xxx: Set bit 3.
Read:
0xF: Indicates there are four bits in claim tag.

Replicator claim tag clear register (REP_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMCLR[3:0]**: Reset claim tag bits
Write:
0000: No effect
xxx1: Clear bit 0.
xx1x: Clear bit 1.
x1xx: Clear bit 2.
1xxx: Clear bit 3.
Read: Returns current value of claim tag.

Replicator lock access register (REP_LAR)

Address offset: 0xFB0

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACCESS_W[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACCESS_W[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **ACCESS_W[31:0]**: CSTF register write access enable
The field enables write access to some CSTF registers by the processor cores (debuggers do not need to unlock the component).
0xC5ACCE55: Enable write access.
Other values: Disable write access.

Replicator lock status register (REP_LSR)

Address offset: 0xFB4

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKTYPE	LOCKGRANT	LOCKEXIST
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **LOCKTYPE**: Size of the CSTF_LAR register
0: 32 bits

Bit 1 **LOCKGRANT**: Current status of lock
This bit always returns 0 when read by an external debugger.
0: Write access is permitted.
1: Write access is blocked. Only read access is permitted.

Bit 0 **LOCKEXIST**: Existence of lock control mechanism
The bit indicates whether a lock control mechanism exists. It always returns 0 when read by an external debugger.
0: No lock control mechanism exists.
1: Lock control mechanism is implemented.

Replicator authentication status register (REP_AUTHSTAT)

Address offset: 0xFB8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **SNID[1:0]**: Security level for secure non-invasive debug
0x0: Not implemented

Bits 5:4 **SID[1:0]**: Security level for secure invasive debug
0x0: Not implemented

Bits 3:2 **NSNID[1:0]**: Security level for nonsecure non-invasive debug
0x0: Not implemented

Bits 1:0 **NSID[1:0]**: Security level for nonsecure invasive debug
0x0: Not implemented

Replicator CoreSight device identity register (REP_DEVID)

Address offset: 0xFC8

Reset value: 0x0000 0002

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PORTNUM[3:0]			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PORTNUM[3:0]**: Number of master ports implemented
0x2: Two master ports

Replicator CoreSight device type identity register (REP_DEVTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0022

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEVTYPEID[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **DEVTYPEID[7:0]**: Device type identifier
0x22: Trace replicator

Replicator CoreSight peripheral identity register 4 (REP_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: Register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm JEDEC code

Replicator CoreSight peripheral identity register 0 (REP_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0009

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number field, bits [7:0]
0x09: Replicator part number

Replicator CoreSight peripheral identity register 1 (REP_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0xB: Arm JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number field, bits [11:8]
0x9: REP part number

Replicator CoreSight peripheral identity register 2 (REP_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 002B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x2: r0p1

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x3: Arm JEDEC code

Replicator CoreSight peripheral identity register 3 (REP_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

Replicator CoreSight component identity register 0 (REP_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value

Replicator CoreSight component identity register 1 (REP_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0x9: CoreSight component

Bits 3:0 **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

Replicator CoreSight component identity register 2 (REP_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

Replicator CoreSight component identity register 3 (REP_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value

Trace replicator register map

Table 896. Replicator register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	REP_IDFILTER0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ID0_70_7F	ID0_60_6F	ID0_50_5F	ID0_40_4F	ID0_30_3F	ID0_20_2F	ID0_10_1F	ID0_0_F
	Reset value																									0	0	0	0	0	0	0	0
0x004	REP_IDFILTER1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ID1_70_7F	ID1_60_6F	ID1_50_5F	ID1_40_4F	ID1_30_3F	ID1_20_2F	ID1_10_1F	ID1_0_F
	Reset value																									0	0	0	0	0	0	0	0
0x008 - 0xF9C	Reserved	Reserved																															
0xFA0	REP_CLAIMSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET [3:0]			
	Reset value																													1	1	1	1
0xFA4	REP_CLAIMCLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR [3:0]			
	Reset value																													0	0	0	0
0xFA8- 0xFAC	Reserved	Reserved																															
0xFB0	REP_LAR	ACCESS_W[31:0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0xFB4	REP_LSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCTYPE	LOCKGRANT	LOCKEXIST
	Reset value																														0	1	1

Table 896. Replicator register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFB8	REP_AUTHSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
	Reset value																									0	0	0	0	0	0	0	0
0xFC8	REP_DEVID	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PORTNUM [3:0]			
	Reset value																													0	0	1	0
0xFCC	REP_DEVTYPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEVTYPEID[7:0]							
	Reset value																									0	0	1	0	0	0	1	0
0xFD0	REP_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON [3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4-0xFDC	Reserved	Reserved																															
0xFE0	REP_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	0	0	1	0	0	1
0xFE4	REP_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	1	0	0	1
0xFE8	REP_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	1	0	1	0	1	1
0xFEC	REP_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	REP_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	REP_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	REP_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	REP_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

78.12.16 Serial-wire output (SWO)

The SWO is a CoreSight component that formats the trace stream from the Cortex-M55 processor ITM and outputs it on the single wire TRACESWO output.

Compared to the TPIU, the SWO contains:

- no formatter
- no pattern generator
- an 8-bit ATB input
- no synchronous trace output, that is, no TRACEDATA or TRACECLK pins
- no support for flush, because this is not required
- no support for triggering

The SWO output supports Manchester encoded and UART NRZ formats.

For more information about the serial-wire output CoreSight component, refer to the Arm CoreSight components Technical Reference Manual *[8]*.

78.12.17 SWO registers

The register file base address for the SWO is defined in *Table 880: Trace subsystem ROM table*.

SWO current output divisor register (SWO_CODR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	PRESCALER[12:0]												
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:13 Reserved, must be kept at reset value.

Bits 12:0 **PRESCALER[12:0]**: SWO baud rate scaling

The baud rate is the trace clock frequency divided by (PRESCALER - 1). The baud rate changes instantly, so it is recommended to stop the trace source and wait until the port is idle before writing to this register.

SWO selected pin protocol register (SWO_SPPR)

Address offset: 0x0F0

Reset value: 0x0000 0001

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PPROT[1:0]	
														rw	rw

Bits 31:2 Reserved, must be kept at reset value.

Bits 1:0 **PPROT[1:0]**: Pin protocol
0x1: Manchester
0x2: NRZ
Others: Reserved

SWO formatter and flush status register (SWO_FFSR)

Address offset: 0x300

Reset value: 0x0000 0008

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FTNONSTOP	TCPRESENT	FTSTOPPED	FLINPROG
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bit 3 **FTNONSTOP**: Change of settings without stopping formatter
1: Change of settings is allowed with formatter running.

Bit 2 **TCPRESENT**: TRACECTL pin present on SWO
0: TRACECTL pin not present

Bit 1 **FTSTOPPED**: Formatter stopped
0: Formatter running
The bit always returns 0 as the SWO formatter cannot be stopped in this device.

Bit 0 **FLINPROG**: Flush in progress
0: Flush is not in progress
The bit always returns 0 as SWO flushing is not supported in this device.

SWO claim tag set register (SWO_CLAIMSET)

Address offset: 0xFA0

Reset value: 0x0000 000F

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMSET[3:0]**: Set claim tag bits
Write:
0000: No effect
xxx1: Set bit 0.
xx1x: Set bit 1.
x1xx: Set bit 2.
1xxx: Set bit 3.
Read:
0xF: Indicates there are four bits in claim tag.

SWO claim tag clear register (SWO_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLAIMCLR[3:0]**: Reset claim tag bits
Write:
0000: No effect
xxx1: Clear bit 0.
xx1x: Clear bit 1.
x1xx: Clear bit 2.
1xxx: Clear bit 3.
Read: Returns current value of claim tag.

SWO lock access register (SWO_LAR)

Address offset: 0xFB0

Reset value: 0xXXXX XXXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ACCESS_W[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACCESS_W[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **ACCESS_W[31:0]**: SWO register write access enable
Enables write access to some SWO registers by the processor cores (debuggers do not need to unlock the component)
0xC5ACCE55: Enable write access.
Other values: Disable write access.

SWO lock status register (SWO_LSR)

Address offset: 0xFB4

Reset value: 0x0000 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKTYPE	LOCKGRANT	LOCKEXIST
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **LOCKTYPE**: Size of the SWO_LAR register
0: 32 bits

Bit 1 **LOCKGRANT**: Current status of lock
This bit always returns 0 when read by an external debugger.
0: Write access is permitted.
1: Write access is blocked. Only read access is permitted.

Bit 0 **LOCKEXIST**: Implementation of a lock control mechanism
The bit indicates whether a lock control mechanism is implemented. It always returns 0 when read by an external debugger.
0: No lock control mechanism available
1: Lock control mechanism is implemented.

SWO authentication status register (SWO_AUTHSTAT)

Address offset: 0xFB8

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **SNID[1:0]**: Security level for secure non-invasive debug
0x0: Not implemented

Bits 5:4 **SID[1:0]**: Security level for secure invasive debug
0x0: Not implemented

Bits 3:2 **NSNID[1:0]**: Security level for nonsecure non-invasive debug
0x0: Not implemented

Bits 1:0 **NSID[1:0]**: Security level for nonsecure invasive debug
0x0: Not implemented

SWO device configuration register (SWO_DEVID)

Address offset: 0xFC8

Reset value: 0x0000 0EA0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	SWOUARTNRZ	SWOMAN	TCLKDATA	FIFOSIZE[2:0]			CLKRELAT	MUXNUM[4:0]				
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bit 11 **SWOUARTNRZ**: SWO UART or NRZ
1: Supported

Bit 10 **SWOMAN**: SWO Manchester format
1: Supported

Bit 9 **TCLKDATA**: Trace clock plus data
1: Supported

Bits 8:6 **FIFOSIZE[2:0]**: FIFO size in powers of two
0x2: FIFO size = 4 (16 bytes)

Bit 5 **CLKRELAT**: ATB clock to traceclkin relation
This bit indicates the relationship between the ATB clock and traceclkin (synchronous or asynchronous).
1: Asynchronous

Bits 4:0 **MUXNUM[4:0]**: Number/type of ATB input port multiplexing
0x0: None

SWO device type identifier register (SWO_DEVTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0011

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE[3:0]				MAJORTYPE[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUBTYPE[3:0]**: Subclassification
0x1: Trace port component

Bits 3:0 **MAJORTYPE[3:0]**: Major classification
0x1: Trace sink component

SWO CoreSight peripheral identity register 4 (SWO_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Register file size
0x0: Register file occupies a single 4-Kbyte region.

Bits 3:0 **JEP106CON[3:0]**: JEP106 continuation code
0x4: Arm JEDEC code

SWO CoreSight peripheral identity register 0 (SWO_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0014

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PARTNUM[7:0]**: Part number field, bits [7:0]
0x14: SWO part number

SWO CoreSight peripheral identity register 1 (SWO_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B9

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID[3:0]				PARTNUM[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **JEP106ID[3:0]**: JEP106 identity code field, bits [3:0]
0xB: Arm JEDEC code

Bits 3:0 **PARTNUM[11:8]**: Part number field, bits [11:8]
0x9: SWO part number

SWO CoreSight peripheral identity register 2 (SWO_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 002B

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	JEP106ID[6:4]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Component revision number
0x2: r0p2

Bit 3 **JEDEC**: JEDEC assigned value
1: Designer ID specified by JEDEC

Bits 2:0 **JEP106ID[6:4]**: JEP106 identity code field, bits [6:4]
0x3: Arm JEDEC code

SWO CoreSight peripheral identity register 3 (SWO_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Metal fix version
0x0: No metal fix

Bits 3:0 **CMOD[3:0]**: Customer modified
0x0: No customer modifications

SWO CoreSight component identity register 0 (SWO_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[7:0]**: Component ID field, bits [7:0]
0x0D: Common ID value

SWO CoreSight component identity register 1 (SWO_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE[11:8]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component ID field, bits [15:12] - component class
0x9: CoreSight component

Bits 3:0 **PREAMBLE[11:8]**: Component ID field, bits [11:8]
0x0: Common ID value

SWO CoreSight component identity register 2 (SWO_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[19:12]**: Component ID field, bits [23:16]
0x05: Common ID value

SWO CoreSight component identity register 3 (SWO_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PREAMBLE[27:20]**: Component ID field, bits [31:24]
0xB1: Common ID value

SWO register map

Table 897. SWO register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x010	SWO_CODR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRESCALER[12:0]													
	Reset value																				0	0	0	0	0	0	0	0	0	0	0	0	0
0x014-0x0EC	Reserved	Reserved																															
0x0F0	SWO_SPPR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PPROT[1:0]	
	Reset value																															0	1
0x0F4 - 0x2FC	Reserved	Reserved																															
0x300	SWO_FFSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FTNONSTOP	TCPRESENT	FTSTOPPED	FLINPROG
	Reset value																													1	0	0	0
0x304-0xF9C	Reserved	Reserved																															
0xFA0	SWO_CLAIMSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMSET [3;0]			
	Reset value																													1	1	1	1
0xFA4	SWO_CLAIMCLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLAIMCLR [3;0]			
	Reset value																													0	0	0	0
0xFA8 to 0xFAC	Reserved	Reserved																															

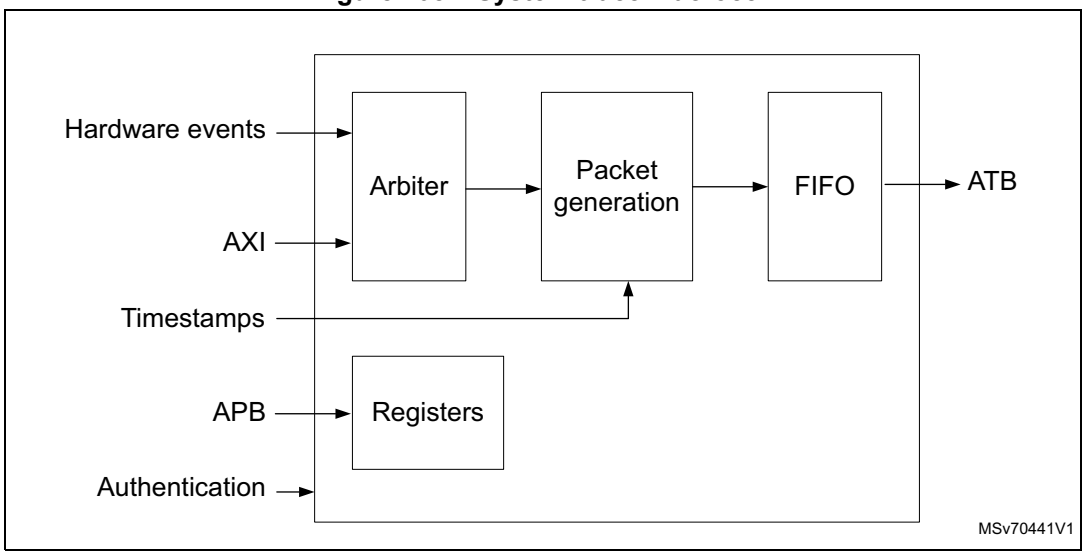
Table 897. SWO register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFB0	SWO_LAR	ACCESS_W[31;0]																															
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
0xFB4	SWO_LSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	LOCKTYPE	LOCKGRANT	LOCKEXIST
	Reset value																														0	1	1
0xFB8	SWO_AUTHSTAT	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1;0]		NSNID[1:0]		NSID[1:0]	
	Reset value																									0	0	0	0	0	0	0	0
0xFBC-0xFC4	Reserved	Reserved																															
0xFC8	SWO_DEVID	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWOUARTNRZ	SWOMAN	TCLKDATA	FIFOSIZE [2:0]			CLKRELAT	MAXNUM[4:0]				
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	0	1	0	0	0	0	0
0xFCC	SWO_DEVTYPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUBTYPE [3:0]				MAJORTYPE [3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1
0xFD0	SWO_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				JEP106CON [3:0]			
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0xFD4-0xFDC	Reserved	Reserved																															
0xFE0	SWO_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PARTNUM[7:0]							
	Reset value																									0	0	0	1	0	1	0	0
0xFE4	SWO_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	JEP106ID [3:0]				PARTNUM [11:8]			
	Reset value																									1	0	1	1	1	0	0	1
0xFE8	SWO_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	JEP106ID [6:4]		
	Reset value																									0	0	1	0	1	0	1	1
0xFEC	SWO_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	SWO_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	SWO_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PREAMBLE [11:8]			
	Reset value																									1	0	0	1	0	0	0	0
0xFF8	SWO_CIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[19:12]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	SWO_CIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PREAMBLE[27:20]							
	Reset value																									1	0	1	1	0	0	0	1

78.12.18 System trace macrocell (STM)

The system trace macrocell is a CoreSight component which allows instrumentation data generated by software to be incorporated in the trace output. The software writes instrumentation data to the STM via a memory-mapped port on the AXI system interconnect. Data are formatted and time-stamped for output on the ATB trace bus, where they are merged with the trace coming from other sources (ETM, ITM).

Figure 1087. System trace macrocell



The STM is configured with the following values to minimize the footprint in gates count:

- DATA_FIFO_DEPTH = 4
- CHN_FIFO_DEPTH = 4
- HW_EVOBIF_PRESENT = TRUE

The various AXI masters can generate trace by writing to the extended stimulus ports of the STM via its AXI slave port, mapped in a 1-Mbyte window of address space (see *Section 2.3: Memory organization*). A 256-byte extended stimulus port comprises 16 x 64-bit locations or registers. Writes to different locations generate different types of trace packets, as shown in the table below.

The base address for stimulus port n is the STM base address + n × 256. A master can use any number of the 4096 available stimulus ports mapped in the 1-Mbyte window.

Table 898. STM extended stimulus port memory map

Address offset	Name	Description
0x00	G_DMTS	Data, marked with timestamp, guaranteed
0x08	G_DM	Data, marked, guaranteed
0x10	G_DTS	Data, with timestamp, guaranteed
0x18	G_D	Data, guaranteed
0x20-0x58	-	Reserved
0x60	G_FLAGTS	Flag with timestamp, guaranteed
0x68	G_FLAG	Flag, guaranteed

Table 898. STM extended stimulus port memory map (continued)

Address offset	Name	Description
0x70	G_TRIGTS	Trigger with timestamp, guaranteed
0x78	G_TRIG	Trigger, guaranteed
0x80	I_DMTS	Data, marked with timestamp, invariant timing
0x88	I_DM	Data, marked, invariant timing
0x90	I_DTS	Data, with timestamp, invariant timing
0x98	I_D	Data, invariant timing
0xA0-0xD8	-	Reserved
0xE0	I_FLAGTS	Flag with timestamp, invariant timing
0xE8	I_FLAG	Flag, invariant timing
0xF0	I_TRIGTS	Trigger with timestamp, invariant timing
0xF8	I_TRIG	Trigger, invariant timing

The trace packets contain the 8-bit identity of the master that triggered them. This identity is decoded from the nonsecure master ID carried on the AWID wires of the AXI interconnect. The ID also includes the security status of the master at the time.

The ID is mapped as follows:

- Bit [7] = 0
- Bit [6] = secure/nonsecure state
- Bit [5:0] = master ID

A list of valid IDs and how they map to the originating master is given in the table below.

Table 899. STM trace packet ID mapping to AXI masters

Master	Secure access ID	Nonsecure access ID
Cortex-M55	0x00	0x40
HPDMA1	0x01	0x41
AXI-AP	0x02	0x42

The STM can also generate trace packets or triggers when certain hardware events are detected. Events can be selected from one of two banks, each bank containing up to 32 events. The bank is selected in the STM_HEBSR register, and the 32 events can be individually enabled in STM_HETER register. All events are edge sensitive.

The events for each banks are shown in *Table 900*.

Table 900. Hardware event mapping

Bank	Event	EXTMUX(7:0) = 0	-	Bank	Event	EXTMUX(7:0) = 0
HEBS = 0	0	HWEVENTA ↑	-	HEBS = 1	0	HPDMA0_ch0_irq
	1	HWEVENTA ↓	-		1	HPDMA0_ch1_irq
	2	HWEVENTB ↑	-		2	HPDMA0_ch2_irq
	3	HWEVENTB ↓	-		3	HPDMA0_ch3_irq
	4	-	-		4	HPDMA0_ch4_irq
	5	NPU_DBG[0]	-		5	HPDMA0_ch5_irq
	6	NPU_DBG[1]	-		6	HPDMA0_ch6_irq
	7	NPU_DBG[2]	-		7	HPDMA0_ch7_irq
	8	NPU_DBG[3]	-		8	HPDMA0_ch8_irq
	9	NPU_DBG[4]	-		9	HPDMA0_ch9_irq
	10	NPU_DBG[5]	-		10	HPDMA0_ch10_irq
	11	NPU_DBG[6]	-		11	HPDMA0_ch11_irq
	12	NPU_DBG[7]	-		12	HPDMA0_ch12_irq
	13	-	-		13	HPDMA0_ch13_irq
	14	GPU_irq	-		14	HPDMA0_ch14_irq
	15	-	-		15	HPDMA0_ch15_irq
	16	EXTI0_irq	-		16	GPDMA0_ch0_irq
	17	EXTI1_irq	-		17	GPDMA0_ch1_irq
	18	EXTI2_irq	-		18	GPDMA0_ch2_irq
	19	EXTI3_irq	-		19	GPDMA0_ch3_irq
	20	EXTI4_irq	-		20	GPDMA0_ch4_irq
	21	EXTI5_irq	-		21	GPDMA0_ch5_irq
	22	EXTI6_irq	-		22	GPDMA0_ch6_irq
	23	EXTI7_irq	-		23	GPDMA0_ch7_irq
	24	EXTI8_irq	-		24	GPDMA0_ch8_irq
	25	EXTI9_irq	-		25	GPDMA0_ch9_irq
	26	EXTI10_irq	-		26	GPDMA0_ch10_irq
	27	EXTI11_irq	-		27	GPDMA0_ch11_irq
	28	EXTI12_irq	-		28	GPDMA0_ch12_irq
	29	EXTI13_irq	-		29	GPDMA0_ch13_irq
	30	EXTI14_irq	-		30	GPDMA0_ch14_irq
	31	EXTI15_irq	-		31	GPDMA0_ch15_irq

For more information on the system trace macrocell, refer to the Arm CoreSight STM-500 System Trace Macrocell Technical Reference Manual [*6*].

78.12.19 STM registers

The register file base address for the STM is defined in *Table 880*.

STM hardware event enable register (STM_HEER)

Address offset: 0xD00

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HEE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HEE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **HEE[31:0]**: Hardware event enable

HEE is used to enable hardware events to generate trace. The register defines one bit per hardware event.

0: Hardware event disabled

1: Hardware event enabled

STM hardware event trigger enable register (STM_HETER)

Address offset: 0xD20

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HETE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HETE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **HETE[31:0]**: Trigger generation enable

HETE is a bit mask used to enable trigger generation on hardware events. This bitfield defines one bit per hardware event.

0: Disabled

1: Enabled

STM hardware event bank select register (STM_HEBSR)

Address offset: 0xD60

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HEBS
															rw

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **HEBS**: Hardware event bank selector

HEBS is used to select the bank of 32 hardware events to control (see *Table 900*).

0: Bank 0

1: Bank 1

STM hardware event main control register (STM_HEMCR)

Address offset: 0xD64

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATBTRIGEN	TRIGCLEAR	TRIGSTATUS	TRIGCTL	Res.	ERRDETECT	COMPEN	EN
								rw	rw	r	rw		r	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **ATBTRIGEN**: ATB trace trigger enable

When set, this bit enables the STM to use the ATID value of 0x7D when a trigger event on match using STMHETER occurs.

0: Do not output ATB trace triggers.

1: Use ATB trace triggers.

Bit 6 **TRIGCLEAR**: Clear trigger status

When TRIGCTL indicates single-shot mode, this bit is used to clear TRIGSTATUS. Writing a 1 to this bit when in multi-shot mode is unpredictable.

0: No effect

1: Clear TRIGSTATUS (if equal to 1).

Bit 5 **TRIGSTATUS**: trigger status

When TRIGCTL indicates single-shot mode, this indicates whether the single trigger has occurred.

0: Trigger has not occurred.

1: Trigger has occurred.

Bit 4 **TRIGCTL**: Trigger control

0: Triggers are multi-shot.

1: Triggers are single-shot.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **ERRDETECT**: Error detection
This bit enables error detection on the hardware event tracing.
0: Disabled
1: Enabled

Bit 1 **COMPEN**: Compression enable
This bit enables leading 0 suppression of hardware event data values in the trace stream.
0: Disabled
1: Enabled

Bit 0 **EN**: Hardware event tracing enable
0: Disabled
1: Enabled

STM hardware event external multiplex control register (STM_HEEXTMUXR)

Address offset: 0xD68

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXTMUX[7:0]							
								rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **EXTMUX[7:0]**: Signals for events selector
This bitfield selects the signals to be mapped onto the 64 hardware events that can be traced by the STM (refer to *Table 900*).

STM hardware event master number register (STM_HEMASTR)

Address offset: 0xDF4

Reset value: 0x0000 0080

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MASTER[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **MASTER[15:0]**: STPv2 master number
This bitfield indicates the STPv2 master number of hardware event trace. This number is the master number presented in STPv2.
0x80: Hardware events are associated with master 0x80.
Others: reserved

STM hardware event features 1 register (STM_HEFEAT1R)

Address offset: 0xDF8

Reset value: 0x3020 0035

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	HEEXTMUXSIZE[2:0]			Res.	Res.	Res.	Res.	NUMHE[8:1]							
	r	r	r					r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMHE[0]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HECOMP[1:0]		HEMASTR	HEERR	Res.	HETER
r										r	r	r	r		r

Bit 31 Reserved, must be kept at reset value.

Bits 30:28 **HEEXTMUXSIZE[2:0]**: EXTMUX bitfield size
This bitfield indicates the size of EXTMUX bitfield in STM_HEEXTMUXR.
0x3: 8-bit wide (up to 256 multiplexers)
Others: Reserved

Bits 27:24 Reserved, must be kept at reset value.

Bits 23:15 **NUMHE[8:0]**: Number of hardware events supported by the STM
0x40: 64 hardware events

Bits 14:6 Reserved, must be kept at reset value.

Bits 5:4 **HECOMP[1:0]**: Data compression on hardware event tracing support
0x3: Data compression support is programmable. COMPEN in STM_HEMCR is implemented.

Bit 3 **HEMASTR**: STMHEMASTR support
0: STMHEMASTR is read-only.

Bit 2 **HEERR**: Hardware event error detection support
1: Hardware event error detection implemented. ERRDETECT in STM_HEMCR is implemented.

Bit 1 Reserved, must be kept at reset value.

Bit 0 **HETER**: STM_HETER support
1: Implemented

STM hardware event features ID register (STM_HEIDR)

Address offset: 0xDFC

Reset value: 0x0000 0011

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	VENDSPEC[3:0]				CLASSREV[3:0]				CLASS[3:0]			
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:12 Reserved, must be kept at reset value.

Bits 11:8 **VENDSPEC[3:0]**: Vendor specific
This bitfield identifies any vendor specific modifications or mappings.
0: Vendor specific information
Others: Reserved

Bits 7:4 **CLASSREV[3:0]**: Programmer model revision
1: Revision
Others: Reserved

Bits 3:0 **CLASS[3:0]**: Programmer model
1: Hardware event control
Others: Reserved

STM stimulus port enable register (STM_SPER)

Address offset: 0xE00

Reset value: 0x0000 0000

This register is used to enable the stimulus registers to generate the trace.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SPE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SPE[31:0]**: Stimulus port enable
This register is used to enable STM registers to generate the trace. It defines one bit per stimulus register. Writing one enables the appropriate stimulus port. Writing 0 disables the appropriate stimulus port. This register is used in conjunction with STM_SPSCR.
0: Disabled
1: Enabled

STM stimulus port trigger enable register (STM_SPTER)

Address offset: 0xE20

Reset value: 0x0000 0000

This register is used to enable trigger generation on writes to enabled stimulus port registers.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SPTE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPTE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **SPTE[31:0]**: Trigger generation bit mask

This register is a bit mask used to enable the trigger generation on writes to enabled stimulus port registers. It defines one bit per stimulus port register.

0: Disabled

1: Enabled

STM stimulus port select configuration register (STM_SPSCR)

Address offset: 0xE60

Reset value: 0x0000 0000

This register allows a debugger to program which stimulus ports the STM_SPER and STM_SPTER registers apply to.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	PORTSEL[7:0]								Res.	Res.	Res.	Res.
				rw	rw	rw	rw	rw	rw	rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PORTCTL[1:0]	
														rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:20 **PORTSEL[7:0]**: Stimulus port selector

This bitfield defines which group of 32 stimulus ports, the STM_SPTER and/or STM_SPER registers apply to.

0000000: Reserved

nnnn nnn1: Select 1 group, 0bnnn nnnn.

nnnn nn10: Select 2 groups,0bnnn nnn0 to 0bnnn nnn1.

nnnn n100: Select 4 groups, 0bnnn nn00 to 0bnnn nn11.

..

n100 0000: Select 64 groups, 0bn00 0000 to 0bn11 1111.

1000 0000: Select all 128 groups.

Bits 19:2 Reserved, must be kept at reset value.

Bits 1:0 **PORTCTL[1:0]**: Port selection control mode

This bitfield defines how the port selection programmed in PORTSEL[7:0] is applied.

0x0: Port selection not used. STM_PER and STM_PTER apply to all groups.

0x1: Port selection applies only to STM_SPTER. STM_PER applies to all groups

0x2: Reserved

0x3: Port selection applies to both STM_SPER and STM_SPTER registers.

STM stimulus port master select configuration register (STM_SPMSCR)

Address offset: 0xE64

Reset value: 0x0000 0000

This register allows a debugger to program which masters the STM_SPSCR register applies to.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASTSEL[7:1]						
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MASTSEL[0]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASTCTL
rw															rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:15 **MASTSEL[7:0]**: Master selection

This bitfield defines which masters the STM_SPSCR register applies to. Six masters are supported (see *Table 899* for the master ID mapping).
0000 0001: Select master 0x00.
0000 0011: Select master 0x01.
0000 0101: Select master 0x02.
0000 0010: Select master 0x00 and master 0x01.
0000 0100: Select master 0x00, master 0x01 and master 0x02.
1000 0001: Select master 0x40.
1000 0011: Select master 0x41.
1000 0101: Select master 0x42.
1000 0010: Select master 0x40 and master 0x41.
1000 0100: Select master 0x40, master 0x41 and master 0x42.
1000 0000: Select all masters.
Others: Reserved

Bits 14:1 Reserved, must be kept at reset value.

Bit 0 **MASTCTL**: Master control mode

This bitfield defines how the master is applied.
0: Master selection not used. STM_SPSCR applies to all masters.
1: Master selection applies to STM_SPSCR.

STM stimulus port override control register (STM_SPOVERRIDER)

Address offset: 0xE68

Reset value: 0x0000 0000

This register allows a debugger to override various features of the STM.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	PORTSEL[12:1]											
				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PORTSEL[0]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVERTS	OVERCTL[1:0]	
rw													rw	rw	rw

Bits 31:28 Reserved, must be kept at reset value.

Bits 27:15 **PORTSEL[12:0]**: Port override select

This bitfield defines which stimulus ports the override controls apply to.
0000 0000 0000 0: Reserved
nnnn nnnn nnnn 1: Select port 0bnnnn nnnn nnnn.
nnnn nnnn nnn1 0: Select ports 0bnnnn nnnn nnn0 to 0bnnnn nnnn nnn1.
nnnn nnnn nn10 0: Select ports 0bnnnn nnnn nn00 to 0bnnnn nnnn nn11.
..
n100 0000 0000: Select ports 0bn000 0000 0000 to 0bn111 1111 1111.
1000 0000 0000: Select all 4096 ports.

Bits 14:3 Reserved, must be kept at reset value.

Bit 2 **OVERTS**: Override timestamp request

This override requests all stimulus port writes that cause trace to be traced with a timestamp (when possible). As with normal operation, this does not ensure that all packets are generated with timestamps. This bitfield is independent from OVERCTL and PORTSEL.
0: Override disabled
1: Override enabled

Bits 1:0 **OVERCTL[1:0]**: Override control mode

This bitfield defines how the port selection is applied.
0x0: Override controls disabled
0x1: Ports selected by PORTSEL always behave as guaranteed transactions.
0x2: Ports selected by PORTSEL always behave as invariant timing transactions0
0x3: Reserved

STM stimulus port master override control register (STM_SPMOVERRIDER)

Address offset: 0xE6C

Reset value: 0x0000 0000

This register allows a debugger to select which masters STM_SPOVERRIDER applies to.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASTSEL[7:1]						
									rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MASTSEL[0]	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASTCTL
rw															rw

Bits 31:23 Reserved, must be kept at reset value.

Bits 22:15 **MASTSEL[7:0]**: Master port override selection

This bitfield defines which master ports the override controls apply to. Six masters are supported (see *Table 899* for the master ID mapping).

0000 0001: Select master 0x00.
0000 0011: Select master 0x01.
0000 0101: Select master 0x02.
0000 0010: Select master 0x00 and master 0x01.
0000 0100: Select master 0x00, master 0x01 and master 0x02.
1000 0001: Select master 0x40.
1000 0011: Select master 0x41.
1000 0101: Select master 0x42.
1000 0010: Select master 0x40 and master 0x41.
1000 0100: Select master 0x40, master 0x41 and master 0x42.
1000 0000: Select all masters.
Others: Reserved

Bits 14:1 Reserved, must be kept at reset value.

Bit 0 **MASTCTL**: Master selection control mode

This bitfield defines how the master selection is applied.

0: Override controls disabled. STM_SPOVERRIDER applies equally to all masters.
1: Master selection enabled. STM_SPOVERRIDER applies to the masters selected by MASTSEL[7:0].

STM stimulus port trigger control and status register (STM_SPTRIGCSR)

Address offset: 0xE70

Reset value: 0x0000 0000

This register is used to control the STM triggers induced by STM_SPTER.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATBTRIGEN_DIR	ATBTRIGEN_TE	TRIGCLEAR	TRIGSTATUS	TRIGCTL
											rw	rw	w	r	rw

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **ATBTRIGEN_DIR**: ATB trace on write to TRIG

When set, this bit enables the STM to use the ATID value of 0x7D when a trigger event on writes to TRIG location occurs.

0: Does not output ATB trace triggers on write to TRIG location.
1: Outputs ATB trace trigger for trigger on write to TRIG location.

Bit 3 **ATBTRIGEN_TE**: ATB trace-trigger output

When set, this bit enables the STM to use the ATID value of 0x7D when a trigger event on match using STM_SPTER occurs.

0: Does not output ATB trace trigger for trigger on match using STM_SPTER.
1: Outputs ATB trace trigger for trigger on match using STM_SPTER.

Bit 2 **TRIGCLEAR**: trigger status clear

When TRIGCTL indicates single-shot mode, this TRIGCLEAR bit is used to clear TRIGSTATUS. Writing a one to this bit when in multi-shot mode leads to an unpredictable behavior.

0: No effect

1: Clear TRIGSTATUS if TRIGSTATUS = 1.

Bit 1 **TRIGSTATUS**: Trigger status

When TRIGCTL indicates single-shot mode, this TRIGSTATUS bit indicates whether the single trigger has occurred.

0: Trigger has not occurred.

1: Trigger has occurred.

Bit 0 **TRIGCTL**: Trigger control

0: Triggers are multi-shot.

1: Triggers are single-shot.

STM trace control and status register (STM_TCSR)

Address offset: 0xE80

Reset value: 0x0000 0004

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSY	TRACEID[6:0]						
								r	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMPEN	Res.	Res.	SYNCEN	TSEN	EN
										rw			r	rw	rw

Bits 31:24 Reserved, must be kept at reset value.

Bit 23 **BUSY**: STM busy

This bit indicates if the STM is busy (for example because STM trace FIFO is not empty).

0: STM is not busy.

1: STM is busy.

Bits 22:16 **TRACEID[6:0]**: ATB Trace ID

Bits 15:6 Reserved, must be kept at reset value.

Bit 5 **COMPEN**: Compression enable for stimulus ports

0: Compression disabled. Data transfers are transmitted at the size of the transaction.

1: Compression enabled. Data transfers are compressed to save bandwidth.

Bits 4:3 Reserved, must be kept at reset value.

Bit 2 **SYNCEN**: STM_SYNCR implementation

1: Implemented

Bit 1 **TSEN**: Timestamp requests enable

This bit controls if timestamp requests are ignored or taken into account.

0: Timestamping disabled. Requests for timestamp generation are ignored, and stimulus port writes selecting timestamping are treated as if it were not selected.

1: Timestamping enabled. If stimulus writes select timestamping, a timestamp is output according to STPv2.

Bit 0 **EN**: Global STM enable
0: STM disabled
1: STM enabled

STM timestamp stimulus register (STM_TSSTIMR)

Address offset: 0xE84

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FORCETS
															w

Bits 31:1 Reserved, must be kept at reset value.

Bit 0 **FORCETS**: Force timestamp stimulus
Write a one to this bit requests the next stimulus port write which causes trace to be upgraded to have a timestamp. Writes this bit as 0 are ignored.
1: Force timestamp stimulus

STM timestamp frequency register (STM_TSFREQR)

Address offset: 0xE8C

Reset value: 0x0000 0000

This register is used to indicate the frequency of the timestamp counter. The unit of measurement is increments per second. When the STPv2 protocol is used, this register contains the value output in FREQ and FREQ_TS packets. The timestamp frequency is output in the STPv2 protocol at every synchronization point when TSEN = 1 in STM_TCSR.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
FREQ[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FREQ[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **FREQ[31:0]**: Timestamp frequency
This bitfield indicates the frequency of the timestamp counter (in Hz).

STM synchronization control register (STM_SYNCR)

Address offset: 0xE90

Reset value: 0x0000 0000

This register controls the interval between synchronization packets as a function of the number of bytes of trace generated. It provides a hint of the desired synchronization

frequency since implementations are permitted to be inaccurate. Writing 0x0000 0000 to this register disables the synchronization counter. However any other implementation defined synchronization mechanism continues to operate independently (refer to the next register STM_AUXCR).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	MODE	COUNT[8:0]									Res.	Res.	Res.
			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw			

Bits 31:13 Reserved, must be kept at reset value.

Bit 12 **MODE**: Mode control

0: COUNT[8:0] defines a value N. Synchronization period is 8 x N bytes.

1: COUNT[8:4] defines a value N. Synchronization period is 2^N bytes. N must be in the range of 12 to 27 inclusive. Other values lead to unpredictable results.

Bits 11:3 **COUNT[8:0]**: Number of bytes between synchronization packets

This bitfield indicates the counter value for the number of bytes between synchronization packets.

Bits 2:0 Reserved, must be kept at reset value.

STM auxiliary control register (STM_AUXCR)

Address offset: 0xE94

Reset value: 0x0000 0000

This register is used for implementation defined STM controls.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	QHWEVOVERRIDE	Res.	Res.	Res.	Res.	PRIORINVDIS	ASYNCPE	FIFOAF
								rw					rw	rw	rw

Bits 31:8 Reserved, must be kept at reset value.

Bit 7 **QHWEVOVERRIDE**: Override control for the Q-Channels

0: No override. Q-Channel is able to accept a quiescence request when hardware event tracing is enabled.

1: Override. Q-Channel denies a quiescence request when hardware event tracing is enabled.

Bits 6:3 Reserved, must be kept at reset value.

Bit 2 **PRIORINVDIS**: AXI versus hardware flush arbitration
This bit controls the arbitration between AXI and hardware during flush.
0: Priority inversion. When AXI flush is finished, the hardware gets priority until hardware flush is done.
1: Priority inversion disabled. AXI always has priority over hardware.

Bit 1 **ASYNCPE**: ASYNC priority
0: ASYNC priority is always lower than trace.
1: ASYNC priority increases on second synchronization request.

Bit 0 **FIFOAF**: Auto-flush
0: Disabled
1: Enabled

STM features 1 register (STM_FEAT1R)

Address offset: 0xEA0

Reset value: 0x0065 87D1

This register indicates the features of the STM in conjunction with STM_FEAT2R and STM_FEAT3R.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWOEN[1:0]		SYNCEN[1:0]		HWTEN[1:0]		TSPRESCALE[1:0]	
								r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRIGCTL[1:0]		TRACEBUS[3:0]				SYNC[1:0]		FORCETS	TSFREQ	TS[1:0]		PROT[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:24 Reserved, must be kept at reset value.

Bits 23:22 **SWOEN[1:0]**: SWOEN support
Support of SWOEN in STM_TCSR.
0x1: SWOEN in STM_TCSR is not implemented.

Bits 21:20 **SYNCEN[1:0]**: SYNCEN support
Support of SYNCEN in STM_TCSR.
0x2: SYNCEN in STM_TCSR is implemented but always reads as 1.

Bits 19:18 **HWTEN[1:0]**: HWTEN support
Support of HWTEN in STM_TCSR.
0x1: HWTEN in STM_TCSR is not implemented.

Bits 17:16 **TSPRESCALE[1:0]**: Timestamp prescale support
0x1: Not implemented.

Bits 15:14 **TRIGCTL[1:0]**: Trigger control support
0x2: Multi-shot and single-shot triggers supported. STM_TRIGCSR is implemented.

Bits 13:10 **TRACEBUS[3:0]**: Trace bus support
0x1: CoreSight ATB plus ATB trigger support implemented. TRACEID in STM_TCSR, and ATBTRIGEN_DIR and ATBTRIGEN_TE in STM_SPTRIGCSR, are implemented.

Bits 9:8 **SYNC[1:0]**: STM_SYNCR support
0x3: STM_SYNCR implemented with MODE control

Bit 7 **FORCETS**: STM_TSSTIMR support
1: FORCETS in STM_TSSTIMR is implemented.

Bit 6 **TSFREQ**: Timestamp frequency indication configuration
1: STM_TSFREQR is read-write.

Bits 5:4 **TS[1:0]**: Timestamp support
0x1: Absolute timestamps implemented

Bits 3:0 **PROT[3:0]**: Protocol
0x1: STPv2 protocol

STM features 2 register (STM_FEAT2R)

Address offset: 0xEA4

Reset value: 0x0001 14F2

This register indicates the features of the STM in conjunction with STM_FEAT1R and STM_FEAT3R.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPTYPE[1:0]	
														r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DSIZE[3:0]				Res.	SPTRTYPE[1:0]		PRIVMASK[1:0]		SPOVERRIDE	SPCOMP[1:0]		Res.	SPER	SPTER[1:0]	
r	r	r	r		r	r	r	r	r	r	r		r	r	r

Bits 31:18 Reserved, must be kept at reset value.

Bits 17:16 **SPTYPE[1:0]**: Stimulus port type support
0x1: Only extended stimulus ports implemented

Bits 15:12 **DSIZE[3:0]**: Fundamental data size
0x1: 64-bit data

Bit 11 Reserved, must be kept at reset value.

Bits 10:9 **SPTRTYPE[1:0]**: Stimulus port transaction type support
0x2: Both invariant timing and guaranteed transactions are supported.

Bits 8:7 **PRIVMASK[1:0]**: STM_PRIVMASKR support
0x1: STM_PRIVMASKR is not implemented.

Bit 6 **SPOVERRIDE**: STM_OVERRIDER support
1: STM_SPOVERRIDER and STM_SPMOVERRIDER are implemented.

Bits 5:4 **SPCOMP[1:0]**: Data compression on stimulus ports support
0x3: Data compression support is programmable. COMPEN in STM_TCSR is implemented.

Bit 3 Reserved, must be kept at reset value.

Bit 2 **SPER**: STM_SPER presence
0: STM_SPER is implemented.

Bits 1:0 **SPTER[1:0]**: STM_SPTER support
0x2: STM_SPTER is implemented.

STM features 3 register (STM_FEAT3R)

Address offset: 0xEA8

Reset value: 0x0000 007F

This register indicates the features of the STM in conjunction with STM_FEAT1R and STM_FEAT2R.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NUMMAST[6:0]						
									r	r	r	r	r	r	r

Bits 31:7 Reserved, must be kept at reset value.

Bits 6:0 **NUMMAST[6:0]**: Number of stimulus ports masters implemented, minus 1
0x7F: 128 masters are implemented (only three masters are connected).

STM claim tag set register (STM_CLAIMSET)

Address offset: 0xFA0

Reset value: 0x0000 000F

This register is used in conjunction with STM_CLAIMCLR. It forms one half of the claim tag value. This location allows individual bits to be set (write), and returns the number of bits that can be set (read).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SET[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **SET[3:0]**: Claim tag bit implementation
0xF: Claim tag bits are present within the claim tag field.

STM claim tag clear register (STM_CLAIMCLR)

Address offset: 0xFA4

Reset value: 0x0000 0000

This register is used in conjunction with STM_CLAIMSET. It forms one half of the claim tag value. This location enables individual bits to be cleared (write), and returns the current claim tag value (read).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLR[3:0]			
												rw	rw	rw	rw

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **CLR[3:0]**: Claim tag current setting

STM lock access register (STM_LAR)

Address offset: 0xFB0

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
KEY[31:16]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
KEY[15:0]															
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

Bits 31:0 **KEY[31:0]**: Device register write-access enable

A write of 0xC5ACCE55 enables further write access to the device. An invalid write removes write access.

0xC5ACCE55: Unlocks the protection register to enable write access.

STM lock status register (STM_LSR)

Address offset: 0xFB4

Reset value: 0x0000 0003

This register indicates the status of the lock control mechanism that prevents accidental writes by the code under debug. The lock mechanism does not impact accesses to the extended stimulus port registers. This register is needed even in the absence of any lock access control mechanism. Where this register is present and locked, the lock mechanism blocks the write accesses to any register but the STM_LAR. The lock mechanism is only present for accesses with PADDRDBG31 signal low.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NTT	SLK	SLI
													r	r	r

Bits 31:3 Reserved, must be kept at reset value.

Bit 2 **NTT**: STM_LAR size indicator
This bit indicates if the STM implements the STM_LAR as 8 or 32 bits.
0: 32 bits

Bit 1 **SLK**: Lock current status
0: Enable write access to the STM. For read accesses to STM_LSR when the PADDRDBG31 signal is high, this bit is always 0.
1: Block write access to the STM. The STM ignores all write accesses to the registers when the PADDRDBG31 signal is low. Reads are permitted.

Bit 0 **SLI**: lock control mechanism existence
Indicates that a lock control mechanism exists for this device.
0: No lock control mechanism exists. The STM ignores writes to the STM_LAR. For read accesses to STM_LSR when the PADDRDBG31 signal is high, this bit is always 0.
1: Lock control mechanism is present. For read accesses to STM_LSR when the PADDRDBG31 signal is low, this bit is always 1.

STM authentication status register (STM_AUTHSTATUS)

Address offset: 0xFB8

Reset value: 0x0000 00AA

This register reports the required security level and current status of the authentication interface.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID[1:0]		SID[1:0]		NSNID[1:0]		NSID[1:0]	
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:6 **SNID[1:0]**: Security level for secure non-invasive debug
0x2: Disabled
0x3: Enabled

Bits 5:4 **SID[1:0]**: Security level for secure invasive debug
0x2: Disabled
0x3: Enabled

Bits 3:2 **NSNID[1:0]**: Security level for nonsecure non-invasive debug
0x2: Disabled
0x3: Enabled

Bits 1:0 **NSID[1:0]**: Security level for nonsecure invasive debug
0x2: Disabled
0x3: Enabled

STM device architecture register (STM_DEVARCH)

Address offset: 0xFBC

Reset value: 0x4771 0A63

This register indicates the architect and architecture of the STM.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ARCHITECT[10:0]											PRESENT	REVISION[3:0]			
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	ARCHID[14:0]														
	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:21 **ARCHITECT[10:0]**: component architect

This bitfield defines the architect of the component. Bits[31:28] indicate the DEP106 continuation code of the architect. Bits[27:21] indicate the JEP106 identification code of the architect. Refer to the Standard Manufacturers Identification Code for information about JEP106.

0x23B: the architect of the STM-500 is Arm.

Bit 20 **PRESENT**: STM_DEVARCH register presence

1: The STM_DEVARCH register is present.

Bits 19:16 **REVISION[3:0]**: Architecture revision

This bitfield returns the revision of the architecture that ARCHID specifies.

0x1: STMv1.1 architecture (for the STM-500)

Bit 15 Reserved, must be kept at reset value.

Bits 14:0 **ARCHID[14:0]**: Architecture ID

This bitfield returns a value that identifies the architecture of the component.

0x0A63: STMv1 architecture (for the STM-500)

STM device configuration register (STM_DEVID)

Address offset: 0xFC8

Reset value: 0x0001 0000

This register indicates the capabilities of the CoreSight STM.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NUMSP[16]
															r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NUMSP[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:17 Reserved, must be kept at reset value.

Bits 16:0 **NUMSP[16:0]**: Number of stimulus ports implemented

0x10000: 65 536 stimulus ports are implemented (however only 4096 are accessible).

STM device type identifier register (STM_DEVTYPE)

Address offset: 0xFCC

Reset value: 0x0000 0063

This register provides a debugger with component information when the part number is not recognized. The debugger can then report this information.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUB[3:0]				MAJOR[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SUB[3:0]**: Sub-classification within the major category

0x6: CoreSight STM. The component generates trace based on software and hardware stimulus.

Bits 3:0 **MAJOR[3:0]**: Major classification for debug or trace component

0x3: The component is a trace source.

STM peripheral ID4 register (STM_PIDR4)

Address offset: 0xFD0

Reset value: 0x0000 0004

This register belongs to the set of peripheral identification registers. It contains part of the designer identity and the memory footprint indicator.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				DES_2[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **SIZE[3:0]**: Component memory-size indicator

This bitfield indicates the total contiguous size of the memory window used by the component, in powers of two from the standard 4 Kbytes. If a component only requires the standard 4 Kbytes, this bitfield must be read as 0x0.

For 8 Kbytes, it is set to 0x1. For 16 Kbytes, it set to 0x2. For 32 Kbytes, it is set to 0x3, and similarly for larger memory windows.

0x0: The device only occupies 4 Kbytes of the memory.

Bits 3:0 **DES_2[3:0]**: JEDEC continuation code

This bitfield indicates the designer of the component, together with the identity code.

0x4: Indicate that Arm JEDEC identity code is on the 5th bank.

STM peripheral ID0 register (STM_PIDR0)

Address offset: 0xFE0

Reset value: 0x0000 0063

This register belongs to the set of peripheral identification registers. It contains part of the designer specific part number.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PART_0[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PART_0[7:0]**: bits [7:0] of the component part number
This bitfield is specified by the designer of the component.
0x63: CoreSight STM part number[7:0]. Eight least significant bits of the part number (0x963).

STM peripheral ID1 register (STM_PIDR1)

Address offset: 0xFE4

Reset value: 0x0000 00B9

This register belongs to the set of peripheral identification registers. It contains part of the designer specific part number and part of the designer identity.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DES_0[3:0]				PART_1[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **DES_0[3:0]**: bits [3:0] of the JEDEC identity code
This bitfield indicates the designer of the component, together with the continuation code.
0xB: least significant four bits of the JEP106 identity code

Bits 3:0 **PART_1[3:0]**: bits [11:8] of the component part number
Specified by the designer of the component.
0x9: CoreSight STM part number[11:8]. Most significant four bits of the part number (0x963).

STM peripheral ID2 register (STM_PIDR2)

Address offset: 0xFE8

Reset value: 0x0000 001B

This register belongs to the set of peripheral identification registers. It contains part of the designer identity and the product revision.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION[3:0]				JEDEC	DES_1[2:0]		
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVISION[3:0]**: Incremental component design version

An incremental value starting from 0x0 for the first design of this component. The value increases by one at both major and minor revisions ,and is used as a look-up to establish the exact major and minor revisions.

0x1: This device is at r0p1.

Bit 3 **JEDEC**: JEDEC assigned value usage

This bit indicates the use of a JEDEC assigned value. This bit is always set.

1: The designer ID is specified by JEDEC (refer to http://www.jedec.org).

Bits 2:0 **DES_1[2:0]**: bits [6:4] of the JEDEC identity code

This bitfield indicates the designer of the component, together with the continuation code.

0x3: Most significant three bits of the JEP106 identity code

STM peripheral ID3 register (STM_PIDR3)

Address offset: 0xFEC

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND[3:0]				CMOD[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **REVAND[3:0]**: Minor fix indicator

This bitfield indicates minor errata fixes specific to the design, for example metal fixes after implementation. In most cases, this field is 0x0. Arm recommends that the component designers ensure that this bitfield can be changed by a metal fix if required (for example by driving this bitfield from registers that reset to 0).

0x0: No metal fix in the component

Bits 3:0 **CMOD[3:0]**: Customer modification indicator

When the component is a reusable peripheral, this value indicates if the customer has modified the behavior of the component. In most cases this field is 0x0.

0x0: No modification made

STM component ID0 register (STM_CIDR0)

Address offset: 0xFF0

Reset value: 0x0000 000D

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRMBL_0[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PRMBL_0[7:0]**: bits [31:24] of component identification
This bitfield indicates that the identification registers are present.
0x0D: Identification value

STM component ID1 register (STM_CIDR1)

Address offset: 0xFF4

Reset value: 0x0000 0090

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PRMBL_1[3:0]			
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:4 **CLASS[3:0]**: Component class
This bitfield indicates the component class (for example, ROM table or CoreSight).
0x9: CoreSight component

Bits 3:0 **PRMBL_1[3:0]**: bits [19:16] of component identification
This bitfield indicates that the identification registers are present.
0x0: Identification value

STM component ID2 register (STM_CIDR2)

Address offset: 0xFF8

Reset value: 0x0000 0005

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRMBL_2[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PRMBL_2[7:0]**: bits [15:8] of component identification
This bitfield indicates that the identification registers are present.
0x05: Identification value

STM component ID3 register (STM_CIDR3)

Address offset: 0xFFC

Reset value: 0x0000 00B1

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRMBL_3[7:0]							
								r	r	r	r	r	r	r	r

Bits 31:8 Reserved, must be kept at reset value.

Bits 7:0 **PRMBL_3[7:0]**: bits [7:0] of component identification
This bitfield indicates that the identification registers are present.
0xB1: Identification value

STM register map

Table 901. STM register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xD00	STM_HEER	HEE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xD04-0xD1C	Reserved	Reserved																															
0xD20	STM_HETER	HETE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xD24-0xD5C	Reserved	Reserved																															
0xD60	STM_HEBSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HEBS
	Reset value																																0
0xD64	STM_HEMCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATBTRIGEN	TRIGCLEAR	TRIGSTATUS	TRIGCTL	Res.	ERRDETECT	COMPEN	EN
	Reset value																									0	0	0	0		0	0	0
0xD68	STM_HEEXTMUXR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	EXTMUX[7:0]							
	Reset value																									0	0	0	0	0	0	0	0

Table 901. STM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xD6C-0xDF0	Reserved	Reserved																															
0xDF4	STM_HEMASTR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASTER[15:0]															
	Reset value																	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0xDF8	STM_HEFEAT1R	Res.	HEEXTMUXSIZE [2:0]			Res.	Res.	Res.	Res.	NUMHE[8:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	HECOMP[1:0]		HEMASTR	HEERR	Res.	HETER
	Reset value		0	1	1					0	0	1	0	0	0	0	0	0										1	1	0	1		1
0xDFC	STM_HEIDR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	VENDSPEC [3:0]				CLASSREV [3:0]				CLASS[3:0]			
	Reset value																					0	0	0	0	0	0	0	1	0	0	0	1
0xE00	STM_SPER	SPE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xE04-0xE1C	Reserved	Reserved																															
0xE20	STM_SPTER	SPTE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xE24-0xE5C	Reserved	Reserved																															
0xE60	STM_SPSCR	Res.	Res.	Res.	Res.	PORTSEL[7:0]								Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PORTCTL [1:0]	
	Reset value					0	0	0	0	0	0	0	0																			0	0
0xE64	STM_SPMSCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASTSEL[7:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASTCTL
	Reset value										0	0	0	0	0	0	0	0															0
0xE68	STM_ SPOVERRIDER	Res.	Res.	Res.	Res.	PORTSEL[12:0]												Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	OVERTS	OVERCTL[1:0]	
	Reset value					0	0	0	0	0	0	0	0	0	0	0	0	0												0	0	0
0xE6C	STM_ SPMOVERRIDER	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASTSEL[7:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MASTCTL
	Reset value										0	0	0	0	0	0	0	0															0
0xE70	STM_SPTRIGCSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	ATBTRIGEN_DIR	ATBTRIGEN_TE	TRIGCLEAR	TRIGSTATUS	TRIGCTL
	Reset value																												0	0	0	0	0

Table 901. STM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xE74-0xE7C	Reserved	Reserved																															
0xE80	STM_TCSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	BUSY	TRACEID[6:0]							Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	COMPEN	Res.	Res.	SYNCEN	TSEN	EN
	Reset value									0	0	0	0	0	0	0	0											0			1	0	0
0xE84	STM_TSSTIMR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	FORCETS
	Reset value																																0
0xE88	Reserved	Reserved																															
0xE8C	STM_TSFREQR	FREQ[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xE90	STM_SYNCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	MODE	COUNT[8:0]								Res.	Res.	Res.
	Reset value																				0	0	0	0	0	0	0	0	0	0			
0xE94	STM_AUXCR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	QHWEVOVERRIDE	Res.	Res.	Res.	Res.	PRIORINVDIS	ASYNCPE	FIFOAF
	Reset value																									0					0	0	0
0xE98-0xE9C	Reserved	Reserved																															
0xEA0	STM_FEAT1R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SWOEN[1:0]		SYNCEN[1:0]		HWTEN[1:0]		TSPRESCALE [1:0]		TRIGCTL[1:0]		TRACEBUS [3:0]				SYNC[1:0]		FORCETS	TSFREQ	TS [1:0]		PROT[3:0]			
	Reset value									0	1	1	0	0	1	0	1	1	0	0	0	0	0	1	1	1	1	0	1	0	0	0	1
0xEA4	STM_FEAT2R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SPTYPE[1:0]		DSIZE[3:0]				Res.	SPTRTYPE [1:0]		PRIVMASK [1:0]		SPOVERRIDE	SPCOMP[1:0]		Res.	SPER	SPTER[1:0]	
	Reset value															0	1	0	0	0	1		1	0	0	1	1	1	1		0	1	0
0xEA8	STM_FEAT3R	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NUMMAST[6:0]						
	Reset value																										1	1	1	1	1	1	1
0xEAC-0xF9C	Reserved	Reserved																															
0xFA0	STM_CLAIMSET	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SET[3:0]			
	Reset value																													1	1	1	1
0xFA4	STM_CLAIMCLR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLR[3:0]			
	Reset value																													0	0	0	0

Table 901. STM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFA8-0xFAC	Reserved	Reserved																															
0xFB0	STM_LAR	KEY[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xFB4	STM_LSR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NTT	SLK	SLI
	Reset value																														0	1	1
0xFB8	STM_ AUTHSTATUS	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SNID [1:0]		SID [1:0]		NSNID [1:0]		NSID [1:0]	
	Reset value																									1	0	1	0	1	0	1	0
0xFBC	STM_DEVARCH	ARCHITECT[10:0]											PRESENT	REVISION [3:0]				Res.	ARCHID[14:0]															
	Reset value	0	1	0	0	0	1	1	1	0	1	1	1	0	0	0	1		0	0	0	1	0	1	0	0	1	1	0	0	0	1	1
0xFC0-0xFC4	Reserved	Reserved																															
0xFC8	STM_DEVID	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	NUMSP[16:0]															
	Reset value																1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0xFCC	STM_DEVTYPE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SUB[3:0]				MAJOR[3:0]			
	Reset value																									0	1	1	0	0	0	1	1
0xFD0	STM_PIDR4	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	SIZE[3:0]				DES_2[3:0]			
	Reset value																									0	0	0	0	0	1	0	0
0xFD4-0xFDC	Reserved	Reserved																															
0xFE0	STM_PIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PART_0[7:0]							
	Reset value																									0	1	1	0	0	0	1	1
0xFE4	STM_PIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DES_0[3:0]				PART_1 [3:0]			
	Reset value																									1	0	1	1	1	0	0	1
0xFE8	STM_PIDR2	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVISION [3:0]				JEDEC	DES_1 [2:0]		
	Reset value																									0	0	0	1	1	0	1	1
0xFEC	STM_PIDR3	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	REVAND [3:0]				CMOD[3:0]			
	Reset value																									0	0	0	0	0	0	0	0
0xFF0	STM_CIDR0	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PRMBL_0[7:0]							
	Reset value																									0	0	0	0	1	1	0	1
0xFF4	STM_CIDR1	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	CLASS[3:0]				PRMBL_1 [3:0]			
	Reset value																									1	0	0	1	0	0	0	0

Table 901. STM register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0xFF8	STM_CIDR2	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	PRMBL_2[7:0]							
	Reset value																									0	0	0	0	0	1	0	1
0xFFC	STM_CIDR3	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	Res	PRMBL_3[7:0]							
	Reset value																									1	0	1	1	0	0	0	1

78.12.20 Microcontroller debug unit (DBGMCU)

The DBGMCU is a component containing a number of registers which control the power and clock behavior in debug mode. Specifically it allows the debugger, or debug software:

- to maintain the clock and power to the processor and/or peripherals when in low-power modes (Sleep, Stop, or Standby)
- to maintain the clock and power to the system debug and trace components when in low-power modes
- to stop the clock to certain peripherals (CAN, SMBUS timeout, watchdogs, timers, RTC, DMAs) when the processor is halted in debug mode

Legacy DBGMCU features (dbg/trace_en, dbg_sleep/stop/stby, dbg_freeze) are disabled when debug is closed. Only registers required for authentication procedure are accessible.

Low-power mode emulation in debug

When entering a low-power state (Sleep, Stop, Standby), the clock to the processor, the system clock or, in the case of Standby, the core power supply can be switched off. This prevents any debug accesses to the unclocked domains.

To simplify debugging of applications which use these power saving features, the clock and power can be maintained or not, according to the setting of DBG_STANDBY, DBG_STOP, and DBG_SLEEP bits in DBGMCU_CR.

There are no restrictions from a security perspective to access these control bits, as the overall system behavior is not modified, except that the power and clocks remain on.

If the system enters one of the power saving modes, and the corresponding register bit is set, the system state in the EXTI is not affected: the system enters and exits the low-power state as normal. However the clock and power remain active, allowing the debugger full access.

Note: If the TPIU is used, loss of trace data can occur if the system goes into Standby mode, and switches off the core power before the TPIU FIFO is empty. The software can check the FIFO state to avoid this. If the DBG_STANDBY bit is set while using the trace port, the trace continues to flush during Standby mode, and no specific software action is required.

Peripheral clock “freeze” in debug

When the processor is halted in debug state, by default the peripherals continue to operate. The system state, at the moment the breakpoint was taken, is overwritten and cannot be recovered. Furthermore, certain peripherals require software intervention on a regular basis, or they can time out, and generate a reset (watchdogs, for example).

By setting the corresponding bits in the DBGMCU peripheral freeze registers, certain peripherals clocks can be stopped automatically as soon as processor enters debug state. The clocks remain stopped until the processor is restarted.

The following peripherals support this feature:

- all DMA engines (HPDMA3 and GPDMA1)
- all timers (TIMx, LPTIMx)
- all I2C/I3C peripherals (to prevent SMBus timeout)
- FDCAN
- all watchdog timers (WWDGx, IWDGx)
- real time clock (RTC)

The peripheral accesses can be controlled from a RISUP for peripherals (see *Section 6: Resource isolation framework security controller (RIFSC)* for more details).

DBGMCU freeze register bits inherit the security level of the peripheral from the RISUP, or from the peripheral itself if it is TrustZone-aware (such as DMA) as follows:

- When spiden = 1 (secure privilege debug enabled), all debug freeze bits can be modified.
- When spiden = 0 (secure privilege debug disabled), only the bits corresponding to non-secure peripherals can be modified.

78.12.21 DBGMCU registers

The register file base address for the DBGMCU is defined in *Table 870*.

The DBGMCU registers are reset only by a power-on-reset.

DBGMCU identity code register (DBGMCU_IDCODE)

Address offset: 0x000

Reset value: 0xXXXX 6XXX

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REV_ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DEV_ID[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **REV_ID[15:0]**: Revision
0x1000: revision A
0x2000: revision B

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DEV_ID[11:0]**: Device ID
0x486: STM32N6x5/x7xx

DBGMCU configuration register (DBGMCU_CR)

Address offset: 0x004

Reset value: 0x8000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HLT_TSGEN_EN	Res.	Res.	DBTRGOEN	Res.	Res.	Res.	Res.	Res.	Res.	TRACECLKEN	DBGCLKEN	Res.	Res.	Res.	Res.
rw			rw							rw	rw				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_STANDBY	DBG_STOP	DBG_SLEEP
													rw	rw	rw

Bit 31 **HLT_TSGEN_EN**: TSGEN halt enable
1: TSGEN stops counting when processor is in halt.
0: TSGEN keeps on counting when processor is in halt.

Bits 30:29 Reserved, must be kept at reset value.

Bit 28 **DBTRGOEN**: DBTRGIO connection control
0: DBTRGIO connected to DBTRGIN
1: DBTRGIO connected to DBTRGOUT

Bits 27:22 Reserved, must be kept at reset value.

Bit 21 **TRACECLKEN**: TPIU export clock enable through software
0: TPIU clock is off.
1: TPIU clock is on.

Bit 20 **DBGCLKEN**: Debug clock enable through software
0: Debug clock is off.
1: Debug clock is on.

Bits 19:3 Reserved, must be kept at reset value.

Bit 2 **DBG_STANDBY**: Debug in Standby mode
0: Normal operation. All clocks are disabled. The V_{DD} domain is automatically powered down in Standby mode.
1: Automatic clock stop/power down disabled. All active clocks and oscillators continue to run during Standby mode. The V_{DD} domain supply is maintained, allowing full debug capability.
Note: On exit from Standby mode, a system reset is performed.

Bit 1 **DBG_STOP**: Debug in Stop mode
0: Normal operation. All clocks are disabled automatically in Stop mode.
1: Automatic clock stop disabled. All active clocks and oscillators continue to run during Stop mode, allowing full debug capability.
Note: On exit from Stop mode, the clock settings is set to the Stop mode exit state.

Bit 0 **DBG_SLEEP**: Debug in Sleep mode
0: Normal operation. Peripheral clocks are automatically stopped in Sleep mode.
1: Automatic clock stop disabled. Peripheral clocks continue to run, allowing full debug capability.

DBGMCU APB1L peripheral freeze register (DBGMCU_APB1LFZR)

Address offset: 0x010

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	DBG_I3C2_STOP	DBG_I3C1_STOP	DBG_I2C3_STOP	DBG_I2C2_STOP	DBG_I2C1_STOP	Res.	Res.	Res.	Res.	Res.
						rw	rw	rw	rw	rw					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	DBG_TIM11_STOP	DBG_TIM10_STOP	DBG_WWDG1_STOP	Res.	DBG_LPTIM1_STOP	DBG_TIM14_STOP	DBG_TIM13_STOP	DBG_TIM12_STOP	DBG_TIM7_STOP	DBG_TIM6_STOP	DBG_TIM5_STOP	DBG_TIM4_STOP	DBG_TIM3_STOP	DBG_TIM2_STOP
		rw	rw	rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:26 Reserved, must be kept at reset value.

Bit 25 **DBG_I3C2_STOP**: I3C2 SMBUS timeout stop in debug
0: Normal operation. I3C2 SMBUS timeout continues to operate while Cortex-M55 is in debug mode.
1: Stop in debug. I3C2 SMBUS timeout is frozen while Cortex-M55 is in debug mode.

Bit 24 **DBG_I3C1_STOP**: I3C1 SMBUS timeout stop in debug
0: Normal operation. I3C1 SMBUS timeout continues to operate while Cortex-M55 is in debug mode.
1: Stop in debug. I3C1 SMBUS timeout is frozen while Cortex-M55 is in debug mode.

Bit 23 **DBG_I2C3_STOP**: I2C3 SMBUS timeout stop in debug
0: Normal operation. I2C3 SMBUS timeout continues to operate while Cortex-M55 is in debug mode.
1: Stop in debug. I2C3 SMBUS timeout is frozen while Cortex-M55 is in debug mode.

Bit 22 **DBG_I2C2_STOP**: I2C2 SMBUS timeout stop in debug
0: Normal operation. I2C2 SMBUS timeout continues to operate while Cortex-M55 is in debug mode.
1: Stop in debug. I2C2 SMBUS timeout is frozen while Cortex-M55 is in debug mode.

Bit 21 **DBG_I2C1_STOP**: I2C1 SMBUS timeout stop in debug
0: Normal operation. I2C1 SMBUS timeout continues to operate while Cortex-M55 is in debug mode.
1: Stop in debug. I2C1 SMBUS timeout is frozen while Cortex-M55 is in debug mode.

Bits 20:14 Reserved, must be kept at reset value.

Bit 13 **DBG_TIM11_STOP**: TIM11 stop in debug
0: Normal operation. TIM11 continues to operate while Cortex-M55 is in debug mode.
1: Stop in debug. TIM11 is frozen while Cortex-M55 is in debug mode.

Bit 12 **DBG_TIM10_STOP**: TIM10 stop in debug
0: Normal operation. TIM10 continues to operate while Cortex-M55 is in debug mode.
1: Stop in debug. TIM10 is frozen while Cortex-M55 is in debug mode.

Bit 11 **DBG_WWDG1_STOP**: WWDG1 stop in debug
0: Normal operation. WWDG1 continues to operate while Cortex-M55 is in debug mode.
1: Stop in debug. WWDG1 is frozen while Cortex-M55 is in debug mode.

Bit 10 Reserved, must be kept at reset value.

Bit 9 **DBG_LPTIM1_STOP**: LPTIM1 stop in debug
0: Normal operation. LPTIM1 continues to operate while Cortex-M55 is in debug mode.
1: Stop in debug. LPTIM1 is frozen while Cortex-M55 is in debug mode.

Bit 8 **DBG_TIM14_STOP**: TIM14 stop in debug
0: Normal operation. TIM14 continues to operate while Cortex-M55 is in debug mode.
1: Stop in debug. TIM14 is frozen while Cortex-M55 is in debug mode.

Bit 7 **DBG_TIM13_STOP**: TIM13 stop in debug
0: Normal operation. TIM13 continues to operate while Cortex-M55 is in debug mode.
1: Stop in debug. TIM13 is frozen while Cortex-M55 is in debug mode.

Bit 6 **DBG_TIM12_STOP**: TIM12 stop in debug
0: Normal operation. TIM12 continues to operate while Cortex-M55 is in debug mode.
1: Stop in debug. TIM12 is frozen while Cortex-M55 is in debug mode.

Bit 5 **DBG_TIM7_STOP**: TIM7 stop in debug
0: Normal operation. TIM7 continues to operate while Cortex-M55 is in debug mode.
1: Stop in debug. TIM7 is frozen while Cortex-M55 is in debug mode.

Bit 4 **DBG_TIM6_STOP**: TIM6 stop in debug
0: Normal operation. TIM6 continues to operate while Cortex-M55 is in debug mode.
1: Stop in debug. TIM6 is frozen while Cortex-M55 is in debug mode.

Bit 3 **DBG_TIM5_STOP**: TIM5 stop in debug
0: Normal operation. TIM5 continues to operate while Cortex-M55 is in debug mod.e
1: Stop in debug. TIM5 is frozen while Cortex-M55 is in debug mode.

Bit 2 **DBG_TIM4_STOP**: TIM4 stop in debug
0: Normal operation. TIM4 continues to operate while Cortex-M55 is in debug mode.
1: Stop in debug. TIM4 is frozen while Cortex-M55 is in debug mode.

Bit 1 **DBG_TIM3_STOP**: TIM3 stop in debug
0: Normal operation. TIM3 continues to operate while Cortex-M55 is in debug mode.
1: Stop in debug. TIM3 is frozen while Cortex-M55 is in debug mode.

Bit 0 **DBG_TIM2_STOP**: TIM2 stop in debug
0: Normal operation. TIM2 continues to operate while Cortex-M55 is in debug mode.
1: Stop in debug. TIM2 is frozen while Cortex-M55 is in debug mode.

DBGMCU APB1H peripheral freeze register (DBGMCU_APB1HFZR)

Address offset: 0x014

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_ FDCAN _STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
							rw								

Bits 31:9 Reserved, must be kept at reset value.

Bit 8 **DBG_FDCAN_STOP**: FDCAN stop in debug
0: Normal operation. FDCAN continues to operate while Cortex-M55 is in debug mode.
1: Stop in debug. FDCAN is frozen while Cortex-M55 is in debug mode.

Bits 7:0 Reserved, must be kept at reset value.

DBGMCU APB2 peripheral freeze register (DBGMCU_APB2FZR)

Address offset: 0x018

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_ TIM9_ STOP	DBG_ TIM17_ STOP	DBG_ TIM16_ STOP	DBG_ TIM15_ STOP
												rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DBG_ TIM18_ STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_ TIM8_ STOP	DBG_ TIM1_ STOP
rw														rw	rw

Bits 31:20 Reserved, must be kept at reset value.

Bit 19 **DBG_TIM9_STOP**: TIM9 stop in debug
0: Normal operation. TIM9 continues to operate while Cortex-M55 is in debug mode.
1: Stop in debug. TIM9 is frozen while Cortex-M55 is in debug mode.

Bit 18 **DBG_TIM17_STOP**: TIM17 stop in debug
0: Normal operation. TIM17 continues to operate while Cortex-M55 is in debug mode.
1: Stop in debug. TIM17 is frozen while Cortex-M55 is in debug mode.

Bit 17 **DBG_TIM16_STOP**: TIM16 stop in debug
0: Normal operation. TIM16 continues to operate while Cortex-M55 is in debug mode.
1: Stop in debug. TIM16 is frozen while Cortex-M55 is in debug mode.

Bit 16 **DBG_TIM15_STOP**: TIM15 stop in debug
0: Normal operation. TIM15 continues to operate while Cortex-M55 is in debug mode.
1: Stop in debug. TIM15 is frozen while Cortex-M55 is in debug mode.

Bit 15 **DBG_TIM18_STOP**: TIM18 stop in debug
0: Normal operation. TIM18 continues to operate while Cortex-M55 is in debug mode.
1: Stop in debug. TIM18 is frozen while Cortex-M55 is in debug mode.

Bits 14:2 Reserved, must be kept at reset value.

Bit 1 **DBG_TIM8_STOP**: TIM8 stop in debug
0: Normal operation. TIM8 continues to operate while Cortex-M55 is in debug mode.
1: Stop in debug. TIM8 is frozen while Cortex-M55 is in debug mode.

Bit 0 **DBG_TIM1_STOP**: TIM1 stop in debug
0: Normal operation. TIM1 continues to operate while Cortex-M55 is in debug mode.
1: Stop in debug. TIM1 is frozen while Cortex-M55 is in debug mode.

DBGMCU APB4 peripheral freeze register (DBGMCU_APB4FZR)

Address offset: 0x01C

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_IWDG_STOP	Res.	DBG_RTC_STOP
													rw		rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	DBG_LPTIM5_STOP	DBG_LPTIM4_STOP	DBG_LPTIM3_STOP	DBG_LPTIM2_STOP	DBG_I2C4_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
			rw	rw	rw	rw	rw								

Bits 31:19 Reserved, must be kept at reset value.

Bit 18 **DBG_IWDG_STOP**: WWDG stop in debug
0: Normal operation. IWDG continues to operate while Cortex-M55 is in debug mode.
1: Stop in debug. IWDG is frozen while Cortex-M55 is in debug mode.

Bit 17 Reserved, must be kept at reset value.

Bit 16 **DBG_RTC_STOP**: RTC clock is suspended in debug
0: Normal operation. RTC clock continues to operate while Cortex-M55 is in debug mode.
1: Stop in debug. RTC clock is suspended while Cortex-M55 is in debug mode.

Bits 15:13 Reserved, must be kept at reset value.

Bit 12 **DBG_LPTIM5_STOP**: LPTIM5 stop in debug
0: Normal operation. LPTIM5 continues to operate while Cortex-M55 is in debug mode.
1: Stop in debug. LPTIM5 is frozen while Cortex-M55 is in debug mode.

Bit 11 **DBG_LPTIM4_STOP**: LPTIM4 stop in debug
0: Normal operation. LPTIM4 continues to operate while Cortex-M55 is in debug mode.
1: Stop in debug. LPTIM4 is frozen while Cortex-M55 is in debug mode.

Bit 10 **DBG_LPTIM3_STOP**: LPTIM3 stop in debug
0: Normal operation. LPTIM3 continues to operate while Cortex-M55 is in debug mode.
1: Stop in debug. LPTIM3 is frozen while Cortex-M55 is in debug mode.

Bit 9 **DBG_LPTIM2_STOP**: LPTIM2 stop in debug
0: Normal operation. LPTIM2 continues to operate while Cortex-M55 is in debug mode.
1: Stop in debug. LPTIM2 is frozen while Cortex-M55 is in debug mode.

Bit 8 **DBG_I2C4_STOP**: I2C4 stop in debug
0: Normal operation. I2C4 continues to operate while Cortex-M55 is in debug mode.
1: Stop in debug. I2C4 is frozen while Cortex-M55 is in debug mode.

Bits 7:0 Reserved, must be kept at reset value.

DBGMCU APB5 peripheral freeze register (DBGMCU_APB5FZR)

Address offset: 0x020

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_GFXTIM_STOP	Res.	Res.	Res.	Res.
											rw				

Bits 31:5 Reserved, must be kept at reset value.

Bit 4 **DBG_GFXTIM_STOP**: GFXTIM stop in debug
0: Normal operation. GFXTIM continues counting while Cortex-M55 is in debug mode.
1: Stop in debug. GFXTIM is frozen while Cortex-M55 is in debug mode.

Bits 3:0 Reserved, must be kept at reset value.

DBGMCU AHB1 peripheral freeze register (DBGMCU_AHB1FZR)

Address offset: 0x024

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DBG_GPDMA1_CH15_STOP	DBG_GPDMA1_CH14_STOP	DBG_GPDMA1_CH13_STOP	DBG_GPDMA1_CH12_STOP	DBG_GPDMA1_CH11_STOP	DBG_GPDMA1_CH10_STOP	DBG_GPDMA1_CH9_STOP	DBG_GPDMA1_CH8_STOP	DBG_GPDMA1_CH7_STOP	DBG_GPDMA1_CH6_STOP	DBG_GPDMA1_CH5_STOP	DBG_GPDMA1_CH4_STOP	DBG_GPDMA1_CH3_STOP	DBG_GPDMA1_CH2_STOP	DBG_GPDMA1_CH1_STOP	DBG_GPDMA1_CH0_STOP
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **DBG_GPDMA1_CHn_STOP**: GPDMA1_CHn suspend in debug (n = 15 to 0)
0: Normal operation. GPDMA1_CHn continues to operate while Cortex-M55 is in debug mode.
1: Stop in debug. GPDMA1_CHn is suspended while Cortex-M55 is in debug mode.

DBGMCU AHB5 peripheral freeze register (DBGMCU_AHB5FZR)

Address offset: 0x028

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_NPU_STOP
															rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DBG_HPDMA1_CH15_STOP	DBG_HPDMA1_CH14_STOP	DBG_HPDMA1_CH13_STOP	DBG_HPDMA1_CH12_STOP	DBG_HPDMA1_CH11_STOP	DBG_HPDMA1_CH10_STOP	DBG_HPDMA1_CH9_STOP	DBG_HPDMA1_CH8_STOP	DBG_HPDMA1_CH7_STOP	DBG_HPDMA1_CH6_STOP	DBG_HPDMA1_CH5_STOP	DBG_HPDMA1_CH4_STOP	DBG_HPDMA1_CH3_STOP	DBG_HPDMA1_CH2_STOP	DBG_HPDMA1_CH1_STOP	DBG_HPDMA1_CH0_STOP
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:17 Reserved, must be kept at reset value.

Bit 16 **DBG_NPU_STOP**: NPU stop in debug mode
0: Normal operation. The NPU continues to operate while Cortex-M55 is in debug mode.
1: Stop in debug. NPU is suspended while Cortex-M55 is in debug mode.

Bits 15:0 **DBG_HPDMA1_CHn_STOP**: HPDMA3_CHn suspend in debug (n = 15 to 0)
0: Normal operation. HPDMA3_CHn continues to operate while Cortex-M55 is in debug mode.
1: Stop in debug. HPDMA3_CHn is suspended while Cortex-M55 is in debug mode.

DBGMCU status register (DBGMCU_SR)

Address offset: 0x0FC

Reset value: 0x0001 0003

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AP1_ENABLE	AP0_ENABLE
														r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AP1_PRESENT	AP0_PRESENT
														r	r

Bits 31:18 Reserved, must be kept at reset value.

Bit 17 **AP1_ENABLE**: Access point 1 enable
0: AP disabled (debug access locked)
1: AP enabled (debug access open)

Bit 16 **AP0_ENABLE**: Access point 0 enable
1: Always enable

Bits 15:2 Reserved, must be kept at reset value.

Bit 1 **AP1_PRESENT**: Access point 1 presence
1: AP present

Bit 0 **AP0_PRESENT**: Access point 0 presence
1: AP present

DBGMCU host authentication register (DBGMCU_DBG_AUTH_HOST)

Address offset: 0x100

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MESSAGE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MESSAGE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:0 **MESSAGE[31:0]**: Mailbox between debugger and processor
Read/write from debugger; read only from processor

DBGMCU device authentication register (DBGMCU_DBG_AUTH_DEV)

Address offset: 0x104

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MESSAGE[31:16]															
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MESSAGE[15:0]															
rw	rw	rw	rw	rw	rw	rw	rs	rw	rw	rw	rw	rw	rw	rw	rs

Bits 31:0 **MESSAGE[31:0]**: Mailbox between debugger and processor
Read/write from processor, read only from debugger

DBGMCU message read acknowledge authentication register (DBGMCU_DBG_AUTH_ACK)

Address offset: 0x108

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEVICE_ACK	HOST_ACK
														r	r

Bits 31:2 Reserved, must be kept at reset value.

Bit 1 **DEVICE_ACK**: Access status to DBG_AUTH_DEV register
1: Processor has written DBG_AUTH_DEV
0: Debugger has read DBG_AUTH_DEV

Bit 0 **HOST_ACK**: Access status to DBG_AUTH_HOST register
1: Debugger has written DBG_AUTH_HOST
0: Processor has read DBG_AUTH_HOST

DBGMCU register map

Table 902. DBGMCU register map and reset values

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x000	DBGMCU_IDCODE	REV_ID[15:0]																Res.	Res.	Res.	Res.	DEV_ID[11:0]											
	Reset value	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X					X	X	X	X	X	X	X	X	X	X	X	X

Table 902. DBGMCU register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x004	DBGMCU_CR	HLT_TSGEN_EN	Res.	Res.	DBTRGOEN	Res.	Res.	Res.	Res.	Res.	Res.	TRACECLKEN	DBGCLKEN	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_STANDBY	DBG_STOP	DBG_SLEEP
	Reset value	1			0							0	0																		0	0	0
0x008-0x00C	Reserved	Reserved																															
0x010	DBGMCU_APB1LFZR	Res.	Res.	Res.	Res.	Res.	Res.	DBG_I3C2_STOP	DBG_I3C1_STOP	DBG_I2C3_STOP	DBG_I2C2_STOP	DBG_I2C1_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_TIM11_STOP	DBG_TIM10_STOP	DBG_WWDG1_STOP	Res.	DBG_LPTIM1_STOP	DBG_TIM14_STOP	DBG_TIM13_STOP	DBG_TIM12_STOP	DBG_TIM7_STOP	DBG_TIM6_STOP	DBG_TIM5_STOP	DBG_TIM4_STOP	DBG_TIM3_STOP	DBG_TIM2_STOP
	Reset value							0	0	0	0	0								0	0	0		0	0	0	0	0	0	0	0	0	0
0x014	DBGMCU_APB1HFZR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_FDCAN_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value																								0								
0x018	DBGMCU_APB2FZR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_TIM9_STOP	DBG_TIM17_STOP	DBG_TIM16_STOP	DBG_TIM15_STOP	DBG_TIM18_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_TIM8_STOP	DBG_TIM1_STOP
	Reset value													0	0	0	0	0														0	0
0x01C	DBGMCU_APB4FZR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_IWDG_STOP	Res.	DBG_RTC_STOP	Res.	Res.	Res.	DBG_LPTIM5_STOP	DBG_LPTIM4_STOP	DBG_LPTIM3_STOP	DBG_LPTIM2_STOP	DBG_I3C4_STOP	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
	Reset value														0		0				0	0	0	0	0								
0x020	DBGMCU_APB5FZR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_GFXTIM_STOP	Res.	Res.	Res.	Res.
	Reset value																												0				

Table 902. DBGMCU register map and reset values (continued)

Offset	Register name	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x024	DBGMCU_ AHB1FZR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_GPDMA1_CH15_STO	DBG_GPDMA1_CH14_STO	DBG_GPDMA1_CH13_STO	DBG_GPDMA1_CH12_STO	DBG_GPDMA1_CH11_STOP	DBG_GPDMA1_CH10_STO	DBG_GPDMA1_CH9_STOP	DBG_GPDMA1_CH8_STOP	DBG_GPDMA1_CH7_STOP	DBG_GPDMA1_CH6_STOP	DBG_GPDMA1_CH5_STOP	DBG_GPDMA1_CH4_STOP	DBG_GPDMA1_CH3_STOP	DBG_GPDMA1_CH2_STOP	DBG_GPDMA1_CH1_STOP	DBG_GPDMA1_CH0_STOP
	Reset value																	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x028	DBGMCU_ AHB5FZR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DBG_NPU_STOP	DBG_HPDMA1_CH15_STOP	DBG_HPDMA1_CH14_STOP	DBG_HPDMA1_CH13_STOP	DBG_HPDMA1_CH12_STOP	DBG_HPDMA1_CH11_STOP	DBG_HPDMA1_CH10_STOP	DBG_HPDMA1_CH9_STOP	DBG_HPDMA1_CH8_STOP	DBG_HPDMA1_CH7_STOP	DBG_HPDMA1_CH6_STOP	DBG_HPDMA1_CH5_STOP	DBG_HPDMA1_CH4_STOP	DBG_HPDMA1_CH3_STOP	DBG_HPDMA1_CH2_STOP	DBG_HPDMA1_CH1_STOP	DBG_HPDMA1_CH0_STOP
	Reset value																0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x02C-0x0F8	Reserved	Reserved																															
0x0FC	DBGMCU_SR	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AP1_ENABLE	AP0_ENABLE	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	AP1_PRESENT	AP0_PRESENT
	Reset value															0	1															1	1
0x100	DBGMCU_DBG _AUTH_HOST	MESSAGE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x104	DBGMCU_DBG _AUTH_DEV	MESSAGE[31:0]																															
	Reset value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x108	DBGMCU_DBG _AUTH_ACK	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	DEVICE_ACK	HOST_ACK
	Reset value																															0	0

78.13 References

1. IHI 0031C (ID080813) - Arm Debug Interface Architecture Specification ADIv5.0 to ADIv5.2, Issue C, 8th Aug 2013.
2. DDI 0480F (ID100313) - Arm CoreSight SoC-400 r3p1 Technical Reference Manual, Issue F, 26th Sept 2013.
3. DDI 0461B (ID010111) - Arm CoreSight Trace Memory Controller r0p1 Technical Reference Manual, Issue B, 10 Dec 2010
4. Arm Cortex-M55 Processor r0p1 Technical Reference Manual, Issue 0001-05, 31 March 2020
5. Arm CoreSight ETM-M55 r0p2 Technical Reference Manual, Issue 0002-01, 17 July 2020
6. DDI 0528B (ID062514) - Arm CoreSight STM-500 System Trace Macrocell r0p1 Technical Reference Manual, Issue B, 11 March 2014
7. IHI 0022E (ID022613) - AMBA AXI and ACE Protocol Specification, Issue E, 22 February 2013
8. DDI 0314H - Arm CoreSight Components Technical Reference Manual, Issue H, 10 July 2009

79 Device electronic signature

The device electronic signature is set in hardware or stored in the OTP (visible in BSEC registers), and can be read using the debug interface or by the CPU. It contains factory-programmed identification data that allow the user firmware or other external devices to automatically match the characteristics of the component.

79.1 Unique device ID register (96 bits) (UID)

The unique device identifier is ideally suited to be used:

- as serial number (USB string serial number, or other end applications)
- as part of the security keys, to increase the security of code while using and combining this unique ID with software cryptographic primitives and protocols before programming the memory
- during processes such as secure boot

The 96-bit unique device identifier provides a reference number, unique for a given device and in any context. These bits cannot be altered by the user.

Base address: 0x4600 9000 (BSEC base address on APB4)

Address offset: 0x014 (BSEC_OTP_DATA5)

Read only = 0xXXXX XXXX where X is factory-programmed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UID[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **UID[31:0]:** 31:0 unique ID bits

Address offset: 0x018 (BSEC_OTP_DATA6)

Read only = 0xXXXX XXXX where X is factory-programmed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UID[63:48]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UID[47:32]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **UID[63:32]:** 63:32 unique ID bits

Address offset: 0x01C (BSEC_OTP_DATA7)

Read only = 0xXXXX XXXX, where X is factory-programmed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
UID[95:80]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
UID[79:64]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **UID[95:64]:** 95:64 unique ID bits

79.2 Device part number (RPN)

Base address: 0x4600 9000 (BSEC base address on APB4)

Address offset: 0x024 (BSEC_OTP_DATA9)

Read only = 0xXXXX XXXX where X is factory-programmed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RPN_Coding[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RPN_Coding[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **RPN_Coding[31:0]:**
0x0000 2000: STM32N657
0x0000 2200: STM32N647
0x0000 6000: STM32N655
0x0000 6200: STM32N645
Others: Reserved

Base address: 0x4600 9000 (BSEC base address on APB3)

Address offset: 0x020 (BSEC_OTP_DATA8)

Read only = 0xXXXX XXXX where X is factory-programmed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Options[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Options[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **Options[31:0]:**
0x00000000: No options

79.3 Package data register (PKG)

Base address: 0x4600 9000 (BSEC base address on APB4)

Address offset: 0x1E8 (BSEC_OTP_DATA122)

Read only = 0xXXXX XXXX where X is factory-programmed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	Res.	PKG[3:0]			
												r	r	r	r

Bits 31:4 Reserved, must be kept at reset value.

Bits 3:0 **PKG[3:0]**: Package type
0010: BGA142
0100: BGA169
0110: BGA178
1000: BGA198
1010: BGA223
1100: BGA264
Others: Reserved

79.4 Device version

Base address: 0x4400 1000 (DBGMCU is defined in *Table 870: System ROM table*)

Address offset: 0x000 (DBGMCU_IDCODE)

Read only = 0xXXXX 6XXX where X is factory-programmed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REV_ID[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Res.	Res.	Res.	Res.	DEV_ID[11:0]											
				r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:16 **REV_ID[15:0]**: Silicon revision
0x1000: Revision A

Bits 15:12 Reserved, must be kept at reset value.

Bits 11:0 **DEV_ID[11:0]**: Device identifier
0x486: STM32N6xx

79.5 Boot ROM version

Base address: 0x1800 1000

Address offset: 0x0000

Read only = 0xXXXX XXXX where X is factory-programmed (data are in little endian format).

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ChipVersion[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ChipVersion[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ChipVersion[31:0]**:
0x0000 8604: STM32N6

Base address: 0x1800 1000

Address offset: 0x0004

Read only = 0xXXXX XXXX where X is factory-programmed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CutVersion[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CutVersion[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **CutVersion[31:0]**:
0x0000 0200: Cut 2.0

Base address: 0x1800 1000

Address offset: 0x0008

Read only = 0xXXXX XXXX where X is factory-programmed.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ROM mask version[31:16]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ROM mask version[15:0]															
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

Bits 31:0 **ROM mask version[31:0]**:
0x0000 0001: version 1

80 Important security notice

The STMicroelectronics group of companies (ST) places a high value on product security, which is why the ST product(s) identified in this documentation may be certified by various security certification bodies and/or may implement our own security measures as set forth herein. However, no level of security certification and/or built-in security measures can guarantee that ST products are resistant to all forms of attacks. As such, it is the responsibility of each of ST's customers to determine if the level of security provided in an ST product meets the customer needs both in relation to the ST product alone, as well as when combined with other components and/or software for the customer end product or application. In particular, take note that:

- ST products may have been certified by one or more security certification bodies, such as Platform Security Architecture (www.psacertified.org) and/or Security Evaluation standard for IoT Platforms (www.trustcb.com). For details concerning whether the ST product(s) referenced herein have received security certification along with the level and current status of such certification, either visit the relevant certification standards website or go to the relevant product page on *www.st.com* for the most up to date information. As the status and/or level of security certification for an ST product can change from time to time, customers must re-check security certification status/level as needed. If an ST product is not shown to be certified under a particular security standard, customers must not assume it is certified.
- Certification bodies have the right to evaluate, grant and revoke security certification in relation to ST products. These certification bodies are therefore independently responsible for granting or revoking security certification for an ST product, and ST does not take any responsibility for mistakes, evaluations, assessments, testing, or other activity carried out by the certification body with respect to any ST product.
- Industry-based cryptographic algorithms (such as AES, DES, or MD5) and other open standard technologies which may be used in conjunction with an ST product are based on standards which were not developed by ST. ST does not take responsibility for any flaws in such cryptographic algorithms or open technologies or for any methods which have been or may be developed to bypass, decrypt or crack such algorithms or technologies.
- While robust security testing may be done, no level of certification can absolutely guarantee protections against all attacks, including, for example, against advanced attacks which have not been tested for, against new or unidentified forms of attack, or against any form of attack when using an ST product outside of its specification or intended use, or in conjunction with other components or software which are used by customer to create their end product or application. ST is not responsible for resistance against such attacks. As such, regardless of the incorporated security features and/or any information or support that may be provided by ST, each customer is solely responsible for determining if the level of attacks tested for meets their needs, both in relation to the ST product alone and when incorporated into a customer end product or application.
- All security features of ST products (inclusive of any hardware, software, documentation, and the like), including but not limited to any enhanced security features added by ST, are provided on an "AS IS" BASIS. AS SUCH, TO THE EXTENT PERMITTED BY APPLICABLE LAW, ST DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, unless the applicable written and signed contract terms specifically provide otherwise.

81 Revision history

Chapter 57.7.19

Table 903. Document revision history

Date	Revision	Changes
28-Nov-2024	1	Initial release.
27-Jan-2025	2	Updated *Table 3: Peripheral register boundary addresses*, *Table 28: Peripheral indexes in IAC*, *Table 55: Functionalities depending on system operating mode*, *Table 63: PWR register map and reset values*, *Table 762: Ethernet peripheral pins*, and *Table 852: Ethernet MAC register map and reset values*. Updated *Section 3.1: Key security features*, *Section 13.2: PWR main features*, *Section 13.5: Power supply supervision*, *Section 13.5.8: Battery voltage thresholds*, *Section 13.9.6: PWR backup domain control register 1 (PWR_BDCR1)*, *Section 14.6.9: General clock concept overview*, *Section 20: Neural-ART accelerator™ (NPU)* and its subsections, *Section 57.7.19: TIM15 alternate function register 1 (TIM15_AF1)*, and *Section 57.8.17: TIMx alternate function register 1 (TIMx_AF1)(x = 16 to 17)*. Updated *Figure 15: Power control block diagram* and *Figure 28: V_{08CAP} thresholds*. Minor text edits across the whole document.

Table 903. Document revision history

Date	Revision	Changes
15-Dec-2025	3	Document scope extended to STM32N645/655xx devices. Updated *Section 2.1.2: Bus architecture*, *Section 2.2.2: NPU_NIC network interconnect*, *Section 7.3: RISAF implementation*, *Configuring base regions in RISAF*, *Section 10.3.1: Internal SRAM features*, *Section 10.3.2: FLEXRAM control*, *Section 12.3: CACHEAXI implementation*, *Section 12.4.8: AXI traffic to slave SRAM port*, *Section 12.7.1: CACHEAXI control register 1 (CACHEAXI_CR1)*, *Section 13.5.5: V_{DDCORE} monitoring*, *Section 14.5.3: NRST reset*, *Section 26.3.3: CRC operation*, *Section 29.5.8: A single XSPI drives two external memories*, *Section 32.6.5: ADC configuration register 2 (ADC_CFGR2)*, *Section 37.5.3: High-resolution CSI2 camera and DCMIPP*, *Section 39.4.1: DCMIPP common configuration*, *Section 39.14.15: DCMIPP common frame counter register (DCMIPP_CMFRCR)*, *Section 39.14.18: DCMIPP common status register 2 (DCMIPP_CMSR2)*, *Section 39.14.19: DCMIPP common interrupt clear register (DCMIPP_CMFCR)*, *Section 39.14.98: DCMIPP Pipe1 status register (DCMIPP_P1SR)*, *Section 39.14.99: DCMIPP Pipe1 interrupt clear register (DCMIPP_P1FCR)*, *Section 45.1: VENC introduction*, *Section 47.7.1: RNG control register (RNG_CR)*, *Section 51.6.5: MCE illegal access interrupt enable register (MCE_IAIER)*, *Section 52.3.7: PKA error management*, *Section 52.4.1: Introduction*, *Section 62.2: TAMP main features*, *Section 68.6.5: SAI slot register (SAI_ASLOTR)*, and *Section 68.6.13: SAI slot register (SAI_BSLOTR)*. Register SYSCFG_VDDIOxCCCR replaced by registers SYSCFG_VDDIO2CCCR, SYSCFG_VDDIO3CCCR, SYSCFG_VDDIO4CCCR, and SYSCFG_VDDIO5CCCR. Register SYSCFG_VDDIOxCCSR replaced by registers SYSCFG_VDDIO2CCSR, SYSCFG_VDDIO3CCSR, SYSCFG_VDDIO4CCSR, and SYSCFG_VDDIO5CCSR. Replaced timx_tgo_cktim and timx_tgo2_cktim, respectively, by timx_trgo_cktim and timx_trgo2_cktim in *Section 32: Analog-to-digital converters (ADC)*. Internal signals gpu2d_irq and gpu2d_irqsys renamed, respectively, gpu2d_it and gpu2d_sys_it in *Section 44: Neo-Chrom graphic processor (GPU2D)*. Updated text, tables and figures in *Section 64: Improved inter-integrated circuit (I3C)* and *Section 76: Ethernet (ETH): gigabit media access control (GMAC) with DMA controller*. Updated *Table 1: Memory map based on IDAU mapping*, *Table 2: Memory map and peripheral register boundary addresses*,*Table 3: Peripheral register boundary addresses*, *Table 18: OTP fuse description (lower OTP region)*, *Table 20: RISUP indexes*, *Table 24: RISAF resource assignment*, *Table 28: Peripheral indexes in IAC*, *Table 33: Internal SRAM features*, *Table 50: PWR input/output signals connected to package pins or balls*, *Table 64: RCC input/output signals connected to package pins or balls*, *Table 72: Peripheral clock distribution summary*, *Table 82: SYSCFG register map and reset values*, *Table 83: Connectivity matrix*, *Table 130: GFXTIM internal signals*, *Table 132: Graphic timer interrupt requests*, *Table 134: STM32N6x5/x7xx vector table*, *Table 204: Use cases*, *Table 241: ADC1/2 interconnection*, *Table 364: DCMIPP interrupts*, and *Table 398: RNG configurations*. Updated *Figure 10: FLEXMEM versus retention*, *Figure 17: System supply configurations*, *Figure 58: Clock distribution for XSPIs and MCE1/2/3*, *Figure 131: Neural-ART 14 integration*, *Figure 140: GFXTIM block diagram*, *Figure 177: Synchronous multiplexed read mode waveforms - NOR, PSRAM (CRAM)*, *Figure 178: Synchronous multiplexed write mode waveforms - PSRAM (CRAM)*, figures *220* to *224*, *Figure 263: ADC1 connectivity*, *Figure 264: ADC2 connectivity*, *Figure 387: Camera subsystem clock diagram*, *Figure 541: Combined PWM mode on channel 1 and 3*, *Figure 736: Edge-aligned PWM waveforms (ARR = 8)*, and *Figure 985: SOF connectivity (SOF trigger output to TIM and ITR1 connection)*. Minor text edits across the whole document.

IMPORTANT NOTICE – READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice.

In the event of any conflict between the provisions of this document and the provisions of any contractual arrangement in force between the purchasers and ST, the provisions of such contractual arrangement shall prevail.

The purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgment.

The purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of the purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

If the purchasers identify an ST product that meets their functional and performance requirements but that is not designated for the purchasers’ market segment, the purchasers shall contact ST for more information.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, refer to *www.st.com/trademarks*. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2025 STMicroelectronics – All rights reserved